

# Introducción a la programación

JavaScript

# Introducción a la programación

## Algoritmo

Un algoritmo es una secuencia ordenada de pasos que sirven para resolver un problema.

► ¿Sabrías explicar el algoritmo de la suma?

► ¿Cómo saldrías del aula?

¿Se podría hacer un algoritmo para ello?



# Introducción a la programación

## Programa

Cuando el problema a resolver lo puede realizar un ordenador entonces el algoritmo “se convierte” en programa.



- Al igual que un algoritmo se descompone en una secuencia de pasos, un programa se compone de una secuencia de instrucciones.

# Introducción a la programación

## Ciclo de desarrollo de Software



# Introducción a la programación

## Lenguajes de programación

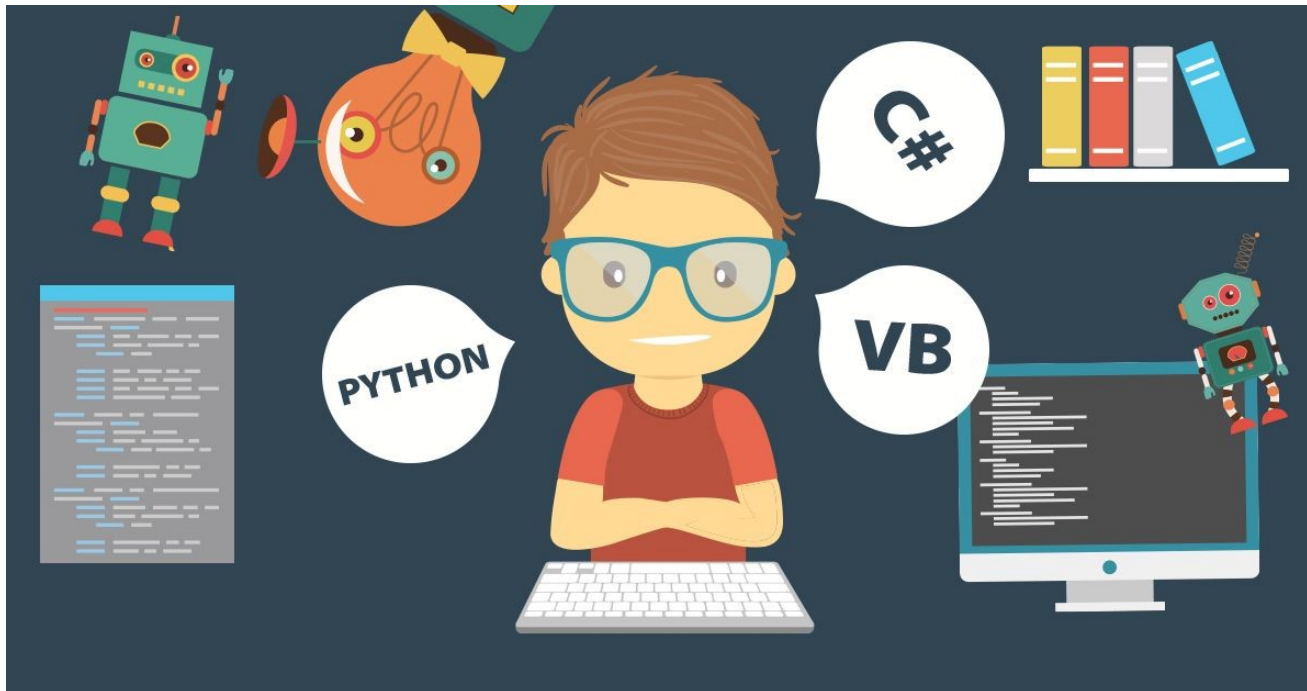
- Lenguaje máquina. El lenguaje máquina o código máquina es el sistema de códigos directamente interpretable por un circuito microprogramable, como el microprocesador de un ordenador.

```
11001010 00010111 11110101 00101011
00010111 11110101 00101011 00101011
11001010 00010111 11110101 00101011
00010111 11110101 00101011 00101011
11001010 11110101 00101011 00101011
11001010 11001010 11110101 00101011
11001010 11110101 00101011 00101011
11001010 00010111 11110101 00101011
00010111 11110101 00101011 00101011
11001010 11110101 00101011 00101011
```

