



Principios de programación

Método Deductivo

El método deductivo aplica principios generales para resolver problemas específicos. En programación, esto se traduce en utilizar algoritmos o fórmulas conocidas para abordar tareas concretas.

Ejemplo: Cálculo del área de un triángulo

Sabemos que el área de un triángulo se calcula con la fórmula general:

$$\text{Área} = \frac{\text{base} \times \text{altura}}{2}$$

Aplicando este principio general, podemos crear un algoritmo en PSeInt que solicite la base y la altura de un triángulo y calcule su área:

Algoritmo CalcularAreaTriangulo

Definir base, altura, area Como Real

Escribir "Ingrese la base del triángulo:"

Leer base

Escribir "Ingrese la altura del triángulo:"

Leer altura

area <- (base * altura) / 2

Escribir "El área del triángulo es: ", area

FinAlgoritmo



Ejemplo: Conversión de temperaturas de Celsius a Fahrenheit

Sabemos que la fórmula general para convertir una temperatura de grados Celsius a grados Fahrenheit es:

$$\text{Fahrenheit} = (\text{Celsius} \times \frac{9}{5}) + 32$$

Aplicando esta fórmula, podemos crear un algoritmo en PSeInt que solicite una temperatura en grados Celsius y la convierta a grados Fahrenheit:

Algoritmo ConvertirCelsiusAFahrenheit

Definir celsius, fahrenheit Como Real

Escribir "Ingrese la temperatura en grados Celsius:"

Leer celsius

fahrenheit <- (celsius * 9 / 5) + 32

Escribir "La temperatura en grados Fahrenheit es: ", fahrenheit

FinAlgoritmo

En este caso, partimos de una fórmula general y la aplicamos a un valor específico ingresado por el usuario.



Método Inductivo

El método inductivo se basa en observar casos específicos para derivar conclusiones generales. En programación, esto implica analizar patrones en datos concretos y desarrollar algoritmos que generalicen dichos patrones.

Ejemplo: Generación de la secuencia de Fibonacci

La secuencia de Fibonacci comienza con 0 y 1, y cada número siguiente es la suma de los dos anteriores. Observando este patrón en casos específicos, podemos generalizar un algoritmo que genere los primeros 'n' números de la secuencia:

Algoritmo GenerarFibonacci

Definir n, i Como Entero

Definir a, b, siguiente Como Entero

Escribir "Ingrese la cantidad de términos de la secuencia de Fibonacci:"

Leer n

a <- 0

b <- 1

Escribir "Secuencia de Fibonacci:"

Escribir a

Escribir b

Para i <- 3 Hasta n Hacer

 siguiente <- a + b

 Escribir siguiente

 a <- b

 b <- siguiente

FinPara

FinAlgoritmo

En este algoritmo, observamos los primeros términos de la secuencia y, a partir de ellos, derivamos una regla general para generar los siguientes términos.

**Ejemplo: Determinar si un número es primo**

Observando números específicos, sabemos que un número primo es aquel que solo es divisible por 1 y por sí mismo. A partir de esta observación, podemos generalizar un algoritmo que determine si un número dado es primo:

Algoritmo VerificarNumeroPrimo

Definir numero, i Como Entero

Definir esPrimo Como Logico

Escribir "Ingrese un número entero positivo:"

Leer numero

esPrimo <- Verdadero

Si numero <= 1 Entonces

esPrimo <- Falso

Sino

Para i <- 2 Hasta numero - 1 Con Paso 1 Hacer

Si numero % i = 0 Entonces

esPrimo <- Falso

Romper

FinSi

FinPara

FinSi

Si esPrimo Entonces

Escribir numero, " es un número primo."

Sino

Escribir numero, " no es un número primo."

FinSi

FinAlgoritmo

En este algoritmo, analizamos el comportamiento de números específicos para identificar un patrón (divisibilidad) y, a partir de ello, desarrollamos una solución general que determina si un número es primo.

Estos ejemplos demuestran cómo los métodos deductivo e inductivo pueden aplicarse en la programación utilizando PSeInt, ya sea partiendo de principios generales o derivando conclusiones a partir de casos específicos.



Lógica

La lógica de programación implica el razonamiento y la capacidad de desglosar un problema en pasos lógicos y secuenciales que una computadora pueda entender y ejecutar. Esto incluye la utilización de estructuras de control, operadores lógicos, variables y funciones para diseñar algoritmos que solucionen problemas de manera efectiva.

Fundamentos de la Lógica de Programación

Para desarrollar una sólida lógica de programación, es fundamental comprender los siguientes conceptos:

Variables y Constantes: Espacios en la memoria que almacenan datos que pueden cambiar (variables) o permanecer constantes durante la ejecución del programa.

Operadores: Símbolos que realizan operaciones sobre variables y constantes. Se dividen en:

- Aritméticos: +, -, *, /, %
- Relacionales: ==, !=, >, <, >=, <=
- Lógicos: && (AND), || (OR), ! (NOT)

Estructuras de Control: Permiten dirigir el flujo de ejecución del programa:

- Condicionales: if, else if, else
- Bucles: for, while, do-while
- Switch: estructura de selección múltiple

Funciones y Procedimientos: Bloques de código reutilizables que realizan tareas específicas y pueden ser llamados desde diferentes partes del programa.

Importancia de la Lógica de Programación

Una buena comprensión de la lógica de programación permite:

- Resolución Eficiente de Problemas: Descomponer problemas complejos en partes más manejables y abordarlas de manera sistemática.
- Código Mantenible y Escalable: Escribir código claro y estructurado facilita su mantenimiento y expansión futura.
- Adaptabilidad: La lógica de programación es independiente del lenguaje; una vez dominada, es más sencillo aprender y adaptarse a diferentes lenguajes de programación.

Algoritmos

Un algoritmo es un conjunto finito y ordenado de instrucciones o pasos que se siguen para resolver un problema específico o realizar una tarea determinada. En el ámbito de la programación, los algoritmos son fundamentales, ya que constituyen la base sobre la cual se desarrollan los programas y aplicaciones informáticas.



Características de los Algoritmos

Para que un conjunto de instrucciones sea considerado un algoritmo, debe cumplir con las siguientes características:

- **Finitud:** El algoritmo debe tener un número limitado de pasos; es decir, debe concluir en algún momento.
- **Precisión:** Cada paso debe estar claramente definido y no debe dar lugar a ambigüedades.
- **Entrada (Input):** Debe aceptar uno o más datos de entrada para procesar.
- **Salida (Output):** Después del procesamiento, debe producir uno o más resultados.
- **Efectividad:** Los pasos deben ser lo suficientemente básicos como para ser ejecutados, en principio, de manera exacta y en un tiempo razonable.

Partes de un Algoritmo

Un algoritmo típico consta de tres partes principales:

- **Entrada:** Datos iniciales necesarios para comenzar el proceso.
- **Proceso:** Conjunto de operaciones o instrucciones que transforman los datos de entrada en resultados.
- **Salida:** Resultados obtenidos después de ejecutar el proceso.

Representación de Algoritmos

Los algoritmos pueden representarse de diversas formas, entre las más comunes se encuentran:

- **Pseudocódigo:** Descripción textual que utiliza una sintaxis similar a los lenguajes de programación, pero más abstracta y orientada a la comprensión humana.
- **Diagramas de Flujo:** Representación gráfica que utiliza símbolos para ilustrar el flujo de ejecución de las instrucciones.

Importancia de los Algoritmos en Programación

Los algoritmos son esenciales en la programación debido a que:

- **Eficiencia:** Un buen algoritmo optimiza el uso de recursos como tiempo y memoria.
- **Reutilización:** Pueden aplicarse a diferentes problemas con modificaciones mínimas.
- **Mantenibilidad:** Facilitan la comprensión y modificación del código.
- **Escalabilidad:** Permiten manejar incrementos en la cantidad de datos o en la complejidad del problema de manera efectiva.



Estructura Básica de un Algoritmo en PSeInt

Un algoritmo en PSeInt sigue una estructura general que incluye:

- **Nombre del Algoritmo:** Identifica el propósito del algoritmo.
- **Declaración de Variables:** Define las variables que se utilizarán .
- **Instrucciones:** Secuencia de pasos o acciones a ejecutar.
- **Fin del Algoritmo:** Indica la terminación del algoritmo.

Ejemplo de estructura básica:

Algoritmo NombreDelAlgoritmo

// Declaración de variables

Definir variable1 Como Tipo

Definir variable2 Como Tipo

// Instrucciones

Escribir "Mensaje al usuario"

Leer variable1

variable2 <- variable1 + 10

Escribir "El resultado es: ", variable2

FinAlgoritmo



Variables

En programación, una variable es un espacio en la memoria de la computadora que se utiliza para almacenar datos que pueden cambiar durante la ejecución de un programa. Cada variable tiene un nombre único que la identifica y un tipo de dato que determina el tipo de información que puede contener, como números, texto o valores booleanos. Las variables son fundamentales en la programación, ya que permiten a los desarrolladores manejar y manipular datos de manera eficiente.

Características de las Variables

Las variables en programación presentan las siguientes características:

Nombre (Identificador): Es el nombre único que se asigna a la variable para referirse a ella en el código. Este nombre debe seguir ciertas reglas, como comenzar con una letra o un guion bajo (_), no contener espacios y evitar el uso de palabras reservadas del lenguaje de programación.

Tipo de Dato: Define el tipo de información que la variable puede almacenar, como números enteros, números con decimales, cadenas de texto o valores lógicos (booleanos).

Valor: Es el contenido actual de la variable, que puede cambiar a lo largo de la ejecución del programa.

Alcance (Scope): Determina la parte del programa donde la variable es accesible. Las variables pueden ser locales (accesibles solo dentro de una función o bloque específico) o globales (accesibles desde cualquier parte del programa).

Tipos de Variables

Los tipos de variables pueden variar según el lenguaje de programación, pero comúnmente se clasifican en:

Variables Numéricas:

Enteras: Almacenan números enteros, sin decimales.

Real: Almacenan números con decimales.

Variables de Carácter:

Carácter: Almacenan un solo símbolo o letra.

Cadena de Caracteres: Almacenan secuencias de caracteres, como palabras o frases.

Variables Booleanas: Almacenan valores lógicos, como verdadero (true) o falso (false).



Declaración de Variables en PSeInt

Antes de utilizar una variable en PSeInt, es necesario declararla, especificando su nombre y tipo de dato. La sintaxis general para declarar una variable es:

Definir nombre_variable Como tipo_de_dato

Por ejemplo, para declarar una variable llamada edad de tipo entero:

Definir edad Como Entero

Tipos de Datos en PSeInt

PSeInt soporta varios tipos de datos básicos:

Entero: Almacena números enteros, sin decimales.

Ejemplo: Definir cantidad Como Entero

Real: Almacena números con decimales.

Ejemplo: Definir precio Como Real

Cadena: Almacena secuencias de caracteres o texto.

Ejemplo: Definir nombre Como Cadena

Lógico: Almacena valores booleanos: Verdadero o Falso.

Ejemplo: Definir esMayorDeEdad Como Lógico

Asignación de Valores a Variables

Una vez declarada una variable, se le puede asignar un valor utilizando el operador <=:

nombre_variable = valor

Por ejemplo:

Definir edad Como Entero

edad = 25



Ejercicios

1. Declara una variable llamada AnioNacimiento y asígnale el valor 2000. Luego, muestra el año de nacimiento en pantalla.
2. Declara una variable llamada CantidadProductos y asígnale el valor 15. Luego, muestra la cantidad en pantalla.
3. Crea un programa que solicite la edad del usuario y luego la muestre en pantalla.
4. Crea un programa que pida al usuario su número favorito y lo muestre en pantalla.
5. Declara una variable llamada Temperatura y asígnale el valor 30.5. Luego, muestra la temperatura en pantalla.
6. Crea un programa que solicite al usuario una frase y luego la muestre en pantalla.
7. Declara una variable llamada Edad y asígnale el valor 25. Luego, muestra la edad en pantalla.
8. Declara una variable llamada Precio y asígnale el valor 19.99. Luego, muestra el precio en pantalla
9. Crea un programa que pida al usuario la ciudad donde vive y la muestre en pantalla.
10. Crea un programa que solicite el nombre y apellido del usuario y luego lo muestre en pantalla.
11. Crea un programa que pida al usuario su color favorito y lo muestre en pantalla.
12. Declara una variable llamada Inicial y asígnale el valor A. Luego, muestra la inicial en pantalla.
13. Crea un programa que solicite al usuario su comida favorita y la muestre en pantalla.
14. Crea un programa que pida al usuario su día de la semana favorito y lo muestre en pantalla.
15. Declara una variable llamada Pais y asígnale el valor Costa Rica. Luego, muestra el país en pantalla.
16. Crea un programa que pida al usuario su país de origen y lo muestre en pantalla.
17. Declara una variable llamada Numero y asígnale el valor 10. Luego, muestra el valor en pantalla.
18. Declara una variable llamada Nombre y asígnale el valor Carlos. Luego, muestra el nombre en pantalla.
19. Declara una variable llamada Calificacion y asígnale el valor 85. Luego, muestra la calificación en pantalla.



Intercambio de Valores entre Variables

1. Intercambio Básico de Dos Variables

Descripción: Tienes dos variables, A y B, con valores iniciales de 5 y 10 respectivamente. Intercambia los valores de manera que A termine con el valor de B y viceversa.

Pasos a seguir:

Declaración de variables:

A = 5

B = 10

Uso de una variable auxiliar:

temp = A (Guarda el valor de A en temp)

A = B (Asigna el valor de B a A)

B = temp (Asigna el valor de temp a B)

Resultado esperado:

A = 10

B = 5

2. Intercambio sin Variable Auxiliar

Descripción: Intercambia los valores de dos variables, X y, sin utilizar una variable auxiliar. Supongamos que X = 15 y Y = 25.

Pasos a seguir:

Declaración de variables:

X = 15

Y = 25

Intercambio utilizando operaciones aritméticas:

X = X + Y (Ahora X = 40)

Y = X - Y (Ahora Y = 15)

X = X - Y (Ahora X = 25)

Resultado esperado:

X = 25

Y = 15



3. Intercambio de Tres Variables

Descripción: Tienes tres variables, A, B y C, con valores 1, 2 y 3 respectivamente. Realiza un intercambio circular de manera que A tome el valor de B, B el valor de C y C el valor de A.

Pasos a seguir:

Declaración de variables:

A = 1

B = 2

C = 3

Uso de una variable auxiliar:

temp = A (Guarda el valor de A en temp)

A = B (Asigna el valor de B a A)

B = C (Asigna el valor de C a B)

C = temp (Asigna el valor de temp a C)

Resultado esperado:

A = 2

B = 3

C = 1



Operadores

En programación, los operadores son símbolos o palabras reservadas que indican al compilador o intérprete la realización de operaciones específicas sobre uno o más operandos. Estos operandos pueden ser variables, constantes o expresiones, y los operadores permiten manipular y procesar datos de diversas maneras.

Tipos de Operadores en Programación

A continuación, se detallan los tipos más comunes de operadores utilizados en programación:

Operadores Aritméticos

Estos operadores se utilizan para realizar operaciones matemáticas básicas:

Suma (+): Adiciona dos valores.

Ejemplo: `resultado = a + b` asigna a resultado la suma de a y b.

Resta (-): Sustraer un valor de otro.

Ejemplo: `diferencia = a - b` asigna a diferencia la resta de b de a.

Multiplicación (*): Multiplica dos valores.

Ejemplo: `producto = a * b` asigna a producto el resultado de multiplicar a por b.

División (/): Divide un valor entre otro.

Ejemplo: `divi = a / b` asigna a divi el resultado de dividir a entre b.

Módulo (% o MOD): Devuelve el residuo de la división entera.

Ejemplo: `resto = a MOD b` asigna a resto el residuo de dividir a entre b.

Trunc o DIV : Devuelve el cociente de la división entera.

Ejemplo: `coci = trunc(a / b) o a DIV b` asigna a coci el cociente de dividir a entre b.

Potenciación (^): Eleva un número a la potencia de otro.

Ejemplo: `potencia = a ^ b` asigna a potencia el resultado de a elevado a la b.

Ejercicios con operadores aritméticos

1. Suma de Variables

Instrucción:

Si $a = 25$ y $b = 15$, calcula el valor de suma utilizando la siguiente expresión:

$\text{suma} = a + b$

Pregunta:

¿Cuál es el valor de suma? 40



— 2. Resta de Variables

Instrucción:

Si $x = 100$ y $z = 45$, determina el valor de resta con la expresión:

$\text{resta} = x - z$

Pregunta:

¿Qué resultado se obtiene en resta? 55

✖ 3. Multiplicación de Variables

Instrucción:

Dados los valores $m = 8$ y $n = 7$, calcula el valor de producto mediante la siguiente operación:

$\text{producto} = m * n$

Pregunta:

¿Cuál es el valor de producto? 56

÷ 4. División de Variables

Instrucción:

Si $p = 48$ y $q = 5$, halla el valor de division usando la expresión:

$\text{division} = p / q$

Pregunta:

¿Cuál es el valor de division? 9.6

÷ 5. División de Variables(Calculo del cociente)

Instrucción:

Si $p = 48$ y $q = 5$, halla el valor de division usando la expresión:

$\text{division} = \text{trunc}(p / q)$ o $p \text{ DIV } q$

Pregunta:

¿Cuál es el valor de division? 9



6. Cálculo del Módulo (Residuo de la División)

Instrucción:

Si $a = 48$ y $b = 5$, calcula el valor de residuo aplicando la operación de módulo:

$\text{residuo} = a \% b$ o $a \text{ MOD } b$

Pregunta:

¿Qué residuo se obtiene al dividir a entre b ? 3

7. Potenciación

Instrucción:

Dadas las variables $\text{base} = 3$ y $\text{exponente} = 4$, determina el valor de potencia con la operación:

$\text{potencia} = \text{base} ^ \text{exponente}$

Pregunta:

¿Cuál es el resultado de elevar base a la potencia de exponente ? 81

Ejercicios en Pseint

1. Declare las variables $n1$, $n2$ y sum con el tipo de dato entero. Asigna el valor 56 a la variable $n1$ y el valor 98 a la variable $n2$. Realiza la suma de $n1$ y $n2$ y almacena el resultado en la variable sum . Muestre en pantalla el resultado de la suma, acompañado de un mensaje descriptivo.
2. Declare las variables $n1$, $n2$ y rest con el tipo de dato entero. Asigna el valor 115 a la variable $n1$ y el valor 45 a la variable $n2$. Realiza la resta de $n1$ y $n2$ y almacena el resultado en la variable rest . Muestre en pantalla el resultado de la resta, acompañado de un mensaje descriptivo.
3. Declare las variables $n1$, $n2$ y multi con el tipo de dato entero. Asigna el valor 456 a la variable $n1$ y el valor 85 a la variable $n2$. Realiza la multiplicación de $n1$ y $n2$ y almacena el resultado en la variable multi . Muestre en pantalla el resultado de la multiplicación, acompañado de un mensaje descriptivo.



4. Declare las variables n1, n2 y divi con el tipo de dato real para permitir resultados decimales. Asigna el valor 1024 a la variable n1 y el valor 8 a la variable n2. Realiza la división de n1 entre n2 y almacena el resultado en la variable divi. Muestra en pantalla el resultado de la división, acompañado de un mensaje descriptivo.
5. Declare las variables n1, n2 y residuo con el tipo de dato entero. Asigna el valor 157 a la variable n1 y el valor 10 a la variable n2. Realiza la operación módulo (MOD) para obtener el residuo de la división de n1 entre n2 y almacena el resultado en la variable residuo. Muestra en pantalla el resultado de la operación módulo, acompañado de un mensaje descriptivo.
6. Declare las variables n1, n2 y cociente con el tipo de dato entero. Asigna el valor 157 a la variable n1 y el valor 10 a la variable n2. Calcula el cociente (parte entera de la división) de n1 entre n2 utilizando la función Trunc o DIV y almacena el resultado en la variable cociente. Muestra en pantalla el resultado del cociente, acompañado de un mensaje descriptivo.
7. Declare las variables base, exponente y resultado con el tipo de dato real. Asigna el valor 5 a la variable base y el valor 3 a la variable exponente. Realiza la potenciación de base elevado a exponente utilizando el operador ^ y almacena el resultado en la variable resultado. Muestra en pantalla el resultado de la potenciación, acompañado de un mensaje descriptivo.
8. Solicitar al usuario que ingrese dos números. Almacenar los números ingresados en las variables n1 y n2. Realizar la suma de los números y almacenar el resultado en la variable suma. Mostrar en pantalla el resultado de la suma, acompañado de un mensaje descriptivo que brinde contexto al usuario.
9. Solicitar al usuario que ingrese dos números. Almacenar los números ingresados en las variables n1 y n2. Realizar la resta del primer número menos el segundo y almacenar el resultado en la variable resta. Mostrar en pantalla el resultado de la resta, acompañado de un mensaje descriptivo que brinde contexto al usuario.



10. Solicitar al usuario que ingrese dos números. Almacenar los números ingresados en las variables n1 y n2. Realizar la multiplicación de los números y almacenar el resultado en la variable multi. Mostrar en pantalla el resultado de la multiplicación, acompañado de un mensaje descriptivo que brinde contexto al usuario.
11. Solicitar al usuario que ingrese dos números. Almacenar los números ingresados en las variables n1 y n2. Realizar la división de n1 entre n2 y almacenar el resultado en la variable divi. Mostrar en pantalla el resultado de la división, acompañado de un mensaje descriptivo que brinde contexto al usuario.
12. Solicitar al usuario que ingrese dos números enteros. Almacenar los números ingresados en las variables n1 y n2. Realizar la operación módulo (MOD) para obtener el residuo de la división de n1 entre n2 y almacenar el resultado en la variable residuo. Mostrar en pantalla el resultado de la operación módulo, acompañado de un mensaje descriptivo que brinde contexto al usuario.
13. Solicitar al usuario que ingrese dos números enteros. Almacenar los números ingresados en las variables n1 y n2. Calcular el cociente (parte entera de la división) de n1 entre n2 utilizando la función Trunc o la operación DIV y almacena el resultado en la variable cociente. Muestra en pantalla el resultado del cociente, acompañado de un mensaje descriptivo que brinde contexto al usuario.
14. Solicitar al usuario que ingrese la base y el exponente. Almacenar los números ingresados en las variables base y exponente. Calcular la potenciación de base elevado a exponente utilizando el operador ^ y almacenar el resultado en la variable resultado. Mostrar en pantalla el resultado de la potenciación, acompañado de un mensaje descriptivo que brinde contexto al usuario



Operadores Relacionales

Permiten comparar dos valores y devuelven un resultado lógico (Verdadero o Falso):

Igual que (=): Verifica si dos valores son iguales.

Ejemplo: $a = b$ es Verdadero si a es igual a b .

Distinto de (<> o !=): Verifica si dos valores son diferentes.

Ejemplo: $a <> b$ es Verdadero si a es diferente de b .

Mayor que (>): Verifica si un valor es mayor que otro.

Ejemplo: $a > b$ es Verdadero si a es mayor que b .

Menor que (<): Verifica si un valor es menor que otro.

Ejemplo: $a < b$ es Verdadero si a es menor que b .

Mayor o igual que (>=): Verifica si un valor es mayor o igual a otro.

Ejemplo: $a >= b$ es Verdadero si a es mayor o igual a b .

Menor o igual que (<=): Verifica si un valor es menor o igual a otro.

Ejemplo: $a <= b$ es Verdadero si a es menor o igual a b .

Ejercicios

$a = 10$

$b = 20$

$a < b$:

$x = 50$

$y = 50$

$x = y$:

$m = 30$

$n = 25$

$m >= n$:

$p = 7$

$q = 14$

$p < q$:



c = 100

d = 50

c != d:

g = 0

h = 1

g <= h:

edad1 = 18

edad2 = 21

edad1 > edad2:

a = 10

b = 10

a =b:

x = 5

y = 3

x != y:

c = 25

d = 25

c >= d:

e = 10

f = 15

e <= f:



Operadores Lógicos

Se utilizan para combinar expresiones lógicas:

Y (AND): Devuelve Verdadero si ambas expresiones son verdaderas.

Ejemplo: $(a > 0)$ Y $(b < 5)$ es Verdadero si a es mayor que 0 y b es menor que 5.

Tabla de la verdad:

A	B	A AND B
Falso	Falso	Falso
Falso	Verdadero	Falso
Verdadero	Falso	Falso
Verdadero	Verdadero	Verdadero

O (OR): Devuelve Verdadero si al menos una de las expresiones es verdadera.

Ejemplo: $(a > 0)$ O $(b < 5)$ es Verdadero si a es mayor que 0 o b es menor que 5.

Tabla de la verdad:

A	B	A OR B
Verdadero	Verdadero	Verdadero
Verdadero	Falso	Verdadero
Falso	Verdadero	Verdadero
Falso	Falso	Falso

No ($\sim$ o NO): Invierte el valor lógico de la expresión.

Ejemplo: $\text{NO}(a > 0)$ es Verdadero si a no es mayor que 0.

Tabla de la verdad:

Estado inicial	Resultado (NOT)
Verdadero	Falso
Falso	Verdadero



Ejercicios

$n = 15$

$(n \geq 10) \text{ AND } (n \leq 20)$:

$y = 4$

$(y \leq 5) \text{ OR } (y \geq 15)$:

$x = 8$

$(x > 0) \text{ AND } (x \% 2 = 0)$:

$z = 9$

$(z \% 3 = 0) \text{ OR } (z \% 5 = 0)$:

$a = 0$

$\text{NOT } (a < 0)$:

$\text{edad} = 20$

$\text{ciudad} = \text{"San José"}$

$(\text{edad} > 18) \text{ AND } (\text{ciudad} = \text{"San José"})$:

$\text{membresia} = \text{False}$

$\text{monto} = 120$

$(\text{membresia} = \text{True}) \text{ OR } (\text{monto} > 100)$:

$\text{hora} = 18$

$(\text{hora} < 9) \text{ OR } (\text{hora} > 17)$



es_estudiante = True

tiene_id_valida = False

(es_estudiante = True) AND (tiene_id_valida = True)

tiene_licencia = True

vehiculo_asegurado = True

(tiene_licencia = True) AND (vehiculo_asegurado = True)

Prioridad de los operadores

La prioridad de operadores determina el orden en que se evalúan las operaciones en una expresión. Comprender esta jerarquía es esencial para escribir algoritmos que produzcan los resultados esperados. Una computadora, para dar el resultado de una operación matemática, realiza cálculos teniendo siempre en cuenta el orden de los operadores que la conforman. Este orden es el que se conoce como jerarquía de los operadores, la cual efectúa las operaciones de forma estricta a como se muestra a continuación:

Jerarquía de Operadores:

1. **Paréntesis ()**
2. **Potenciación ^**
3. **Multiplicación *, División decimal /**
4. **Division Entera (Div) y Operador Modulo (Mod)**
5. **Suma y Resta**
6. **Operadores Relacionales**
7. **Operadores Lógicos**

Al resolver cualquier expresión matemática, se desarrollan las operaciones que están entre paréntesis, y en el interior se van aplicando cada uno de los niveles que se muestran. Es importante tener en cuenta que un paréntesis desaparece cuando no tenga operaciones para realizar en su interior.

Luego, se repasan los niveles de arriba hacia abajo hasta reducir a un solo término.

Ejemplos de jerarquía de los operadores

Para realizar cualquier operación matemática es necesario utilizar la jerarquía de los operadores antes mencionada, debido a que esta indica el orden a seguir en el desarrollo de cada una de las operaciones, dependiendo de los operadores aritméticos que la componen.

Ejemplo 1. La expresión $2 + 5 * 3$ podrá generar duda al momento de evaluar cuál operación se ejecuta primero: ¿multiplicación o suma?, si se suma primero $2 + 5$ da como resultado 7 y al multiplicarlo por 3, da como respuesta 21; pero si primero se multiplica $5 * 3$ da como resultado 15 y luego de sumarle 2, da como respuesta 17. Entonces, ¿cuál es la respuesta correcta: 21 o 17?



Para no tener dudas al momento de resolver una operación matemática, se deben seguir las instrucciones de la jerarquía de los operadores, donde se observa que primero se ejecuta la multiplicación y luego la suma, por lo tanto, la respuesta correcta será 17, quedando así:

$$2 + 5 * 3$$

$$2 + 15$$

$$17$$

Ejemplo 2. Para dar el resultado a la expresión $40 / 5 + 8 ^ 2 * 3$, se debe hacer lo siguiente:

$40 / 5 + 8 ^ 2 * 3$ Primero se resuelve la potenciación, que es el operador de mayor jerarquía (debido a que no tiene paréntesis), dando como resultado: $40 / 5 + 64 * 3$.

$40 / 5 + 64 * 3$ En este momento se pasa a dar solución a la división, que está al mismo nivel que la multiplicación, puesto que operaciones del mismo nivel se desarrollan de izquierda a derecha, dando como resultado: $8 + 64 * 3$.

$8 + 64 * 3$ Luego se resuelve la multiplicación, que está por encima de la suma en la jerarquía, dando como resultado: $8 + 192$.

$8 + 192$ Por último, se realiza la suma dando como resultado 200

Ejemplo 3:

$$5+3\times(2+4)$$

Solución:

Paréntesis primero: $2+4=6$

Multiplicación: $3\times6=18$

Suma final: $5+18=23$

Resultado: 23

Ejemplo 4:

$$(7+2)\times3-4\div2$$

Solución:

Paréntesis: $7+2=9$

Multiplicación: $9\times3=27$

División: $4\div2=2$

Resta final: $27-2=25$

Resultado: 25



Ejemplo 5:

Expresión: $12 \div (2+4) + 5 \times 3$

Solución:

Paréntesis: $2+4=6$

División: $12 \div 6=2$

Multiplicación: $5 \times 3=15$

Suma final: $2+15=17$

Resultado: 17

Ejercicios resueltos

1. **Ejercicio 1:** $3^2 + (5 \times 4) - 8/2 = 25$
2. **Ejercicio 2:** $(2+3)^3 - 4 \times (6-2) + 10/5 = 111$
3. **Ejercicio 3:** $7 + 4 \times 3^2 - (9/3) = 40$
4. **Ejercicio 4:** $(8/2)^2 + 5 \times (3+7) - 6 = 60$
5. **Ejercicio 5:** $5 \times (2^3 + 4) - 9 + 12/3 = 55$
6. **Ejercicio 6:** $(10-3) \times 2^2 + 6/2 = 31$
7. **Ejercicio 7:** $9 + 3 \times (4^2 - 7) - 8/4 = 34$
8. **Ejercicio 8:** $(6+2)/2 + 3^3 - 5 \times 2 = 21$
9. **Ejercicio 9:** $4 \times (5+3)^2 - 7/1 = 249$
10. **Ejercicio 10:** $(9-4) \times 3 + 2^3/2 = 19$



Condicionales

En programación, las estructuras condicionales son fundamentales para la toma de decisiones dentro de un algoritmo. Permiten que un programa ejecute diferentes bloques de código según se cumplan o no ciertas condiciones, haciendo que el flujo de ejecución sea dinámico y adaptable a diversas situaciones.

Los ejemplos que se muestran a continuación presentan condiciones expresadas en español y su respectiva conversión a expresiones lógicas que se utilizan en programación o pseudocódigo. Estas condiciones suelen usarse en estructuras condicionales como if en muchos lenguajes de programación.

Ejemplos Destacados

Condiciones sobre edad y estado civil

Lenguaje natural: Si una persona es mayor de edad

Algoritmo: Si (Edad $\geq$ 18) Entonces

Se usa la comparación $\geq$ 18 para verificar si la edad es 18 o más.

Lenguaje natural: Si una persona es casada

Algoritmo: Si (EstadoCivil = "Casado") Entonces

Se usa una comparación de igualdad para determinar si el estado civil es "Casado".

Condiciones sobre números

Lenguaje natural: Si un número es positivo

Algoritmo: Si (Numero $>$ 0) Entonces

Se usa $>$ 0 para comprobar si el número es mayor que cero.

Lenguaje natural: Si un número es par

Algoritmo: Si (Numero Mod 2 = 0) Entonces

Se usa Mod 2 = 0 para verificar si el residuo de la división entre 2 es cero, lo que indica que es un número par.

Lenguaje natural: Si un número es múltiplo de 5

Algoritmo: Si (Numero Mod 5 = 0) Entonces

Se usa Mod 5 = 0 para comprobar si el número es divisible por 5.

Lenguaje natural: Si un número no es múltiplo de 3

Algoritmo: Si (Numero Mod 3 $\neq$ 0) Entonces

Se usa $\neq$ (diferente de) para verificar que el número no sea divisible por 3.



Condiciones sobre intervalos y rangos

Lenguaje natural: Si un número está entre 10 y 20

Algoritmo: Si (Numero $\geq$ 10 Y Numero $\leq$ 20) Entonces

Se usa una doble condición con el operador Y (AND en algunos lenguajes) para comprobar que el número esté dentro del rango.

Lenguaje natural: Si una nota está entre 2.0 y 4.0

Algoritmo: Si (Nota $\geq$ 2.0 Y Nota $\leq$ 4.0) Entonces

Similar al caso anterior, se verifica que la nota esté dentro del intervalo de valores permitidos.

Condiciones sobre figuras geométricas

Lenguaje natural: Si un triángulo es equilátero

Algoritmo: Si ($L1 = L2$ Y $L1 = L3$) Entonces

Se verifica que los tres lados sean iguales.

Lenguaje natural: Si un triángulo es escaleno

Algoritmo: Si ($L1 \neq L2$ Y $L1 \neq L3$ Y $L2 \neq L3$) Entonces

Se comprueba que los tres lados sean diferentes.

Tipos de Estructuras Condicionales

Condicional Simple (Si ... Entonces): Ejecuta un bloque de código únicamente cuando una condición específica es verdadera.

Sintaxis:

Si <condición> Entonces

<instrucciones>

FinSi

Ejemplo:

Algoritmo VerificarMayorDeEdad

Definir edad Como Entero

Escribir "Ingrese su edad:"

Leer edad

Si edad $\geq$ 18 Entonces

Escribir "Usted es mayor de edad."

FinSi

FinAlgoritmo



Ejercicios

1. **Ejercicio 1:** Determinar si un número es positivo. Solicitar al usuario que ingrese un número y determinar si es positivo.
2. **Ejercicio 2:** Verificar si un número es par. Pedir al usuario un número entero y verificar si es par.
3. **Ejercicio 3:** Comprobar si un número es múltiplo de 5. Pedir al usuario un número y verificar si es múltiplo de 5.
4. **Ejercicio 4:** Verificar si una persona es menor de edad. Solicitar la edad de una persona y mostrar un mensaje si es menor de 18 años.
5. **Ejercicio 5:** Determinar si un número es igual a cero. Pedir al usuario que ingrese un número y verificar si es igual a cero.

Condicional Doble (Si ... Entonces ... Sino): Ofrece dos caminos: uno si la condición es verdadera y otro si es falsa.

Sintaxis:

Si <condición> Entonces

<instrucciones_si_verdadero>

Sino

<instrucciones_si_falso>

FinSi

Ejemplo:

Algoritmo VerificarMayorMenorDeEdad

Definir edad Como Entero

Escribir "Ingrese su edad:"

Leer edad

Si edad >= 18 Entonces

Escribir "Usted es mayor de edad."