# Introducción a la programación

## Lenguajes de programación

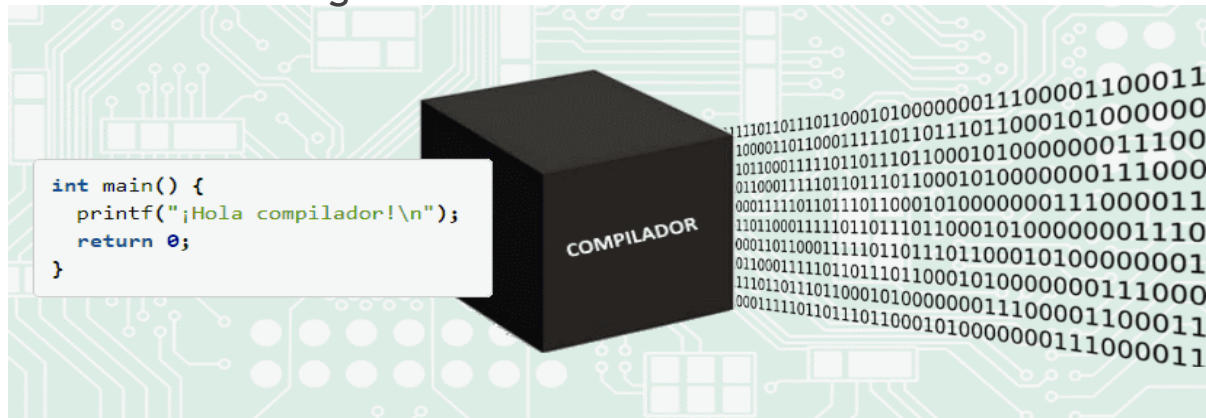
- Lenguaje de alto nivel. Un lenguaje de programación de alto nivel se caracteriza por expresar los algoritmos de una manera adecuada a la capacidad cognitiva humana, en lugar de la capacidad con que los ejecutan las máquinas.



# Introducción a la programación

## Lenguajes de programación

- **Compilador.** Un compilador es un tipo de traductor que transforma un programa entero de un lenguaje de programación a otro. Usualmente el lenguaje objetivo es código máquina, aunque también puede ser traducido a un código intermedio o a texto.
- **Intérprete.** Un intérprete es un programa informático capaz de analizar y ejecutar otros programas. Los intérpretes se diferencian de los compiladores o de los ensambladores en que los intérpretes sólo realizan la traducción a medida que sea necesaria, típicamente, instrucción por instrucción, y normalmente no guardan el resultado de dicha traducción.



# JavaScript

## Lenguajes de programación

- ▶ Es un lenguaje interpretado.
- ▶ Se utiliza principalmente en su forma del lado del cliente.
- ▶ Se utiliza principalmente en páginas web, aunque cada vez más fuera de ellas (documentos PDF, widgets de escritorio, etc.).



```
<!DOCTYPE html>
<html>
<head>
  <title>Apenas un gui3n</title>
</head>
<body>
<h1>
  <script language="javascript" type="text/javascript">
    document.write ("¡Hola Mundo!")
  </script>
</h1>
</body>
</html>
```

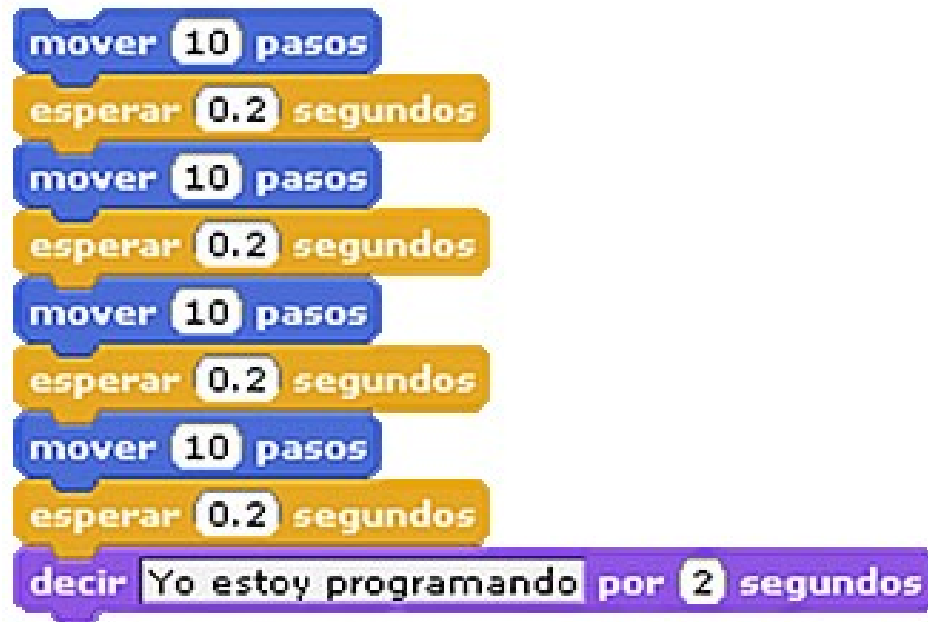
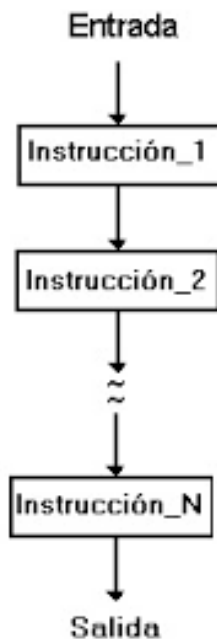


# JavaScript

## Ejecución secuencial

JavaScript es un lenguaje secuencial, es decir, las instrucciones se ejecutan en secuencia, una detrás de otra según el orden en que están escritas.

Existen instrucciones que permiten alterar la secuencia de ejecución que veremos más adelante.



# Javascript

## Empezando con Javascript

El código Javascript se sitúa:

- ▶ En la etiqueta <head>
- ▶ En la etiqueta <body>
- ▶ En un archivo externo (.js)

El código Javascript se ejecuta:

- ▶ Al terminar de cargar la página.
- ▶ Ante un evento concreto (por ejemplo una pulsación del ratón)

```
<script type="text/javascript">  
  alert("Este es el mensaje de bienvenida");  
  document.write("Este es mi primer ejemplo");  
</script>
```

```
<script type="text/javascript" src="js/ej1.js"></script>
```

# Javascript

## Empezando con Javascript

Para ver los errores en el código:

- ▶ Internet Explorer:

- ▶ Herramientas -> Opciones de Internet -> Opciones Avanzadas -> “Mostrar una notificación sobre cada error de script”
- ▶ Herramientas -> Herramientas de desarrollo.

- ▶ Firefox:

- ▶ Herramientas -> Consola de error
- ▶ Desarrollador web -> Consola de errores

- ▶ Chrome:

- ▶ Herramientas -> Herramientas para desarrolladores -> Consola

# JavaScript

## Instrucciones para escribir en pantalla

Queremos que nuestro programa pueda escribir un mensaje o un resultado en la pantalla para que lo pueda ver el usuario que utiliza el programa. Para ello usaremos la siguiente instrucción.

```
document.write ("¡Hola Mundo!");
```

Algunos aspectos a tener en cuenta:

- ▶ `document.write` es el nombre de la instrucción y se debe escribir siempre tal cual, sin mayúsculas ni espacios ni ninguna otra modificación.
- ▶ El mensaje que quiero escribir se escribe dentro de los paréntesis que siguen a la instrucción.
- ▶ Si el mensaje se compone de varias partes o “trozos”, separo cada una de ellas con una coma (,).