SiNo

Escribir "Usted es menor de edad."

FinSi

FinAlgoritmo



Ejercicios

1. **Ejercicio 1:** Verificar si un número es par o impar. Pedir al usuario un número entero y verificar si es par o impar.
2. **Ejercicio 2:** Determinar si un número es positivo o negativo. Solicitar al usuario que ingrese un número y determinar si es positivo o negativo.
3. **Ejercicio 3:** Comparar dos números y mostrar el mayor. Pedir al usuario que ingrese dos números y mostrar cuál es el mayor.
4. **Ejercicio 4:** Calcular el descuento en una compra. Solicitar el monto de una compra y aplicar un descuento del 10% si el monto es mayor o igual a \$100; sino no, no se aplica descuento.
5. **Ejercicio 5:** Calcular el índice de masa corporal (IMC). Solicitar al usuario su peso en kilogramos y su altura en metros, luego calcular su IMC y determinar si tiene un peso normal o no. Para un peso normal su IMC debe estar entre 18.5 y 24.9. **Formula para calcular el IMC= peso ÷ altura²**
6. **Ejercicio 6:** Solicitar al usuario el monto de una compra y aplicar un impuesto del 13% si el monto es mayor o igual a \$100. Si es menor, no se aplica el impuesto.
7. **Ejercicio 7:** Calcular una propina en un restaurante. Solicitar el total de la cuenta y calcular una propina si el monto es mayor o igual a \$50 aplicar el 15% sino si el monto es menor a \$50 aplicar el 10%.
8. **Ejercicio 8:** Calcular un aumento de salario. Solicitar el salario actual de un empleado y aplicar un aumento. Si el salario es menor a \$1000, aumentar 10%. Si el salario es mayor o igual a \$1000, aumentar 5%.
9. **Ejercicio 9:** Calcular una bonificación por antigüedad. Solicitar los años de servicio de un empleado y calcular su bonificación. Si tiene 5 años o más, se le otorga un bono del 10% de su salario. Si tiene menos de 5 años, se le otorga un bono del 5%.



Condicional anidada(o)

La estructura condicional anidada permite evaluar múltiples condiciones de forma jerárquica dentro de una estructura Si...Entonces...Sino. Esto significa que dentro de una condición Si, se pueden agregar otras estructuras condicionales Si para evaluar más opciones.

¿Cómo funciona una Condición Anidada?

Cuando se anidan condicionales, el programa evalúa la primera condición, y si es falsa, revisa la siguiente condición dentro de la estructura Sino. Este proceso se repite hasta encontrar una condición verdadera o llegar a la última opción.

Si (condición1) Entonces

// Instrucciones si condición1 es verdadera

Sino

Si (condición2) Entonces

// Instrucciones si condición2 es verdadera

Sino

Si (condición3) Entonces

// Instrucciones si condición3 es verdadera

Sino

// Instrucciones si ninguna de las condiciones anteriores es verdadera

FinSi

FinSi

FinSi

Ejemplo 1: Desarrollar un algoritmo en PSeInt que permita al usuario ingresar una calificación numérica y mostrar su clasificación correspondiente de acuerdo con la siguiente tabla:

Rango de calificación	Clasificación
90 - 100	Excelente
80 - 89	Notable
70 - 79	Bueno
60 - 69	Suficiente
0 - 59	Reprobado



Algoritmo Clasificar_Calificacion

Definir nota Como Entero

Escribir "Ingrese la calificación (0-100):"

Leer nota

Si nota ≥ 90 Entonces

Escribir "Excelente"

Sino

Si nota ≥ 80 Entonces

Escribir "Notable"

Sino

Si nota ≥ 70 Entonces

Escribir "Bueno"

Sino

Si nota ≥ 60 Entonces

Escribir "Suficiente"

Sino

Escribir "Reprobado"

FinSi

FinSi

FinSi

FinSi

FinAlgoritmo



Ejemplo 2: Desarrollar un algoritmo en PSeInt que solicite al usuario que ingrese tres valores correspondientes a los lados de un triángulo y determine su clasificación aplicando las siguientes condiciones:

Si los tres lados son iguales, el triángulo es Equilátero.

Si solo dos lados son iguales, el triángulo es Isósceles.

Si los tres lados son diferentes, el triángulo es Escaleno.

Algoritmo Tipo_de_Triangulo

Definir lado1, lado2, lado3 Como Real

Escribir "Ingrese el primer lado:"

Leer lado1

Escribir "Ingrese el segundo lado:"

Leer lado2

Escribir "Ingrese el tercer lado:"

Leer lado3

Si lado1 = lado2 Y lado2 = lado3 Entonces

Escribir "El triángulo es equilátero"

Sino

Si lado1 = lado2 O lado1 = lado3 O lado2 = lado3 Entonces

Escribir "El triángulo es isósceles"

Sino

Escribir "El triángulo es escaleno"

FinSi

FinSi

FinAlgoritmo



Ejercicios

1. **Ejercicio 1:** Verificar si un número es par, impar o cero. Pedir al usuario un número entero y verificar si es par, impar o cero.
2. **Ejercicio 2:** Determinar si un número es positivo, negativo o cero. Solicitar al usuario que ingrese un número y determinar si es positivo, negativo o cero.
3. **Ejercicio 3:** Comparar tres números y mostrar el mayor. Pedir al usuario que ingrese tres números y mostrar cuál es el mayor.
4. **Ejercicio 4:** Imagina que queremos clasificar a las personas según su edad. Sabemos que dependiendo de la edad, una persona puede ser niño, adolescente, adulto o adulto mayor. El objetivo de este programa es pedirle a un usuario que ingrese su edad y que el programa le diga en qué grupo se encuentra:
 - ✓ Si la edad es menor a 0, mostrar "Edad inválida"
 - ✓ Sino Si la edad es menor de 12 años, mostrar "Eres un niño"
 - ✓ Sino Si la edad es de 12 a 17 años, mostrar "Eres un adolescente"
 - ✓ Sino Si la edad es de 18 a 59 años, mostrar "Eres un adulto"
 - ✓ Sino Si la edad es 60 años o más mostrar "Eres un adulto mayor"
5. **Ejercicio 5:** Imagina que una tienda quiere aplicar descuentos a sus clientes dependiendo del monto que gasten en su compra. Queremos crear un programa que calcule automáticamente el descuento y el total a pagar según las siguientes reglas:
 - ✓ Si el monto es menor a \$50, el cliente no tiene descuento.
 - ✓ Si el monto es entre \$50 y \$100, se aplica un 5% de descuento.
 - ✓ Si el monto es mayor a \$100 pero menor o igual a \$200, se aplica un 10% de descuento.
 - ✓ Si el monto es mayor a \$200, se aplica un 15% de descuento.

El programa debe calcular cuánto es el descuento y cuánto es el total a pagar después del descuento.



Condicional de Selección Múltiple(Según Hacer): Permite evaluar varias condiciones en secuencia hasta encontrar la que sea verdadera.

Segun (variable) Hacer

Opcion 1:

// Instrucciones para la opción 1

Opcion 2:

// Instrucciones para la opción 2

De Otro Modo:

// Instrucciones si ninguna opción coincide

FinSegun



Ejemplos Prácticos en PSeInt

Algoritmo Menu_DiasDeLaSemana

Definir dia Como Entero

Escribir "===== MENÚ DÍAS DE LA SEMANA ====="

Escribir "1. Lunes"

Escribir "2. Martes"

Escribir "3. Miercoles"

Escribir "4. Jueves"

Escribir "5. Viernes"

Escribir "6. Sábado"

Escribir "7. Domingo"

Escribir "8. Salir"

Escribir "Ingrese un número del 1 al 7 para ver el día de la semana:"

Leer dia

Borrar Pantalla

Segun dia Hacer

1:

Escribir "Lunes"

2:

Escribir "Martes"

3:

Escribir "Miércoles"

4:

Escribir "Jueves"

5:

Escribir "Viernes"

6:

Escribir "Sábado"



7:

Escribir "Domingo"

8:

Escribir "Presione cualquier tecla para salir del programa"

De Otro Modo:

Escribir "Número inválido. Debe ser un número entre 1 y 7."

FinSegun

FinAlgoritmo

Algoritmo Menu_EstacionesDelAño

Definir opcion Como Entero

Escribir "===== MENÚ DE ESTACIONES ====="

Escribir "1. Primavera"

Escribir "2. Verano"

Escribir "3. Otoño"

Escribir "4. Invierno"

Escribir "5. Salir"

Escribir "Ingrese un número del 1 al 5 para ver la estación:"

Leer opcion

Borrar Pantalla

Segun opcion Hacer

1:

Escribir "Primavera: La estación de las flores y el clima templado."

2:

Escribir "Verano: Días calurosos ideales para la playa y las vacaciones."

3:

Escribir "Otoño: Hojas caen, clima fresco y colores dorados en los árboles."

4:

Escribir "Invierno: Frío, nieve en algunos lugares y ambiente acogedor."



5:

Escribir "Presione cualquier tecla para salir del programa"

De Otro Modo:

Escribir "Número inválido. Debe ser un número entre 1 y 5."

FinSegun

FinAlgoritmo

Algoritmo Menu_ComidaSemanal

Definir opcion Como Entero

Escribir "===== MENÚ DE COMIDAS ====="

Escribir "1. Lunes - Sopa de pollo"

Escribir "2. Martes - Ensalada César"

Escribir "3. Miércoles - Arroz con pollo"

Escribir "4. Jueves - Pasta Alfredo"

Escribir "5. Viernes - Pescado frito"

Escribir "6. Sábado - Hamburguesa casera"

Escribir "7. Domingo - Asado familiar"

Escribir "8. Salir"

Escribir "Ingrese un número del 1 al 7 para ver el plato del día:"

Leer opcion

Borrar Pantalla

Segun opcion Hacer

1:

Escribir "El plato del lunes es: Sopa de pollo."

2:

Escribir "El plato del martes es: Ensalada César."



3:

Escribir "El plato del miércoles es: Arroz con pollo."

4:

Escribir "El plato del jueves es: Pasta Alfredo."

5:

Escribir "El plato del viernes es: Pescado frito."

6:

Escribir "El plato del sábado es: Hamburguesa casera."

7:

Escribir "El plato del domingo es: Asado familiar."

8:

Escribir "Presione cualquier tecla para salir del programa"

De Otro Modo:

Escribir "Número inválido. Debe ser un número entre 1 y 8."

FinSegun

FinAlgoritmo



Ejercicios:

Planteamiento del Problema: Menú de Actividades Semanales

Contexto:

En un centro educativo, se busca fomentar la participación de los estudiantes en actividades extracurriculares para mejorar su bienestar físico, mental y social. Para facilitar la consulta de estas actividades, se necesita desarrollar un programa en PSeInt que muestre un menú interactivo de actividades semanales.

Problema a Resolver:

Los estudiantes desean conocer qué actividad se realizará en un día específico de la semana. Sin embargo, actualmente no tienen una forma rápida y sencilla de acceder a esta información. Por ello, se requiere un programa que permita al usuario ingresar un número correspondiente a un día de la semana y, a cambio, mostrar la actividad programada para ese día.

Menú de Actividades Semanales

El programa debe mostrar el siguiente menú al usuario:

===== MENÚ DE ACTIVIDADES SEMANALES =====

- 1. Lunes - Clase de yoga**
- 2. Martes - Sesión de pintura**
- 3. Miércoles - Caminata al aire libre**
- 4. Jueves - Taller de cocina**
- 5. Viernes - Noche de cine**
- 6. Sábado - Partido de fútbol**
- 7. Domingo - Día de picnic**
- 8. Salir**

Ingrese un número del 1 al 7 para ver la actividad del día:

Después de que el usuario ingrese un número:

Si elige 1, se muestra: "La actividad del lunes es: Clase de yoga."

Si elige 5, se muestra: "La actividad del viernes es: Noche de cine."

Si elige 8, se muestra: "Presione cualquier tecla para salir del programa" y el programa termina.

Si ingresa un número fuera del rango (1-8), el programa debe mostrar un mensaje de advertencia:

"Número inválido. Debe ser un número entre 1 y 8."



Requisitos del Programa

Interfaz amigable: El programa debe presentar un menú con las actividades de cada día de la semana.

Selección por número: El usuario ingresará un número del 1 al 7 para conocer la actividad de un día específico.

Opción de salida: Se debe incluir la opción 8 para salir del programa.

Manejo de errores: Si el usuario ingresa un número fuera del rango (1-8), el programa deberá mostrar un mensaje de advertencia.

Claridad en la salida: Una vez seleccionada una opción válida, el programa deberá limpiar la pantalla antes de mostrar la actividad.

Finalización adecuada: Al elegir la opción 8, el programa debe mostrar un mensaje de despedida antes de cerrarse.



Planteamiento del Problema: Menú de Rutinas de Ejercicio Semanales

Contexto:

Un gimnasio quiere ofrecer a sus clientes una guía rápida sobre las rutinas de ejercicio diarias. Actualmente, los clientes deben preguntar a los entrenadores qué ejercicio realizar cada día, lo que genera demoras. Para solucionar esto, se necesita un programa en PSeInt que muestre un menú interactivo con la rutina de ejercicio de cada día de la semana.

Problema a Resolver:

Los clientes del gimnasio desean conocer qué rutina de ejercicios les corresponde según el día de la semana. Sin embargo, no tienen una forma fácil de obtener esta información. Se requiere un programa que permita al usuario ingresar un número correspondiente a un día de la semana y que le muestre la rutina programada para ese día.

Menú de Rutinas Semanales

El programa debe mostrar el siguiente menú al usuario:

===== MENÚ DE RUTINAS DE EJERCICIO =====

- 1. Lunes - Entrenamiento de piernas**
- 2. Martes - Rutina de espalda y bíceps**
- 3. Miércoles - Cardio y abdominales**
- 4. Jueves - Entrenamiento de pecho y tríceps**
- 5. Viernes - Rutina de cuerpo completo**
- 6. Sábado - Día de estiramientos y yoga**
- 7. Domingo - Descanso activo**
- 8. Salir**

Ingrese un número del 1 al 7 para ver la rutina del día:

Después de que el usuario ingrese un número:

Si elige 1, se muestra: "La rutina del lunes es: Entrenamiento de piernas."

Si elige 5, se muestra: "La rutina del viernes es: Rutina de cuerpo completo."

Si elige 8, se muestra: "Presione cualquier tecla para salir del programa" y el programa termina.

Si ingresa un número fuera del rango (1-8), el programa debe mostrar un mensaje de advertencia:

"Número inválido. Debe ser un número entre 1 y 8."



Requisitos del Programa

Interfaz clara: Debe presentar un menú con las rutinas de cada día de la semana.

Selección por número: El usuario ingresará un número del 1 al 7 para conocer la rutina de un día específico.

Opción de salida: Se debe incluir la opción 8 para salir del programa.

Manejo de errores: Si el usuario ingresa un número fuera del rango (1-8), el programa deberá mostrar un mensaje de advertencia.

Claridad en la salida: Una vez seleccionada una opción válida, el programa deberá limpiar la pantalla antes de mostrar la rutina.

Finalización adecuada: Al elegir la opción 8, el programa debe mostrar un mensaje de despedida antes de cerrarse.



Planteamiento del Problema: Calculadora Básica

Contexto:

Un grupo de estudiantes de programación está aprendiendo sobre operaciones matemáticas básicas y necesita una herramienta sencilla para realizar cálculos sin necesidad de usar una calculadora física o una aplicación avanzada. Para ello, se requiere desarrollar un programa en PSeInt que permita realizar operaciones de suma, resta, multiplicación y división de manera interactiva.

Problema a Resolver:

Actualmente, los estudiantes deben realizar cálculos básicos manualmente o utilizando herramientas externas. Para facilitar su aprendizaje, se necesita un programa que les permita seleccionar una operación matemática e ingresar dos números para obtener el resultado de manera inmediata.

Menú de la Calculadora

El programa debe mostrar el siguiente menú al usuario:

===== CALCULADORA BÁSICA =====

1. Sumar
2. Restar
3. Multiplicar
4. Dividir
5. Salir

Seleccione una opción:

Después de que el usuario elija una opción:

Si elige 1, se le pedirá ingresar dos números y se mostrará el resultado de la suma.

Si elige 2, se le pedirá ingresar dos números y se mostrará el resultado de la resta.

Si elige 3, se le pedirá ingresar dos números y se mostrará el resultado de la multiplicación.

Si elige 4, se le pedirá ingresar dos números y se mostrará el resultado de la división.

Si el divisor es 0, el programa debe mostrar un mensaje de error: "No se puede dividir entre 0."

Si elige 5, se mostrará el mensaje: "Presione cualquier tecla para salir del programa" y el programa terminará.

Si ingresa un número fuera del rango (1-5), el programa deberá mostrar el mensaje:

"Opción inválida. Debe seleccionar un número entre 1 y 5."



Requisitos del Programa

Interfaz clara y simple: El usuario podrá seleccionar la operación que desea realizar.

Entrada de datos: Se solicitarán dos números al usuario para operar.

Validación de la división: Si el divisor es 0, el programa debe evitar la operación y mostrar un mensaje de error.

Opción de salida: Si el usuario elige la opción 5, el programa deberá finalizar correctamente.

Manejo de errores: Si el usuario ingresa una opción incorrecta, el programa mostrará un mensaje de advertencia.

Claridad en la salida: Después de mostrar el resultado, el programa deberá esperar una tecla antes de finalizar.



Asignación de Información

El documento explica cómo se asigna información a una variable en un algoritmo. Se menciona que existen tres formas de asignación:

- ✓ Asignación interna
- ✓ Asignación externa
- ✓ Actualización

El enfoque principal de la asignación interna ocurre cuando se asigna directamente un valor a una variable utilizando el signo igual (=).

Ejemplos de Asignación Interna:

$X = 15 \rightarrow$ Asigna el valor 15 a la variable X.

$X = A \rightarrow$ Asigna el valor de la variable A a X.

El valor que está a la derecha del signo igual se almacena en la variable de la izquierda.

Asignación Externa

La asignación externa ocurre cuando se asigna un valor a una variable mediante la entrada de datos del usuario utilizando la instrucción Leer. Este valor solo se almacena en la variable cuando el programa se ejecuta o cuando se realiza una prueba de escritorio.

En PSeInt, se usa Leer para guardar los datos. La asignación externa permite que el usuario ingrese valores en tiempo de ejecución, lo que hace que los programas sean más interactivos.

Ejemplos de Asignación Externa:

Pedir un número y mostrarlo. Se pide al usuario que ingrese un número y luego se muestra en pantalla.

Algoritmo Asignacion_Externa_Numero

Definir numero Como Entero

Escribir "Ingrese un número:"

Leer numero

Escribir "El número ingresado es: ", numero

FinAlgoritmo



Actualización de variables

La actualización de una variable ocurre cuando su valor cambia en algún momento durante la ejecución del algoritmo. Esto significa que una variable puede tomar diferentes valores a lo largo del programa, dependiendo de cómo se modifique con nuevas asignaciones.

En otras palabras, una variable puede almacenar un valor inicial, pero este valor puede cambiar en cualquier momento si se usa una nueva instrucción de asignación.

Importancia de la Actualización en Programación

En programación, la actualización de variables es muy común y necesaria en diferentes situaciones, como:

- ✓ Contadores: Para llevar la cuenta de cuántas veces ocurre algo.
- ✓ Acumuladores: Para sumar valores en un ciclo o bucle.
- ✓ Cálculos dinámicos: Para cambiar valores en función de operaciones previas.

Aunque los contadores y acumuladores se parecen porque ambos usan variables que se actualizan dentro de un ciclo, tienen propósitos distintos. Aquí te explico la diferencia claramente:

Contador

¿Qué hace? Lleva la cuenta de cuántas veces ocurre algo.

¿Cómo se actualiza? Se incrementa generalmente en 1.

Valor inicial típico: 0.

Es una variable que se utiliza para llevar la cuenta del número de veces que se realiza una operación o se cumple una condición. Por lo general, se incrementa o decrementa en una cantidad constante, comúnmente en una unidad.

Acumulador

¿Qué hace? Suma (o acumula) valores a lo largo del tiempo.

¿Cómo se actualiza? Se incrementa en el valor de una variable.

Valor inicial típico: 0.

Acumulador: Es una variable que acumula el resultado de una operación repetitiva, sumando, restando o multiplicando valores que pueden variar en cada iteración.

Diferencia clave:

El contador incrementa de manera constante, normalmente en 1.

El acumulador incrementa de manera variable, dependiendo del valor que se le suma.



✓ Fórmulas para identificar

12 Contador

✦ Estructura típica:

$\text{contador} = \text{contador} + 1$

¿Qué hace? Suma un valor constante (normalmente 1 puede ser 2 o 3 o 4 dependiendo el programa) cada vez que ocurre un evento.

Uso típico: Llevar la cuenta de cuántas veces pasa algo.

+ Acumulador

✦ Estructura típica:

$\text{acumulador} = \text{acumulador} + \text{valor}$

¿Qué hace? Suma un valor variable (puede ser el número ingresado, una nota, un precio, etc.).

Uso típico: Sumar o acumular cantidades para luego sacar promedios, totales, etc.

Ejercicios y ejemplos del contador

Ejemplo 1: Contador manual (sumar de 1 en 1 sin condicionales ni ciclos)

Enunciado:

Simular un contador que suma 1 en cada paso, hasta llegar a 3 pasos. Mostrar el resultado final.

Algoritmo ContadorManual

Definir contador Como Entero

contador = 0

contador = contador + 1

contador = contador + 1

contador = contador + 1

Escribir "Valor final del contador: ", contador

FinAlgoritmo



Ejemplo 2: Contar pasos en un proceso de cocina

Enunciado:

Simular que estás haciendo una receta que tiene 4 pasos. Usa un contador para representar cuántos pasos has completado. Mostrar el total de pasos realizados.

Algoritmo ContarPasosReceta

Definir contador Como Entero

contador = 0

// Paso 1: Lavar los ingredientes

contador = contador + 1

// Paso 2: Picar los ingredientes

contador = contador + 1

// Paso 3: Cocinar

contador = contador + 1

// Paso 4: Servir

contador = contador + 1

Escribir "Pasos completados en la receta: ", contador

FinAlgoritmo



Ejemplo 3: Contar escenas de una obra de teatro

Enunciado:

Una obra tiene 5 escenas. Simula un contador que registra cuántas escenas se han presentado.

Algoritmo ContarEscenas

Definir escenas_presentadas Como Entero

escenas_presentadas = 0

escenas_presentadas = escenas_presentadas + 1

escenas_presentadas = escenas_presentadas + 1

escenas_presentadas = escenas_presentadas + 1

escenas_presentadas = escenas_presentadas + 1

escenas_presentadas = escenas_presentadas + 1

Escribir "Escenas presentadas: ", escenas_presentadas

FinAlgoritmo

Ejemplo 4: Contar paquetes empacados

Enunciado:

Un trabajador empacó 3 paquetes en la mañana y 2 en la tarde. Usa un contador para calcular cuántos empacó en total.

Algoritmo ContarPaquetes

Definir paquetes Como Entero

paquetes = 0

// Mañana

paquetes = paquetes + 1

paquetes = paquetes + 1

paquetes = paquetes + 1

// Tarde

paquetes = paquetes + 1

paquetes = paquetes + 1

Escribir "Total de paquetes empacados: ", paquetes

FinAlgoritmo



Ejercicio 1

Problema: Operaciones Básicas con Contador de Ejecuciones

Objetivo:

Desarrollar un algoritmo en PSeInt que realice las cuatro operaciones matemáticas básicas (suma, resta, multiplicación y división) entre dos números ingresados por el usuario. Además, el programa debe llevar un conteo de cuántas operaciones se han realizado y mostrar este total al finalizar.

Descripción del problema:

Se requiere implementar un algoritmo que realice las siguientes acciones en secuencia:

Inicialización:

Declarar una variable contador de tipo real e inicializarla en 0. Esta variable llevará la cuenta de las operaciones realizadas.

Declarar las variables n1 y n2 de tipo real para almacenar los números ingresados por el usuario.

Declarar las variables suma, resta, mult y divi de tipo real para almacenar los resultados de las operaciones.

Operaciones matemáticas:

Suma:

Solicitar al usuario que ingrese dos números (n1 y n2).

Calcular la suma de n1 y n2.

Mostrar el resultado de la suma.

Incrementar el contador en 1.

Resta:

Solicitar al usuario que ingrese dos números (n1 y n2).

Calcular la resta de n1 menos n2.

Mostrar el resultado de la resta.

Incrementar el contador en 1.



Multiplicación:

Solicitar al usuario que ingrese dos números ($n1$ y $n2$).

Calcular la multiplicación de $n1$ por $n2$.

Mostrar el resultado de la multiplicación.

Incrementar el contador en 1.

División:

Solicitar al usuario que ingrese dos números ($n1$ y $n2$).

Verificar que $n2$ no sea cero para evitar una división por cero.

Si $n2$ es diferente de cero, calcular la división de $n1$ entre $n2$ y mostrar el resultado.

Si $n2$ es cero, mostrar un mensaje de error indicando que no se puede dividir entre cero.

Incrementar el contador en 1, independientemente de si la división fue válida o no.

Mostrar resultado final:

Al finalizar todas las operaciones, mostrar en pantalla el número total de operaciones realizadas utilizando la variable contador.



Ejercicio 2

Problema: Registro de Estudiantes y Conteo de Inscripciones

Objetivo:

Desarrollar un algoritmo en PSeInt que permita registrar los nombres de una cantidad fija de estudiantes y utilice un contador para llevar la cuenta del número total de estudiantes registrados. Al finalizar, el programa debe mostrar el número total de estudiantes inscritos.

Descripción del problema:

Se requiere implementar un algoritmo que realice las siguientes acciones:

Inicialización:

Declarar una variable contador de tipo entero e inicializarla en 0. Esta variable llevará la cuenta de los estudiantes registrados.

Declarar una variable estudiante de tipo texto para almacenar temporalmente el nombre de cada estudiante.

Registro de estudiantes:

Solicitar al usuario que ingrese el nombre del primer estudiante.

Leer y almacenar el nombre en la variable estudiante.

Incrementar el contador en 1.

Repetir los pasos anteriores para un total de 4 estudiantes.

Mostrar resultado:

Al finalizar las inscripciones, mostrar en pantalla el número total de estudiantes registrados utilizando la variable contador



Ejercicio 3

Problema: Registro de Ingredientes para la Preparación del Gallo Pinto

Objetivo:

Desarrollar un algoritmo en PSeInt que permita al usuario ingresar cinco ingredientes específicos necesarios para la preparación del Gallo Pinto, mostrando cada ingrediente ingresado y utilizando un contador para llevar la cuenta del número total de ingredientes añadidos. Al finalizar, el programa debe mostrar el número total de ingredientes registrados.

Descripción del problema:

Se requiere implementar un algoritmo que realice las siguientes acciones en secuencia:

Inicialización:

Declarar una variable contador de tipo entero e inicializarla en 0. Esta variable llevará la cuenta de los ingredientes registrados.

Declarar una variable ingrediente de tipo texto para almacenar temporalmente el nombre de cada ingrediente.

Registro de ingredientes:

Solicitar al usuario que ingrese el nombre del primer ingrediente.

Leer y almacenar el nombre en la variable ingrediente.

Incrementar el contador en 1.

Mostrar el nombre del ingrediente ingresado.

Añadir una línea en blanco para mejorar la legibilidad de la salida.

Repetir los pasos anteriores para un total de 5 ingredientes.

Mostrar resultado:

Al finalizar las inscripciones, mostrar en pantalla el número total de ingredientes registrados utilizando la variable contador.



Ejercicio 4

Problema: Registro de Productos en el Inventario

Objetivo:

Desarrollar un algoritmo en PSeInt que permita al usuario ingresar los nombres de seis productos para un inventario, mostrando cada producto ingresado y utilizando un contador para llevar la cuenta del número total de productos añadidos. Al finalizar, el programa debe mostrar el número total de productos registrados.

Descripción del problema:

Se requiere implementar un algoritmo que realice las siguientes acciones en secuencia:

Inicialización:

Declarar una variable contador de tipo entero e inicializarla en 0. Esta variable llevará la cuenta de los productos registrados.

Declarar una variable producto de tipo texto para almacenar temporalmente el nombre de cada producto.

Registro de productos:

Solicitar al usuario que ingrese el nombre del primer producto.

Leer y almacenar el nombre en la variable producto.

Incrementar el contador en 1.

Mostrar el nombre del producto ingresado.

Añadir una línea en blanco para mejorar la legibilidad de la salida.

Repetir los pasos anteriores para un total de 6 productos.

Mostrar resultado:

Al finalizar las inscripciones, mostrar en pantalla el número total de productos registrados utilizando la variable contador.



Docente: Msc. Miguel Gómez Sánchez

Ejemplos del acumulador

Acumulador de asignación directa

Ejemplo 1: Cálculo de la suma de tres números

Objetivo: Sumar tres números ingresados por el usuario y mostrar el resultado.

Algoritmo SumaTresNumeros

Definir numero1, numero2, numero3, suma Como Real

Escribir "Ingrese un número:"

Leer numero1

Escribir "Ingrese un número:"

Leer numero2

Escribir "Ingrese un número:"

Leer numero3

$\text{suma} = \text{numero1} + \text{numero2} + \text{numero3}$

Escribir "La suma de los tres números es:", suma

FinAlgoritmo

Ejemplo 2: Cálculo del total de una factura con tres productos

Objetivo: Calcular el total a pagar por tres productos ingresados por el usuario.

Algoritmo TotalFactura

Definir precio1, precio2, precio3, total Como Real

Escribir "Ingrese el precio del producto:"

Leer precio1

Escribir "Ingrese el precio del producto:"

Leer precio2

Escribir "Ingrese el precio del producto:"

Leer precio3

$\text{total} = \text{precio1} + \text{precio2} + \text{precio3}$

Escribir "El total a pagar es:", total

FinAlgoritmo



Acumulador progresivo

Ejemplo 1: Cálculo de la suma de tres números

Objetivo: Sumar tres números ingresados por el usuario y mostrar el resultado.

Algoritmo SumaTresNumeros

Definir numero1, numero2, numero3, suma Como Real

Escribir "Ingrese un número:"

Leer numero1

Escribir "Ingrese un número:"

Leer numero2

Escribir "Ingrese un número:"

Leer numero3

suma = 0

suma = suma + numero1

suma = suma + numero2

suma = suma + numero3

Escribir "La suma de los tres números es:", suma

FinAlgoritmo

Ejemplo 2: Cálculo del total de una factura con tres productos

Objetivo: Calcular el total a pagar por tres productos ingresados por el usuario.

Algoritmo TotalFactura

Definir precio1, precio2, precio3, total Como Real

Escribir "Ingrese el precio del producto:"

Leer precio1

Escribir "Ingrese el precio del producto:"

Leer precio2

Escribir "Ingrese el precio del producto:"

Leer precio3

total= 0

total = total + precio1

total = total + precio2

total = total + precio3

Escribir "El total a pagar es:", total

FinAlgoritmo



Ejercicios

Utilice un acumulador de asignación directa para los siguientes ejercicios:

1. Cálculo del total de puntos en un juego de tres rondas. Objetivo: Calcular el total de puntos obtenidos en tres rondas de un juego.
2. Cálculo del gasto total en un viaje con tres gastos diferentes. Objetivo: Calcular el gasto total de un viaje considerando tres tipos de gastos: transporte, alojamiento y alimentación.
3. Cálculo del total de horas trabajadas en una semana. Objetivo: Calcular el total de horas trabajadas en una semana, sumando las horas trabajadas cada día.

Utilice un acumulador progresivo para los siguientes ejercicios:

1. Cálculo del total de puntos en un juego de tres rondas. Objetivo: Calcular el total de puntos obtenidos en tres rondas de un juego.
2. Cálculo del gasto total en un viaje con tres gastos diferentes. Objetivo: Calcular el gasto total de un viaje considerando tres tipos de gastos: transporte, alojamiento y alimentación.
3. Cálculo del total de horas trabajadas en una semana. Objetivo: Calcular el total de horas trabajadas en una semana, sumando las horas trabajadas cada día.

Ciclo Para (For)

¿Qué es el ciclo Para y cómo funciona?

El ciclo Para en PSeInt es una estructura de control repetitiva (un bucle) que permite ejecutar un bloque de instrucciones un número fijo de veces. A diferencia de otros bucles como Mientras o Repetir, el **bucle Para** está controlado por un contador: la propia estructura inicializa automáticamente una variable de control (contador) con un valor inicial, la incrementa o decrementa en cada iteración según un paso definido y finaliza cuando la variable alcanza un valor final preestablecido.

En otras palabras, la instrucción Para repite la secuencia de instrucciones un número determinado de veces, recorriendo la variable contador desde un valor de inicio hasta un valor de término

En cuanto a su funcionamiento interno, se puede describir en los siguientes pasos básicos:

Inicialización: La variable de control del ciclo toma el valor inicial especificado al comenzar el bucle.

Verificación de condición: Antes de cada iteración (o después de inicializar, según corresponda), se verifica si la variable de control no ha superado el valor final establecido. Si aún está dentro del rango (por ejemplo, $\leq$ valor final para un incremento positivo), el ciclo puede continuar.

Ejecución del cuerpo: Si la condición es verdadera (es decir, si todavía no se alcanzó el límite final), se ejecuta el bloque de instrucciones dentro del ciclo Para.

Actualización (incremento/decremento): Al final de cada iteración, la variable de control se modifica automáticamente sumándole el valor del paso (incremento si es positivo, o decremento si es negativo).

Repetición o fin: Tras incrementar, se vuelve a verificar la condición. El ciclo se repite mientras la variable no haya superado el valor final. Cuando la variable excede el límite final, el bucle termina y la ejecución continúa con la instrucción siguiente a FinPara

Gracias a este mecanismo, Para es ideal cuando se conoce de antemano el número exacto de iteraciones que se deben realizar.

El bucle se autocontrola con su contador, evitando la necesidad de escribir manualmente la lógica de incremento y comparación en cada iteración (como sí ocurre con un Mientras). En resumen, se configura el punto de inicio, el punto final y el tamaño del paso, y PSeInt se encarga de iterar automáticamente hasta completar el ciclo.

Nota: Si el parámetro de paso no se especifica, PSeInt asume por defecto un paso de 1 (incremento unitario en cada iteración)

Además, es posible usar pasos negativos para invertir el sentido del conteo (decrementos). Por ejemplo, para iterar en orden descendente se puede iniciar con un valor mayor y usar un paso negativo – el bucle controlará que la variable vaya disminuyendo hasta que supere (por debajo) el valor final. En ambos casos, el ciclo finalizará automáticamente cuando el contador sobrepase el límite establecido.

Sintaxis del ciclo Para en PSeInt

La sintaxis formal del ciclo Para en PSeInt es muy clara y debe incluir su inicio y fin explícitamente. En forma general, se escribe así:

```
Para <variable> = <valorInicial> Hasta <valorFinal> [Con Paso <paso>] Hacer  
    <instrucciones del bloque>  
FinPara
```

Donde cada parte corresponde a:

Elemento	Descripción
Para	Palabra clave que inicia la estructura repetitiva. Indica el comienzo del bucle.
<variable>	Nombre de la variable de control (contador) que tomará distintos valores en cada iteración. Debe ser una variable numérica (comúnmente definida como Entero).
<- <valorInicial>	Asignación del valor inicial a la variable de control. Indica con qué valor comienza el contador.
Hasta <valorFinal>	Indica el valor final (límite) hasta el cual iterará la variable de control. El bucle se ejecutará mientras la variable no supere este valor (inclusive lo alcanza).
[Con Paso <paso>]	Opcional: especifica el incremento (o decremento) que se le suma a la variable de control en cada iteración. Si se omite, por defecto el paso es 1. Puede ser positivo (conteo ascendente) o negativo (conteo descendente).
Hacer	Palabra clave que indica el comienzo del bloque de instrucciones a repetir. Lo que sigue en líneas subsecuentes serán las acciones que se repiten en cada iteración.
<instrucciones del bloque>	Una o varias instrucciones indentadas debajo del Para, que constituyen el cuerpo del bucle. Pueden incluir cualquier operación válida (cálculos, lecturas, escrituras, llamadas a subprocesos, etc.).
FinPara	Palabra clave de cierre del bucle Para. Señala el final del bloque repetitivo. Ninguna instrucción del bucle debe aparecer después de FinPara.



Docente: Msc. Miguel Gómez Sánchez

Ejemplo simple de sintaxis: Para ilustrar, un bucle que vaya de 1 a 5 imprimiendo el valor del contador puede escribirse como:

Algoritmo EjemploParaBasico

```
Definir i Como Entero      // Declaración de la variable de control
Para i = 1 Hasta 5 con paso 1 Hacer    // i toma valores 1,2,3,4,5 en cada iteración
    Escribir i                // Instrucción repetida: muestra el valor actual de i
FinPara
FinAlgoritmo
```

En este ejemplo, la variable i empieza en 1 y se incrementa automáticamente en 1 en cada iteración hasta llegar a 5 (valor final). El resultado sería la impresión de los números del 1 al 5, uno por línea. Obsérvese cómo la estructura comienza con Para y termina con FinPara, y el cuerpo del bucle (la instrucción Escribir i) está indentado dentro de estos.

Más ejemplos:

1. Escriba un algoritmo que imprima en pantalla los números del 1 al 10 en orden ascendente.

Algoritmo imprimirnumeros1a10

```
Para i=1 Hasta 10 Con Paso 1 Hacer
    escribir i
FinPara
FinAlgoritmo
```

2. Escriba un algoritmo que imprima en pantalla los números del 10 al 1 en orden descendente.

Algoritmo imprimirnumeros1a10

```
Para i=10 Hasta 1 Con Paso -1 Hacer
    escribir i
FinPara
FinAlgoritmo
```

3. Escriba un algoritmo que imprima en pantalla los números pares del 1 al 10.

Algoritmo ImprimirPares1a10

```
Definir i Como Entero;
Para i = 2 Hasta 10 Con Paso 2 Hacer
    Escribir i;
FinPara
FinAlgoritmo
```

Algoritmo ImprimirPares1a10

```
Definir i Como Entero;
Para i = 1 Hasta 10 Con Paso 1 Hacer
    si i mod 2=0 Entonces
        Escribir i
    FinSi
FinPara
FinAlgoritmo
```



4. Realizar un programa que permita al usuario ingresar 3 números y luego muestre cuántos números fueron ingresados.

Algoritmo contarNumeros

Definir i,n,c Como Real

c= 0

Para i=1 Hasta 3 Con Paso 1 Hacer

Escribir "Ingrese un número:"

Leer n

c=c+1

Fin Para

Escribir "La cantidad de números ingresados son:",c

FinAlgoritmo

5. Escriba un algoritmo que imprima en pantalla la cantidad de números pares desde 1 hasta al 10.

Algoritmo ContarPares

Definir i, contador Como Entero;

contador = 0;

Para i = 1 Hasta 20 Hacer

Si i MOD 2 = 0 Entonces

contador = contador + 1;

FinSi

FinPara

Escribir "La cantidad de números pares entre 1 y 20 es: ", contador;

FinAlgoritmo

6. Desarrollar un programa que permita al usuario ingresar 3 números y calcule la suma total de esos números.

Algoritmo sumarnumeros

Definir i,n,s Como Real

s= 0

Para i=1 Hasta 3 Con Paso 1 Hacer

Escribir "Ingrese un número:"

Leer n

s=s+n

Fin Para

Escribir "La suma de los números ingresados es:",s

FinAlgoritmo

7. Realizar un programa que calcule e imprima la suma de los números desde el 1 hasta el 20

Algoritmo SumarNumeros

Definir i, suma Como Entero;

suma = 0;

Para i = 1 Hasta 20 Hacer

suma = suma + i;

FinPara

Escribir "La suma de los números del 1 al 20 es: ", suma;

FinAlgoritmo



8. Realizar un programa que calcule e imprima la suma de los números pares desde el 1 hasta el 20

Algoritmo SumarPares

Definir i, suma Como Entero;

suma = 0;

Para i = 2 Hasta 20 Con Paso 2 Hacer

 suma = suma + i;

FinPara

 Escribir "La suma de los números pares del 1 al 20 es: ", suma;

FinAlgoritmo

Algoritmo SumarPares

Definir i, suma Como Entero;

suma = 0

Para i = 1 Hasta 20 Hacer

 Si $i \text{ MOD } 2 = 0$ Entonces

 suma = suma + i

 FinSi

FinPara

 Escribir "La suma de los números pares del 1 al 20 es: ", suma;

FinAlgoritmo



Ejercicios

1. Desarrolle un algoritmo que solicite un número entero N al usuario y luego muestre en pantalla todos los números desde 1 hasta N en orden ascendente.
2. Desarrolle un algoritmo que solicite un número entero N al usuario y luego muestre en pantalla todos los números desde N hasta 1 en orden descendente.
3. Escriba un algoritmo que imprima en pantalla los números impares del 1 al 30.
4. Desarrolle un algoritmo que solicite un número entero N al usuario y luego muestre en pantalla todos los números pares desde 1 hasta N
5. Desarrolle un algoritmo que solicite un número entero N al usuario y luego muestre en pantalla todos los números impares desde 1 hasta N
6. Escriba un algoritmo que imprima en pantalla la cantidad de números impares desde 1 hasta al 10.
7. Desarrolle un algoritmo que solicite un número entero N al usuario y luego muestre en pantalla la cantidad de números pares desde 1 hasta N
8. Desarrolle un algoritmo que solicite un número entero N al usuario y luego muestre en pantalla la cantidad de números impares desde 1 hasta N
9. Desarrolle un algoritmo que solicite un número entero N al usuario y luego calcule e imprima la suma de los números desde el 1 hasta N
10. Realizar un programa que calcule e imprima la suma de los números impares desde el 1 hasta el 15
11. Desarrolle un algoritmo que solicite un número entero N al usuario y luego calcule e imprima la suma de los números impares desde el 1 hasta N
12. Una tienda desea calcular el total de una factura para un cliente que va a comprar 3 productos. Se requiere un programa que solicite el precio de cada producto, sume los precios y al final muestre el total a pagar.



Ciclo Repetir

El bucle "Repetir...Hasta que" es una estructura de control repetitiva que ejecuta un conjunto de instrucciones al menos una vez y las repite hasta que una condición específica se evalúa como verdadera. Esto significa que la condición de terminación se verifica después de la ejecución del bloque de instrucciones, garantizando que el cuerpo del bucle se ejecute al menos una vez.

Sintaxis en Pseudocódigo

La estructura general del bucle "Repetir...Hasta que" en pseudocódigo es la siguiente:

```
Repetir  
    <instrucciones>  
Hasta Que <condición>
```

Donde:

Repetir: Indica el inicio del bucle.

<instrucciones>: Conjunto de acciones que se ejecutarán en cada iteración.

Hasta Que <condición>: Especifica la condición que, al evaluarse como verdadera, finalizará el bucle.

Funcionamiento

1. Se ejecuta el bloque de instrucciones dentro del bucle.
2. Al finalizar la ejecución del bloque, se evalúa la condición especificada.
3. Si la condición es falsa, el bucle se repite y vuelve al paso 1.
4. Este proceso continúa hasta que la condición se evalúa como verdadera, momento en el cual el bucle termina y el control del programa pasa a la siguiente instrucción después del bucle.

Es importante destacar que, debido a que la condición se evalúa después de ejecutar el bloque de instrucciones, el cuerpo del bucle siempre se ejecutará al menos una vez, independientemente de si la condición es verdadera desde el principio.



Ejemplo en Pseudocódigo

Supongamos que queremos crear un programa que solicite al usuario ingresar un número positivo. Utilizando el bucle "Repetir...Hasta que", el pseudocódigo sería:

Algoritmo SolicitarNumeroPositivo

Definir numero Como Entero;

Repetir

 Escribir "Introduce un número positivo (negativo para terminar) :";

 Leer numero;

Hasta Que numero < 0

Fin Algoritmo

En este ejemplo:

El programa solicita al usuario que ingrese un número positivo.

El bucle se repite hasta que el usuario ingresa un número negativo.

Más ejemplos:

1. Imprimir los números del 1 al 10 en orden ascendente

Algoritmo NumerosAscendentes

Definir numero Como Entero

numero = 1

Repetir

 Escribir numero

 numero = numero + 1

Hasta Que numero > 10

Fin Algoritmo

2. Imprimir los números del 10 al 1 en orden descendente

Algoritmo NumerosDescendentes

Definir numero Como Entero

numero = 10

Repetir

 Escribir numero

 numero = numero - 1

Hasta Que numero < 1

Fin Algoritmo



3. Imprimir los números pares del 1 al 10

Algoritmo NumerosPares

Definir numero Como Entero

numero = 2

Repetir

 Escribir numero

 numero = numero + 2

Hasta Que numero > 10

FinAlgoritmo

Algoritmo NumerosPares

Definir numero Como Entero

numero = 1

Repetir

 Si numero mod 2=0

 Escribir numero

 Finsi

 numero=numero+1

Hasta Que numero > 10

FinAlgoritmo

4. Permitir al usuario ingresar 3 números y mostrar cuántos números fueron ingresados

Algoritmo ContarNumerosIngresados

Definir contador, numero Como Entero

contador = 0

Repetir

 Escribir "Ingrese un número:"

 Leer numero

 contador = contador + 1

Hasta Que contador = 3

Escribir "Cantidad de números ingresados:", contador

Fin Algoritmo



5. Imprimir la cantidad de números pares desde 1 hasta 10

Algoritmo ContarNumerosPares

Definir numero, contadorPares Como Entero

numero = 1

contadorPares = 0

Repetir

Si numero MOD 2 = 0 Entonces

contadorPares = contadorPares + 1

FinSi

numero = numero + 1

Hasta Que numero > 10

Escribir "Cantidad de números pares entre 1 y 10:", contadorPares

FinAlgoritmo

6. Permitir al usuario ingresar 3 números y calcular la suma total de esos números

Algoritmo SumarTresNumeros

Definir contador, numero, suma Como Entero

contador = 0

suma = 0

Repetir

Escribir "Ingrese un número:"

Leer numero

suma = suma + numero

contador = contador + 1

Hasta Que contador = 3

Escribir "La suma total de los números ingresados es:", suma

FinAlgoritmo

7. Calcular e imprimir la suma de los números desde el 1 hasta el 20

Algoritmo SumarUnoAlVeinte

Definir numero, suma Como Entero

numero = 1

suma = 0

Repetir

suma = suma + numero

numero = numero + 1

Hasta Que numero > 20

Escribir "La suma de los números del 1 al 20 es:", suma

FinAlgoritmo



8. Calcular e imprimir la suma de los números pares desde el 1 hasta el 20

Algoritmo SumarNumerosPares

Definir numero, sumaPares Como Entero

numero = 2

sumaPares = 0

Repetir

 sumaPares = sumaPares + numero

 numero = numero + 2

Hasta Que numero > 20

Escribir "La suma de los números pares del 1 al 20 es:", sumaPares

FinAlgoritmo

Algoritmo SumarNumerosPares

Definir numero, sumaPares Como Entero

numero = 1

sumaPares = 0

Repetir

 Si numero mod 2=0

 sumaPares = sumaPares + numero

 Finsi

 numero=numero+1

Hasta Que numero > 20

Escribir "La suma de los números pares del 1 al 20 es:", sumaPares

FinAlgoritmo



Ejercicios

1. Desarrolle un algoritmo que solicite un número entero N al usuario y luego muestre en pantalla todos los números desde 1 hasta N en orden ascendente.
2. Desarrolle un algoritmo que solicite un número entero N al usuario y luego muestre en pantalla todos los números desde N hasta 1 en orden descendente.
3. Escriba un algoritmo que imprima en pantalla los números impares del 1 al 30.
4. Desarrolle un algoritmo que solicite un número entero N al usuario y luego muestre en pantalla todos los números pares desde 1 hasta N
5. Desarrolle un algoritmo que solicite un número entero N al usuario y luego muestre en pantalla todos los números impares desde 1 hasta N
6. Escriba un algoritmo que imprima en pantalla la cantidad de números impares desde 1 hasta al 10.
7. Desarrolle un algoritmo que solicite un número entero N al usuario y luego muestre en pantalla la cantidad de números pares desde 1 hasta N
8. Desarrolle un algoritmo que solicite un número entero N al usuario y luego muestre en pantalla la cantidad de números impares desde 1 hasta N
9. Desarrolle un algoritmo que solicite un número entero N al usuario y luego calcule e imprima la suma de los números desde el 1 hasta N
10. Realizar un programa que calcule e imprima la suma de los números impares desde el 1 hasta el 15
11. Desarrolle un algoritmo que solicite un número entero N al usuario y luego calcule e imprima la suma de los números impares desde el 1 hasta N
12. Una tienda desea calcular el total de una factura para un cliente que va a comprar 3 productos. Se requiere un programa que solicite el precio de cada producto, sume los precios y al final muestre el total a pagar.



Ciclo Mientras

El ciclo Mientras (equivalente al while en otros lenguajes) permite ejecutar repetidamente un bloque de instrucciones mientras una condición determinada sea verdadera. Es especialmente útil cuando no se conoce de antemano cuántas veces debe repetirse una acción.

Mientras <condición> Hacer
<instrucciones>
FinMientras

<condición>: Una expresión lógica que se evalúa antes de cada iteración.

<instrucciones>: El bloque de código que se ejecutará repetidamente mientras la condición sea verdadera.

Importante: Si la condición es falsa desde el principio, el bloque de instrucciones no se ejecutará ni una sola vez.

Ejemplo práctico: Contador simple

Algoritmo ContadorSimple

Definir contador Como Entero

contador = 1

Mientras contador <= 5 Hacer

Escribir "Contador: ", contador

contador = contador + 1

FinMientras

FinAlgoritmo

Explicación:

Se inicializa la variable contador en 1.

Mientras contador sea menor o igual a 5, se imprime su valor y se incrementa en 1.

El ciclo se repite hasta que contador sea mayor que 5.



Ejemplo con acumulador: Suma de números positivos

Algoritmo SumaNumeros

Definir numero, suma Como Entero

suma = 0

Escribir "Ingrese un número positivo (negativo para terminar):"

Leer numero

Mientras numero >= 0 Hacer

 suma = suma + numero

 Escribir "Ingrese otro número positivo (negativo para terminar):"

 Leer numero

FinMientras

Escribir "La suma total es: ", suma

FinAlgoritmo

Explicación:

Se solicita al usuario ingresar números positivos.

Cada número ingresado se suma a la variable suma.

El ciclo continúa hasta que el usuario ingresa un número negativo.

Más ejemplos

1. Imprimir los números del 1 al 10 en orden ascendente

Algoritmo ImprimirNumerosDel1Al10

Definir numero Como Entero

numero = 1

Mientras numero <= 10 Hacer

 Escribir numero

 numero = numero + 1

FinMientras

Fin Algoritmo



2. Imprimir los números del 10 al 1 en orden descendente

Algoritmo ImprimirDescendente

Definir numero Como Entero

numero = 10

Mientras numero $\geq$ 1 Hacer

 Escribir numero

 numero = numero - 1

FinMientras

FinAlgoritmo

3. Imprimir los números pares del 1 al 10

Algoritmo ImprimirPares1a10

Definir numero Como Entero

numero = 1

Mientras numero $\leq$ 10 Hacer

 Si numero MOD 2 = 0 Entonces

 Escribir numero

 FinSi

 numero = numero + 1

FinMientras

FinAlgoritmo

4. Permitir al usuario ingresar 3 números y mostrar cuántos números fueron ingresados

Algoritmo ContarTresNumeros

Definir numero, contador Como Entero

contador = 0

 Mientras contador < 3 Hacer

 Escribir "Ingrese un número:"

 Leer numero

 contador = contador + 1

 FinMientras

 Escribir "Se ingresaron ", contador, " números."

FinAlgoritmo

5. Imprimir la cantidad de números pares desde 1 hasta 10

Algoritmo ContarPares1a10

Definir numero, contador Como Entero

numero = 1

contador = 0

Mientras numero $\leq$ 10 Hacer

 Si numero MOD 2 = 0 Entonces

 contador = contador + 1

 FinSi

 numero = numero + 1

FinMientras

 Escribir "Cantidad de números pares entre 1 y 10: ", contador

FinAlgoritmo



6. Permitir al usuario ingresar 3 números y calcular la suma total de esos números

Algoritmo SumarTresNumeros

Definir numero, suma, contador Como Entero

suma = 0

contador = 1

Mientras contador <= 3 Hacer

 Escribir "Ingrese el número ", contador, ":"

 Leer numero

 suma = suma + numero

 contador = contador + 1

FinMientras

Escribir "La suma total de los 3 números es: ", suma

FinAlgoritmo

7. Calcular e imprimir la suma de los números desde el 1 hasta el 20

Algoritmo Sumar1a20

Definir numero, suma Como Entero

numero = 1

suma = 0

 Mientras numero <= 20 Hacer

 suma = suma + numero

 numero = numero + 1

FinMientras

 Escribir "La suma de los números del 1 al 20 es: ", suma

FinAlgoritmo

8. Calcular e imprimir la suma de los números pares desde el 1 hasta el 20

Algoritmo SumarPares1a20

Definir numero, suma Como Entero

numero = 1

suma = 0

Mientras numero <= 20 Hacer

 Si numero MOD 2 = 0 Entonces

 suma = suma + numero

 FinSi

 numero = numero + 1

FinMientras

Escribir "La suma de los números pares del 1 al 20 es: ", suma

FinAlgoritmo



Ejercicios

1. Desarrolle un algoritmo que solicite un número entero N al usuario y luego muestre en pantalla todos los números desde 1 hasta N en orden ascendente.
2. Desarrolle un algoritmo que solicite un número entero N al usuario y luego muestre en pantalla todos los números desde N hasta 1 en orden descendente.
3. Escriba un algoritmo que imprima en pantalla los números impares del 1 al 30.
4. Desarrolle un algoritmo que solicite un número entero N al usuario y luego muestre en pantalla todos los números pares desde 1 hasta N
5. Desarrolle un algoritmo que solicite un número entero N al usuario y luego muestre en pantalla todos los números impares desde 1 hasta N
6. Escriba un algoritmo que imprima en pantalla la cantidad de números impares desde 1 hasta al 10.
7. Desarrolle un algoritmo que solicite un número entero N al usuario y luego muestre en pantalla la cantidad de números pares desde 1 hasta N
8. Desarrolle un algoritmo que solicite un número entero N al usuario y luego muestre en pantalla la cantidad de números impares desde 1 hasta N
9. Desarrolle un algoritmo que solicite un número entero N al usuario y luego calcule e imprima la suma de los números desde el 1 hasta N
10. Realizar un programa que calcule e imprima la suma de los números impares desde el 1 hasta el 15
11. Desarrolle un algoritmo que solicite un número entero N al usuario y luego calcule e imprima la suma de los números impares desde el 1 hasta N
12. Una tienda desea calcular el total de una factura para un cliente que va a comprar 3 productos. Se requiere un programa que solicite el precio de cada producto, sume los precios y al final muestre el total a pagar.

Evolución de la Web y el Lenguaje de Marcado de Documentos Web

Tabla de contenido

Concepto de Internet y Web	3
Internet en la industria	3
Evolución de la World Wide Web	4
Fundamentos de la Web	5
Secuencias de comandos del lado del cliente y del servidor	5
Jerarquía de datos	6
Sistemas Operativos	7
Lenguajes de programación.....	7
Tecnologías de objetos (Programación orientada a objetos).....	8
Lenguaje de Marcado (HTML)	9
Atributos en HTML	10
Componentes para formularios	11
Elementos de entrada (tipos de <input>)	12
Listas en HTML.....	14
Autocompletado	15
Estructura de una página web	17

Concepto de Internet y Web

Internet es una red global de redes de computadoras que se comunican mediante un conjunto de protocolos comunes (TCP/IP). En términos simples, Internet es conocido como una “red de redes” que interconecta millones de computadoras a nivel mundial

Por su parte, la World Wide Web (WWW o simplemente "Web") es un sistema de información distribuido que opera sobre Internet. La Web se compone de páginas y recursos vinculados mediante hipertexto (enlaces); los usuarios acceden a este contenido a través de navegadores web usando principalmente el protocolo HTTP. En otras palabras, la Web es un servicio que funciona sobre Internet, permitiendo acceder y compartir documentos y medios a través de páginas HTML interconectadas

Es importante diferenciar ambos conceptos: Internet es la infraestructura de red, mientras que la Web es uno de los servicios más populares que utiliza dicha infraestructura para ofrecer información de forma hipermedia (texto, imágenes, videos enlazados). Tim Berners-Lee, creador de la Web a finales de los 80, la concibió como un conjunto universal de documentos enlazados; desde entonces la Web ha crecido exponencialmente y se ha convertido en sinónimo de Internet para muchos usuarios comunes.

Internet en la industria

La incorporación de Internet en la industria ha transformado profundamente los procesos productivos, comerciales y de gestión en las últimas décadas. Gracias a la conectividad global, las empresas pueden comunicarse y acceder a información en tiempo real desde cualquier lugar del mundo. Esto ha permitido, por ejemplo, que un gerente supervise el estado de su fábrica o negocio remotamente mediante su smartphone, aprovechando sistemas interconectados

La cuarta revolución industrial (Industria 4.0) se caracteriza justamente por la integración masiva de Internet, sensores y datos en los procesos industriales para mejorar la eficiencia y la toma de decisiones. Industrias tradicionales se han visto impulsadas por tecnologías como el Internet de las Cosas (IoT), que conecta máquinas y dispositivos a Internet, posibilitando la automatización, el mantenimiento predictivo y la optimización de cadenas de suministro. Asimismo, sectores como el comercio han dado paso al e-commerce, la educación ha adoptado plataformas en línea, y prácticamente toda empresa moderna depende de Internet para sus operaciones diarias (desde la comunicación por email hasta servicios en la nube). En conclusión, Internet en la industria ha aumentado la productividad y ha permitido nuevos modelos de negocio digitales, siendo un pilar fundamental de la economía global actual

Evolución de la World Wide Web

Desde su creación a inicios de los 90, la Web ha pasado por distintas etapas evolutivas comúnmente denominadas Web 1.0, Web 2.0 y Web 3.0, entre otras. Web 1.0 es la primera generación de la Web, caracterizada principalmente por páginas estáticas de solo lectura. En esta etapa inicial, los sitios web proveían contenido fijo que los usuarios podían leer pero no modificar; no había interacción significativa ni contenido generado por usuarios. Estos sitios Web 1.0 tenían información que no se actualizaba dinámicamente, lo cual era una limitación cuando los datos necesitaban refrescarse con frecuencia

Hacia principios de los 2000, emergió la Web 2.0, un término popularizado en 2004 para describir una Web mucho más interactiva y social. La Web 2.0 marcó el paso de usuarios pasivos a usuarios activos: las personas comenzaron a participar colaborativamente creando y compartiendo contenido en blogs, wikis, redes sociales y otras plataformas

Sitios como Wikipedia, YouTube, Facebook o Twitter son ejemplos de esta etapa, donde la característica fundamental es la interacción usuario-usuario y usuario-sitio (comentarios, publicaciones, contenido generado por el usuario). La Web 3.0, a menudo llamada Web Semántica, es la siguiente fase en la evolución. Su objetivo es que la información en la web pueda ser entendida y procesada de forma más efectiva por las máquinas. Para ello, la Web 3.0 propone el uso de metadatos estandarizados y ontologías que describan el significado de los datos, de modo que los ordenadores puedan “comprender” mejor la información y ofrecer respuestas más inteligentes

En la práctica, la Web Semántica integra datos de distintas fuentes y añade contexto, lo que habilitaría aplicaciones más inteligentes (por ejemplo, asistentes que interpretan las preferencias del usuario). Cabe mencionar que en años recientes el término “Web 3.0” también se ha asociado a ideas de una web más descentralizada (uso de blockchain, redes peer-to-peer, etc.) y con mayor privacidad, aunque estos conceptos siguen en desarrollo. En resumen, la Web ha evolucionado de páginas estáticas (Web 1.0) a una plataforma social y dinámica (Web 2.0) y apunta hacia una web de datos interconectados e inteligentes (Web 3.0), reflejando el creciente rol de los usuarios y de las máquinas en la construcción de la experiencia web.

Fundamentos de la Web

Los fundamentos del funcionamiento de la Web se basan en una arquitectura cliente-servidor. En este modelo, un cliente (por ejemplo, nuestro navegador web) realiza una petición a un servidor (un ordenador que hospeda un sitio web) para acceder a cierto recurso (una página web, una imagen, un archivo). Esta petición normalmente se hace mediante el protocolo HTTP o su versión segura HTTPS. El servidor, al recibir la petición, prepara una respuesta que suele incluir el documento solicitado (frecuentemente en HTML) y la envía de vuelta al cliente. El navegador entonces interpreta el código HTML y lo renderiza en forma de página web visible e interactiva para el usuario. Durante este proceso básico entran en juego tres tecnologías esenciales que son la base del desarrollo web: HTML, CSS y JavaScript. HTML define la estructura y el contenido de la página; CSS (Hojas de Estilo en Cascada) define la presentación o el estilo visual; JavaScript se encarga del comportamiento dinámico y la interactividad en el lado del cliente. En síntesis, HTML estructura el contenido (por ejemplo, coloca títulos, párrafos, imágenes, enlaces), CSS se ocupa de la apariencia (colores, fuentes, diseños responsivos), y JavaScript permite responder a acciones del usuario o actualizar el contenido de forma dinámica. Todas las páginas web utilizan este trío de tecnologías de una forma u otra. Otro fundamento clave es el sistema de URLs y DNS: cada página tiene una dirección URL única, y los navegadores usan el DNS para traducir los nombres de dominio (ej. www.ejemplo.com) en direcciones IP del servidor correspondiente. Con estos componentes, la Web funciona de manera que, independientemente de la ubicación física, un usuario puede acceder a información alojada en cualquier servidor del mundo, siempre que esté conectado a Internet y conozca (o encuentre mediante un buscador) la dirección adecuada. En resumen, los fundamentos web combinan protocolos (HTTP), formatos de contenido (HTML/CSS/JS) y un modelo cliente-servidor de comunicación para hacer posible la experiencia de navegar por sitios web.

Secuencias de comandos del lado del cliente y del servidor

En el desarrollo web distinguimos entre scripting del lado del cliente y scripting del lado del servidor, según dónde se ejecuta el código de un sitio o aplicación web. El lado del cliente se refiere al navegador web del usuario final. Un script del lado del cliente es típicamente código JavaScript que se envía junto con la página web y que el navegador ejecuta localmente. Estos scripts pueden manipular la página (ej. validar formularios instantáneamente, mostrar animaciones, actualizar contenido sin recargar la página usando AJAX, etc.) y responden directamente a las acciones del usuario, todo sin necesidad de volver a contactar al servidor para cada interacción. Por otro lado, el lado del servidor corresponde al código que se ejecuta en el servidor web antes de que la página sea enviada al cliente. Lenguajes como PHP, Python (Django, Flask), Ruby (Rails), Java (Spring), JavaScript (Node.js) u otros se usan en el servidor para procesar la lógica de negocio, acceder a bases de datos, y

generar dinámicamente las páginas HTML que luego recibirá el cliente. La diferencia fundamental radica en el entorno de ejecución: JavaScript tradicionalmente opera en el navegador del usuario (cliente), mientras que lenguajes como PHP operan en el servidor; es decir, el código JavaScript se procesa en la máquina del usuario, mientras que el código del servidor genera o prepara el contenido en la máquina servidora antes de enviarlo

Ambos tipos de scripting suelen trabajar en conjunto: por ejemplo, en un formulario web, puede haber validación del lado del cliente (rápida, para mejorar la experiencia de usuario) y validación del lado del servidor (más segura, en el servidor, para garantizar la integridad de los datos). En resumen, cliente y servidor son dos entornos diferentes. El código del lado del cliente mejora la interactividad y experiencia en la interfaz de usuario, y el código del lado del servidor se encarga de manejar datos, reglas de negocio y respuesta a peticiones, antes de que la página llegue al navegador. Comprender esta diferencia es esencial para estructurar aplicaciones web modernas que sean eficientes y seguras.

Jerarquía de datos

En informática se define una jerarquía de datos clásica, que organiza la información en niveles desde el más básico hasta estructuras más complejas. Esta jerarquía de menor a mayor es:

Bit: la unidad mínima de información, puede tener valor 0 o 1 (valor binario básico).

Byte: conjunto de 8 bits. Un byte usualmente representa un carácter (una letra, número o símbolo) en sistemas informáticos, o puede almacenar números pequeños.

Campo: grupo de bytes que conforman un dato con significado, por ejemplo un nombre, una fecha o un importe. Un campo suele ser la unidad de información más pequeña con sentido en un contexto (equivale a una columna en una tabla de base de datos).

Registro: conjunto de campos relacionados entre sí. Por ejemplo, los campos “nombre”, “apellido”, “edad” y “DNI” juntos forman el registro de una persona. Un registro representa típicamente una entidad completa (una fila en una base de datos).

Archivo (Fichero): conjunto de registros (o también podría considerarse un conjunto de campos) que comparten una relación. Por ejemplo, un archivo puede contener todos los registros de personas (una lista completa de empleados, cada uno con sus datos). Es una colección de datos almacenada como una unidad (como un documento, una hoja de cálculo, etc.).

Base de datos: conjunto organizado de archivos o registros relacionados, almacenados de forma estructurada. Una base de datos integra múltiples archivos

o tablas con información vinculada, permitiendo consultas y operaciones complejas sobre los datos en conjunto. Es el nivel más alto en esta jerarquía clásica de datos

Este orden jerárquico (bit < byte < campo < registro < fichero < base de datos) ayuda a entender cómo se construye la información en los sistemas: los bits forman bytes, los bytes forman campos, varios campos constituyen registros, los registros se agrupan en archivos, y los archivos relacionados conforman bases de datos. En el contexto de la Web, por ejemplo, cuando un usuario rellena un formulario HTML, cada campo de entrada (como “nombre” o “edad”) podría considerarse un campo en esta jerarquía, y el conjunto de campos enviados podría representar un registro que luego se guarda en una base de datos en el servidor.

Sistemas Operativos

Un Sistema Operativo (SO) es el software fundamental que administra los recursos de una computadora y sirve de base para que otros programas puedan funcionar. En términos formales, es un conjunto de programas que gestiona el hardware (CPU, memoria, dispositivos de entrada/salida) y provee servicios básicos a las aplicaciones de usuario

El sistema operativo actúa como intermediario entre los componentes físicos del equipo y las aplicaciones, asegurándose de que cada programa obtenga tiempo de procesador, memoria y acceso a dispositivos de forma adecuada y segura. Algunos ejemplos de sistemas operativos son Microsoft Windows, macOS (de Apple) y GNU/Linux en computadoras personales, o Android e iOS en dispositivos móviles. En servidores web, son muy comunes las variantes de Linux (por su estabilidad y eficiencia) aunque también existen servidores Windows. El sistema operativo maneja funciones críticas como la gestión de archivos (sistema de archivos donde se almacenan documentos y datos), la gestión de procesos (qué programas se están ejecutando y con qué prioridad), el control de periféricos (teclado, impresora, disco duro, red) y proporciona la interfaz básica para el usuario (ya sea una interfaz gráfica de ventanas o una consola de comandos). En resumen, sin un sistema operativo los dispositivos serían incapaces de funcionar de manera útil, pues es el SO quien orquesta todos los recursos y crea la plataforma sobre la cual corren tanto aplicaciones de escritorio como servicios de servidor y, en el contexto web, los servidores web y navegadores mismos son programas que se ejecutan sobre un sistema operativo.

Lenguajes de programación

Un lenguaje de programación es un lenguaje formal que los desarrolladores utilizan para escribir instrucciones que una computadora puede ejecutar. En otras palabras, es la herramienta que permite “comunicarle” órdenes a la máquina para que realice tareas específicas. Estas instrucciones siguen una lógica (algoritmos) diseñados

para controlar el comportamiento de un sistema informático o resolver un problema dado

Existen muchos lenguajes de programación (C, C++, Java, Python, JavaScript, etc.), cada uno con su sintaxis y particularidades, pero todos comparten el objetivo de permitirnos crear software (programas). Conviene destacar que un lenguaje de marcado como HTML no es un lenguaje de programación en sentido estricto

HTML estructura datos pero no ejecuta lógica o algoritmos por sí mismo. A diferencia de HTML, los lenguajes de programación son Turing completos (permiten implementar cualquier algoritmo concebible) e incluyen elementos como variables, bucles, condiciones, funciones, etc. Para que el computador entienda estas instrucciones de alto nivel, suelen existir compiladores o intérpretes que traducen el código fuente escrito por el programador a código máquina (los bits 0 y 1 que la CPU realmente procesa). Por ejemplo, un programador puede escribir en C++ una serie de pasos para calcular la nómina de empleados; el compilador C++ traducirá ese código a lenguaje máquina ejecutable por la computadora. Los lenguajes de programación pueden clasificarse de diversas formas: de bajo nivel (cercaos al hardware) vs alto nivel (cercaos al lenguaje humano), compilados vs interpretados, procedurales vs orientados a objetos, etc. En el desarrollo web conviven múltiples lenguajes de programación: JavaScript (ejecutado en el navegador, y también en servidores con Node.js), lenguajes de servidor como PHP, Python, Ruby o Java, e incluso lenguajes para bases de datos como SQL. Todos ellos permiten implementar la lógica de las aplicaciones web. En resumen, aprender un lenguaje de programación implica aprender su sintaxis (cómo escribir las instrucciones correctamente) y su semántica (cómo esas instrucciones controlan el flujo lógico), con el fin de decirle al ordenador qué hacer y cómo hacerlo.