1		JavaScript	
2	document.write("Hola, ", "¿cómo estás?");		Hola, ¿cómo estás?
3			

# JavaScript

## Instrucciones para escribir en pantalla

Otras consideraciones:

- ▶ El texto que compone un mensaje y que quiero que se escriba de manera literal debe escribirse entre comillas dobles(") o simples (').
- ▶ Si una parte del mensaje contiene solo un número, una expresión matemática, una variable o el nombre de una función NO se escribe entre comillas.
- ▶ Para escribir un salto de línea debemos utilizar el código HTML correspondiente, que es `<br>`.

```
document.write("Tengo ", 18, " años, <br> es decir, ", 18*365," días.")
```

Tengo 18 años,  
es decir, 6570 días.

# JavaScript

## Instrucciones para escribir en pantalla

Realiza los siguientes programas JavaScript a modo de ejemplo:

1. Escribe un programa que muestre el siguiente mensaje:  
*Estoy aprendiendo JavaScript y estoy muy emocionado.*
2. Modifica el programa anterior para escribir el mismo mensaje pero ahora en tres “trozos”.
3. Escribe un programa que muestre el mensaje: *El día tiene 24 horas*. Hazlo de forma que el número 24 sea un “trozo” por sí solo del mensaje.
4. Modifica el programa anterior para que además el mensaje indique cuantos segundos tiene un día.  
NOTA: No calcules previamente cuántos segundos tiene un día, para multiplicar valores en JavaScript usamos el asterisco (\*).
5. Escribe un programa que muestre el nombre de los miembros de tu familia o de un grupo de amigos. Cada nombre debe aparecer en una línea diferente.

# JavaScript

## Variables

En los programas utilizamos frecuentemente valores que, a priori, desconocemos o que podrían cambiar durante la ejecución.

Para almacenarlos y utilizarlos usamos variables. Las variables son un espacio de memoria RAM que utilizamos para nuestro programa.

Una variable se identifica por su nombre que debe comenzar por una letra y contener letras, número y el carácter guion bajo (\_).



Variable



Memoria RAM

# JavaScript

## Variables

En primer lugar debemos declarar la variable, es decir, indicar al programa que vamos a utilizar un determinado espacio de memoria y qué nombre le vamos a dar.

```
var mi_variable
```

```
var nombre, edad, direccion
```

NOTA: Aunque JavaScript lo acepta, no es recomendable utilizar caracteres no ANSI, es decir, Ñ o tildes, en los nombres de las variables.





# JavaScript

## Variables

Antes de poder utilizar una variable se le debe asignar un valor. En nuestro ejemplo sería como introducir algo en la caja.

Para asignar un valor a una variable se utiliza el signo igual (=).

```
mi_variable = 35
```

```
var nombre = "Josema", edad
```

NOTA: Cuando asignamos un texto a una variable debemos encerrarlo entre comillas dobles (") o simples(').



# JavaScript

## Variables

Para utilizar el valor de una variable basta con escribir su nombre en el programa. En nuestro ejemplo sería como mirar qué hay en la caja.

```
document.write ("Me llamo ", nombre)
```



# JavaScript

## Obtener información del usuario

En muchas ocasiones vamos a necesitar, como programadores, interactuar con el usuario de nuestro programa.

Imagina que queremos hacer un programa que salude a un usuario por su nombre. Al programarlo es imposible que yo sepa como se va a llamar la persona que lo ejecute. Además, dependiendo de quién lo haga tendrá un nombre u otro.

La solución es preguntar al usuario su nombre y guardarlo en una variable, para ello utilizamos la instrucción prompt.