Tecnologías de objetos (Programación orientada a objetos)

El concepto de tecnologías de objetos se refiere a la programación orientada a objetos (POO) y al ecosistema de herramientas y lenguajes que implementan este paradigma. La POO es un modelo de desarrollo de software que se popularizó ampliamente en la industria durante los años 1990, llegando a reemplazar en gran medida a los enfoques procedurales clásicos debido a sus ventajas en organización y reutilización de código

En la programación orientada a objetos, se estructuran los programas definiendo “clases” y creando “objetos”. Una clase es como un plano o plantilla que describe las propiedades (atributos o datos) y comportamientos (métodos o funciones) que ciertos objetos tendrán. Un objeto es una instancia concreta de una clase, que agrupa datos y las funciones que operan sobre esos datos en una sola entidad. Este enfoque permite modelar entidades del mundo real de forma más natural en el código. Por ejemplo, se puede tener una clase Coche con atributos como color, modelo, velocidad, y métodos como acelerar() o frenar(); y crear múltiples objetos

coche con distintos valores pero mismo comportamiento. Las tecnologías de objetos traen consigo principios fundamentales como encapsulación (cada objeto controla sus propios datos internos, exponiendo solo lo necesario), herencia (una clase puede heredar características de otra, facilitando la reutilización y extensión de código), y polimorfismo (objetos de distintas clases se pueden tratar de forma uniforme si comparten una interfaz común). Lenguajes de programación muy usados en la web y software general —como Java, C++, C#, Python, JavaScript (en su modelo basado en prototipos), PHP, Ruby, etc.— soportan la orientación a objetos. Gracias a ello, hoy en día el desarrollo de aplicaciones web complejas se beneficia de bibliotecas y frameworks contruidos con POO, donde componentes de software son objetos que interactúan entre sí. Por ejemplo, en un framework web orientado a objetos, podríamos tener objetos que representan usuarios, productos, pedidos, cada uno con sus métodos para cargar, guardar, procesar información, etc. El paradigma de objetos facilita manejar la complejidad dividiendo el problema en piezas (objetos) más manejables. En resumen, las tecnologías de objetos han proporcionado un marco sólido para construir software modular, mantenible y extensible, por lo que dominar la programación orientada a objetos es clave para desarrolladores, incluidos aquellos enfocados en aplicaciones web modernas

Lenguaje de Marcado (HTML)

Un lenguaje de marcado es un conjunto de reglas y sintaxis que se utilizan para describir la estructura y elementos de un documento. En el contexto de la web, el principal lenguaje de marcado es HTML (HyperText Markup Language), que es el estándar para crear y estructurar páginas web. Funciona a través de etiquetas (tags) que rodean o delimitan el contenido, indicando qué significado o rol tiene cada fragmento en la página

Por ejemplo, una etiqueta `<p>` indica un párrafo de texto, `<h1>` indica un encabezado principal, `<a>` define un hipervínculo, etc. Estas etiquetas no se ven en la página final, sino que le dicen al navegador cómo estructurar y presentar el contenido al usuario. HTML por sí mismo no es un lenguaje de programación (no maneja lógica, condiciones o cálculos); es más bien un lenguaje de marcado descriptivo que le comunica al navegador la jerarquía y tipo de información en el documento

Cada página HTML está compuesta por texto plano con etiquetas que el navegador interpreta para construir la página visual. Las etiquetas generalmente vienen en pares de apertura y cierre (por ejemplo `<title>` para abrir y `</title>` para cerrar el título de la página), aunque hay etiquetas auto-cerradas para contenido vacío, como `<br>` para un salto de línea o `<img>` para insertar una imagen. HTML es un lenguaje en constante evolución (vamos por HTML5 en la actualidad) y forma, junto con CSS y JavaScript, la base de todo desarrollo web front-end. En resumen, HTML proporciona el esqueleto o la estructura de la página web, indicando “qué es qué” en el contenido (qué es un encabezado, qué es una lista, dónde empieza una tabla,

etc.), de manera que los navegadores puedan mostrar la información con el formato apropiado.

Atributos en HTML

Las etiquetas HTML pueden llevar asociados unos parámetros adicionales llamados atributos. Un atributo proporciona información adicional sobre un elemento o modifica su comportamiento por defecto. Se escriben dentro de la etiqueta de apertura de un elemento, generalmente en formato `nombre="valor"`. **Por ejemplo, en ``, `src` y `alt` son atributos de la etiqueta `<img>`: `src` especifica la ruta o fuente de la imagen, y `alt` proporciona un texto alternativo (descripción) en caso de que la imagen no cargue.** Los atributos permiten configurar casi cualquier aspecto de un elemento HTML. Algunos atributos son globales (aplican a casi todas las etiquetas), como `id` (identificador único del elemento), `class` (clase CSS para aplicar estilos), `style` (estilos en línea), `title` (texto emergente de título), entre otros

Otros son específicos de ciertos elementos: por ejemplo, un enlace `<a>` utiliza el atributo `href` para la URL de destino; un elemento de formulario `<input>` puede usar atributos como `type` (tipo de campo: texto, checkbox, radio, etc.), `value` (valor del campo), `placeholder` (texto de sugerencia), `required` (para marcarlo como obligatorio), etc. Los atributos siempre van en la apertura de la etiqueta y pueden ser múltiples, separados por espacios. Su orden no suele importar, pero es buena práctica usar un orden lógico para legibilidad. Es importante también que los valores de los atributos vayan entre comillas. En HTML5 a veces es permitido omitir las comillas si el valor no tiene espacios, pero incluirlas siempre evita problemas. Veamos un ejemplo sencillo combinando varios atributos en una etiqueta:

```
<a href="https://www.ejemplo.com" target="_blank" title="Ir a Ejemplo">Visitar sitio</a>
```

En este enlace, `href` indica la URL a visitar, `target="_blank"` le dice al navegador que abra el enlace en una pestaña/nueva ventana, y `title` define un texto descriptivo que aparecerá al pasar el cursor. Otro ejemplo con un elemento de imagen:

```

```

Aquí `width` y `height` establecen las dimensiones de la imagen mostrada en píxeles, además de los ya mencionados `src` y `alt`. Como se observa, los atributos enriquecen las etiquetas HTML, permitiendo especificar detalles y afinando cómo se comportan o se presentan los elementos en la página web. Sin atributos, HTML sería mucho menos flexible. Siempre que usemos un elemento HTML, es recomendable conocer qué atributos soporta para aprovechar al máximo su potencial.

Componentes para formularios

Los formularios en HTML son la principal vía de interacción del usuario con un sitio web, permitiendo enviar datos (como entradas de texto, selecciones, opciones, etc.) al servidor. Un formulario se define con la etiqueta `<form>` y dentro de él se colocan diversos controles de formulario (campos y botones) para la entrada de datos. Entre los componentes típicos de un formulario encontramos: campos de texto, áreas de texto, casillas de verificación, botones de opción, menús desplegables, selectores de fecha, botones de envío, etc. Cada uno de estos componentes se logra mediante diferentes elementos HTML. Los más usados son:

<input>: elemento genérico para varios tipos de campos de entrada. Dependiendo de su atributo `type`, puede ser una caja de texto (`type="text"`), un campo de contraseña (`"password"`), un checkbox (`"checkbox"`), un radio button (`"radio"`), un selector de archivos (`"file"`), un botón (`"button"` o `"submit"`), entre otros.

<textarea>: crea un campo de texto de varias líneas (un área de texto grande, por ejemplo para comentarios o descripciones).

<select>: genera un menú desplegable de opciones. Dentro de `<select>` se colocan múltiples `<option>` cada una representando una opción seleccionable.

<label>: etiqueta descriptiva para un campo de formulario. Al hacer clic en el texto de un `<label>` asociado, el foco pasa al campo correspondiente, mejorando la accesibilidad. Se asocia usando el atributo `for` (que debe coincidir con el `id` del campo `<input>`).

<button>: representa un botón clicable. Puede usarse como botón de envío de formulario (con `type="submit"`) o botón genérico que se maneja con scripts (con `type="button"`).

Todos estos elementos van contenidos dentro de `<form>...</form>`. La etiqueta `<form>` además lleva atributos importantes como `action` (la URL o ruta a la que se enviarán los datos del formulario cuando se envíe) y `method` (el método HTTP de envío, comúnmente `"GET"` o `"POST"`). Por ejemplo:

<form action="/registro" method="POST">

<label for="usuario">Nombre de usuario:</label>

<input type="text" id="usuario" name="usuario" placeholder="Escribe tu nombre" required>

<label for="pais">País:</label>

<select id="pais" name="pais">

<option value="CR">Costa Rica</option>

<option value="MX">México</option>

```

        <option value="ES">España</option>
    </select>

    <label><input type="checkbox" name="terminos" required>

        Acepto los términos y condiciones

    </label>

    <button type="submit">Registrarse</button>

</form>

```

En este ejemplo práctico, definimos un formulario que enviará datos vía POST a la URL `/registro`. Contiene un campo de texto para nombre de usuario (con placeholder y marcado como requerido), un menú desplegable para país, una casilla de verificación para aceptar términos, y un botón de tipo submit para enviar el formulario. Los componentes para formularios trabajan en conjunto: cuando el usuario rellena los campos y pulsa el botón, el navegador recopila cada campo con su nombre (name) y valor ingresado, y los envía al servidor según la configuración dada. La versatilidad de los formularios HTML ha permitido la creación de funcionalidades como registros, inicios de sesión, búsquedas, encuestas, compra de productos (cestas de compra), entre otros. Cabe destacar que HTML5 añadió nuevos tipos de input (email, url, number, date, etc.) y atributos como required, pattern (para validaciones), que hacen los formularios más poderosos y útiles incluso antes de recurrir a JavaScript o validación en servidor.

Elementos de entrada (tipos de `<input>`)

Como mencionamos, el elemento `<input>` de HTML es muy versátil y soporta múltiples tipos de entrada mediante su atributo type. Cada tipo de `<input>` brinda un control diferente para la entrada de datos del usuario. A continuación, algunos de los tipos más comunes de campos de entrada y su propósito

text (Texto): campo de texto de una sola línea para ingresar texto libre (letras, números, etc.). Es el tipo por defecto de `<input>` si no se especifica otro. Ideal para nombres, búsquedas, etc.

password (Contraseña): similar al tipo texto, pero oculta los caracteres ingresados (suelen mostrarse como puntos o asteriscos) para mantener la confidencialidad. Usado en campos de contraseña.

email: campo de texto optimizado para introducir direcciones de correo electrónico. Los navegadores suelen validar que el formato parezca un email (contenga '@', etc.) antes de permitir el envío.

number (Número): campo que admite solo números. Algunos navegadores muestran pequeñas flechas para incrementar/decrementar el valor.

checkbox (Casilla de verificación): presenta una casilla cuadrada que puede estar marcada o no. Sirve para opciones booleanas (sí/no) o selecciones múltiples (ej: “Acepto términos”, selección de varias preferencias, etc.).

radio (Botón de opción): presenta un círculo que puede marcarse. Los radio buttons se agrupan por nombre (name igual) y solo uno del mismo grupo puede estar seleccionado a la vez (ej: elegir una de varias opciones exclusivas).

file (Archivo): campo que al hacer clic permite al usuario seleccionar un archivo de su sistema (se abre el explorador de archivos). Se usa para subir archivos al servidor. Suele mostrarse con un botón “Examinar...” y el nombre del archivo elegido.

date (Fecha): control para selección de fecha. En navegadores modernos aparece un calendario emergente para elegir día, mes y año. (También existen datetime-local, month, etc., para variantes de fecha/hora).

range (Rango): muestra un control tipo deslizador (slider) para seleccionar un valor dentro de un rango numérico. Por ejemplo, escoger un número entre 0 y 100 moviendo una perilla.

submit (Enviar): botón de envío de formulario. Al pulsarlo, el formulario se envía (ejecuta la acción especificada en <form>). Por defecto suele mostrarse como un botón con el texto “Submit” o “Enviar” (personalizable con CSS o value).

button (Botón): un botón genérico sin acción predeterminada. Se utiliza cuando se necesita un botón que, por ejemplo, ejecute alguna función de JavaScript en el lado del cliente en lugar de enviar el formulario.

Además de estos, HTML5 introdujo otros tipos útiles como tel (teléfono), url (URL web), color (selector de color), etc., cada uno proporcionando interfaces adecuadas en dispositivos compatibles. Por ejemplo, en móviles, un <input type="tel"> puede mostrar un teclado numérico optimizado para teléfonos, o email muestra un teclado con la arroba a mano. Es importante destacar que, aunque estos campos aportan cierta validación o interfaz por defecto (como el calendario para fecha), siempre se debe implementar validación robusta en el servidor. Los elementos de entrada <input> no tienen contenido interno (son elementos vacíos, no llevan texto entre etiqueta de inicio y fin, de hecho en HTML5 la etiqueta de cierre es opcional o innecesaria para ellos). Para saber qué dato envió el usuario, cada <input> (o control de formulario) debe tener un atributo name que actúa como identificador del dato al ser enviado. Por ejemplo, <input type="text" name="usuario"> enviará el texto escrito bajo la clave “usuario”. Combinando diversos tipos de <input> y otros controles podemos crear formularios completos para casi cualquier necesidad de interacción. La llegada de nuevos tipos en HTML5 mejoró la experiencia del usuario (con validaciones y widgets nativos del navegador), reduciendo la necesidad de escribir tanto JavaScript para validación básica.

Listas en HTML

HTML ofrece elementos especiales para representar listas de ítems o elementos similares, de manera estructurada. Existen principalmente dos tipos de listas: listas no ordenadas y listas ordenadas, además de las listas de definición (útiles para glosarios o descripciones).

Lista no ordenada (): Presenta una colección de ítems donde el orden no es relevante. Por defecto, el navegador muestra cada ítem con viñetas (puntos, círculos o otros símbolos) a modo de listado. Cada elemento de la lista se define con (List Item). Por ejemplo:

Manzana

Banana

Cereza

Esto generaría una lista con viñetas: • Manzana, • Banana, • Cereza. Las listas no ordenadas se usan para enumeraciones simples, menús de navegación, y en general cualquier conjunto de elementos donde la secuencia no implica prioridad.

Lista ordenada (): Es similar a pero aquí sí importa el orden, y el navegador por defecto numera cada elemento en secuencia (1., 2., 3., etc. o letras, según el estilo). También utiliza para cada elemento. Ejemplo:

Precalentar el horno a 180°C

Mezclar harina y azúcar

Agregar huevos y batir

Esto mostrará una lista numerada 1, 2, 3 con los pasos de una receta, por ejemplo. Las listas ordenadas son útiles para instrucciones paso a paso, rankings, o cualquier lista donde el índice de cada elemento tiene significado (posición en el orden).

Lista de definición (<dl> junto con <dt> y <dd>): Es un tipo diferente de lista, utilizada para pares de término-definición. Dentro de <dl> (Definition List), se emplea <dt> (Definition Term) para el término o concepto, y <dd> (Definition Description) para la descripción o definición del término inmediatamente anterior. Por ejemplo:

<dl>

<dt>HTML</dt>

<dd>Lenguaje de marcado para estructurar páginas web.</dd>

<dt>CSS</dt>

<dd>Lenguaje de estilos en cascada para diseñar la apariencia de las páginas web.</dd>

</dl>

Aquí tendríamos dos términos "HTML" y "CSS" con sus definiciones sangradas debajo. Este tipo de lista es ideal para glosarios, FAQs (pregunta-respuesta) o describir pares de información.

Internamente, todos los tipos de lista utilizan para cada ítem, excepto la de definiciones que usa la estructura <dt>/<dd>. Las listas pueden anidarse: por ejemplo, una lista puede contener dentro de un otra lista u para sub-listas (y se verán con sangría adicional). También se les puede dar estilo con CSS para cambiar los marcadores (p. ej., usar un cuadrado en vez de punto, o imágenes como viñetas, o en listas ordenadas cambiar números por letras romanas, etc.). En resumen, las listas permiten estructurar contenidos en forma de conjuntos de elementos relacionados, mejorando la organización visual y lógica del contenido. HTML provee esta semántica de lista para que el contenido se entienda como tal (por lectores de pantalla, agentes de usuario, etc.), en lugar de, por ejemplo, usar guiones manualmente o números escritos, que serían menos semánticos.

Autocompletado

El autocompletado es una función muy útil asociada principalmente a formularios, donde el navegador puede rellenar automáticamente campos en base a información previamente ingresada por el usuario u ofrecida por el sitio. En HTML, el atributo autocomplete controla esta característica en formularios y campos de entrada. Por defecto, la mayoría de navegadores recuerdan lo que el usuario escribió en ciertos campos (por ejemplo, nombre, correo, dirección) y pueden sugerir o completar esos valores en futuras ocasiones para ahorrar tiempo.

Cuando autocomplete="on" en un formulario o input, se indica al navegador que permita el autocompletado usando datos almacenados. Por ejemplo, un campo de nombre con autocomplete activado puede mostrar sugerencias de nombres previamente ingresados.

Si se establece `autocomplete="off"`, el navegador no ofrecerá completar automáticamente ese campo (esto puede ser útil para campos sensibles, como números de tarjeta de crédito, o para forzar que el usuario introduzca datos frescos).

El atributo `autocomplete` se puede poner a nivel de `<form>` (afecta todos sus campos) o en cada `<input>` individual. Además, HTML5 definió valores específicos de autocompletado para indicar el tipo de dato esperado. Por ejemplo, `autocomplete="name"` en un campo sugiere al navegador que es un nombre personal, `autocomplete="email"` que es un correo electrónico, `autocomplete="address-line1"` para dirección, `autocomplete="country"` para país, etc. Estos valores guían a la función de autocompletar para que coloque la información apropiada (muchos navegadores almacenan perfiles básicos con nombre, email, dirección, teléfono del usuario para rellenarlos automáticamente en formularios). En campos de login, `autocomplete="current-password"` o `"new-password"` pueden utilizarse para gestionar cómo se guardan contraseñas.

Un ejemplo de uso:

```
<form action="/buscar" method="GET" autocomplete="on">
    <input type="text" name="query" autocomplete="off"
    placeholder="Buscar...">
    <button type="submit">Buscar</button>
</form>
```

En este formulario de búsqueda, hemos activado autocompletar para el formulario, pero desactivado (off) para el campo específico de texto. Esto podría tener sentido si no queremos que se guarden las búsquedas. En otros casos, como un formulario de envío de dirección, convendría dejar `autocomplete="on"` para facilitar al usuario.

Además del atributo `autocomplete`, existe el elemento `<datalist>` que proporciona una lista predefinida de opciones de autocompletado. Un `<input>` de tipo texto puede asociarse a un `<datalist>` mediante el atributo `list`. Por ejemplo:

```
<input type="text" name="ciudad" list="sugerencias-ciudad" placeholder="Tu ciudad">
    <datalist id="sugerencias-ciudad">
        <option value="San José">
        <option value="Cartagena">
        <option value="Buenos Aires">
        <option value="México D.F.">
    </datalist>
```

En este ejemplo, al escribir en el campo "ciudad", el navegador ofrecerá las opciones definidas dentro del `<datalist>` como posibles autocompletados. A diferencia de autocomplete general, estas opciones las define explícitamente el desarrollador.

En resumen, el autocompletado mejora la experiencia de usuario al reducir la necesidad de volver a teclear información recurrente. HTML nos da control sobre esta función mediante el atributo `autocomplete` (para habilitarla, deshabilitarla o afinar qué datos esperar).

Y herramientas como `<datalist>` para sugerir valores. No obstante, es bueno recordar que algunos usuarios pueden desactivar el autocompletado en sus navegadores por privacidad, y que para datos críticos siempre se debe permitir que el usuario los ingrese manualmente si así lo desea.

Estructura de una página web

Toda página web en HTML sigue una estructura básica estandarizada que le permite al navegador interpretar correctamente el contenido. En su forma más esencial, un documento HTML se compone de tres secciones o elementos contenedores fundamentales: `<html>`, `<head>` y `<body>`

Antes de estos elementos, al inicio del archivo, se incluye además la declaración de tipo de documento `<!DOCTYPE html>` que indica al navegador que usaremos HTML5 (esta declaración no es una etiqueta HTML, sino una instrucción para el navegador). A continuación describimos cada parte de la estructura:

`<!DOCTYPE html>`: Debe estar en la primera línea de la página. Es una declaración que informa el tipo de documento. En HTML5 simplemente se escribe así (en versiones anteriores era más complejo). Garantiza que los navegadores trabajen en modo estándar.

`<html>`: Es el elemento raíz que engloba todo el contenido de la página (exceptuando el DOCTYPE). Representa el documento en sí. Suele llevar el atributo `lang` para indicar el idioma principal del contenido (por ejemplo `<html lang="es">` para español). La etiqueta de cierre `</html>` va al final de todo el documento. Dentro de `<html>` van dos secciones: el `<head>` y el `<body>`.

`<head>`: Es la sección de encabezado del documento HTML. Aquí no va contenido visible de la página, sino meta-información y configuraciones. Por ejemplo, dentro del `<head>` se colocan: el título de la página (`<title>`), que se muestra en la pestaña del navegador; metadatos como `<meta charset="UTF-8">` (para indicar la codificación de caracteres, UTF-8 es estándar para soportar caracteres internacionales), meta description y palabras clave (usadas por buscadores), enlaces a hojas de estilo CSS externas (`<link rel="stylesheet" href="estilos.css">`), enlaces a scripts JavaScript externos (`<script src="app.js" defer></script>`), estilos o scripts internos (con `<style>` o `<script>`), y cualquier otra

información que no sea directamente contenido a mostrar. El `<head>` comienza justo después de abrir `<html>` y termina antes de abrir `<body>`. Aunque su contenido no se ve en la página, es muy importante para el correcto funcionamiento y SEO del sitio.

<body>: Es el cuerpo del documento, donde reside todo el contenido visible de la página web. Todo lo que queremos que se muestre al usuario (texto, imágenes, videos, enlaces, formularios, etc.) debe estar dentro del `<body>`. Cuando el navegador renderiza la página, procesa el `<head>` para configuraciones y luego dibuja el contenido encontrado en `<body>` en la ventana. El `<body>` suele empezar después del cierre del `</head>` y se cierra justo antes de `</html>`. Dentro del body podemos estructurar el contenido usando diversos elementos HTML (encabezados `<h1>`-`<h6>`, párrafos `<p>`, divs `<div>` para secciones genéricas, navs `<nav>` para menús de navegación, artículos `<article>`, secciones `<section>`, pies de página `<footer>`, etc., además de los elementos inline como `<span>`, imágenes `<img>`, listas `<ul>`/`<ol>`, tablas `<table>`, y etcétera según la necesidad).

Juntando todo esto, una estructura básica mínima de una página HTML5 se ve así:

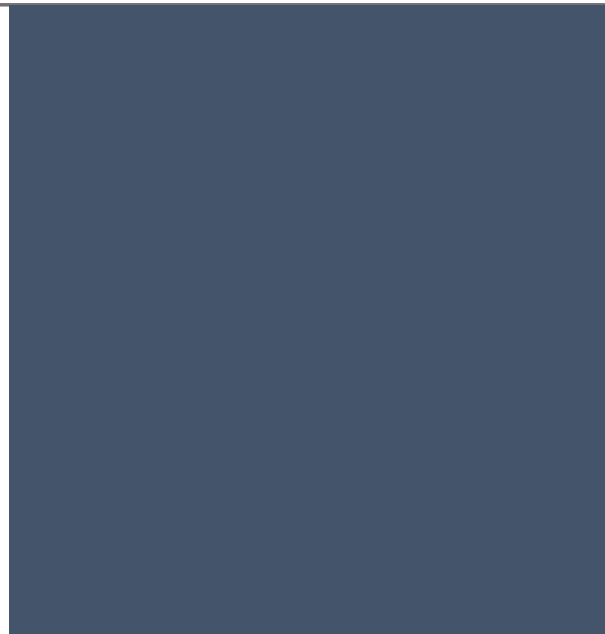
```
<!DOCTYPE html>  
<html lang="es">  
  <head>  
    <meta charset="UTF-8">  
    <title>Título de la página</title>  
  </head>  
  <body>  
    <h1>¡Hola, mundo!</h1>  
    <p>Esta es mi primera página web.</p>  
  </body>  
</html>
```

En este código de ejemplo, definimos el idioma como español, declaramos la codificación UTF-8, asignamos un título ("Título de la página"), y en el cuerpo colocamos un encabezado nivel 1 y un párrafo de texto. Esta sería la base; por supuesto, las páginas reales incluirán mucho más dentro del `<body>` y usualmente más metadatos en el `<head>` (como hojas de estilo CSS, enlaces a scripts, descripciones, etc.). Sin embargo, incluso esta estructura sencilla ya constituye un documento HTML válido. Todos los elementos mostrados cumplen una jerarquía: `<html>` es padre de `<head>` y `<body>`; `<head>` y `<body>` son hermanos; el `<title>` va dentro del `<head>`, el `<h1>` y `<p>` van dentro del `<body>`, etc. Mantener esta

estructura es crucial: por ejemplo, no se debe poner un `<title>` fuera del `head`, o contenido visible antes del `<body>`, ya que violaría la sintaxis esperada.

En resumen, la estructura de una página web HTML consta de un encabezado con información técnica/meta y un cuerpo con el contenido principal

Esta separación permite organizar correctamente el documento. A medida que la Web ha evolucionado, se han introducido etiquetas semánticas para secciones comunes (`header`, `main`, `footer`, `nav`, etc.), pero todas ellas siempre estarán dentro del `<body>`. Dominar esta estructura base es el primer paso para crear páginas web, pues es el esqueleto sobre el cual luego aplicaremos estilos CSS y comportamientos con JavaScript para obtener páginas web completas y atractivas.



Guía Técnica de Desarrollo Web

Miguel Gómez Sánchez



Que es HTML?

HTML es el acrónimo de Hyper Text Markup Language o Lenguaje para marcado de hipertexto un lenguaje estándar que se emplea para crear los contenidos de páginas y sitios web. Y al decir que es un lenguaje estándar, significa que ha sido estructurado basado en una serie de normas reconocidas universalmente por todas las entidades relacionadas con el mundo de la Internet lo cual quiere decir que una página escrita mediante este lenguaje se debe visualizar de forma similar en cualquier navegador web y bajo distintos sistemas operativos. También se define como un lenguaje de marcado porque dispone de marcas o instrucciones que permiten definir los textos, formularios, imágenes, vídeos y otros elementos contenidos en una página web. Además, este lenguaje se basa en el concepto de hipertexto, un sistema que permite enlazar varias páginas o documentos entre sí lo cual facilita la navegación a través de las mismas. Es importante aclarar que HTML no es un lenguaje de programación pues para que un lenguaje sea considerado como tal debe cumplir con ciertos criterios por ejemplo debe ser capaz de generar y ejecutar flujos de procesos en función de varios parámetros y además debe ser capaz de repetir dichos procesos un determinado número de veces. También debe permitir trabajar con variables y funciones aritméticas y HTML no cumple con tales características razón por la cual no es posible definir la funcionalidad de una página web mediante este lenguaje. Para esto debemos utilizar lenguajes de programación como JavaScript, PHP, ASP entre muchos otros más, pero tampoco HTML debe ser empleado para definir la apariencia o aspecto visual de una página o sitio web, a pesar de disponer de una serie de instrucciones que pueden ser empleadas para tales fines, sin embargo esto no es recomendado y en su lugar debemos hacer uso de un lenguaje de estilos como CSS. Entonces si con HTML no podemos definir la funcionalidad ni el aspecto visual de una web para qué sirve exactamente este lenguaje. La respuesta es que HTML es el lenguaje que utilizamos como base para crear los contenidos de una web, es decir con HTML definimos párrafos, listas con viñetas, tablas, formularios, reproductores de audio y video, imágenes, botones y cualquier otro elemento que pueda contener una web y todo esto lo hacemos por medio de una serie de instrucciones que debemos escribir en un archivo de texto llamado comúnmente como documento HTML, el cual va alojado o guardado ya sea en un servidor remoto o bien en el disco duro de nuestro computador para luego ser mostradas a los usuarios por medio de los navegadores web como Firefox, Google Chrome, Opera, Safari, Edge entre otros más, los cuales no muestran estas instrucciones de forma literal sino que las interpretan para luego renderizarlas, es decir para mostrarlas en forma de contenido a través de las pantallas de los dispositivos que empleamos para conectarnos a Internet.

Desde su creación en 1991 se han publicado cinco versiones de este lenguaje la más reciente es HTML5 publicada a finales de 2014.

Esta versión incorpora una serie de elementos y características que hasta ese momento eran imposibles de implementar sin el uso de complementos de terceros

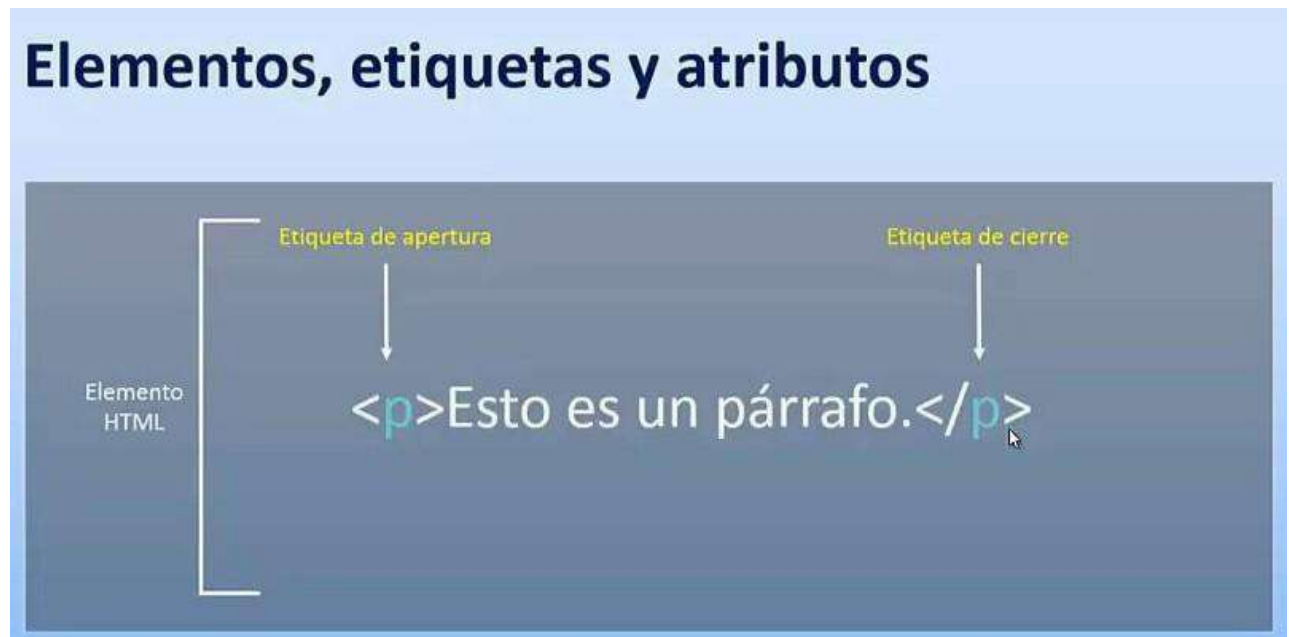
como por ejemplo: reproducción de audio y videos independencia de plugins como Adobe Flash o Microsoft Silverlight, asimismo integración de canvas para crear imágenes dinámicas mediante Javascript y también integración de SVG para crear gráficos vectoriales, además incorporación de nuevos elementos semánticos que permiten definir el esquema principal de un documento o página HTML, también aceleración gráfica en los navegadores, incorporación de APIs o interfaces de programación de aplicaciones, compatibilidad con todos los navegadores web, mejoras en los formularios y muchas otras más.

Elementos, etiquetas y atributos

HTML cuenta con varios elementos y cada uno de ellos se utiliza para un fin específico.

Por ejemplo, tenemos elementos para mostrar un encabezado o un párrafo o elementos para insertar una imagen, para crear un formulario, un botón, una tabla y muchos otros más.

Sin embargo, para poder trabajar con HTML y agregar contenido a un sitio web es importante conocer la estructura básica de un elemento HTML, generalmente un elemento HTML está compuesto por las siguientes partes.



En primer lugar, tenemos las etiquetas que son el componente base de HTML y se emplean para marcar el inicio y el final de un elemento.

Las etiquetas se crean mediante corchetes angulares, que se representan con los signos de < /> dentro de éstos debemos cerrar una letra o palabra clave que identifica a cada elemento. Esto significa que no podemos colocar cualquier letra o

palabra como etiqueta sino aquellas que han sido reservadas para un propósito específico.

En la mayoría de los casos las etiquetas se presentan en pares, es decir tenemos una etiqueta de apertura(<) y una etiqueta de cierre(>). La etiqueta de apertura tiene la función de declarar el inicio del elemento mientras que la etiqueta de cierre indica el final del mismo.

Se diferencia de la primera porque lleva una barra diagonal delante de la letra o palabra clave que identifica al elemento.

Pero además de las etiquetas normalmente un elemento HTML cuenta con algún contenido el cual se coloca entre la etiqueta de apertura y la etiqueta de cierre y consiste en un fragmento de texto que se verá afectado por el elemento que lo contiene.

3. Elementos, etiquetas y atributos

Elementos, etiquetas y atributos



Por ejemplo, dependiendo de la etiqueta empleada un texto se mostrará en negrita, cursiva, subrayado etc.

Es bueno aclarar que no todos los elementos HTML encierran un contenido y a estos se les conoce como elementos vacíos y dado que no lleva un contenido, estos elementos también prescinden de la etiqueta de cierre es decir se muestran solamente con la etiqueta de apertura.

Pero además del contenido un elemento HTML puede disponer o no de un atributo, es decir una palabra clave cuya función es la de agregar nuevas características a las etiquetas.

El atributo siempre deberá escribirse dentro de la etiqueta de apertura separándolo del nombre del elemento mediante un espacio en blanco.

Los atributos se componen de dos partes, en primer lugar tenemos el nombre del atributo el cual define la propiedad que queremos agregar a la etiqueta que lo contiene.

Por otro lado, tenemos el valor del atributo mismo que siempre deberá escribirse entre comillas dobles.

Este valor puede ser una palabra clave, una url e incluso un número, un elemento puede tener asociado varios atributos e incluso un atributo puede admitir más de un valor a la vez.

Elementos, etiquetas y atributos



También cabe aclarar que no todos los elementos deben llevar asociado un atributo y del mismo modo no todos los atributos pueden ser utilizados con cualquier etiqueta.

También es importante tener en cuenta que las etiquetas y los atributos no son sensibles a las mayúsculas y minúsculas.

Esto significa que podemos escribirlos tanto con letras mayúsculas o minúsculas y el navegador las interpretará correctamente.

Elementos, etiquetas y atributos

```
<P CLASS="INTRO">Esto es un párrafo.</P>  
<p class="intro">Esto es un párrafo.</p>
```

Sin embargo, las buenas prácticas recomiendan siempre escribir las etiquetas y atributos en letras minúsculas, razón por la cual siempre trataremos de evitar las letras mayúsculas para describir los nombres de las etiquetas y de los atributos y sus valores.

HTML también nos permite anidar elementos, es decir incluir elementos dentro de otros elementos que contienen a otro, se denomina elemento padre o elemento contenedor mientras que el o los elementos que éste contiene se denominan elementos hijos.

Elementos, etiquetas y atributos



La anidación de elementos es algo muy común en HTML y de hecho toda la estructura de este lenguaje está basada en elementos anidados.

Por último, es importante saber que HTML clasifica todos los elementos en dos grupos, elementos de bloque y elementos de línea, los elementos de bloque se posicionan uno debajo de otro y ocupan todo el espacio de su elemento padre o de la ventana del navegador.

Los párrafos los encabezados, listas son ejemplos de elementos de bloque pues ocupan todo el ancho disponible de su contenedor.

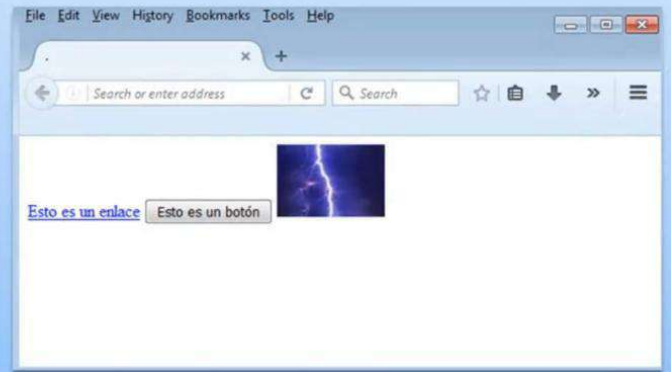
En cambio, los elementos en línea se posicionan uno al lado de otro en una misma línea, los enlaces, las imágenes, los botones son ejemplos de elementos en línea.

Clasificación de los elementos HTML

Elementos de bloque



Elementos en línea



Etiquetas HTML más utilizadas

HTML es un lenguaje que se basa en el uso de etiquetas predefinidas. Esto quiere decir que para poder crear los contenidos de una página web, es necesario conocer y aprender las diferentes etiquetas que componen cada uno de los elementos HTML.

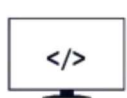
A continuación puedes ver el listado de las etiquetas más utilizadas en HTML.

Etiquetas básicas

Etiqueta de apertura	Etiqueta de cierre	Descripción
<!DOCTYPE>		Define el tipo de documento.
<html>	</html>	Define un documento HTML.
<head>	</head>	Especifica la cabecera del documento HTML.
<meta>		Define metadatos.
<title>	</title>	Define el título de un documento HTML.
<body>	</body>	Define el cuerpo de un documento HTML.
 		Inserta un salto de línea.
<hr>		Traza una línea horizontal.
<!-- -->		Permite agregar un comentario interno en un documento.

Formato de texto

Etiqueta de apertura	Etiqueta de cierre	Descripción
<code><p></code>	<code></p></code>	Define los párrafos que contiene el documento.
<code></code>	<code></code>	Se utiliza para crear una lista desordenada.
<code></code>	<code></code>	Se utiliza para crear una lista ordenada.
<code></code>	<code></code>	Define los elementos dentro de una lista.
<code></code>	<code></code>	Indica énfasis fuerte en un texto determinado.
<code></code>	<code></code>	Representa un texto con un estilo en negrita.
<code><i></code>	<code></i></code>	Marca un texto con un estilo en cursiva o itálica.
<code></code>	<code></code>	Se emplea para indicar énfasis en un texto determinado..
<code><h1></code>	<code></h1></code>	Se utilizan para definir los encabezados en un documento HTML.
<code><h2></code>	<code></h2></code>	
<code><h3></code>	<code></h3></code>	
<code><h4></code>	<code></h4></code>	
<code><h5></code>	<code></h5></code>	
<code><h6></code>	<code></h6></code>	
<code><mark></code>	<code></mark></code>	Resalta un fragmento texto con un color de fondo.
<code><small></code>	<code></small></code>	Se utiliza para mostrar un texto en tamaño pequeño.
<code></code>	<code></code>	Representa contenido que ha sido removido de un documento.



Formato de texto (2)

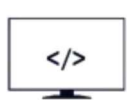
Etiqueta de apertura	Etiqueta de cierre	Descripción
<code><ins></code>	<code></ins></code>	Representa contenido que ha sido insertado al documento
<code><cite></code>	<code></cite></code>	Representa a una cita del título de una obra.
<code><q></code>	<code></q></code>	Permite insertar citas en forma de líneas cortas de texto
<code><blockquote></code>	<code></blockquote></code>	Permite insertar citas en forma de bloques de contenido
<code><sub></code>	<code></sub></code>	Se emplea para representar un subíndice.
<code><sup></code>	<code></sup></code>	Se utiliza para representar un superíndice.
<code></code>	<code></code>	Sirve para aplicar estilo a los textos y para agrupar elementos en línea.

Enlaces

Etiqueta de apertura	Etiqueta de cierre	Descripción
<code><a></code>	<code></code>	Define los enlaces que contiene el documento.
<code><link></code>	<code></link></code>	Enlaza un recurso externo.

Estilos

Etiqueta de apertura	Etiqueta de cierre	Descripción
<code><style></code>	<code></style></code>	Define hojas de estilo dentro del documento HTML.

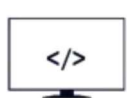


Seccionamiento del contenido

Etiqueta de apertura	Etiqueta de cierre	Descripción
<code><nav></code>	<code></nav></code>	Se utiliza para crear un menú de navegación.
<code><section></code>	<code></section></code>	Define una sección en el documento HTML.
<code><article></code>	<code></article></code>	Define un contenido específico dentro de un documento HTML y que tiene sentido por sí solo.
<code><aside></code>	<code></aside></code>	Se utiliza para visualizar contenido secundario.
<code><header></code>	<code></header></code>	Contiene el código HTML correspondiente a la cabecera del documento o de un bloque de contenido.
<code><footer></code>	<code></footer></code>	Representa el pie de página del documento o de un bloque de contenido..
<code><div></code>	<code></div></code>	Sirve para crear secciones o agrupar un bloque de elementos.

Scripting

Etiqueta de apertura	Etiqueta de cierre	Descripción
<code><script></code>	<code></script></code>	Inserta a un script ejecutable.
<code><noscript></code>	<code></noscript></code>	Para navegadores que no soportan scripts.



Imágenes

Etiqueta de apertura	Etiqueta de cierre	Descripción
<code></code>		Inserta una imagen en un documento.
<code><figure></code>	<code></figure></code>	Incrusta una figura ilustrada.
<code><figcaption></code>	<code></figcaption></code>	Representa la leyenda de un elemento <code><figure></code>

Audio y vídeo

Etiqueta de apertura	Etiqueta de cierre	Descripción
<code><audio></code>	<code></audio></code>	Inserta contenido de audio.
<code><video></code>	<code></video></code>	Permite embeber vídeos en un documento HTML.
<code><source></code>		Especifica recursos multimedia alternativos para los elementos <code><video></code> y <code><audio></code> .
<code><track></code>	<code></track></code>	Agrega subtítulos a elementos multimedia.

Contenido embebido

Etiqueta de apertura	Etiqueta de cierre	Descripción
<code><iframe></code>	<code></iframe></code>	Incorpora recursos o contenido externo en la página actual
<code><object></code>	<code></object></code>	Inserta objetos en un documento.



Contenido embebido (2)

Etiqueta de apertura	Etiqueta de cierre	Descripción
<code><param></code>		Parámetros de ejecución para un objeto.
<code><embed></code>		Permite incrustar una aplicación externa (plug-in).

Tablas

Etiqueta de apertura	Etiqueta de cierre	Descripción
<code><table></code>	<code></table></code>	Define una tabla dentro de un documento HTML.
<code><th></code>	<code></th></code>	Define una celda de encabezado en una tabla.
<code><tr></code>	<code></tr></code>	Inserta una fila dentro de una tabla.
<code><td></code>	<code></td></code>	Inserta una celda de datos dentro de una tabla.
<code><thead></code>	<code></thead></code>	Agrupar las cabeceras de una tabla
<code><tfoot></code>	<code></tfoot></code>	Representa la sección de pie de una tabla
<code><tbody></code>	<code></tbody></code>	Agrupar los contenidos de una tabla.
<code><caption></code>	<code><caption></code>	Asigna un título a una tabla.



Formularios

Etiqueta de apertura	Etiqueta de cierre	Descripción
<code><form></code>	<code></form></code>	Contenedor de un formulario.
<code><input></code>		Define un campo de entrada de datos.
<code><textarea></code>	<code></textarea></code>	Define un campo para edición multi-línea de texto plano.
<code><button></code>	<code></button></code>	Se utiliza para crear botones de un formulario.
<code><select></code>	<code></select></code>	Inserta un menú desplegable.
<code><option></code>	<code></option></code>	Define una opción dentro de un menú desplegable.
<code><label></code>	<code></label></code>	Agrega una etiqueta.
<code><fieldset></code>	<code></fieldset></code>	Agrupar los controles de un formulario.
<code><legend></code>	<code><legend></code>	Crea un título para uno de los grupos de campos (<code><fieldset></code>).



Atributos globales HTML

Los atributos son *keywords* o palabras clave que se agregan a los elementos HTML y sirven para definir nuevas características.

En general, los atributos se dividen en dos grupos:

- **Atributos específicos**, que son propios de un elemento determinado. Los estudiaremos en detalle a medida que vayamos avanzando en este módulo.
- **Atributos globales**, que se aplican a todos los elementos.

En la siguiente tabla se muestra un listado de los principales atributos globales de HTML.

Atributo	Descripción
accesskey	Genera un acceso de teclado para el elemento.
class	Este atributo permite a CSS y JavaScript seleccionar y acceder a elementos específicos.
contenteditable	Indica al navegador si debería permitir al usuario editar los contenidos del elemento.
contextmenu	Es el <i>id</i> de un elemento <code><menu></code> que permite que pueda ser utilizado como menú contextual para este elemento.
data-*	Define datos personalizados para el elemento en el cual ha sido declarado.
dir	Indica la direccionalidad del texto del elemento.
draggable	Indica si el elemento puede ser arrastrado, usando la Drag and Drop API.



Atributo	Descripción
dropzone	Indica qué tipos de contenidos pueden soltarse sobre un elemento “arrastrable”.
hidden	Permite ocultar un elemento.
id	Define un identificador cuyo valor no debe repetirse en todo el documento.
lang	Define el lenguaje utilizado en el contenido del elemento.
spellcheck	Indica si el elemento debería estar sujeto a revisiones ortográficas.
style	Define un conjunto de declaraciones CSS que serán aplicadas al elemento
tabindex	Indica la posición del elemento en la navegación por tabulación, es decir, la navegación a través de aquellos elementos en los que se puede hacer foco presionando la tecla TAB.
title	Muestra una línea de texto que representa información descriptiva acerca del elemento.

HTML dispone de otros *atributos globales* que aún se encuentran en fase experimental, razón por la cual no han sido incluidos en este listado.

Estructura básica de un documento HTML

Vamos crear una estructura básica para un documento HTML.

Lo primero que haremos será crear una carpeta en nuestro computador en la cual guardaremos todos los archivos que vamos a utilizar.

Dentro de la carpeta documentos creamos una carpeta y la nombramos como **frontend** luego dentro de la misma creamos otra carpeta que nombraremos como **html**.

Lo siguiente que haremos será crear nuestro primer documento HTML, para esto necesitamos utilizar el editor de código Visual Studio Code así que abrimos nuestro editor de código y el siguiente paso será abrir la carpeta frontend en el visual studio, damos click en **Archivo** después en **Abrir carpeta**, buscamos la carpeta frontend en documentos click sobre la carpeta y presionamos **seleccionar carpeta** posteriormente crear un nuevo documento presionando sobre la pestaña Archivo de la barra de menú y luego sobre la opción que dice Nuevo archivo de texto, seguidamente vamos a asignar un nombre a este documento, presionamos sobre la pestaña Archivo de la barra de menú y luego sobre la opción que dice Guardar, de inmediato se abre un cuadro de diálogo para que guardemos nuestro documento, asegúrate de guardarlo dentro de la carpeta HTML que acabamos de crear, vamos a asignar un nombre a este documento debes tener en cuenta que existen algunas reglas que debemos seguir al asignar un nombre a un documento HTML. Lo primero es que el nombre siempre deberá escribirse en minúsculas y sin espacios en blanco, ni caracteres especiales como tildes, asteriscos, @, la letra ñ o cualquier carácter especial el único signo permitido es el guion medio.

Al final del nombre siempre deberemos agregar la extensión .html para que nuestro documento pueda ser interpretado correctamente por todos los navegadores web. Por ejemplo, vamos asignarle el nombre de prueba.html presionamos el botón Guardar y con esto hemos creado nuestro primer documento HTML y por lo tanto podemos empezar a crear nuestra estructura.

Ahora vamos a escribir **html:5** pulsamos la **tecla de tabulación o enter** y automáticamente el editor ha creado una estructura básica HTML.

Explicación línea por línea qué hace cada parte de ese código HTML básico:

<!DOCTYPE html>

Significado: Es una declaración que indica al navegador que el documento está escrito en HTML5, la versión más moderna de HTML.

Función: Asegura que el navegador interprete correctamente el código según los estándares actuales del lenguaje HTML.

<html lang="en">

Significado: Abre el documento HTML y establece el idioma del contenido como inglés (lang="en").

Función: Ayuda a los navegadores, motores de búsqueda y lectores de pantalla a entender el idioma del contenido para una mejor accesibilidad y SEO.

Por ejemplo, si nuestro idioma es el español el valor deberá ser igual a ES, si fuese el idioma inglés sería EN, Si fuese francés sería FR y así sucesivamente, en este caso vamos a fijarlo como ES.

<head>

Significado: Es la sección del documento que no se muestra directamente al usuario.

Función: Contiene metadatos, el título de la página, enlaces a hojas de estilo, scripts, etc.

<meta charset="UTF-8">

Significado: Define la codificación de caracteres como UTF-8.

Función: Permite que el sitio pueda mostrar correctamente caracteres especiales, acentos y símbolos (por ejemplo, ñ, é, %, etc.). Como puedes ver este elemento es un elemento vacío lo que significa que no encierra ningún contenido ni finaliza con una etiqueta de cierre dentro de la etiqueta de apertura.

<meta name="viewport" content="width=device-width, initial-scale=1.0">

Significado: Establece la configuración de visualización para dispositivos móviles.

Función: Hace que el diseño se adapte bien a diferentes tamaños de pantalla. Es esencial para que el sitio sea responsivo (responsive). Como puedes ver este elemento es un elemento vacío lo que significa que no encierra ningún contenido ni finaliza con una etiqueta de cierre dentro de la etiqueta de apertura.

<title>Document</title>

Significado: Define el título de la página web.

Función: Este texto aparece en la pestaña del navegador cuando visitas la página.

Este es el título que visualizamos en la pestaña que se encuentra en la parte superior izquierda del navegador así que vamos a añadir un título para nuestro documento en la etiqueta de apertura y la etiqueta de cierre, escribimos el siguiente texto **Mi página web**.

</head>

Significado: Cierra la sección <head>.

<body>

Significado: Abre el cuerpo del documento HTML.

Función: Aquí va todo el contenido visible de la página web: textos, imágenes, botones, enlaces, formularios, etc.

Por ejemplo vamos a agregar un encabezado y un párrafo, esto lo hacemos mediante los elementos **h1** y **p**. Comenzamos con el encabezado escribimos `<h1>` luego la etiqueta de cierre `</h1>`, entre la etiqueta de apertura y la etiqueta de cierre escribimos el texto de nuestro encabezado por ejemplo, **Hola mundo** y debajo añadimos su párrafo, así que pulsamos la tecla Enter y escribimos `<p>` y luego la etiqueta de cierre `</p>`, entre la etiqueta de apertura y la etiqueta de cierre escribimos el texto de nuestro párrafo, en este caso **Esta es mi primera página web** el siguiente paso será ejecutar todo este código.

Para esto debemos guardar los cambios pulsando sobre la pestaña Archivo de la barra de menú y luego sobre la opción que dice Guardar, el siguiente paso será visualizar este documento en el navegador, para esto pulsamos el botón derecho del ratón y luego sobre la opción que dice Open with live server y así es como se visualiza nuestra primera página web.

</body>

Significado: Cierra la sección `<body>`.

</html>

Significado: Cierra el documento HTML.

Formatear textos en HTML

En esta lección veremos cómo formatear texto en HTML pero primero vamos a crear un nuevo documento así que con el editor de código abierto presionamos sobre la pestaña Archivo de la barra de menú y luego sobre la opción que dice Nuevo archivo de texto o presionamos las teclas de control y N, de inmediato vamos a asignar un nombre a este documento presionando sobre la pestaña Archivo de la barra de menú y luego sobre la opción que dice Guardar o presionamos las teclas de control y S asegúrate de guardarlo dentro de la carpeta HTML que creamos anteriormente.

Vamos a nombrar nuestro documento como formato-texto.html presionamos el botón Guardar y de inmediato vamos a proceder a crear una estructura HTML como la que hicimos anteriormente, aquí lo único que tenemos que hacer es modificar el valor del atributo **LANG** y fijarlo como **es** y también modificar el texto del elemento **TITLE**, por ejemplo, vamos a escribir Formatear textos en HTML y con esto ya estamos listos para comenzar a trabajar.

Empecemos con lo más básico y esto es definir un párrafo en HTML.

Para esto necesitamos agregar el elemento P dentro del elemento body o cuerpo de nuestro documento HTML.

Recuerda que anteriormente dijimos que todos los elementos visibles de una página web se deben colocar dentro del elemento Body así que entre la etiqueta de apertura y la etiqueta de cierre de este elemento añadimos un elemento P se escribe `<p> </p>` ahora añadir un texto entre la etiqueta de apertura y la etiqueta de cierre del elemento P, por ejemplo:

```
<p>Esto es un párrafo</p>
<p>Esto también es un párrafo</p>
```

Vamos a guardar los cambios y vamos a visualizar este documento en el navegador, clic derecho sobre cualquier parte del fondo negro del editor de código y luego clic izquierdo sobre Open with Live Server y se verá así

Esto es un párrafo

Esto también es un párrafo

Vamos a regresar al editor de códigos y vamos a colocar ambos párrafos uno al lado de otro así que vamos a seleccionar el segundo párrafo y con el puntero del ratón lo voy a colocar al lado del primero.

```
<p>Esto es un párrafo</p> <p>Esto también es un párrafo</p>
```

Vamos a guardar los cambios presionando las teclas de control y S y regresamos al navegador, clic derecho sobre cualquier parte del fondo negro del editor de código y luego clic izquierdo sobre Open with Live Server, si te fijas el navegador sigue mostrando cada párrafo uno debajo de otro a pesar de que en el editor los colocamos uno al lado de otro. La razón de esto se debe a que los párrafos o elementos en HTML son considerados como elementos de bloque.

Recuerda que anteriormente dijimos que HTML clasifica los elementos en dos grupos elementos de bloque y elementos en línea.

Pues bien, los párrafos son elementos de bloque y una de las características principales de este tipo de elementos es que ocupan el ancho total del elemento que los contiene.



Entidades HTML de uso más común

Las entidades HTML son códigos que se utilizan para insertar símbolos especiales en una página Web. Como regla general, todas las entidades HTML se deben escribir con el signo *et* o *ampersand* (&) al inicio, y con el signo de punto y coma (;) al final.

A continuación, puedes ver un listado con las entidades HTML más utilizadas.

Caracter	Entidad	Código	Descripción
	¦	¦	Barra vertical rota
“	“	“	Apertura de comillas dobles
”	”	”	Cierre de comillas dobles
‘	‘	‘	Apertura de comillas simples
’	’	’	Cierre de comillas simples
«	«	«	Comilla de ángulo (izquierda)
»	»	»	Comilla de ángulo (derecha)
◀	‹	‹	Cita de ángulo izquierdo único
▶	›	›	Cita de ángulo recto individual
§	§	§	Símbolo de sección
¶	¶	¶	Símbolo de párrafo
•	•	•	Lista de punto grande
·	·	·	Lista de punto medio
...	…	…	Elipsis horizontal
		|	Barra vertical
&	&	&	Ampersand



Caracter	Entidad	Código	Descripción
¤	¤t;	¤	Símbolo de moneda genérico
\$		$	Símbolo de dólar
¢	¢	¢	Símbolo de centavo
£	£	£	Libra
¥	¥	¥	Yen
€	€	€	Símbolo del euro
©	©	©	Copyright
®	®	®	Registered symbol
™	™	™	Trademark symbol
<	<	<	Menos que
>	>	>	Mayor que
≤	≤	≤	Menor que o igual a
≥	≥	≥	Mayor que o igual a
×	×	×	Símbolo de multiplicación
÷	÷	÷	Símbolo de división
−	−	−	Símbolo de resta
±	±	±	Símbolo más / menos
≠	≠	≠	No es igual
¹	¹	¹	Superíndice 1



Caracter	Entidad	Código	Descripción
²	²	²	Superíndice 2
³	³	³	Superíndice 3
½	½	½	Fracción ½
¼	¼	¼	Fracción ¼
¾	¾	¾	Fracción ¾
‰	‰	‰	Por milla
°	°	°	Símbolo de grado
√	√	√	Raíz cuadrada
∞	∞	∞	Infinito
←	←	←	Flecha izquierda
↑	↑	↑	Flecha arriba
→	→	→	Flecha derecha
↓	↓	↓	Flecha hacia abajo
↔	↔	↔	Flecha izquierda derecha
↶	↵	↵	Flecha de retorno de carro
⌈	⌈	⌈	Left ceiling
⌋	⌉	⌉	Right ceiling



Caracter	Entidad	Código	Descripción
⌋	⌊	⌊	Left floor
⌋	&rffloor;	⌋	Right floor
♠	♠	♠	Spade
♣	♣	♣	Club
♥	♥	♥	Heart
♦	♦	♦	Diamond
◇	◊	◊	Lozenge
†	†	†	Dagger
‡	‡	‡	Double dagger
¡	¡	¡	Signo de apertura de exclamación
¿	¿	¿	Signo de apertura de interrogación
	 		Non-break space
	&ensp		Single tab space
		 	Six-per-em-space

Formatear textos en HTML(II)

Otra característica de HTML con respecto a los párrafos, es que este lenguaje ignora todos los espacios en blanco que no sean aquellos que separan cada palabra.

Esto significa que si en un texto dejamos más de un espacio en blanco entre palabra y palabra HTML pasará por alto cualquier espacio en blanco adicional y por consiguiente no lo mostrará en el navegador.

Vamos a ver un ejemplo así que regresamos al editor de códigos y lo que haremos será añadir varios espacios en blanco

```
1 <!DOCTYPE html>
2 <html lang="es">
3 <head>
4     <meta charset="UTF-8">
5     <title>Formatear textos en HTML</title>
6 </head>
7 <body>
8     <p>Esto es un párrafo.</p> <p>Esto también es un párrafo.</p>
9
10 </body>
11 </html>
```

vamos a colocar este párrafo debajo del primero para tener un código mucho más organizado

```
1 <!DOCTYPE html>
2 <html lang="es">
3 <head>
4     <meta charset="UTF-8">
5     <title>Formatear textos en HTML</title>
6 </head>
7 <body>
8     <p>Esto es un párrafo.</p>
9     <p>Esto también es un párrafo.</p>
10
11 </body>
12 </html>
```

y entre la palabra **Esto** y la palabra **también** utilizamos la barra espaciadora para añadir espacios en blanco, agreguemos cinco espacios en blanco.

```
1 <!DOCTYPE html>
2 <html lang="es">
3 <head>
4   <meta charset="UTF-8">
5   <title>Formatear textos en HTML</title>
6 </head>
7 <body>
8   <p>Esto es un párrafo.</p>
9   <p>Esto | también es un párrafo.</p>
10
11 </body>
12 </html>
```

Vamos a guardar los cambios presionando las teclas de **Control y S**, vamos a regresar al navegador, clic derecho sobre cualquier parte del fondo negro del editor de código y luego clic izquierdo sobre Open with Live Server y como puedes ver HTML ha ignorado los espacios en blanco que hemos añadido.

Esto es un párrafo.

Esto también es un párrafo.

Para solucionar esto, HTML dispone de un listado de códigos denominados entidades HTML que no son más que un conjunto de caracteres especiales como por ejemplo letras con acentos, pertenecientes o no al alfabeto latino internacional, signos de puntuación o tipográficos, caracteres especiales y por supuesto espacios en blanco. Así que regresamos al editor de códigos, vamos a eliminar los espacios en blanco que hemos añadido

```

1 <!DOCTYPE html>
2 <html lang="es">
3 <head>
4     <meta charset="UTF-8">
5     <title>Formatear textos en HTML</title>
6 </head>
7 <body>
8     <p>Esto es un párrafo.</p>
9     <p>Esto [.....] también es un párrafo.</p>
10
11 </body>
12 </html>

```

y en el lugar donde queremos insertar espacios en blanco escribimos la siguiente entidad HTML, comenzamos con el signo de &, luego el signo de numeral o almohadilla. #, después agregamos el siguiente código numérico 8198.  

```

1 <!DOCTYPE html>
2 <html lang="es">
3 <head>
4     <meta charset="UTF-8">
5     <title>Formatear textos en HTML</title>
6 </head>
7 <body>
8     <p>Esto es un párrafo.</p>
9     <p>Esto &#8198 también es un párrafo.</p>
10
11 </body>
12 </html>

```

I

Vamos a seleccionar este código con el puntero del ratón y lo vamos a copiar 4 veces, para esto presionamos las teclas de **Control y C** dejamos un espacio y lo volvemos a pegar presionando las teclas de **Control y V**.

```

1 <!DOCTYPE html>
2 <html lang="es">
3 <head>
4   <meta charset="UTF-8">
5   <title>Formatear textos en HTML</title>
6 </head>
7 <body>
8   <p>Esto es un párrafo.</p>
9   <p>Esto &#8198 &#8198 &#8198 &#8198 también es un párrafo.</p>
10
11 </body>
12 </html>

```

Ahora si vamos a guardar los cambios y vamos a regresar al navegador, refrescamos la página, si te fijas el navegador ahora muestra los espacios en blanco que hemos definido a través de la entidad HTML que insertamos en el editor de códigos.

Esto es un párrafo.

Esto también es un párrafo.

HTML tampoco nos permite agregar saltos de línea al texto de un párrafo cuando presionamos la tecla **Enter** en el editor tal como ocurriría si estuviésemos escribiendo en algún procesador de palabras. Esto puede suponer un problema si por ejemplo queremos introducir un punto y aparte cuando estábamos escribiendo un texto.

Vamos a ver un ejemplo en el editor de códigos, por ejemplo, vamos a crear un nuevo párrafo y añadirle algo de texto así que pulsamos la tecla Enter y añadimos un elemento P.

Vamos a agregarle un texto de relleno para esto vamos a añadirle un texto Lorem Ipsum que es un texto de relleno que desde hace varios siglos ha sido utilizado por la industria de la impresión gráfica para saber por ejemplo cómo se comportará la página de un material impreso cuando se tenga el texto definitivo.

Tan solo basta con que entre la etiqueta de apertura y la etiqueta de cierre del elemento P escribamos la palabra Lorem y luego pulsemos la tecla de tabulación o tab y fíjate como automáticamente el editor ha añadido un texto de relleno o texto Loreem ipsum.

Vamos a crear nuestro ejemplo y lo que haremos será separar todo este texto como si estuviéramos añadiendo un punto y aparte.

Así que en este punto pulsamos la tecla Enter y haremos lo propio en este punto, vamos a guardar los cambios

```
1 <!DOCTYPE html>
2 <html lang="es">
3 <head>
4   <meta charset="UTF-8">
5   <title>Formatear textos en HTML</title>
6 </head>
7 <body>
8   <p>Esto es un párrafo.</p>
9   <p>Esto &#8198 &#8198 &#8198 &#8198 también es un párrafo.</p>
10  <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit.
11    Dolor architecto consequuntur accusantium, provident repellendus reiciendis unde doloribus I
12    quae voluptatibus ducimus soluta ipsa neque totam aspernatur ut eaque nisi, est obcaecati.</p>
13
14 </body>
15 </html>
```

y vamos a visualizar este documento en el navegador

Esto es un párrafo.

Esto también es un párrafo.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Dolor architecto consequuntur accusantium, provident repellendus reiciendis unde doloribus quae voluptatibus ducimus soluta ipsa neque totam aspernatur ut eaque nisi, est obcaecati.

y si te fijas el navegador ha ignorado los saltos de línea que hemos añadido en el editor.

Para corregir esto, HTML dispone del elemento br el cual nos permite crear un salto de línea entre textos.

Vamos a regresar al editor de códigos y lo que haremos será añadir un elemento br

```
1 <!DOCTYPE html>
2 <html lang="es">
3 <head>
4   <meta charset="UTF-8">
5   <title>Formatear textos en HTML</title>
6 </head>
7 <body>
8   <p>Esto es un párrafo.</p>
9   <p>Esto &#8198 &#8198 &#8198 &#8198 también es un párrafo.</p>
10  <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. <br>
11    Dolor architecto consequuntur accusantium, provident repellendus reiciendis unde doloribus <br>
12    quae voluptatibus ducimus soluta ipsa neque totam aspernatur ut eaque nisi, est obcaecati.</p>
13
14 </body>
15 </html>
```

si te fijas br es un elemento vacío.

Esto quiere decir que no encierra ningún contenido, pero tampoco finaliza con una etiqueta de cierre.

Vamos a guardar los cambios y vamos a regresar al navegador, clic derecho sobre cualquier parte del fondo negro del editor de código y luego clic izquierdo sobre Open with Live Server y como puedes ver ahora el navegador respeta los saltos de línea que hemos añadido

Esto es un párrafo.

Esto también es un párrafo.

Lorem ipsum dolor sit amet, consectetur adipisicing elit.

Dolor architecto consequuntur accusantium, provident repellendus reiciendis unde doloribus quae voluptatibus ducimus soluta ipsa neque totam aspernatur ut eaque nisi, est obcaecati.

Ahora veamos cómo podemos escribir texto en negrita

Para esto HTML dispone de los elementos Strong y B.

`<b> Este texto se muestra en negrita. </b>`

`<strong> Este texto se muestra en negrita. </strong>`

Vamos a guardar los cambios presionando las teclas de control y S y vamos a visualizar este documento en el navegador, clic derecho sobre cualquier parte del fondo negro del editor de código y luego clic izquierdo sobre Open with Live Server, lo primero que podemos notar es que ambos textos se muestran uno al lado de otro. La razón de esto es que tanto los elementos b como los elementos strong son elementos en línea y HTML divide los elementos en dos grupos: elementos de bloque y elementos de línea.

Hola mundo

Esta es mi primera página web

Este texto se muestra en negrita. Este texto se muestra en negrita.

Pues bien, en el caso de los elementos de línea el navegador por defecto nos muestra uno al lado de otro en una misma línea, ahora vamos a añadir un elemento salto de línea para que el navegador muestre ambos elementos uno debajo de otro y así tengamos una mejor visualización

`<b> Este texto se muestra en negrita. </b><br>`

Este texto se muestra en negrita.

Vamos a guardar los cambios, regresamos al navegador, clic derecho sobre cualquier parte del fondo negro del editor de código y luego clic izquierdo sobre Open with Live Server y ahora si ambos textos se muestran uno debajo de otro y si te fijas no existe ninguna diferencia entre los mismos es decir ambos se muestran en negrita.

Hola mundo

Esta es mi primera página web

Este texto se muestra en negrita.

Este texto se muestra en negrita.

Sin embargo, la forma como los navegadores y los motores de búsqueda como Google o Bing se interpretan es totalmente diferente.

Por un lado, cuando encerramos un texto dentro de un elemento B estamos indicando a los navegadores que el texto debe aparecer en negrita sin ningún otro significado.

En cambio, cuando encerramos su texto dentro de un elemento Strong indicamos a los navegadores que deben resaltar el texto otorgándole la máxima importancia.

Esto significa que el elemento Strong es un elemento semántico es decir un elemento que informa al navegador o a cualquier otro sistema que procese una página web sobre la naturaleza del contenido que encierra lo cual ayuda al posicionamiento web.

Por tanto, si queremos que un texto se muestre simplemente en negrita sólo para fines decorativos lo encerramos dentro de un elemento B

En cambio, si queremos que el mismo texto se muestre en negrita, pero con cierto énfasis lo encerramos dentro de un elemento Strong

Ahora veamos cómo podemos formatear texto para que se muestre en un estilo de fuente itálica o cursiva.

Este texto se muestra en estilo itálica o cursiva.

Este texto se muestra en estilo itálica o cursiva.

Antes de guardar los cambios vamos a añadir un elemento br debido a que los elementos i y los elementos em son elementos de línea, para que se muestren uno debajo de otro tenemos que añadir un elemento br, ahora sí vamos a guardar los cambios y visualizar este documento en el navegador, clic derecho sobre cualquier parte del fondo negro del editor de código y luego clic izquierdo sobre Open with

Live Server y como puedes notar visualmente no existe ninguna diferencia entre ambos textos.

Hola mundo

Esta es mi primera página web

Este texto se muestra en negrita.

Este texto se muestra en negrita.

Este texto se muestra en estilo itálica o cursiva.

Este texto se muestra en estilo itálica o cursiva.

Pero tal y como sucede con los elementos Strong y B la forma como los interpretan los navegadores es totalmente distinta. Por un lado, cuando encerramos un texto dentro de un elemento i estamos indicando a los navegadores que dicho texto debe mostrarse en un estilo de fuente itálica o cursiva sin ningún otro significado relevante. En cambio, cuando encerramos un texto dentro de un elemento em establecemos que dicho texto es importante y esto quiere decir que tiene un valor semántico añadido.

Formatear textos en HTML(III)

HTML cuenta con muchos otros elementos para formatear texto.

Veamos algunos de los más utilizados.

La mayoría de estos elementos son elementos de línea, pero para los siguientes ejemplos vamos a anidarlos dentro de un elemento P pues sólo nos interesa formatear un fragmento de cada texto.

De esta forma el navegador presentará cada elemento uno debajo de otro y ya no tendremos que agregar un salto de línea para cada ejemplo.

Comenzamos con el elemento Mark el cual sirve para mostrar un fragmento de texto en forma resaltada como si estuviese marcado con un rotulador de color amarillo.

Vamos a ver un ejemplo:

`<p>La última palabra de este texto se muestra resaltada con un fondo de color`

`<mark>amarillo</mark></p>`

Vamos a guardar los cambios y vamos a visualizar este documento en el navegador, clic derecho sobre cualquier parte del fondo negro del editor de código y luego clic izquierdo sobre Open with Live Server.

Hola mundo

Esta es mi primera página web

Este texto se muestra en negrita.

Este texto se muestra en negrita.

Este texto se muestra en estilo itálica o cursiva.

Este texto se muestra en estilo itálica o cursiva.

La última palabra de este texto se muestra resaltada con un fondo de color **amarillo**

Ahora se puede visualizar la última palabra de este texto resaltada con un fondo de color amarillo como si estuviese marcada con un rotulador y al igual que otros elementos el elemento Mark es un elemento semántico lo que significa que cuando encerramos un texto dentro de este elemento el navegador debe considerarlo como un texto relevante.

También podemos aplicar un tamaño de fuente pequeño a un texto esto lo hacemos mediante el elemento Small.