`nombre = prompt (“¿Cómo te llamas?”)`



Variable para  
guardar el valor  
introducido



Instrucción  
para preguntar  
al usuario



Mensaje que se  
muestra al usuario

# JavaScript

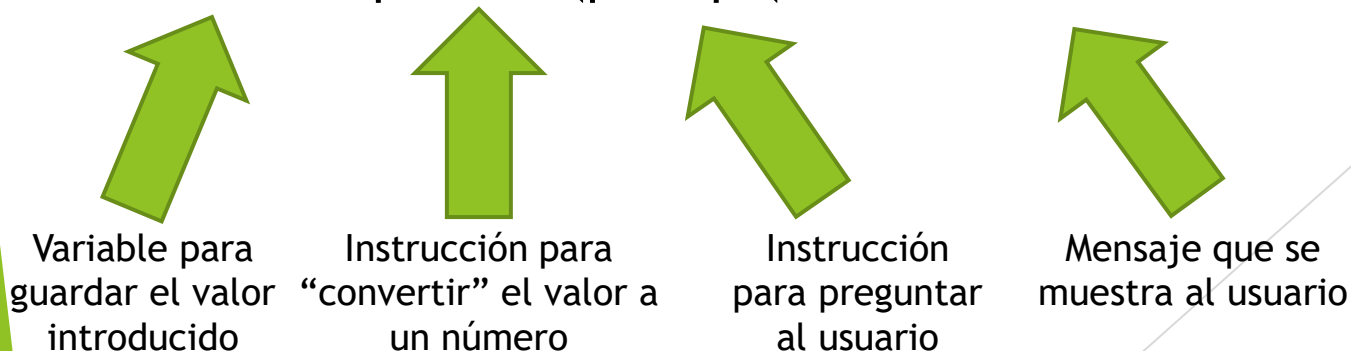
## Obtener información del usuario

Cuando ejecutamos un programa JavaScript en una página web, se interpreta que la información introducida por el usuario mediante la instrucción `prompt` es un texto.

Cuando le pedimos a un usuario un número y queremos realizar alguna operación numérica o de comparación con otro número, en ocasiones, esta interpretación que hace `prompt` nos lleva a un error en el programa.

Necesitamos indicar explícitamente que queremos que ese valor se trate en el programa como un número y para ello usamos la función:

```
numero = parseInt(prompt("Introduce un número:"))
```



# JavaScript

## Variables e introducción de datos de usuario

Realiza los siguientes programas JavaScript a modo de ejemplo:

6. Escribe un programa que pregunte su nombre al usuario, lo guarde en una variable y muestre un mensaje de saludo personalizado.  
*Hola Juan! Encantado de conocerte.*
7. Modifica el programa anterior para que el usuario además introduzca su edad y se incluya en el mensaje de saludo.  
*Hola Juan! Así que tienes 17 años, encantado de conocerte.*
8. Escribe un programa que pregunte su nombre a un usuario y lo guarde en una variable, luego le pregunte a qué se quiere dedicar y lo guarde en otra variable, y por último, muestre un mensaje como el siguiente:  
*Juan, electricista es una profesión con mucho futuro.*
9. Escribe un programa que pida al usuario que introduzca dos números y muestre un mensaje como el siguiente:  
*La suma de 4 y 8 es 12.*

# JavaScript

## Operadores matemáticos y lógicos

En JavaScript se utilizan los siguientes operadores:

- ▶ De asignación: =
- ▶ De concatenación: +
- ▶ Aritméticos:
  - ▶ Suma: +
  - ▶ Resta: -
  - ▶ Multiplicación: \*
  - ▶ División: /
  - ▶ Cociente de la división entera: \
  - ▶ Resto de la división entera: %
- ▶ De comparación:
  - ▶ Igual: ==
  - ▶ Distinto: !=
  - ▶ Menor: <
  - ▶ Mayor: >
  - ▶ Menor o igual: <=
  - ▶ Mayor o igual: >=
- ▶ Lógicos:
  - ▶ Y (and): &&
  - ▶ O (or): ||
  - ▶ No (not): !
- ▶ Incremento y decremento: ++, --

# JavaScript

## Comentarios

Los programas tienen muchas líneas de código, con multitud de instrucciones, variables, funciones y otros elementos. Mientras se escribe un programa todo se ve muy claro, pero si miramos el código fuente de un programa que escribimos hace semanas, meses, o incluso años, es muy probable que no recordemos la intención con la que escribimos el código.