Veamos un ejemplo:

<p><small>Al enviar este formulario aceptas nuestra "Política de privacidad".</small></p>

Vamos a guardar los cambios presionando las teclas de control y S, vamos a visualizar este documento en el navegador, clic derecho sobre cualquier parte del fondo negro del editor de código y luego clic izquierdo sobre Open with Live Server y tal y como puedes ver el texto se muestra en un tamaño de fuente más pequeño que el tamaño regular HTML.

Hola mundo

Esta es mi primera página web

Este texto se muestra en negrita.

Este texto se muestra en negrita.

Este texto se muestra en estilo itálica o cursiva.

Este texto se muestra en estilo itálica o cursiva.

La última palabra de este texto se muestra resaltada con un fondo de color **amarillo**

Al enviar este formulario aceptas nuestra "Política de privacidad".

HTML también nos permite representar contenido que ha sido removido de un documento. Esto es especialmente útil para informar a los usuarios de un sitio web que ciertos contenidos que fueron publicados en un momento dado eran relevantes, pero en el momento actual ya no lo son, pero se muestran para que los lectores o usuarios puedan beneficiarse de conocer los cambios que han sido aplicados, para esto debemos emplear los elementos **del e ins**.

Vamos a ver un ejemplo supongamos que hemos suscrito un artículo en un sitio web de noticias o en un blog especializado en temas legales y en un párrafo determinado escribimos lo siguiente

<p>Para poder aprobar el proyecto de ley se necesita el 45% de los votos.</p>

Pero supongamos que en el momento actual esto ya no es válido y ahora se necesita el 60 por ciento para aprobar el proyecto de ley. Así que en lugar de eliminar todo ese texto lo encerraríamos dentro de un elemento **del** de la siguiente forma

<p>Para poder aprobar el proyecto de ley se necesita el 45% de los votos</p>

Y en este punto dejamos un espacio e insertamos la información actualizada.

<p>Para poder aprobar el proyecto de ley se necesita el 45% de los votos<ins> 60% de los votos</ins></p>

Vamos a guardar los cambios y vamos a visualizar este documento en el navegador clic derecho sobre cualquier parte del fondo negro del editor de código y luego clic izquierdo sobre Open with Live Server.

Hola mundo

Esta es mi primera página web

Este texto se muestra en negrita.

Este texto se muestra en negrita.

Este texto se muestra en estilo itálica o cursiva.

Este texto se muestra en estilo itálica o cursiva.

La última palabra de este texto se muestra resaltada con un fondo de color **amarillo**

Al enviar este formulario aceptas nuestra "Política de privacidad".

Para poder aprobar el proyecto de ley se necesita el ~~45% de los votos~~ 60% de los votos

Y como puedes ver el navegador muestra tachado el texto que hemos encerrado dentro de un elemento **del** y muestra subrayado el texto que encerramos dentro de un elemento **ins**

Ahora veamos cómo podemos crear una división temática de contenido.

Esto lo hacemos mediante el elemento HR el cual nos permite trazar una línea horizontal debajo de un párrafo para indicar a los navegadores que hay un cambio de tema.

Es decir que no se trata de un elemento decorativo sino de un elemento semántico.

Vamos a ver un ejemplo

<p>En los años transcurridos desde la invención de los primeros ordenadores digitales, el poder de la informática y la capacidad de almacenamiento han aumentado de manera exponencial. Los ordenadores personales y teléfonos inteligentes tienen la capacidad de acceder, modificar, almacenar y compartir medios digitales en las manos de miles de millones de personas.</p>

<p>Muchos dispositivos electrónicos, desde cámaras digitales hasta drones tienen la capacidad de crear, transmitir y visualizar los medios digitales. En combinación con la World Wide Web e Internet, los medios digitales ha transformado la sociedad del siglo XXI de una manera que con frecuencia se compara con el impacto cultural, económica y social de la imprenta. El cambio ha sido tan rápido y de tal magnitud que ha puesto en marcha una transición económica de una economía industrial a una economía basada en la información, creando de un nuevo período en la historia

humana conocida como la era de la información o de la revolución digital. La transición ha creado cierta incertidumbre acerca de las definiciones.

En comparación con los medios de comunicación impresos, los medios de comunicación, y otras tecnologías analógicas, los medios digitales son fáciles de copiar, almacenar, compartir y modificar. Esta calidad de los medios digitales ha dado lugar a cambios significativos en muchas industrias, especialmente el periodismo, la edición, la educación, el entretenimiento y el negocio de la música.

Ahora lo que haremos será añadir un elemento HR entre el segundo y el tercer párrafo.

Así que dejamos un espacio y escribimos `<hr>`, HR es un elemento vacío y como ya sabes este tipo de elemento no encierra ningún contenido ni termina con una etiqueta de cierre.

Vamos a guardar los cambios y vamos a visualizar este documento en el navegador, clic derecho sobre cualquier parte del fondo negro del editor de código y luego clic izquierdo sobre Open with Live Server

Hola mundo

Esta es mi primera página web

Este texto se muestra en **negrita**.

Este texto se muestra en **negrita**.

Este texto se muestra en *estilo itálica o cursiva*.

Este texto se muestra en *estilo itálica o cursiva*.

La última palabra de este texto se muestra resaltada con un fondo de color **amarillo**

Al enviar este formulario aceptas nuestra "Política de privacidad".

Para poder aprobar el proyecto de ley se necesita el ~~45% de los votos~~ **60% de los votos**

En los años transcurridos desde la invención de los primeros ordenadores digitales, el poder de la informática y la capacidad de almacenamiento han aumentado de manera exponencial. Los ordenadores personales y teléfonos inteligentes tienen la capacidad de acceder, modificar, almacenar y compartir medios digitales en las manos de miles de millones de personas.

Muchos dispositivos electrónicos, desde cámaras digitales hasta drones tienen la capacidad de crear, transmitir y visualizar los medios digitales. En combinación con la World Wide Web e Internet, los medios digitales ha transformado la sociedad del siglo XXI de una manera que con frecuencia se compara con el impacto cultural, económica y social de la imprenta. El cambio ha sido tan rápido y de tal magnitud que ha puesto en marcha una transición económica de una economía industrial a una economía basada en la información, creando de un nuevo período en la historia humana conocida como la era de la información o de la revolución digital. La transición ha creado cierta incertidumbre acerca de las definiciones.

En comparación con los medios de comunicación impresos, los medios de comunicación, y otras tecnologías analógicas, los medios digitales son fáciles de copiar, almacenar, compartir y modificar. Esta calidad de los medios digitales ha dado lugar a cambios significativos en muchas industrias, especialmente el periodismo, la edición, la educación, el entretenimiento y el negocio de la música.

Y se puede observar en el navegador se ha añadido una línea separadora entre el segundo y el tercer párrafo debido a que hay un cambio de contenido entre los mismos.

También podemos formatear texto para que se muestre como una cita.

Para esto HTML dispone de los elementos blockquote, q y cite.

Veamos algunos ejemplos.

Comenzamos con el elemento Cite el cual se emplea para referenciar un trabajo creativo como puede ser un libro una película una canción o cualquier obra intelectual.

En versiones anteriores de HTML este elemento podía ser utilizado para encerrar el nombre del autor de la obra citada, pero a partir de HTML 5.

Esto no es aconsejable y se recomienda utilizarlo solamente para citar el nombre de la obra.

Vamos a ver un ejemplo

<p>El primer libro de la Trilogía cósmica de C.S Lewis se titula más allá del planeta silencioso (1918)</p>

Lo que haremos será encerrar el nombre de los libros citados dentro de un elemento Cite .

<p>El primer libro de la <cite>Trilogía cósmica</cite> de C.S Lewis se titula <cite>Más allá del planeta silencioso (1918)</cite></p>

Vamos a guardar los cambios y visualizar este documento en el navegador, clic derecho sobre cualquier parte del fondo negro del editor de código y luego clic izquierdo sobre Open with Live Server

Hola mundo

Esta es mi primera página web

Este texto se muestra en **negrita**.

Este texto se muestra en **negrita**.

Este texto se muestra en *estilo itálica o cursiva*.

Este texto se muestra en *estilo itálica o cursiva*.

La última palabra de este texto se muestra resaltada con un fondo de color **amarillo**

Al enviar este formulario aceptas nuestra "Política de privacidad".

Para poder aprobar el proyecto de ley se necesita el ~~45% de los votos~~ **60% de los votos**

En los años transcurridos desde la invención de los primeros ordenadores digitales, el poder de la informática y la capacidad de almacenamiento han aumentado de manera exponencial. Los ordenadores personales y teléfonos inteligentes tienen la capacidad de acceder, modificar, almacenar y compartir medios digitales en las manos de miles de millones de personas.

Muchos dispositivos electrónicos, desde cámaras digitales hasta drones tienen la capacidad de crear, transmitir y visualizar los medios digitales. En combinación con la World Wide Web e Internet, los medios digitales han transformado la sociedad del siglo XXI de una manera que con frecuencia se compara con el impacto cultural, económica y social de la imprenta. El cambio ha sido tan rápido y de tal magnitud que ha puesto en marcha una transición económica de una economía industrial a una economía basada en la información, creando de un nuevo período en la historia humana conocida como la era de la información o de la revolución digital. La transición ha creado cierta incertidumbre acerca de las definiciones.

En comparación con los medios de comunicación impresos, los medios de comunicación, y otras tecnologías analógicas, los medios digitales son fáciles de copiar, almacenar, compartir y modificar. Esta calidad de los medios digitales ha dado lugar a cambios significativos en muchas industrias, especialmente el periodismo, la edición, la educación, el entretenimiento y el negocio de la música.

El primer libro de la *Trilogía cósmica* de C.S Lewis se titula *Más allá del planeta silencioso (1918)*

Y como puedes ver el navegador muestra un estilo de fuente cursiva el nombre de los títulos de los libros que hemos encerrado dentro del elemento cite lo que significa que el elemento cite aporta un valor semántico que ayuda al posicionamiento web.

También podemos formatear texto para que se muestre como una cita corta.

Esto lo hacemos mediante el elemento Q

Regresamos al editor de códigos y vamos a crear un ejemplo

<p>Tal y como decía Aristóteles: Somos lo que hacemos día a día. De modo que la excelencia no es un acto sino un hábito</p>

Lo que haremos será encerrar la frase citada dentro de un elemento.

<p>Tal y como decía Aristóteles:<q>Somos lo que hacemos día a día. De modo que la excelencia no es un acto sino un hábito</q></p>

Guardamos los cambios y regresamos al navegador, clic derecho sobre cualquier parte del fondo negro del editor de código y luego clic izquierdo sobre Open with Live Server y tal y como se puede ver el navegador ha encerrado entre comillas dobles el texto citado resaltándolo como un texto importante.

Tal y como decía Aristóteles:“Somos lo que hacemos día a día. De modo que la excelencia no es un acto sino un hábito”

Por último, veamos el elemento blockquote el cual sirve para insertar citas largas en forma de bloques de contenido.

Vamos a regresar al editor de códigos y vamos a crear un ejemplo y como el elemento blockquote es un elemento de bloque no es necesario que lo insertamos dentro de un párrafo tal y como hemos hecho con otros elementos.

<p>A tales efectos, el experto ha declarado:</p>

<blockquote>"Lorem ipsum dolor sit amet consectetur adipisicing elit. Vel sequi, esse ducimus similique labore, asperiores, delectus excepturi perspiciatis eos quas nisi accusamus optio! Aliquid dolores voluptate laborum magnam, quos debitis?"</blockquote>

Ahora sí, vamos a guardar los cambios y vamos a regresar al navegador, clic derecho sobre cualquier parte del fondo negro del editor de código y luego clic izquierdo sobre Open with Live Server.

A tales efectos, el experto ha declarado:

"Lorem ipsum dolor sit amet consectetur adipisicing elit. Vel sequi, esse ducimus similique labore, asperiores, delectus excepturi perspiciatis eos quas nisi accusamus optio! Aliquid dolores voluptate laborum magnam, quos debitis?"

Como puedes ver el navegador por defecto ha incluido un margen izquierdo al texto que hemos encerrado dentro de un elemento blockquote.

Formatear textos en HTML(IV)

Otra manera de formatear textos mediante HTML consiste en mostrarlos como un subíndice y superíndice.

Generalmente este tipo de texto se utiliza para terminologías científicas y matemáticas.

Veamos algunos ejemplos.

Comenzamos definiendo un subíndice con HTML.

Para esto es necesario emplear el elemento **sub**

```
<p>La fórmula química del agua es H<sub>2</sub>O.</p>
```

Vamos a guardar los cambios y regresar al navegador, clic derecho sobre cualquier parte del fondo negro del editor de código y luego clic izquierdo sobre Open with Live Server y cómo ves el navegador lo muestra como un subíndice.

La fórmula química del agua es H₂O.

Ahora vamos a crear un ejemplo de un superíndice.

```
<p>La expresión de 8 elevado a la 2 sería la siguiente: <p>La expresión de 8 elevado a la 2 es: 8<sup>2</sup></p>
```

Guardamos los cambios y visualizamos en el navegador, clic derecho sobre cualquier parte del fondo negro del editor de código y luego clic izquierdo sobre Open with Live Server y el navegador muestra el número 2 expresado como un super índice.

Por último, veamos el elemento Span con el cual podemos aplicar estilos a un fragmento específico de un texto, por sí solo este elemento no aporta ningún cambio visual en un texto ni añade valor semántico alguno.

Por esta razón siempre debe emplearse en conjunto con algún atributo.

Veamos un ejemplo

```
<p>La última palabra de este texto se muestra en color <span style="color:red">rojo</span></p>
```

Guardamos los cambios y visualizamos en el navegador, clic derecho sobre cualquier parte del fondo negro del editor de código y luego clic izquierdo sobre Open with Live Server, y como puedes notar el navegador muestra la última palabra de este texto en color rojo.

La última palabra de este texto se muestra en color rojo

Comentarios

Ahora veremos cómo agregar comentarios en el código de un documento HTML. Básicamente, un comentario en HTML es una nota o aviso que sirve para explicar cuál es la función del código que hemos escrito.

De esta forma, los diseñadores y desarrolladores podemos tener una mejor comprensión del código fuente de un sitio web. Esto permite que el sitio esté mucho más organizado, especialmente cuando se trabaja en equipo con otros desarrolladores.

Vamos a crear algunos comentarios para el código HTML escrito en el archivo `formato-texto.html`.

Para insertar un comentario, tan solo necesitamos encerrar el texto dentro de los signos correspondientes, de la siguiente forma:

Ejemplos:

```
<!-- Ejemplo de un párrafo en HTML -->
```

```
<!-- Cómo formatear texto en negrita en HTML -->
```

```
<!-- Ejemplo de texto en estilo de fuente cursiva en HTML -->
```

También podemos agregar un comentario de forma más rápida simplemente presionamos las teclas **Ctrl + K** y después **Ctrl + C** por ejemplo, si queremos añadir un nuevo comentario, presionamos estas teclas, y automáticamente el editor insertará las etiquetas correspondientes. Solo queda escribir el contenido del comentario, como:

```
<!-- Otros elementos para formatear texto en HTML -->
```

Como puedes ver, los comentarios nos ayudan a tener un código más organizado. Sin embargo, es importante saber que los comentarios no se muestran en el navegador.

Por ejemplo, si guardamos los cambios y visualizamos el documento, veremos que ningún comentario aparece en pantalla.

No obstante, cualquier usuario puede acceder al código fuente, incluyendo los comentarios, simplemente haciendo clic derecho y seleccionando la opción "Ver código fuente de la página". Esto abre una nueva ventana donde es posible leer los comentarios que hemos escrito, así como todo el contenido del documento HTML.

Por esta razón, nunca debemos incluir información confidencial en los comentarios.

Otra utilidad de los comentarios

Los comentarios también sirven para anular temporalmente un fragmento de código que no queremos eliminar por completo. Esto puede ser útil si estamos haciendo pruebas o queremos observar cómo se ve la página sin ciertos elementos.

Para ello, simplemente encerramos el bloque de código en un comentario. Por ejemplo, en el editor, seleccionamos el código deseado, presionamos Ctrl + K y después Ctrl + C para comentarlo. De esta manera, el navegador lo ignorará, pero seguirá presente en el archivo.

Guardamos los cambios, visualizamos el navegador. El contenido comentado ya no se mostrará, pero permanece en el documento.

Si más adelante deseamos que vuelva a aparecer, simplemente eliminamos las etiquetas de comentario, guardamos, y visualizamos la página nuevamente.

Encabezados HTML

Seguidamente vamos a trabajar con los encabezados en HTML. Para comenzar, creamos un nuevo documento y lo guardamos con el nombre encabezados.html. Luego, procedemos a construir la estructura básica del documento HTML. Modificamos el valor del atributo lang y escribimos "es". También cambiamos el contenido del elemento <title> por "Encabezados en HTML".

HTML cuenta con seis niveles diferentes de encabezados, definidos mediante los elementos <h1>, <h2>, <h3>, <h4>, <h5> y <h6>, conocidos como elementos de encabezado. Para crear un encabezado, simplemente encerramos el texto deseado entre las etiquetas correspondientes.

Por ejemplo, dentro del elemento <body>, agregamos un encabezado de nivel 1:

```
<h1>Esto es un encabezado de nivel 1</h1>
```

Luego, podemos agregar un encabezado de nivel 2:

```
<h2>Esto es un encabezado de nivel 2</h2>
```

A continuación, añadimos un encabezado de nivel 3:

```
<h3>Esto es un encabezado de nivel 3</h3>
```

Después, insertamos un encabezado de nivel 4:

```
<h4>Esto es un encabezado de nivel 4</h4>
```

Continuamos con un encabezado de nivel 5:

```
<h5>Esto es un encabezado de nivel 5</h5>
```

Y finalizamos con un encabezado de nivel 6:

`<h6>Esto es un encabezado de nivel 6</h6>`

Guardamos los cambios y visualizamos la página en el navegador. Verás que cada encabezado tiene un tamaño diferente y está en negrita, lo que permite destacar su importancia dentro del contenido.

Sin embargo, el uso de los encabezados no es solo estético. Cada uno posee un valor semántico: al usarlos, le estamos indicando al navegador y a los motores de búsqueda (como Google o Bing) que ese texto tiene un nivel jerárquico o de relevancia superior.

Esto contribuye a mejorar el posicionamiento del sitio en los resultados de búsqueda.

Importante: No debemos encerrar todo el contenido del documento dentro de encabezados con la intención de mejorar el SEO, ya que un uso excesivo puede ser penalizado por los motores de búsqueda. Solo deben emplearse para delimitar el inicio de cada sección o bloque de contenido.

Buenas prácticas con los encabezados

1. Sigue una jerarquía: No utilices un `<h3>` si antes no has colocado un `<h2>`, ni un `<h2>` si no hay un `<h1>`.
2. Un solo `<h1>` por página: Este debe ser el encabezado principal del documento, como el título del artículo o el nombre del sitio web.
3. Encabezados secundarios con `<h2>`: Úsalos para secciones importantes dentro del documento.
4. Subsecciones con `<h3>` a `<h6>`: Sirven para organizar el contenido que se deriva de los encabezados principales y secundarios.

Ejemplo estructurado

Vamos anular todo este código encerrándolo de una etiqueta para comentarios de la siguiente manera:

`<!-- <h1>Esto es un encabezado de nivel 1</h1>`

`<h2>Esto es un encabezado de nivel 2</h2>`

`<h3>Esto es un encabezado de nivel 3</h3>`

`<h4>Esto es un encabezado de nivel 4</h4>`

`<h5>Esto es un encabezado de nivel 5</h5>`

`<h6>Esto es un encabezado de nivel 6</h6> -->`

Supongamos que estamos redactando un artículo sobre "Animales vertebrados".

El código podría estructurarse así:

```
<h1>Animales vertebrados</h1>
<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit.</p>
<h2>Características</h2>
<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit.</p>
<h2>Anatomía</h2>
<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit.</p>
<h2>Clasificación</h2>
<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit.</p>
<h3>Mamíferos</h3>
<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit.</p>
<h3>Aves</h3>
<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit.</p>
<h3>Peces</h3>
<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit.</p>
<h3>Anfibios y reptiles</h3>
<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit.</p>
```

Guardamos los cambios y visualizamos el documento en el navegador. Notaremos que:

- `<h1>` define el título principal del documento.
- Cada `<h2>` marca una sección importante.
- Cada `<h3>` indica un subtema dentro de la sección correspondiente.

Utilizar correctamente los encabezados en HTML no solo mejora la estructura visual del sitio, sino también su accesibilidad y posicionamiento en buscadores. Por eso, es fundamental respetar la jerarquía y evitar el abuso de estos elementos.

Utilizar varios elementos H1 dentro de un mismo documento HTML

Para finalizar con el tema de los encabezados en HTML, vamos a ver cómo es posible añadir varios elementos `<h1>` dentro de una misma página web.

Es importante aclarar que, en versiones anteriores de HTML, esto no era permitido. El uso del elemento `<h1>` se limitaba a uno por documento, bajo la idea de que solo debía existir un encabezado principal en toda la página.

Sin embargo, con la llegada de la especificación HTML5, esta restricción ha quedado obsoleta. Ahora es totalmente válido utilizar múltiples elementos `<h1>`, siempre que se encuentren contenidos dentro de elementos semánticos estructurales, como `<section>` o `<article>`. Esto permite dividir una página en varias secciones, cada una con su propio encabezado principal.

Ejemplo práctico

Volvamos al editor de código. Justo debajo del contenido actual, agregamos un nuevo elemento semántico:

```
<section>
```

```
</section>
```

El elemento `<section>` se utiliza para agrupar contenido temáticamente relacionado. Lo estudiaremos con más detalle más adelante, pero por ahora lo usaremos para estructurar nuestro ejemplo.

Seleccionamos un bloque de contenido, por ejemplo, desde el encabezado `<h1>` hasta `<h3>`, y lo recortamos con `Ctrl + X`. Luego, lo pegamos entre las etiquetas de apertura y cierre del nuevo elemento `<section>`.

De esta manera, ese encabezado `<h1>` deja de ser el encabezado general del documento y pasa a ser el encabezado único de esa sección específica.

A continuación, agregamos otra sección debajo:

```
<section>
```

```
</section>
```

Copiamos un bloque diferente, que incluya un `<h1>`, un `<h2>` y un `<p>`, y lo pegamos dentro de esta segunda sección. Luego, modificamos el texto del segundo `<h1>` para que diga "Animales invertebrados". Con esto, ahora tenemos dos elementos `<h1>` en el documento, pero cada uno encabezando su respectiva sección temática.

Subdivisiones dentro de secciones

Dentro de la segunda sección, podemos agregar encabezados secundarios con `<h3>`. Por ejemplo:

```
<h3>Artrópodos</h3>
```

```
<p>Lorem ipsum dolor, sit amet consectetur adipisicing elit.</p>
```

```
<h3>Poríferos</h3>
```

```
<p>Lorem ipsum dolor sit amet consectetur adipisicing elit.</p>
```

```
<h3>Anélidos y equinodermos</h3>
```

```
<p>Lorem ipsum dolor sit amet consectetur, adipisicing elit.</p>
```

Esto nos permite mantener una estructura jerárquica dentro de cada sección.

Guardamos los cambios y visualizamos el documento en el navegador.

Ahora, si agregamos un nuevo `<h1>` fuera de cualquier sección semántica, por ejemplo:

```
<h1>Reino animal</h1>
```

Guardamos los cambios y visualizamos el documento en el navegador.

El navegador interpretará este como el encabezado principal de toda la página, y lo mostrará con un tamaño de fuente más grande que los demás encabezados `<h1>` contenidos dentro de secciones.

Gracias a la especificación HTML5, podemos usar múltiples elementos `<h1>` en una sola página de forma válida y semántica, siempre que:

- Cada uno esté contenido dentro de un elemento estructural como `<section>` o `<article>`.
- No se utilicen varios `<h1>` dentro de una misma sección.

Esta práctica no solo organiza mejor nuestro contenido, sino que también mejora la accesibilidad y el posicionamiento SEO del sitio web.



Estructura de una URL

La URL (*Uniform Resource Locator / Localizador de Recursos Uniforme*) tiene la función de nombrar y localizar los recursos de una página web de manera inequívoca. Estos recursos pueden ser otra página web, un archivo PDF, una imagen o cualquier otro tipo de documento.

Generalmente, una URL cuenta con el siguiente formato:

protocolo:// **nombre del servidor** **/ruta/** **recurso** **?page=2** **#url**

Vamos a ver para qué sirve cada componente.

1. Protocolo: también llamado esquema URL, es el mecanismo que utiliza el navegador para acceder al recurso.

Los protocolos más utilizados son:

- **http://** (Hypertext Transfer Protocol o Protocolo de transferencia de hipertexto). Es el protocolo de uso más común para páginas y sitios web.
- **https://** (Hypertext Transfer Protocol Secure / Protocolo seguro de transferencia de hipertexto). Utilizado para cifrar información sensible.



- **ftp://** (File Transfer Protocol / Protocolo de transferencia de ficheros). Orientado a la transferencia de ficheros o archivos.

Al escribir una URL para enlazar un documento HTML con una página externa debes asegurarte de siempre agregar el protocolo correspondiente o de lo contrario el vínculo o enlace podría no funcionar correctamente.

2. Nombre del servidor: se refiere al nombre de la computadora o máquina destino donde se aloja el contenido. Este se compone de dos partes:

- El **nombre** que se le ha asignado al equipo o máquina destino, el cual suele ser representativo del sitio web, o bien puede ser una dirección IP. Algunos ejemplos pueden ser: Google, Yahoo, Youtube, 98.139.183.24, etc.
- El **dominio** de nivel superior o TLD (Top-level domain), que consiste en las últimas letras del nombre del servidor, que se escriben precedidas por un punto, tales como .com, .org, .net, aunque también existen algunas variantes como .tv, .es, .edu, etc.

De forma opcional, algunas URL's pueden contener un subdominio, mismo que se utiliza para que la máquina destino pueda atender a varios nombres. Normalmente,



el subdominio se escribe delante del nombre del servidor. El subdominio más famoso es el *www*.

3. Ruta de directorios: se refiere a la secuencia de carpetas o directorios que definen la ruta a seguir para llegar al recurso que se quiere enlazar.

4. Recurso: El nombre del archivo que contiene al recurso enlazable, generalmente se trata de un documento HTML, aunque también puede ser otro tipo de documento, como un archivo PDF, un video, etc.

5. Query strings: Información adicional para acceder al recurso. Se suele utilizar en lenguajes del lado del servidor como JavaScript. Comienza con el signo de cierre de interrogación (?) seguido de una sucesión de palabras y/o números separados por el signo de igual (=). Por ejemplo:

<https://www.google.com/search?q=restaurant>

6. Ancla: un fragmento de texto precedido por el signo de numeral o almohadilla (#). Se utiliza para acceder directamente a secciones concretas dentro de una misma página. Ejemplo:

<http://misitio.com/pagina.html#quienes-somos>



HTML & CSS

desde cero hasta avanzado

Debes tener en cuenta que no es necesario que una URL tenga todos estos componentes para ser válida. De hecho, muchas URL's suelen tener sólo algunas de estas partes presentes y aún así siguen siendo válidas. Por ejemplo:

- 69.172.201.153 (url sin protocolo, sin nombre de servidor, sin ruta de directorios y sin recurso)
- https://google.com (url con protocolo y nombre del servidor)
- index.html (url con nombre del recurso)
- images/nombre-imagen.jpg (url con ruta de directorio y recurso)

Generalmente, las partes que componen una URL se separan mediante caracteres reservados como el signo de igual (=), signo de ampersand (&), el signo de la barra inclinada (/), entre otros.

En algunos casos, el servidor puede llegar a sustituir estos caracteres por una codificación equivalente denominada *caracteres seguros*. Así, el signo de la barra inclinada lo puede sustituir por el código %2F, o puede cambiar el signo de igual por el código %3D, etc.

Enlaces o hipervínculos

A continuación, veremos cómo agregar enlaces en un documento HTML.

Los enlaces, también llamados vínculos o hipervínculos, son uno de los elementos más importantes de HTML, ya que permiten conectar las diferentes páginas de un sitio web. Además, pueden utilizarse para enlazar otros recursos, como imágenes, archivos PDF, audio, video o incluso páginas externas. También permiten referenciar una parte específica del contenido dentro de una misma página.

Creación de enlaces básicos

Vamos a regresar al editor de código para crear algunos ejemplos prácticos.

Lo primero que haremos será crear un nuevo documento. Para ello, presionamos las teclas Ctrl + N y luego lo guardamos como enlaces.html usando Ctrl + S. Una vez guardado, procedemos a crear la estructura básica del documento HTML. Escribimos html:5 y presionamos TAB para generar la plantilla predeterminada.

Modificamos el valor del atributo lang por "es" y cambiamos el contenido del elemento <title> por:

"Enlaces o hipervínculos en HTML"

Luego, dentro del elemento <body>, agregamos un encabezado de nivel 1:

```
<h1>Ejemplo de enlaces en HTML</h1>
```

Para escribir un enlace en HTML se utiliza el elemento <a>. Su sintaxis básica es:

```
<a href="URL">Texto del enlace</a>
```

Dentro del atributo href, colocamos la dirección a la que debe llevar el enlace.

Enlaces con URLs absolutas

Una URL absoluta incluye todas las partes de la dirección: el protocolo (http o https), el nombre del servidor y, en algunos casos, la ruta y el nombre del recurso. Estas URLs son ideales para enlazar a páginas o archivos alojados en servidores externos.

Por ejemplo:

```
<a href="https://www.google.com">Ir a Google</a>
```

Otro ejemplo más completo, usando una URL de Wikipedia, sería:

```
<a href="https://es.wikipedia.org/wiki/Medios_digitales">Ir a Wikipedia</a>
```

En este caso, la URL contiene el protocolo, el nombre del servidor, la ruta y el nombre del recurso. Guardamos los cambios y visualizamos el documento en el navegador.

Enlaces con URLs relativas

Una URL relativa no incluye el protocolo ni el nombre del servidor. Se utiliza para enlazar recursos que están almacenados en la misma carpeta o subcarpeta del proyecto.

Por ejemplo:

```
<a href="encabezados.html">Ver encabezados</a>
```

En este caso, el archivo encabezados.html debe estar en la misma carpeta que el documento actual. El navegador buscará ese archivo dentro del servidor o, en este caso, en el disco duro local.

Guardamos los cambios con Ctrl + S y visualizamos el documento en el navegador. Al hacer clic sobre el enlace, nos redirige a un archivo local dentro de nuestro proyecto.

Enlaces relativos con subcarpetas

Ahora vamos a crear un ejemplo de URL relativa con ruta.

1. En la carpeta donde estamos guardando los documentos HTML, creamos una subcarpeta llamada recursos.
2. Movemos dentro de esa carpeta un archivo llamado prueba.html.
3. En nuestro documento principal, agregamos un enlace:

```
<a href="recursos/prueba.html">Ir al enlace</a>
```

Este tipo de url relativa se denomina relativa ascendente en este caso, estamos indicando al navegador que debe buscar el archivo prueba.html dentro de la subcarpeta recursos.

Guardamos los cambios, visualizamos el documento en el navegador y al hacer clic sobre este enlace, el navegador nos dirigirá correctamente al documento ubicado en la ruta especificada.

Atributos Target y Download

Además del atributo href, el elemento <a> admite otros atributos útiles. Uno de ellos es el atributo target, el cual especifica dónde debe abrirse la página o recurso enlazado.

Uso del atributo target

Vamos a crear un ejemplo para entender mejor este atributo. Supongamos que queremos que un enlace abra el recurso en una ventana específica. Para esto, escribimos lo siguiente dentro del elemento <a>:

```
<a href="https://www.begomez.com" target="_self">Ir al enlace</a>
```

El valor por defecto de target es `_self`, lo que indica al navegador que el recurso se debe abrir en la misma ventana o pestaña donde se encuentra el enlace. Dado que es el valor por defecto, no es necesario escribirlo explícitamente.

Guardamos los cambios, visualizamos la página en el navegador y, al hacer clic sobre el enlace, veremos que el recurso se abre automáticamente en la misma ventana.

Si deseamos cambiar este comportamiento y hacer que el recurso se abra en una nueva ventana o pestaña, utilizamos el valor `_blank`:

```
<a href="https://www.begomez.com" target="_blank">Ir al enlace</a>
```

Guardamos los cambios, visualizamos la página en el navegador y, al hacer clic sobre el enlace, veremos que el recurso se abre automáticamente en una nueva ventana.

El atributo target también admite otros valores como `_parent` y `_top`, aunque en la práctica su uso es menos común, por lo que no los abordaremos.

Uso del atributo download

Ahora veamos otro atributo muy útil: `download`. Este atributo le indica al navegador que el recurso enlazado debe descargarse automáticamente, en lugar de abrirse en el navegador.

Este atributo fue introducido con HTML5 y es especialmente útil cuando deseamos que los usuarios descarguen documentos como manuales en PDF, archivos de música, videos o cualquier otro archivo ejecutable.

Ejemplo práctico

Supongamos que ya tenemos un archivo en formato PDF guardado en la carpeta recursos.

Ahora escribimos lo siguiente en nuestro documento HTML:

```
<p>Aquí puedes descargar el documento</p>
```

```
<a href="reglamento_de_evaluacion_curso_lectivo_2026.pdf"
```

```
download>Descargar archivo</a>
```

Es importante asegurarse de que la ruta del archivo sea correcta y que incluya la extensión del archivo (en este caso, `.pdf`).

El atributo `download` es un atributo booleano, lo que significa que su sola presencia en el elemento indica un valor "verdadero". En HTML5, los atributos booleanos no requieren valor asignado, por lo tanto, simplemente se escribe:

```
download
```

Guardamos los cambios, visualizamos la página en el navegador y, al hacer clic sobre el enlace, el navegador mostrará un cuadro de diálogo para descargar el archivo. Al hacer clic en Aceptar, el recurso será descargado automáticamente al computador del usuario.



Uso del signo de numeral o almohadilla (#) en los enlaces

El signo de numeral o almohadilla (#) se puede utilizar para varios fines en una URL de destino.

El uso más común es para hacer referencia a puntos concretos dentro de un documento HTML, y tal y como vimos en el video anterior, esto se hace agregando este signo como valor de la URL, seguido del *identificador del punto de destino o anclaje*, que no es más que el valor del atributo *id* del elemento al que se desea apuntar.

Pero también, el signo de numeral puede ser empleado para crear un *enlace de marcador de posición*, que se parece a un enlace tradicional, con la diferencia de que no dirige a ninguna ubicación en concreto, o bien, dependiendo de la posición en la que se encuentre el enlace en el documento HTML, el navegador nos llevará hasta la parte superior de la página web.

Los enlaces de marcador de posición se crean omitiendo el atributo *href* en el elemento `<a>`, o simplemente al añadir el signo de numeral como valor de este atributo. Por ejemplo:



HTML & CSS

desde cero hasta avanzado

```
<h1>Esto es un encabezado</h1>
```

```
<p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Sit deserunt  
ratione error, accusamus quaerat velit eveniet officiis enim accusantium.  
Perspiciatis numquam sint quibusdam, totam odio magni officia recusandae  
sequi exercitationem.</p>
```

```
<p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Cumque  
reiciendis dolor debitis, porro enim eos, cupiditate quos temporibus a quo  
laudantium. Facere omnis nulla numquam impedit voluptas excepturi ut  
illum.</p>
```

```
<p>Lorem ipsum dolor numquam sint quibusdam, totam odio magni officia  
recusandae sequi porro enim eos, cupiditate quos temporibus a quo.</p>
```

```
<a href="#">Ir al inicio</a>
```

En el código anterior, simplemente se ha añadido el signo de numeral o almohadilla como valor del atributo *href* del elemento `<a>`, con lo cual se ha creado un marcador de posición. Así que cuando pulsemos sobre este enlace, el navegador se desplazará automáticamente a la parte superior de la página web, debido a que el enlace se ha colocado al final del contenido..

A lo largo de este curso haremos uso de este tipo enlace para crear ejemplos de vínculos que no estarán conectados con un recurso o contenido en particular.

Enlaces tipo ancla (Atributo ID)

Ahora veremos cómo agregar enlaces tipo ancla en un documento HTML. Este tipo de enlaces permite vincular diferentes secciones dentro de una misma página, lo cual es especialmente útil cuando el contenido de la página es extenso y está dividido en múltiples apartados.

Un buen ejemplo lo podemos ver en artículos largos como los de Wikipedia, donde al inicio del contenido suele haber un índice formado por enlaces ancla. Al hacer clic en uno de estos enlaces, el navegador nos lleva directamente al apartado correspondiente.

¿Cómo se crean los enlaces tipo ancla?

Estos enlaces se construyen utilizando el elemento `<a>`, junto con el atributo `id`, que debe colocarse dentro del elemento que queremos enlazar. El atributo `id` tiene como función identificar de forma única a cada elemento del documento. Es importante que el valor asignado al `id`:

- No se repita en todo el documento.
- No comience con un número, un espacio ni con caracteres especiales.

Ejemplo paso a paso

Crear contenido extenso

Regresamos al editor de código y, como primer paso, comentamos el contenido anterior para que no se muestre en el navegador. Luego, comenzamos a crear contenido de ejemplo. Insertamos un elemento `<h2>` con el texto "Capítulo 1", seguido de un párrafo `<p>` con texto de ejemplo (puede ser Lorem Ipsum).

Duplicar el contenido

Copiamos ese bloque (encabezado + párrafo) 15 veces, cambiando el número de capítulo en cada encabezado: "Capítulo 1", "Capítulo 2", ..., hasta "Capítulo 15".

Esto simula una página larga con varias divisiones.

Agregar identificadores id

Luego, asignamos atributos `id` a los encabezados que deseamos enlazar. Por ejemplo:

```
<h2 id="cap-3">Capítulo 3</h2>
```

```
<h2 id="cap-7">Capítulo 7</h2>
```

```
<h2 id="cap-12">Capítulo 12</h2>
```

Crear los enlaces ancla

En la parte superior del documento (o donde desees colocar el índice), escribimos los enlaces que apuntan a cada sección:

```
<a href="#cap-3">Capítulo 3</a><br>
```

```
<a href="#cap-7">Capítulo 7</a><br>
```

```
<a href="#cap-12">Capítulo 12</a><br>
```

Observa que en el atributo href, el valor corresponde al id asignado a cada sección, precedido por el símbolo # (almohadilla).

También agregamos un salto de línea (
) después de cada enlace para evitar que se muestren todos en la misma línea, ya que el elemento <a> es de tipo en línea.

Guardamos los cambios y visualizamos la página en el navegador. Al hacer clic en cualquiera de los enlaces, el navegador nos lleva automáticamente al encabezado correspondiente:

- Clic en "Capítulo 3" → lleva a la sección con id="cap-3"
- Clic en "Capítulo 7" → lleva a la sección con id="cap-7"
- Clic en "Capítulo 12" → lleva a la sección con id="cap-12"

Los enlaces ancla permiten mejorar la navegación dentro de una misma página, facilitando el acceso rápido a secciones específicas del contenido. Son muy útiles en documentos largos, tutoriales, términos y condiciones, enciclopedias, entre otros.

Insertar imágenes en un documento HTML

Seguidamente aprenderemos cómo insertar imágenes en un documento o página HTML. Primero, creamos un nuevo documento y lo guardamos con el nombre imagenes.html. Una vez guardado, procedemos a crear la estructura básica de HTML utilizando la plantilla html:5. Modificamos el valor del atributo lang a "es" y el título del documento a "Imágenes en HTML"

Dentro de la carpeta principal del proyecto (llamada html, por ejemplo), creamos una subcarpeta llamada images, donde almacenaremos todas las imágenes utilizadas.

Descargamos dos imágenes y las renombramos como:

- leopardo.jpg
- carretera.jpg

Las colocamos dentro de la carpeta images.

Inserción de imágenes en HTML

Regresamos al editor de código. Dentro del elemento `<body>`, escribimos el siguiente código:

```

```

El atributo `src` define la ruta de la imagen.

El atributo `alt` proporciona una descripción alternativa, útil si la imagen no puede cargarse.

Guardamos los cambios y visualizamos la página en el navegador.

Ajuste de dimensiones con atributos HTML

Podemos modificar el tamaño de la imagen usando los atributos `width` y `height`:

```

```

Guardamos los cambios y visualizamos la página en el navegador

Si omitimos uno de los dos atributos, el navegador ajustará automáticamente la proporción de la imagen.

Ajuste de dimensiones con CSS inline

Otra opción es usar el atributo `style`, que permite aplicar estilos CSS directamente:

```

```

En este caso, es obligatorio indicar la unidad de medida (como `px`, `%`, etc.).

Guardamos los cambios y visualizamos la página en el navegador

También podemos usar porcentajes para que la imagen se adapte al contenedor:

```

```

Esto hará que la imagen ocupe el 50% del ancho de la ventana del navegador.

Hacer que una imagen funcione como enlace

Podemos insertar una imagen dentro de un enlace para que, al hacer clic sobre ella, se abra una página web:

<!-- Ejemplo de imagen como enlace -->

```
<a href="https://www.google.com">  
    
</a>
```

En este ejemplo, al hacer clic sobre la imagen, el navegador nos llevará a la página principal de Google.

Listas

Las listas en HTML permiten agrupar elementos relacionados para facilitar su lectura y organización. En esta lección aprenderemos a crear y utilizar listas ordenadas y listas desordenadas.

Primero, creamos un nuevo documento y lo guardamos con el nombre listas.html. Luego, generamos la estructura básica con la plantilla html:5. Modificamos el valor del atributo lang a "es" y cambiamos el contenido del elemento <title> por: "Listas en HTML".

Dentro del elemento <body>, agregamos un encabezado <h1> con el texto: "Ejemplo de listas en HTML".

Listas desordenadas ()

Las listas desordenadas se utilizan cuando los elementos tienen igual importancia y su orden no es relevante. Los elementos aparecen con viñetas en lugar de números.

Ejemplo: Lista de ingredientes

Vamos a simular una lista con los ingredientes de una receta de postre, por ejemplo, Tiramisú de arándanos.

```
<h2>Tiramisú de arándanos</h2>  
<p>Ingredientes:</p>  
<ul>
```

- 250 g de queso mascarpone
- 200 ml de nata para montar
- 100 g de azúcar
- Bizcochos de soletilla
- Café frío
- Licor de amaretto (opcional)
- Arándanos frescos
- Cacao en polvo
- Chocolate rallado

Al visualizar el documento en el navegador, verás que cada ingrediente aparece con una viñeta negra a la izquierda. Más adelante, aprenderemos cómo personalizar o eliminar estas viñetas usando CSS.

Listas ordenadas ()

Las listas ordenadas se utilizan cuando los elementos deben seguir una secuencia lógica, como pasos en un procedimiento. En lugar de viñetas, los elementos se numeran automáticamente.

Ejemplo: Lista de pasos para preparar la receta

<h2>Preparación:</h2>

- Batir la nata con el azúcar hasta que esté firme.
- Mezclar la nata con el queso mascarpone.
- Remojar los bizcochos en café con licor.
- Montar capas alternas de crema, bizcochos y arándanos.
- Espolvorear cacao y chocolate rallado antes de servir.

Cuando se visualiza en el navegador, cada elemento de la lista aparece precedido por un número, indicando el orden en que deben realizarse los pasos.



Listas de descripción

Otro tipo de lista que permite HTML son las denominadas *Listas de descripción o definición*, cuya función es la de asociar un nombre o término con un valor especificado.

Aunque este tipo de lista se utiliza muy poco en el diseño y desarrollo web actual, es importante conocer como podemos emplearlas en un documento HTML, debido a que son elementos semánticos que indican a los navegadores cual es la naturaleza de los elementos de una lista que quedan asociados a su definición.

Básicamente, para crear una lista de descripción debemos emplear tres elementos:

Elemento	Definición
<code><dl></code>	Es el elemento contenedor y se emplea para definir listas de definición
<code><dt></code>	Se utiliza para definir los términos de los elementos de la lista.
<code><dd></code>	Indica la definición o o valor asociado al término de los elementos de la lista.



Veamos un ejemplo práctico del uso de estos elementos:

```
<h1>Glosario de términos de cine</h1>

<dl>
  <dt>Angulación:</dt>
  <dd>Inclinación del eje de la cámara respecto al sujeto
    que ha de ser captado.</dd>

  <dt>Atmósfera</dt>
  <dd> El espacio de influencia de una película, es decir, el
    ambiente que se pretende crear en determinadas escenas: los decorados,
    la música, los movimientos de cámara, el ritmo, la puesta
    en escena, los sonidos ambientales, etc.</dd>

  <dt>Banda sonora</dt>
  <dd> Figuradamente, se refiere al conjunto de sonidos del
    film, esto es, la voz, la música, efectos sonoros y silencio.
    También se suele llamar así a la música de la película para
    su venta en disco.</dd>

  <dt>Blockbuster</dt>
  <dd>Se refiere a un tipo de película que tiene mucho éxito
    en taquilla, sin embargo su calidad es cuestionable. </dd>
</dl>
```

En el código anterior, hemos estructurado un glosario de términos cinematográficos. Como puedes ver, cada término se ha encerrado dentro de un elemento `<dt>`, mientras que las definiciones van contenidas dentro de un elemento `<dd>`. Todo este contenido está englobado dentro de un elemento `<dl>`.



HTML & CSS

desde cero hasta avanzado

Si visualizamos en el navegador, por defecto cada definición contenida dentro de un elemento `<dd>` se mostrará desplazada hacia la izquierda.

Listas anidadas

Además de las listas simples, en HTML también es posible crear estructuras más complejas mediante listas anidadas, es decir, una lista dentro de otra. Este tipo de estructura nos permite representar datos jerárquicos y organizar la información de forma más clara y ordenada.

¿Qué es una lista anidada?

Una lista anidada es una lista dentro de otra. Podemos, por ejemplo, tener una lista ordenada de temas principales, y para cada uno de ellos, incluir una lista desordenada con subtemas relacionados. Este recurso se utiliza frecuentemente para crear menús, esquemas, directorios, entre otros.

Ejemplo práctico: Planetas del Sistema Solar

Vamos a crear una lista ordenada con los planetas del Sistema Solar. Para cada planeta, agregaremos una lista desordenada con dos datos: su diámetro y su distancia con respecto al Sol.

Primero, comentamos el código anterior para que no se muestre en el navegador. Luego, insertamos un encabezado:

```
<h2>Planetas del Sistema Solar</h2>
```

Debajo, iniciamos una lista ordenada:

```
<ol>
  <li>Mercurio
    <ul>
      <li>Diámetro: 4,878 km</li>
      <li>Distancia al Sol: 58 millones de km</li>
    </ul>
  </li>
  <li>Venus
    <ul>
      <li>Diámetro: 12,100 km</li>
      <li>Distancia al Sol: 108 millones de km</li>
    </ul>
  </li>
  <li>Tierra
    <ul>
      <li>Diámetro: 12,756 km</li>
```

```
<li>Distancia al Sol: 150 millones de km</li>
</ul>
</li>
<li>Marte
<ul>
<li>Diámetro: 6,786 km</li>
<li>Distancia al Sol: 228 millones de km</li>
</ul>
</li>
</ol>
```

En este ejemplo, cada planeta se define como un ítem de la lista principal (ordenada), y dentro de cada uno, agregamos una lista secundaria (desordenada) con dos propiedades.

Guardamos los cambios y visualizamos la página en el navegador

Tablas

Consecutivamente aprenderemos a trabajar con tablas en HTML, una herramienta fundamental para presentar y organizar información de forma estructurada, utilizando filas y columnas. Las tablas pueden contener texto, números, imágenes, enlaces y otros elementos complejos.

En versiones anteriores de HTML, las tablas eran comúnmente utilizadas para estructurar la totalidad de un sitio web. Sin embargo, esta práctica ha quedado obsoleta, ya que las tablas deben usarse únicamente para mostrar datos tabulados. La maquetación de páginas se realiza hoy en día mediante CSS.

Crear el documento base

Comenzamos creando un nuevo archivo llamado tablas.html, y generamos la estructura básica de HTML. Luego, modificamos el valor del atributo lang por "es" y el contenido del elemento <title> por:

```
<title>Tablas en HTML</title>
```

Dentro del cuerpo del documento (<body>), insertamos un encabezado principal:

```
<h1>Ejemplo de tablas en HTML</h1>
```

Estructura básica de una tabla

Para crear una tabla, utilizamos el elemento `<table>`, que funciona como contenedor principal. Dentro de este, definimos las filas mediante el elemento `<tr>` (table row) y las celdas de cada fila mediante los elementos `<th>` (table header) y `<td>` (table data).

A continuación, un ejemplo de una tabla con 4 columnas y 4 filas:

```
<table>
  <tr>
    <th>Nombre</th>
    <th>Apellido</th>
    <th>Teléfono</th>
    <th>Email</th>
  </tr>
  <tr>
    <td>Sofía</td>
    <td>G.</td>
    <td>8888-8888</td>
    <td>sofia@email.com</td>
  </tr>
  <tr>
    <td>Mario</td>
    <td>L.</td>
    <td>8777-1234</td>
    <td>mario@email.com</td>
  </tr>
  <tr>
    <td>Lucía</td>
    <td>R.</td>
    <td>8999-4567</td>
    <td>lucia@email.com</td>
  </tr>
</table>
```

Explicación de los elementos

`<table>`: Encierra toda la tabla.

`<tr>`: Define una fila dentro de la tabla.

`<th>`: Define una celda de encabezado (se muestra en negrita y centrada por defecto).

`<td>`: Define una celda de datos.

La primera fila contiene los encabezados de las columnas. Las siguientes filas representan los datos correspondientes a cada persona. Guardamos los cambios y visualizamos el documento en el navegador.

Tablas avanzadas

Además de las tablas básicas, HTML nos permite crear tablas complejas que organizan los datos en secciones temáticas. Para lograrlo, utilizamos elementos adicionales como `<caption>`, `<thead>`, `<tbody>` y `<tfoot>`, que mejoran la estructura y legibilidad de la información.

Ejemplo práctico: Reporte de ventas

Supongamos que queremos replicar en HTML una tabla como la que podemos construir en Excel, con filas fusionadas, secciones diferenciadas y encabezados claros. A continuación, aprenderemos paso a paso cómo hacerlo.

Estructura base y título de la tabla

Primero, comentamos el código anterior para evitar que se muestre en el navegador. Luego, agregamos el elemento `<table>`, y justo después de su apertura, insertamos el elemento `<caption>` para definir el título:

```
<table>
  <caption>Reporte trimestral de ventas</caption>
```

El elemento `<caption>` se utiliza para asignar un título a la tabla y solo puede aparecer una vez por tabla, siempre justo después de la apertura de `<table>`.

Encabezado de la tabla con `<thead>`

A continuación, agregamos el elemento `<thead>` para agrupar las filas de encabezado. Dentro de él, insertamos una fila `<tr>` y tres celdas de encabezado `<th>`:

```
<thead>
  <tr>
    <th>Mes</th>
    <th>Vendedor</th>
    <th>Importe</th>
  </tr>
</thead>
```

Cuerpo de la tabla con <tbody>

El elemento <tbody> se utiliza para agrupar las filas de contenido. Podemos declarar múltiples secciones <tbody> dentro de una misma tabla para organizar los datos por grupo o período. En este ejemplo, usaremos tres secciones, una por cada mes del trimestre:

```
<tbody>
  <tr>
    <td rowspan="4">Enero</td>
    <td>Luis</td>
    <td>350,000</td>
  </tr>
  <tr>
    <td>Joana</td>
    <td>200,000</td>
  </tr>
  <tr>
    <td>Ana</td>
    <td>500,000</td>
  </tr>
  <tr>
    <th>Subtotal</th>
    <td>1,050,000</td>
  </tr>
</tbody>
```

Repetimos este bloque dos veces, uno para febrero y otro para marzo, cambiando los nombres de los meses y los importes correspondientes. El atributo rowspan="4" en la celda del mes indica que esta celda abarca cuatro filas.

Pie de tabla con <tfoot>

Finalmente, añadimos el elemento <tfoot> para representar el pie de la tabla. Dentro, insertamos una fila con una celda de encabezado y una celda de datos:

```
<tfoot>
  <tr>
    <th colspan="2">Total</th>
    <td>3,450,000</td>
  </tr>
</tfoot>
</table>
```

Aquí, el atributo `colspan="2"` en la celda `<th>` indica que dicha celda abarca dos columnas, para alinear correctamente el texto "Total" con la estructura general de la tabla.

Guardamos los cambios y visualizamos el documento en el navegador. Verás que la tabla se muestra estructurada con encabezado, cuerpo y pie claramente definidos.

Para mejorar su apariencia visual —como márgenes, bordes o colores— utilizaremos CSS.

Elementos multimedia

Insertar un reproductor de audio en un documento HTML

Aprenderemos a trabajar con elementos multimedia en HTML. El término multimedia se refiere a la presentación de información mediante una combinación de texto, sonido, imágenes, video y animaciones.

Evolución del soporte multimedia en HTML

En versiones anteriores de HTML, insertar contenido multimedia requería el uso de plugins de terceros, como Flash o QuickTime, los cuales no venían instalados por defecto en los navegadores.

Con la introducción de HTML5, ahora es posible incluir archivos multimedia de forma nativa, utilizando elementos específicos como `<audio>` y `<video>`. Esto mejora la compatibilidad y la experiencia del usuario.

Crear el documento base

Comenzamos creando un nuevo documento llamado `multimedia.html`. Luego, generamos la estructura básica del archivo HTML y modificamos los valores correspondientes.

Dentro del elemento `<title>` escribimos:

```
<title>Elementos multimedia en HTML</title>
```

También agregamos un encabezado de nivel 1:

```
<h1>Multimedia en HTML</h1>
```


Insertar un archivo de audio

Para este ejemplo, necesitaremos un archivo de audio. Puedes utilizar una canción, un fragmento de audiolibro u otro archivo de sonido que tengas disponible. Guarda el archivo dentro de la carpeta llamada recursos.

Código para insertar el reproductor de audio

Volvemos al editor de código y añadimos el siguiente bloque:

```
<h2>Reproductor de audio</h2>

<audio controls>
  <source src="recursos/disparos.mp3" type="audio/mp3">
  <source src="recursos/disparos.ogg" type="audio/ogg">
  <p>No se puede reproducir el archivo. El formato no es compatible con su navegador. Intente con otro navegador o contacte al administrador.</p>
</audio>
```

Explicación:

`<audio>`: elemento contenedor del reproductor.

`controls`: atributo booleano que permite mostrar los botones de reproducción, pausa y volumen.

`<source>`: define la ruta (`src`) y el tipo (`type`) de cada archivo de audio.

`<p>`: mensaje alternativo que se mostrará si el navegador no puede reproducir ninguno de los formatos.

Se recomienda ofrecer más de un formato de audio porque no todos los navegadores son compatibles con los mismos formatos.

Guardamos los cambios y abrimos el documento en el navegador. Verás un reproductor de audio cuya apariencia puede variar ligeramente según el navegador y la versión utilizada.

Al hacer clic en el botón reproducir, el archivo comenzará a sonar.

Insertar un reproductor de video en un documento HTML

Ahora aprenderemos cómo insertar un reproductor de video en un documento HTML de forma nativa, sin necesidad de plugins externos.

Obtener un archivo de video

Antes de comenzar con la codificación, necesitaremos un archivo de video. Si ya cuentas con uno en tu computadora, simplemente cópialo y colócalo dentro de la carpeta recursos.

Estructura del documento

A continuación, agrega un encabezado de segundo nivel para identificar la sección del video:

```
<h2>Reproductor de video</h2>
```

Sintaxis del elemento <video>

Para insertar el video, utilizamos el elemento <video>. Su sintaxis es muy similar al del elemento <audio>. Aquí tienes un ejemplo completo:

```
<video controls width="500" height="281">
  <source src="recursos/tortuga.mp4" type="video/mp4">
  <source src="recursos/tortuga.ogg" type="video/ogg">
  <source src="recursos/tortuga.webm" type="video/webm">
  <p>El video no ha podido reproducirse. Este formato no es compatible con
  su navegador. Intente con otro navegador o contacte al administrador.</p>
</video>
```

Explicación del código:

<video>: Elemento principal del reproductor.

controls: Muestra botones de reproducción, pausa, volumen y pantalla completa.

width y height: Definen el ancho y alto del reproductor (en píxeles).

<source>: Permite declarar múltiples formatos del mismo video para mejorar la compatibilidad con distintos navegadores.

<p>: Mensaje alternativo que se mostrará si el navegador no puede reproducir el video.

Es recomendable incluir al menos tres formatos diferentes (.mp4, .ogg, .webm) porque no todos los navegadores aceptan el mismo formato de video. Una vez guardados los cambios, abrimos el archivo en el navegador. Verás que el reproductor de video se ha insertado correctamente. Si el reproductor aparece demasiado grande o pequeño, puedes ajustar sus dimensiones con los atributos width y height, o bien mediante estilos CSS.



Otros atributos del elemento `<video>`

Además de los atributos que vimos en el video anterior, el elemento `<video>` admite otros atributos:

Poster

Muestra una imagen representativa del video para dar a los usuarios una pista acerca de su contenido.

Preload

Proporciona una sugerencia al navegador acerca de si el recurso debería ser mostrado cuando se cargue el documento. Puede tener uno de los siguientes valores:

- `none`: indica que este video no se debe almacenar en la *cache*.
- `metadata`: permite capturar los metadatos.
- `auto`: indica que, si es necesario, se puede descargar el video completo.
- Una cadena vacía: que es un sinónimo del valor *auto*.



Loop

Un atributo booleano que indica si el navegador debería recomenzar automáticamente la reproducción del video una vez que haya llegado a su fin.

Autoplay

Atributo booleano que especifica que el video comenzará a reproducirse automáticamente tan pronto se cargue la página.

Muted

Un valor booleano que permite que la salida de sonido del video sea silenciada.

Otro elemento multimedia

Además de los elementos <audio> y <video>, HTML también permite incrustar otro tipo de contenido multimedia, como videos de YouTube. Para esto, se utiliza el elemento <iframe>.

Uso del elemento <iframe>

El elemento <iframe> permite incrustar páginas web o servicios externos dentro de un recuadro en nuestra página. Puede ser utilizado para insertar contenido de YouTube, Google Maps, SoundCloud, entre otros.

Ejemplo: incrustar una página web

Añade un título:

<h2>Recurso incrustado</h2>

Inserta el iframe:

```
<iframe      src="https://www.begomez.com"      width="1000"      height="500"
allowfullscreen></iframe>
```

Este código incrusta este sitio web dentro de un recuadro de 1000×500 píxeles.

Incrustar un video de YouTube

<h2>Recurso incrustado video de youtube </h2>

YouTube proporciona directamente el código para embeber sus videos:

Ve a un video en YouTube (por ejemplo: *Las Cuatro Estaciones de Vivaldi*).

Haz clic en Compartir → Incorporar.

Copia el código HTML generado y pégalo en el editor de código.

Formularios

A continuación, aprenderemos a trabajar con formularios en HTML. Los formularios se utilizan en los sitios web para permitir que los visitantes envíen información específica, la cual puede ser procesada por el servidor o enviada por correo electrónico al administrador del sitio.

Introducción

Antes de comenzar, vamos a crear un nuevo documento llamado formularios.html. Una vez guardado, generamos la estructura básica de HTML5, modificando el valor del atributo lang a "es" y el contenido del elemento <title> por:

```
<title>Formularios en HTML</title>
```

Luego, dentro del <body>, agregamos un encabezado:

```
<h1>Cómo crear formularios en HTML</h1>
```

Consideraciones iniciales

Es importante aclarar que únicamente aprenderemos a maquetar formularios con HTML.

Los formularios que crearemos no serán funcionales, ya que para procesar y enviar los datos se requiere programación del lado del servidor.

El elemento <form>

El formulario comienza con el elemento <form>, el cual actúa como contenedor de todos los campos que lo conforman (cuadros de texto, botones de selección, listas desplegables, etc.).

```
<form>
```

```
  <!-- aquí van los campos -->
```

```
</form>
```

Atributos importantes del formulario:

action: Define la dirección o archivo del servidor que procesará los datos del formulario.

```
<form action="procesar.php">
```

Para efectos de este curso, podemos dejar el valor vacío, ya que no se enviarán datos realmente.

method: Establece el método con el que se enviarán los datos al servidor. Puede tener dos valores:

- get: Envía los datos a través de la URL, visibles en la barra de direcciones. Se usa comúnmente en formularios de búsqueda.
- post: Envía los datos de forma oculta, recomendada para formularios que manejan información personal, como inicios de sesión o formularios de contacto.

Ejemplo:

```
<form action="" method="post">
```

Campos dentro del formulario

Dentro del elemento `<form>`, se agregan los diferentes campos o controles que permiten al usuario introducir y enviar información:

- Cuadros de texto
- Áreas de texto
- Casillas de verificación (checkbox)
- Botones de radio
- Listas desplegables
- Botones de envío

Los formularios son fundamentales en el desarrollo web, ya que permiten la interacción entre el usuario y el sitio. Aunque en esta etapa solo trabajaremos con la parte visual utilizando HTML, más adelante podremos complementarlos con JavaScript y lenguajes del lado del servidor para hacerlos completamente funcionales.

Elemento Input

Aprenderemos a definir diferentes campos de formulario en HTML, es decir, los elementos con los que interactúan los usuarios de un sitio web. La mayoría de estos campos se crean utilizando el elemento `<input>`, que permite generar:

- Cuadros de texto
- Campos para contraseñas
- Casillas de verificación
- Botones de opción (radio)
- Campos para correo electrónico
- Botones de envío, entre otros.

Campo de texto básico

Para crear un campo de texto donde el usuario pueda ingresar su nombre, dirección u otro dato general, utilizamos el siguiente código:

```
<input type="text" name="nombre">
```

`type="text"`: Indica que se trata de un campo de texto.

`name="nombre"`: Sirve para identificar el dato que se enviará al servidor.

Etiqueta `<label>` asociada

Para agregar un título visible al campo, usamos el elemento `<label>`, que se puede asociar directamente al campo mediante el atributo `for`, el cual debe coincidir con el valor del atributo `id` del campo correspondiente:

```
<label for="nombre">Nombre:</label>
```

```
<input type="text" id="nombre" name="nombre">
```

Esto mejora la accesibilidad y permite, por ejemplo, que al hacer clic sobre la palabra "Nombre", el cursor se ubique automáticamente en el campo de texto.

Atributos comunes del campo `<input>`

size

Define la anchura del campo en caracteres. Por ejemplo:

```
<input type="text" name="nombre" id="nombre" size="15">
```

Aunque es funcional, hoy en día este tipo de ajustes se hacen principalmente con CSS.

autocomplete

Activa o desactiva las sugerencias automáticas del navegador basadas en datos previamente introducidos por el usuario.

```
<input type="text" name="nombre" id="nombre" size="15" autocomplete="on">
```

```
<input type="text" name="nombre" id="nombre" size="15" autocomplete="off">
```

"on" (valor por defecto): permite sugerencias.

"off": evita que se muestren sugerencias, útil en formularios con información sensible.

autofocus

Hace que el campo esté automáticamente seleccionado al cargar la página:

```
<input type="text" name="nombre" id="nombre" size="15" autocomplete="on" autofocus>
```

maxlength

Limita la cantidad máxima de caracteres que el usuario puede ingresar:


```
<input type="text" name="nombre" id="nombre" size="15" autocomplete="on" autofocus maxlength="10">
```

placeholder

Muestra un texto guía dentro del campo antes de que el usuario escriba:

```
<label for="direccion">Dirección:</label>
```

```
<input type="text" name="direccion" id="direccion" placeholder="Calle, número, ciudad">
```

Este texto desaparece automáticamente cuando el usuario comienza a escribir.

Separación y presentación

Dado que los elementos `<label>` e `<input>` son elementos de línea, por defecto se presentan uno junto al otro. Para mejorar la visualización, se puede usar un salto de línea con la etiqueta `<br>`:

```
<label for="nombre">Nombre:</label>
```

```
<input type="text" id="nombre" name="nombre"><br><br>
```

```
<label for="direccion">Dirección:</label>
```

```
<input type="text" id="direccion" name="direccion" placeholder="Calle, número, ciudad"><br>
```

Guarda los cambios y abre el documento en un navegador. Podrás observar cómo el cursor se posiciona en el campo especificado por `autofocus`, cómo se limita la cantidad de caracteres y cómo aparece el texto guía dentro del campo de dirección.

Otros campos de formulario con el elemento Input

Continuaremos explorando los distintos **atributos** que admite el elemento `<input>`, el cual nos permite crear la mayoría de los campos de un formulario HTML.

Atributo value

El atributo `value` permite definir un **valor inicial** o predeterminado en un campo de formulario.

```
<label for="apellido">Apellido:</label>
```

```
<input type="text" id="apellido" name="apellido" value="Cervantes">
```

Al visualizarlo en el navegador, el campo de texto mostrará por defecto el valor **“Cervantes”**. Este valor puede ser modificado o borrado por el usuario.

Atributo readonly

Si queremos que el usuario **no pueda modificar** el valor de un campo, utilizamos el atributo readonly.

```
<input type="text" value="Cervantes" readonly>
```

Esto bloquea la edición del campo, aunque el contenido sigue siendo visible y se enviará al servidor al momento de enviar el formulario.

Atributo disabled

Para mostrar un campo deshabilitado, se utiliza el atributo disabled. Esto impide al usuario interactuar con el campo y, además, el valor no será enviado al servidor.

```
<input type="text" name="direccion" id="direccion" placeholder="Calle, número, ciudad" disabled>
```

Atributo required

El atributo required indica que el campo es obligatorio. El navegador no permitirá enviar el formulario si este campo está vacío.

```
<input type="text" name="nombre" required>
```

Este comportamiento se observa mejor cuando se agrega un botón de envío, ya que el formulario solo se enviará si el campo cumple con los requisitos.

Campo para correo electrónico

Para validar una dirección de correo electrónico, el atributo type se define como email:

```
<label for="email">Correo electrónico:</label>
```

```
<input type="email" id="email" name="email" placeholder="correo@ejemplo.com" required>
```

El navegador mostrará un mensaje de advertencia si el formato no es válido.

Campo para contraseña

Para ocultar el texto ingresado por el usuario (por ejemplo, contraseñas), usamos el tipo password:

```
<label for="password">Contraseña:</label>
```

```
<input type="password" id="password" name="password">
```

Campo para dirección web

Para campos que deben contener direcciones URL válidas, se usa type="url":

```
<label for="web">Dirección web:</label>
```

```
<input type="url" id="web" name="web" placeholder="https://www.ejemplo.com">
```

Botones de selección (radio)

Los botones tipo radio permiten al usuario elegir una sola opción entre varias.

```
<p>Edad:</p>
```

```
<input type="radio" id="edad1" name="edad" value="menor18" checked >
```

```
<label for="edad1">Menor de 18 años</label><br>
```

```
<input type="radio" id="edad2" name="edad" value="adulto" >
```

```
<label for="edad2">Entre 18 y 60 años</label><br>
```

```
<input type="radio" id="edad3" name="edad" value="mayor60">
```

```
<label for="edad3">Mayor de 60 años</label>
```

Todos los botones de selección o radio tienen el mismo atributo name="edad" → esto los agrupa.

El atributo checked en el primer botón lo deja seleccionado por defecto.

Se agregan saltos de línea (
) para que se muestren en forma de lista vertical.

Más campos de formulario con el elemento Input

El elemento <input> no solo permite crear campos de texto, correos electrónicos y contraseñas, sino también otros tipos de controles que mejoran la interactividad de los formularios. A continuación, exploramos algunos de ellos.

Casillas de verificación (checkbox)

Las casillas de verificación son similares a los botones de selección (radio), con la diferencia de que permiten al usuario seleccionar varias opciones a la vez.

Ejemplo:

```
<p>Pasatiempos:</p>
```

```
<input type="checkbox" id="pasatiempo1" name="pasatiempo1" value="leer">
```

```
<label for="pasatiempo1">Lectura</label><br>
```

```
<input type="checkbox" id="pasatiempo2" name="pasatiempo2" value="deportes">
```

```
<label for="pasatiempo2">Hacer deportes</label><br>
```

```
<input type="checkbox" id="pasatiempo3" name="pasatiempo3" value="musica">
<label for="pasatiempo3">Escuchar música</label><br>
<input type="checkbox" id="pasatiempo4" name="pasatiempo4" value="otros">
<label for="pasatiempo4">Otros</label>
```

Para seleccionar una opción por defecto, se agrega el atributo checked a uno de los <input>:

```
<input type="checkbox" ... checked>
```

Campo numérico con paso (step)

El atributo type="number" permite al navegador mostrar un control gráfico para

Incrementar o disminuir valores.

```
<label for="cantidad">Cantidad:</label>
```

```
<input type="number" id="cantidad" name="cantidad" value="1" min="1" max="5"
step="1">
```

min: valor mínimo permitido.

max: valor máximo permitido.

step: tamaño del incremento (por ejemplo, 3 para que avance de 3 en 3).

Campo para selección de fecha (date)

Permite al usuario seleccionar una fecha en formato año-mes-día.

```
<label for="fecha">Fecha de nacimiento:</label><br>
```

```
<input type="date" id="fecha" name="fecha">
```

Este campo es soportado por la mayoría de los navegadores modernos. En versiones antiguas de Firefox, puede que no esté disponible.

Barra de rango (range)

Se utiliza para seleccionar un valor dentro de un rango predefinido mediante una barra deslizante.

```
<label for="rango">Define un rango:</label>
```

```
<input type="range" id="rango" name="rango" min="0" max="100" value="50" >
```

Subida de archivos (file)

Permite al usuario adjuntar uno o varios archivos desde su dispositivo.

`<form action="" method="post" enctype="multipart/form-data">` **(Utilizar el mismo formulario creado al principio – agregar el atributo enctype)**

`<label for="archivo">Adjuntar archivo:</label><br>`

`<input type="file" id="archivo" name="archivo[]" multiple>`

`</form>`

multiple: permite subir varios archivos.

enctype="multipart/form-data": obligatorio para enviar archivos correctamente al servidor.