Esta situación se agrava si miramos el código escrito por otra persona.

Para aclarar la intención de lo que escribimos usamos comentarios.

```
// Esto es un comentario de una línea en JavaScript
```

```
/* Esto es un comentario de
```

```
varias líneas
```

```
en JavaScript */
```

# JavaScript

## Funciones en JavaScript

- ▶ Una función es un conjunto de instrucciones a las que se asigna un nombre, devuelve o no un valor y se puede ejecutar tantas veces como se desea con solo llamarla con el nombre asignado.
- ▶ A la hora de trabajar con funciones hay que tener en cuenta dos aspectos: la creación de la función...

```
function nombreFuncion(parametro1, parametro2,..., parametroN)
{
    instrucciones;
    return valorRetorno;
}
```

... y llamada (o utilización) de la función.

```
nombreFuncion(parametro1, parametro2,..., parametroN);
```



# JavaScript

## Funciones en JavaScript

Ejemplo:

```
<script type="text/javascript">
  function suma(A,B)
  {
    var C;
    C = parseInt(A) + parseInt(B);
    return C;
  }

  document.write(suma(2,3));
</script>
```

# JavaScript

## Eventos en JavaScript

- ▶ Los eventos son las “cosas” que ocurren en una página web.
- ▶ Un evento de JavaScript por sí mismo carece de utilidad. Para que los eventos resulten útiles, se deben asociar funciones o código JavaScript a cada evento.
- ▶ Algunos eventos:
  - ▶ onclick
  - ▶ onmouseover
  - ▶ onmousemove
  - ▶ onresize

# JavaScript

## Eventos en JavaScript

- ▶ Se puede asociar un evento a una función JavaScript utilizando un atributo con el nombre del evento cuyo valor es el nombre de la función que se quiere ejecutar. Se puede hacer de dos formas:
- ▶ Con código HTML.

```
<body>
...
  <input type="button" value="Sumar" onclick="muestraResultado();"/>
...
</body>
```