Otros atributos que admite el elemento input

Además de los atributos que hemos visto en los videos anteriores, el elemento `<input>` incluye otros atributos que se muestran a continuación:

Accept

Se utiliza para definir los tipos de archivos que el servidor puede aceptar. Los valores que admite este atributo pueden ser una lista de contenidos separados por una coma, tales como un media type con una extensión de archivo (como jpeg, pdf, mp4, etc), o una de las cadenas audio/*, video/* e image/* para archivos de audio, video e imagen, respectivamente. Ejemplo:

```
<form action="" method="">
  <input type="file" accept="image/*">
  <input type="submit">
</form>
```

En este ejemplo, el formulario permitirá cargar archivos de imágenes con cualquier formato disponible.



Form

Este atributo permite insertar elementos `<input>` en cualquier parte dentro de un documento HTML, y no solamente dentro de un elemento `<form>`. El valor del atributo deberá ser el mismo asignado a un atributo `id` de un elemento `<form>` en el mismo documento.

```
<form action="datos.php" id="form1">
  Nombre: <input type="text" name="fname"><br>
  <input type="submit" value="Enviar">
</form>
Apellido: <input type="text" name="lname" form="form1">
```

Formation

Si el elemento es de tipo `<submit>`, se utiliza para reemplazar al atributo *action* del formulario al que pertenece el elemento.

```
<form action="datos.php">
  Nombre: <input type="text" name="fname"><br>
  Apellido: <input type="text" name="lname"><br>
  <input type="submit" value="Enviar"><br>
  <input type="submit" formaction="datos2.php" value="Enviar a otra página">
</form>
```



En este ejemplo si se pulsa el segundo botón se procesará un *action* diferente.

Formenctype

Si el elemento es de tipo `<submit>`, este atributo reemplaza al atributo *enctype* del formulario al que pertenece el elemento. Los valores posibles son:

- `application/x-www-form-urlencoded`: el valor predeterminado si el atributo no es especificado.
- `multipart/form-data`: no se aplica ninguna codificación.
- `text/plain`

```
<form action="datos.php">
  Nombre: <input type="text" name="fname">
  Cargar imagen: <input type="file" name="fimage">
  <input type="submit" value="Enviar con imagen"
  formenctype="multipart/form-data">
  <input type="submit" value="Enviar sin imagen">
</form>
```

En el código de este ejemplo, el segundo botón le indica al navegador que envíe el formulario sin cargar la imagen adjunta.



Formmethod

Si el elemento `input` es un botón de tipo `<submit>`, este atributo reemplaza al atributo *method* del formulario al que pertenece el elemento. Los valores posibles son:

- `post`: Los datos del formulario son incluidos en el cuerpo del formulario, y enviados al servidor.
- `get`: Los datos del formulario son añadidos a la URL del formulario, precedido por un signo de interrogación ("`?`") como separador.

```
<form action=".datos.php">
  Nombre: <input type="text" name="fname">
  <input type="submit" value="Enviar con GET" formmethod="get">
  <input type="submit" value="Enviar con POST" formmethod="post">
</form>
```

En este ejemplo, al pulsar sobre cualquiera de los formularios el navegador enviará los datos con el método especificado.

Formnovalidate

Si el elemento `input` es de tipo *submit*, este atributo *booleano* le indica al navegador no validar los datos del formulario durante su envío.



```
<form action="datos.php">
  E-mail: <input type="email" name="femail"><br>
  <input type="submit" value="Enviar"><br>
  <input type="submit" formnovalidate="formnovalidate"
  value="Enviar sin validar">
</form>
```

En el ejemplo de arriba, al pulsar sobre el primer botón el formulario se enviará y se realizarán las validaciones correspondientes. Si se pulsa sobre el segundo botón, formulario se enviará sin las validaciones pertinentes.

Formtarget

Si el elemento input es de tipo *submit*, este atributo indica dónde mostrar la respuesta que se recibe después de enviar el formulario. Los valores que admite este atributo son:

- `_self`: la respuesta se muestra en el mismo contexto que contiene al formulario. Este valor es el predeterminado.
- `_blank`: la respuesta se muestra en una nueva ventana/pestaña.
- `_parent`: carga la respuesta en el contexto de navegación padre del actual.
- `_top`: carga la respuesta en el más alto contexto de navegación (el contexto más exterior que contiene al formulario).



HTML & CSS

desde cero hasta avanzado

```
<form action="datos.php">
  Nombre: <input type="text" name="fname"><br>
  Apellido: <input type="text" name="flastname"><br>
  <input type="submit" value="Enviar">
  <input type="submit" formtarget="_blank" value="Enviar en una nueva ventana">
</form>
```

En este ejemplo, al pulsar sobre el segundo botón el formulario se procesará en una nueva /pestaña.

List

Permite establecer una lista de valores predefinidos en etiquetas de tipo input. El valor de este atributo debe ser el mismo del *id* asignado a un elemento `<datalist>` en el documento.

```
Escribe el nombre de una fruta: <input list="frutas">

<datalist id="frutas">
  <option value="Manzana">
  <option value="Uva">
  <option value="Fresa">
  <option value="Pera">
  <option value="Cereza">
</datalist>
```



HTML & CSS

desde cero hasta avanzado

En el ejemplo de arriba, al escribir el nombre de una fruta en el campo `<input>` el navegador mostrará un listado de opciones predefinidas como sugerencias al usuario.

Pattern

Este atributo permite delimitar mediante *expresiones regulares* el contenido insertado por el usuario en los inputs de los formularios. Se puede usar el atributo *title* con un texto que sirva de ayuda al usuario.

Una *expresión regular* es una secuencia de caracteres que forman un patrón de búsqueda. Para que sea válido, este patrón debe coincidir con el valor completo y no sólo una parte.

Este atributo aplica solamente cuando el valor del atributo `type` es igual a *text*, *search*, *tel*, *url*, *email*, o *password*.

```
<form action="datos.php" method="POST">
  Contraseña: <input type="password" name="contrasena"
  pattern="[A-Za-z]{5-10}"
  title="La contraseña debe tener entre 5 y 10 caracteres">
  <input type="submit" value="Enviar">
</form>
```



HTML & CSS

desde cero hasta avanzado

En el código anterior, el atributo *pattern* indica al navegador que sólo se admite una contraseña que contenga entre 5 y 10 caracteres escritos en letras mayúsculas o minúsculas.

Listas desplegables y áreas de texto

El elemento `<input>` también permite crear un campo de búsqueda, útil para que los usuarios realicen consultas sobre contenidos dentro de un sitio web.

Para ello, simplemente se debe establecer el atributo `type="search"`. Este tipo de campo suele colocarse dentro de un elemento `<form>`, acompañado de un botón de envío. En la mayoría de los casos, el método de envío (`method`) se establece como `get`, que es el valor por defecto.

Es importante aclarar que el siguiente ejemplo de buscador no será funcional, ya que para ello se requiere el uso de un lenguaje de programación del lado del servidor. Aquí únicamente se explicará cómo maquetar el campo de búsqueda en HTML.

Ejemplo de campo de búsqueda:

```
<form method="get"> (Utilizar el mismo formulario creado al principio)

  <label for="buscador">Buscar:</label>

  <input type="search" id="buscador" name="buscador" placeholder="Buscar...">

</form>
```

Al visualizarlo en el navegador, se mostrará un campo de texto con estilo de buscador y un mensaje guía.

Listas desplegables (`<select>`)

HTML proporciona el elemento `<select>` para crear listas desplegables que permiten seleccionar una o más opciones.

Ejemplo: Selección de país

```
<label for="pais">País:</label><br>

<select id="pais" name="pais">

  <option value="argentina" selected >Argentina</option>
  <option value="chile" >Chile</option>
  <option value="colombia">Colombia</option>
  <option value="estados_unidos">Estados Unidos</option>
  <option value="mexico">México</option>
  <option value="panama">Panamá</option>
  <option value="republica_dominicana">República Dominicana</option>
```

</select>

El atributo selected indica cuál opción debe aparecer seleccionada por defecto.

Para permitir la selección múltiple, se utiliza el atributo multiple y se deben agregar [] al nombre:

<select name="países[]" multiple>

Por ejemplo:

<label for="países">Países:</label>

<select id="países" name="países[]" multiple >

<option value="argentina" selected >Argentina</option>

<option value="chile" >Chile</option>

<option value="colombia">Colombia</option>

<option value="estados_unidos">Estados Unidos</option>

<option value="mexico">México</option>

<option value="panama">Panamá</option>

<option value="republica_dominicana">República Dominicana</option>

</select>

De este modo, el usuario podrá seleccionar varias opciones manteniendo presionada la tecla **Ctrl** (Windows) o **Cmd** (Mac).

Áreas de texto (<textarea>)

Para campos donde se espera que el usuario introduzca comentarios o textos largos, se emplea el elemento <textarea>.

Ejemplo:

<label for="mensaje">Comentarios:</label>

<textarea id="mensaje" name="mensaje" rows="10" cols="30"></textarea>

- rows: número de líneas visibles.
- cols: número de caracteres visibles por línea.

En caso de que el usuario escriba más líneas de las visibles, el navegador mostrará automáticamente una barra de desplazamiento vertical para acceder al contenido oculto.

Botones en formularios HTML

Ahora, veremos cómo crear diferentes tipos de **botones** en un formulario HTML. Existen dos formas principales para hacerlo: mediante el uso del elemento `<input>` con distintos valores en su atributo `type`, y mediante el uso del elemento `<button>`.

Botones con el elemento `<input>`

Con el elemento `<input>`, es posible crear botones para enviar un formulario, restablecer los campos o simplemente un botón genérico. A continuación se muestran ejemplos para cada uno:

Botón para enviar datos:

```
<!-- Botón para enviar datos -->
```

```
<input type="submit" name="enviar_datos" value="Enviar formulario">
```

Este tipo de botón se utiliza para enviar los datos del formulario al servidor.

Botón para restablecer campos:

```
<!-- Botón para limpiar campos -->
```

```
<input type="reset" name="resetear_formulario" value="Limpiar campos">
```

Este botón permite restablecer todos los campos del formulario a sus valores iniciales.

Botón genérico:

```
<!-- Botón sin acción predefinida -->
```

```
<input type="button" name="boton_generico" value="Pulsar aquí">
```

Este botón no ejecuta ninguna acción por defecto. Para que funcione, debe ser programado con algún lenguaje del lado del cliente, como JavaScript.

Prueba en el navegador

Al rellenar algunos campos del formulario (nombre, correo electrónico, contraseña, sitio web, edad, pasatiempos, cantidad, país), y pulsar el botón **"Limpiar campos"**, el navegador restablecerá todos los valores automáticamente.

Por otro lado, si se presiona **"Enviar formulario"**, el navegador verificará si los campos obligatorios se han completado correctamente. Si algún campo marcado como `required` está vacío, el envío se bloqueará hasta que se corrija.

Botones con el elemento <button>

Otra forma de crear botones es utilizando el elemento <button>. Este permite una mayor flexibilidad, ya que puede contener otros elementos HTML como texto en negrita, imágenes o iconos.

Ejemplo de botón genérico:

```
<button type="button">Este es un <strong>botón</strong></button>
```

En este caso, el botón muestra la palabra "botón" en negrita. El atributo type puede asumir los mismos valores que en el caso del elemento <input>: submit, reset o button.

Una ventaja del elemento <button> es que permite tener múltiples botones de envío dentro del mismo formulario, cada uno con una función diferente, según la lógica que se programe.



Atributos que admite el elemento button

Tal y como vimos en el video anterior, el elemento `<button>` permite insertar un botón estándar en cualquier parte de una página HTML.

Cuando se emplea este elemento es necesario declarar el atributo *type*, el cual puede tomar uno de los siguientes valores:

- **submit**: representa un botón para enviar el formulario.
- **reset**: crea un botón que restaura todos los campos del formulario a sus valores iniciales.
- **button**: indica al navegador que debe proveer un botón genérico que no tiene asociada ninguna acción, a no ser que sea programado mediante un lenguaje como JavaScript.

Se recomienda siempre definir el atributo *type* dentro de un elemento `<button>`, debido a que si se omite cada navegador le aplicará de forma predeterminada un comportamiento diferente.



Dependiendo del valor asignado al atributo *type*, el elemento `<button>` puede admitir otros atributos que definen ciertos parámetros relativos a cómo el formulario debe ser enviado, tal y como veremos a continuación.

Button (type=submit)

Cuando el elemento `<button>` tiene declarado el atributo *type* con el valor *submit*, se pueden definir cualquiera de los siguientes atributos específicos:

- autofocus
- disabled:
- form
- formaction
- formenctype
- formmethod
- formtarget
- formnovalidate
- name
- value

Estos atributos ya han sido definidos de forma detallada en otras lecciones y archivos PDF, por tanto no los vamos a tratar en este documento.



Button (type=reset)

Si el elemento `<button>` tiene el atributo *type* con el valor *reset*, también puede admitir los siguientes atributos específicos:

- autofocus
- disabled
- form
- name
- value

Button (type=button)

Si en el elemento `<button>` declaramos el atributo *type* con el valor *button*, también podemos definir los siguientes atributos específicos:

- autofocus
- disabled
- form
- name
- value



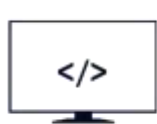
Otros campos de formulario

HTML dispone de otros elementos para insertar campos de formulario. Estos son:

Optgroup

Representa un grupo de opciones, organizadas temáticamente, que forman parte de un elemento `<select>`. Por ejemplo:

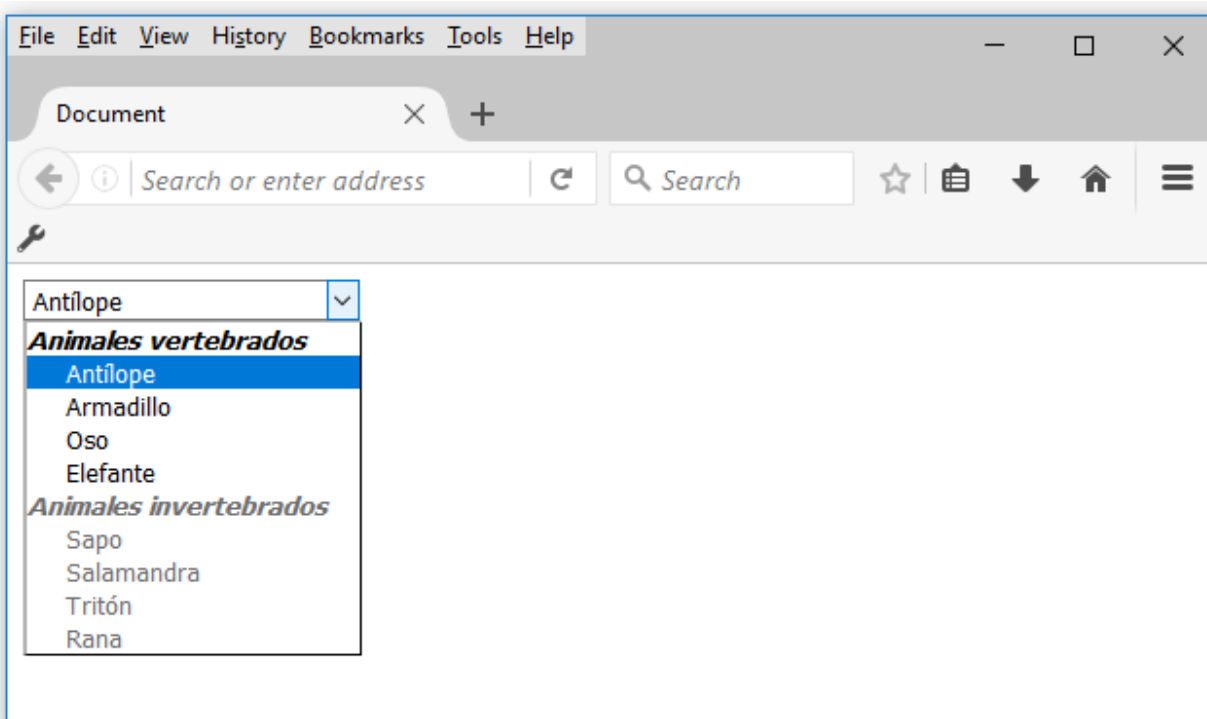
```
<select>
  <optgroup label="Animales vertebrados">
    <option value="antilope">Antílope</option>
    <option value="armadillo">Armadillo</option>
    <option value="oso">Oso</option>
    <option value="elefante">Elefante</option>
  </optgroup>
  <optgroup label="Animales invertebrados">
    <option value="sapo">Sapo</option>
    <option value="salamandra">Salamandra</option>
    <option value="tritón">Tritón</option>
    <option value="rana ">Rana </option>
  </optgroup>
</select>
```



HTML & CSS

desde cero hasta avanzado

Y este sería el resultado en la pantalla del navegador:



Este elemento admite dos atributos:

- **label**, que inserta una línea de texto que identifica al grupo de opciones.
- **disabled**, indica que el grupo y sus opciones se encuentran deshabilitados.



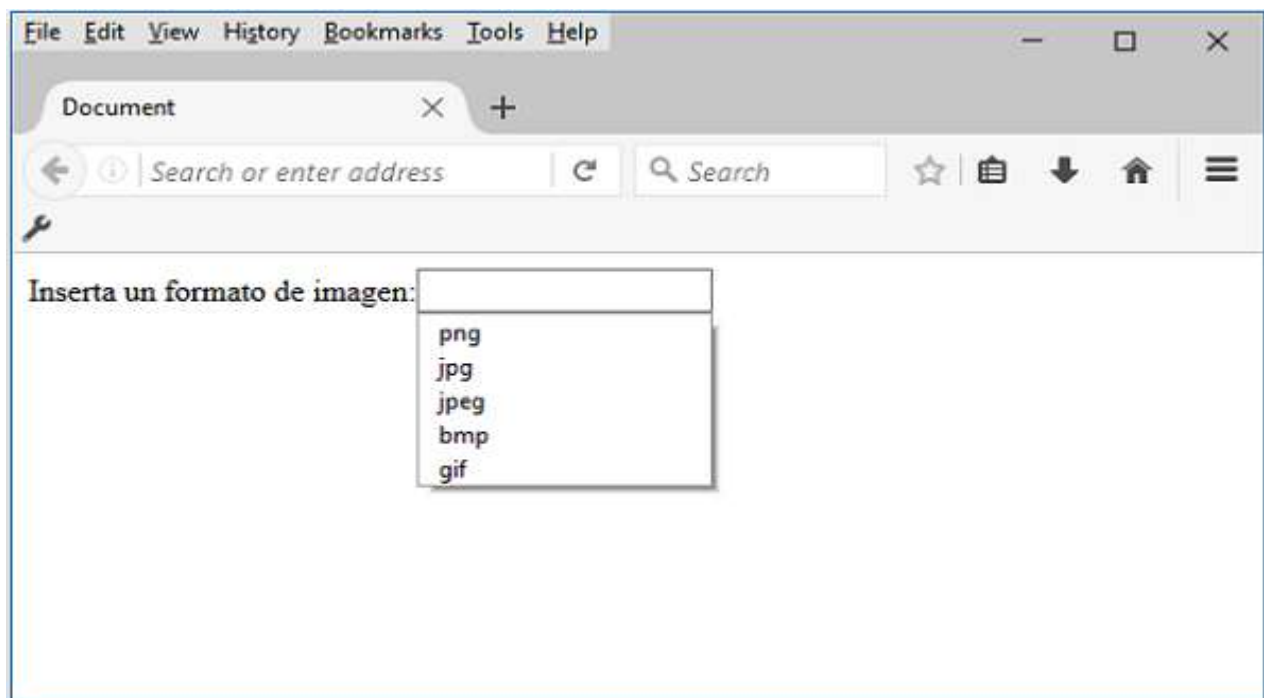
HTML & CSS

desde cero hasta avanzado

Datalist

Inserta una lista de opciones con el fin de proveer sugerencias a los usuarios. Estas sugerencias son definidas mediante el elemento `<option>`.

```
Inserta un formato de imagen:<input list="formatos">
<datalist id="formatos">
  <option value="png">
  <option value="jpg">
  <option value="jpeg">
  <option value="bmp">
  <option value="gif">
</datalist>
```





Este elemento no tiene otros atributos mas que aquellos que son comunes a todos los elementos.

Output

Se utiliza para mostrar los resultados de un cálculo llevado a cabo en el documento HTML. Este cálculo se realiza mediante un lenguaje de programación del lado cliente como JavaScript.

```
<form oninput="resultado.value=parseInt(a.value)+parseInt(b.value)">
  <input type="range" name="rango" value="50" /> +
  <input type="number" name="numero" value="10" /> =
  <output name="resultado">60</output>
</form>
```

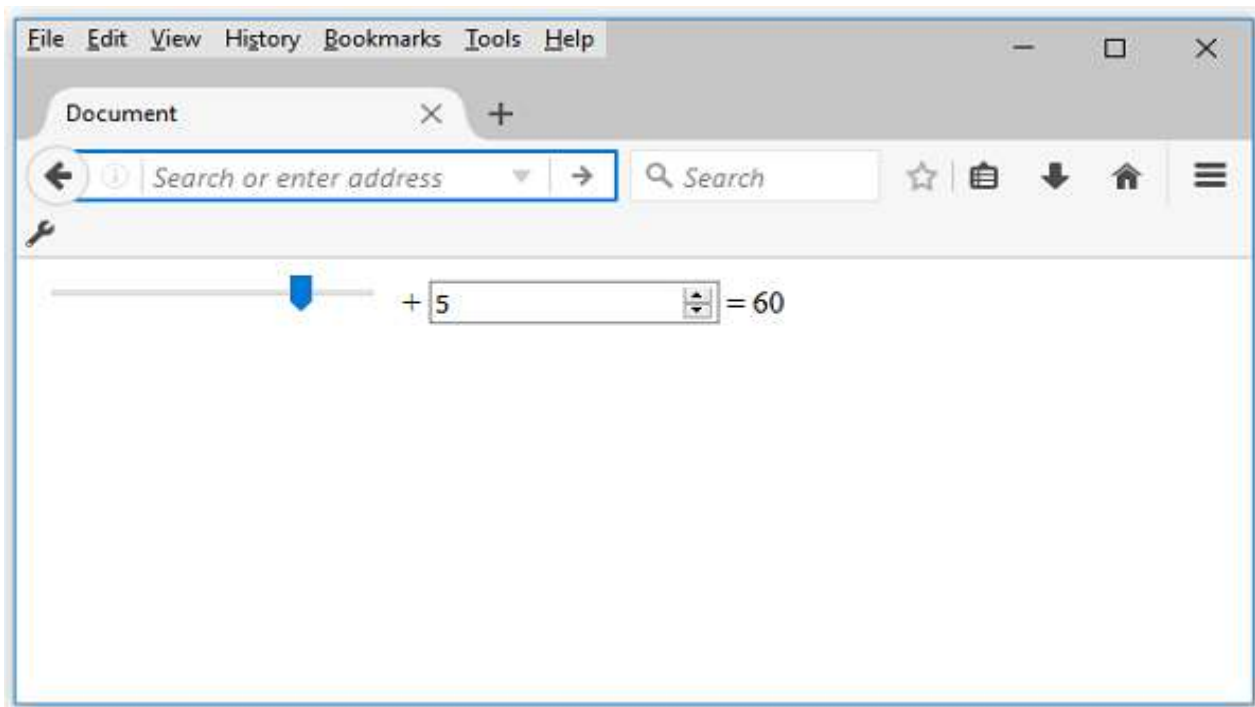
En el código de arriba, se muestra un formulario que cuenta con un campo de rango numérico, conjuntamente con un campo de entrada de datos dentro del cual podemos ingresar valores numéricos de forma manual. Cada vez que cambian los valores de cada campo el resultado se muestra en el elemento `<output>`.

Y así es como se muestra en el navegador:



HTML & CSS

desde cero hasta avanzado



Los atributos que admite este elemento son:

- **For:** define una lista de identificadores que van separados por espacios, cada uno de los cuales debería coincidir con el atributo *id* de uno de los campos utilizados en el cálculo.
- **Form:** define el formulario al cual está asociado un elemento.
- **Name:** el nombre que será usado en el envío del formulario.



Fieldset y Legend

El elemento `<fieldset>` agrupa un conjunto de campos de formulario bajo un mismo nombre. Por defecto, el navegador muestra un marco alrededor de los campos agrupados.

Por su parte, el elemento `<legend>` crea un título para los campos agrupados.

El elemento `<fieldset>` admite los atributos *name*, *disabled* y *form*.

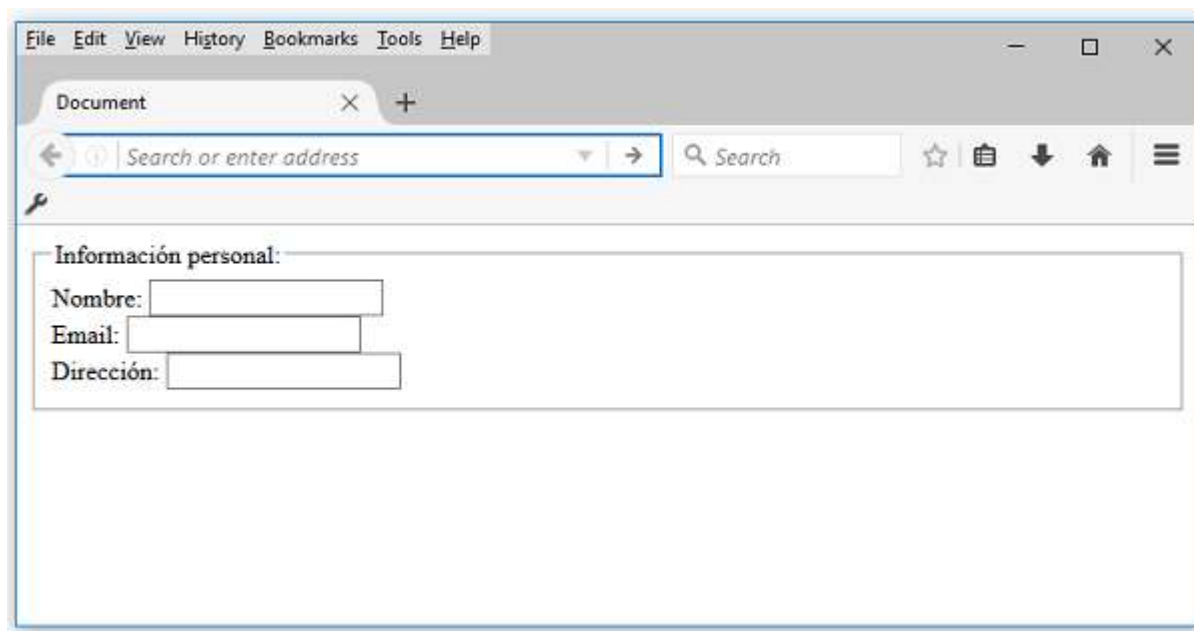
El elemento `<legend>` no tiene atributos específicos.

```
<form>
  <fieldset>
    <legend>Información personal:</legend>
    Nombre: <input type="text"><br>
    Email: <input type="email"><br>
    Dirección: <input type="text">
  </fieldset>
</form>
```



HTML & CSS

desde cero hasta avanzado



Meter

Se utiliza para representar una medida dentro de un rango conocido.

Este elemento es muy útil en situaciones en las cuáles debemos mostrar información que se puede representar mediante un medidor, por ejemplo, para indicar que una contraseña tiene un nivel de seguridad bajo, medio o alto, o también para mostrar el espacio disponible en un disco duro, etc.

Los atributos que soporta este elemento son:

Value: representa el valor actual del medidor.



HTML & CSS

desde cero hasta avanzado

Min y Max: representan los límites inferior y superior del rango.

```
Valor actual y valor mínimo y máximo: <br><br>
<meter min="2" max="20" value="4">valor 4</meter> <br>
<meter min="2" max="20" value="10">valor 10</meter> <br>
<meter min="2" max="20" value="19">valor 19</meter> <br><br>
```

Low y High: definen valores considerados como bajos y altos, respectivamente.

```
Valor optimum <br><br>
<meter min="10" max="30" value="15" optimum="20">valor 15</meter> <br>
<meter min="10" max="30" value="25" optimum="20">valor 25</meter> <br>
<meter min="10" max="30" value="20" optimum="20">valor 20</meter>
<br><br>
```

Optimum: indica cuál es el valor óptimo del medidor.

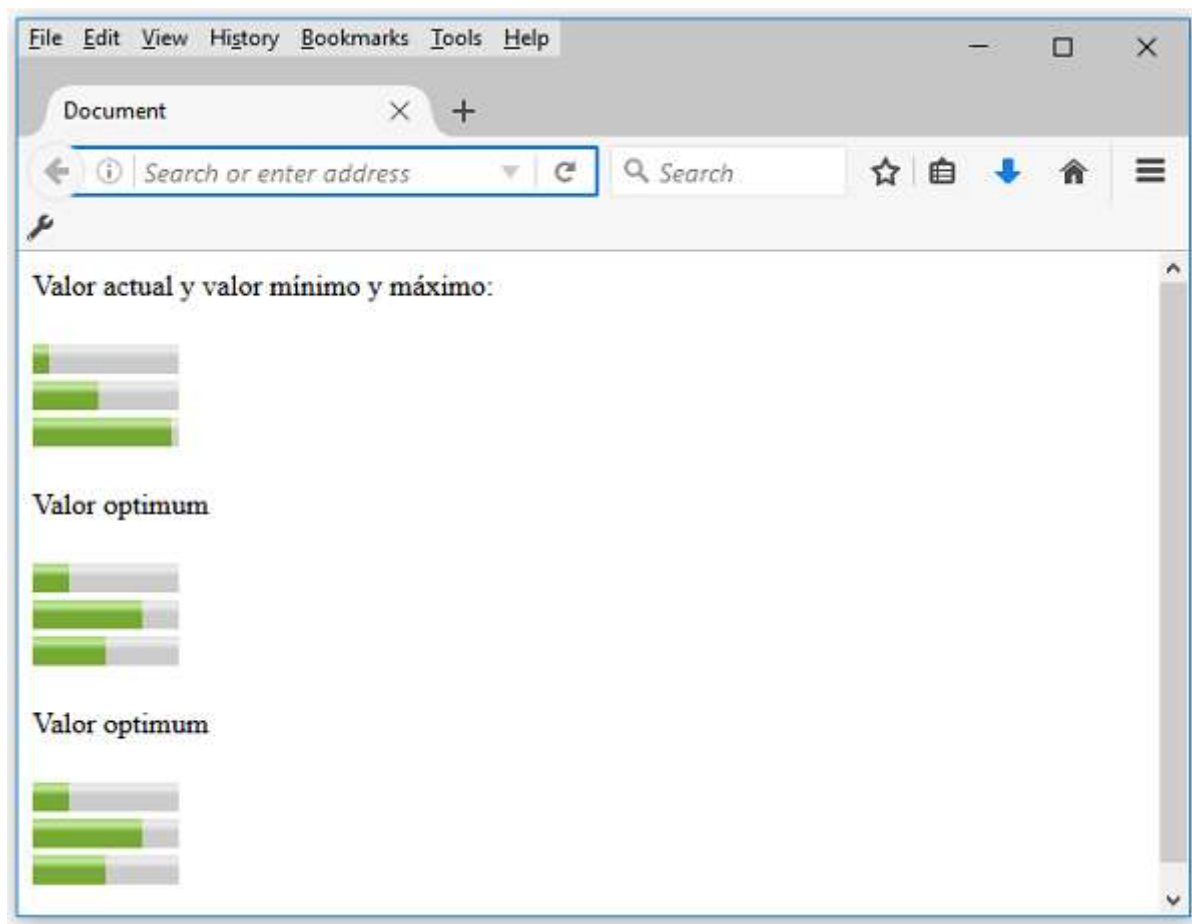
```
Valor optimum <br><br>
<meter min="10" max="30" value="15" optimum="20">valor 15</meter> <br>
<meter min="10" max="30" value="25" optimum="20">valor 25</meter> <br>
<meter min="10" max="30" value="20" optimum="20">valor 20</meter>
<br><br>
```

Y así es como se muestran los bloques de códigos anteriores en el navegador:



HTML & CSS

desde cero hasta avanzado



Progress

Se utiliza para indicar de manera gráfica el progreso de una tarea o de un proceso. Este campo de formulario se representa por una barra de progreso que se mueve de izquierda a derecha, de acuerdo con el progreso de la operación.

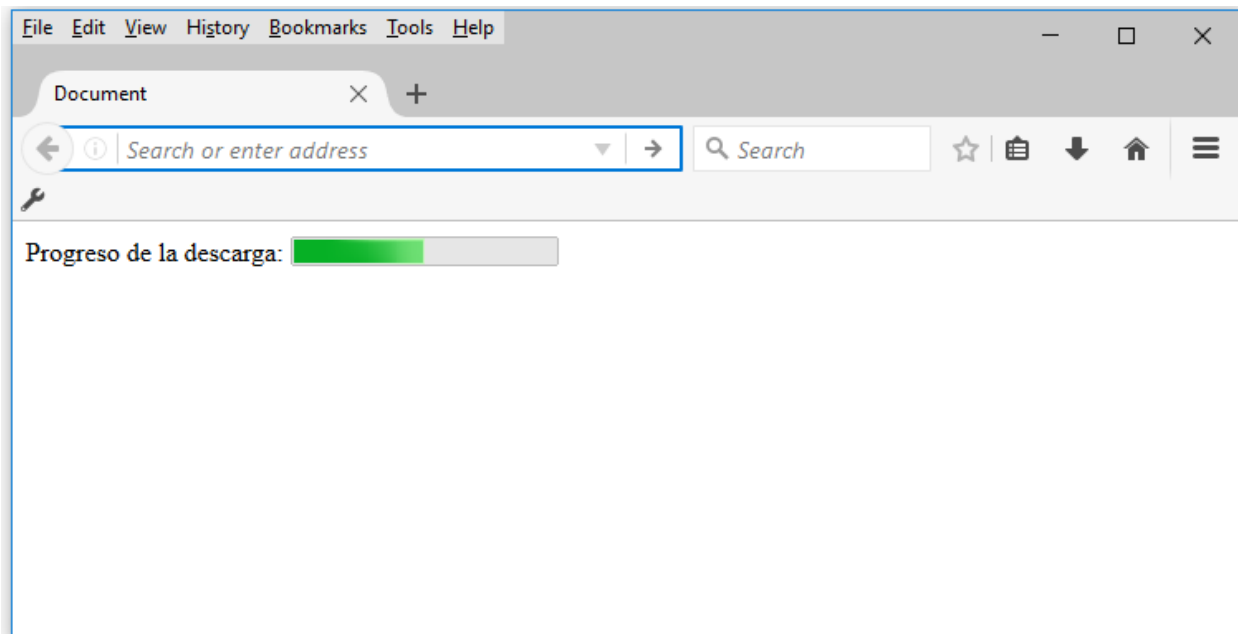


HTML & CSS

desde cero hasta avanzado

Mediante el atributo *max* podemos definir el valor que debe tener la barra de progreso para estar completa, mientras que el atributo *value* permite definir el valor actual de la barra de progreso.

```
Progreso de la descarga: <progress value="50" max="100"></progress>
```



Por defecto, las barras de progreso se muestran sin interactividad, pero podemos hacer que los valores aumente de manera dinámica mediante JavaScript.