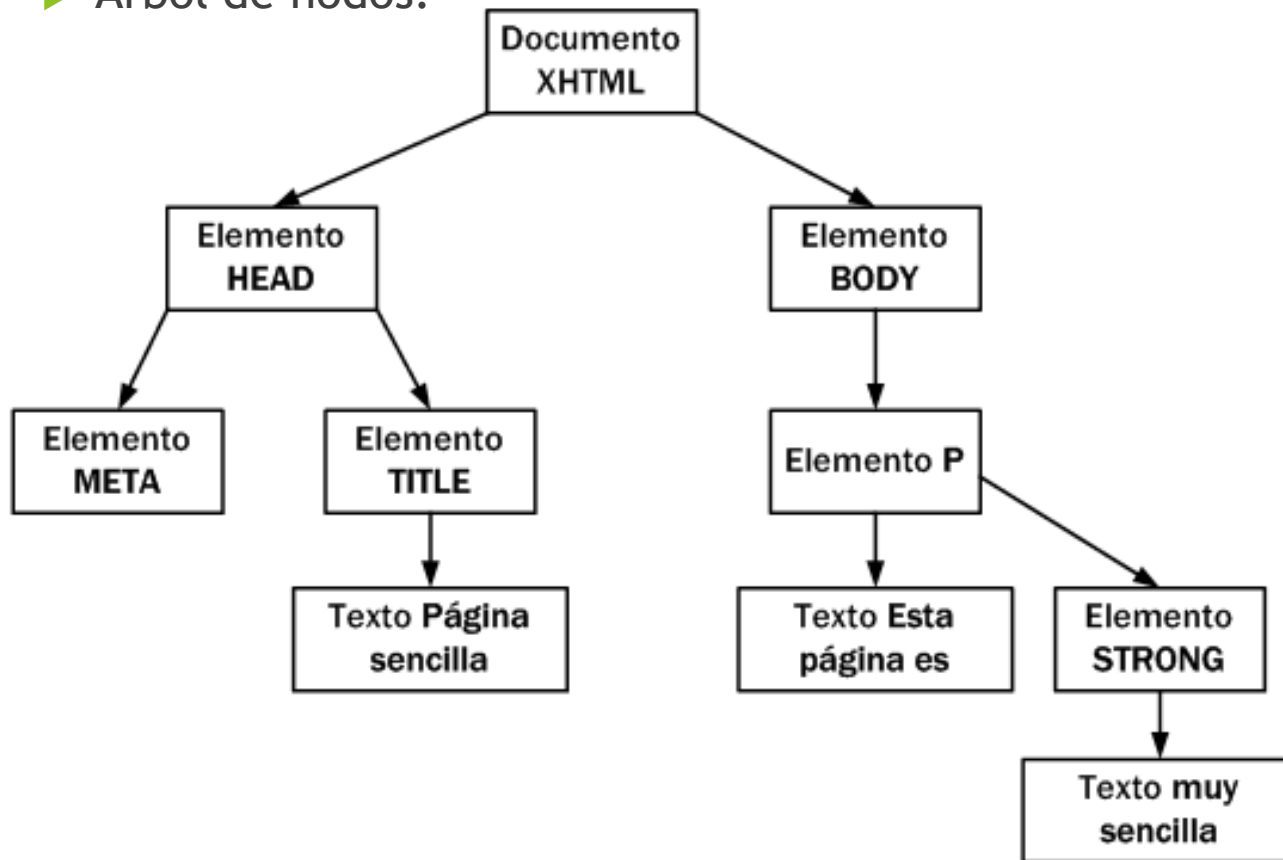
- ▶ Con código JavaScript.

```
document.getElementById("boton").onclick=muestraResultado;
```

# JavaScript

## DOM (Document Object Model)

### ► Árbol de nodos.



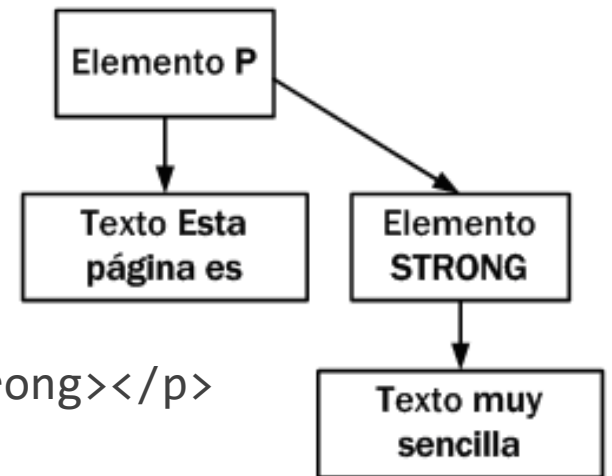
# JavaScript

## DOM (Document Object Model)

Ejemplo:

`<p>Esta página es <strong>muy sencilla</strong></p>`

- ▶ Genera los siguientes nodos:
- ▶ Nodo de tipo *"Elemento"* correspondiente a la etiqueta `<p>`.
- ▶ Nodo de tipo *"Texto"* con el contenido textual de la etiqueta `<p>`.
- ▶ Como el contenido de `<p>` incluye en su interior otra etiqueta HTML, la etiqueta interior se transforma en un nodo de tipo *"Elemento"* que representa la etiqueta `<strong>` y que deriva del nodo anterior.
- ▶ El contenido de la etiqueta `<strong>` genera a su vez otro nodo de tipo *"Texto"* que deriva del nodo generado por `<strong>`.



# JavaScript

## Acceso con JavaScript a los elementos de un formulario

- ▶ Cuando utilizamos formularios y se desea acceder a los elementos del formulario para comprobar los valores introducidos por el usuario o realizar acciones sobre ellos disponemos de una serie de propiedades.

- ▶ Para hacer referencia a un elemento concreto:

`document.nombreFormulario.nombreElemento.value`

donde:

- **document** se refiere a la página que contiene el formulario.
- **nombreFormulario** es el valor de la propiedad name de la etiqueta `<form>`.
- **nombreElemento** es el valor de la propiedad name del elemento concreto del formulario.
- **value** es la propiedad que contiene el valor de ese elemento del formulario.

# JavaScript

## Acceso con JavaScript a elementos HTML de la página

- ▶ La función `getElementById()` devuelve el elemento HTML cuyo atributo *id* coincide con el parámetro indicado en la función.
- ▶ Como el atributo *id* debe ser único para cada elemento de una misma página, la función devuelve únicamente el nodo deseado.

```
<div id="cabecera">  
  <a href="/" id="logo">...</a>  
</div>
```

```
var cabecera = document.getElementById("cabecera");
```

# JavaScript

## Acceso con JavaScript a elementos HTML de la página

- ▶ El contenido de un elemento HTML se puede obtener mediante el atributo *innerHTML*.
- ▶ Se considera el contenido de un elemento todo el código HTML contenido entre una etiqueta y su correspondiente cierre.

```
<p id="p1">
Puedes seguir este <a id="enlace" href="http://www...com">Enlace</a>
</p>
```

```
var texto= document.getElementById("p1");
alert(texto.innerHTML);

// muestra Puedes seguir este <a id="enlace"
href="http://www...com">Enlace</a>
```



# JavaScript

## Acceso con JavaScript a elementos HTML de la página

- ▶ Los atributos HTML de los elementos de la página se transforman automáticamente en propiedades de los nodos.
- ▶ Para acceder a su valor, simplemente se indica el nombre del atributo HTML detrás del nombre del nodo.

```
<a id="enlace" href="http://www...com">Enlace</a>
```

```
var enlace = document.getElementById("enlace");  
alert(enlace.href); // muestra http://www...com
```

# JavaScript

## Acceso con JavaScript a elementos HTML de la página

- ▶ Las propiedades CSS no son tan fáciles de obtener como los atributos HTML.
- ▶ Para obtener el valor de cualquier propiedad CSS del nodo, se debe utilizar el atributo *style*.
- ▶ El siguiente ejemplo obtiene el valor de la propiedad *margin* de la imagen:

```

```

```
var imagen = document.getElementById("imagen");  
alert(imagen.style.margin);
```

# JavaScript

## Acceso con JavaScript a los elementos de un formulario

- Para llamar a una función JavaScript con un enlace se escribe de la siguiente manera :

```
<a href="javascript:nombreFuncion();">Enlace.</a>
```

- Para llamar a una función JavaScript desde un botón de un formulario se utiliza el evento *onclick*.

```
<input type="button" value="TextoBoton" onclick="nombreFuncion();" />
```

# JavaScript

Acceso con JavaScript a los elementos de un formulario

Ejemplo:

```
<head>
<script type="text/javascript">
  function muestraResultado()
  {
    var num1, num2;
    num1 = document.sumador.numero1.value;
    num2 = document.sumador.numero2.value;
    alert(parseInt(num1)+parseInt(num2));
  }
</script>
</head>
<body>
<form name="sumador">
  Número 1:<input type="number" name="numero1"><br>
  Número 2:<input type="number" name="numero2"><br>
  <input type="button" value="Sumar" onclick="muestraResultado();" />
</form>
</body>
```