



Argentina  
programa  
4.0

# Introducción a Javascript

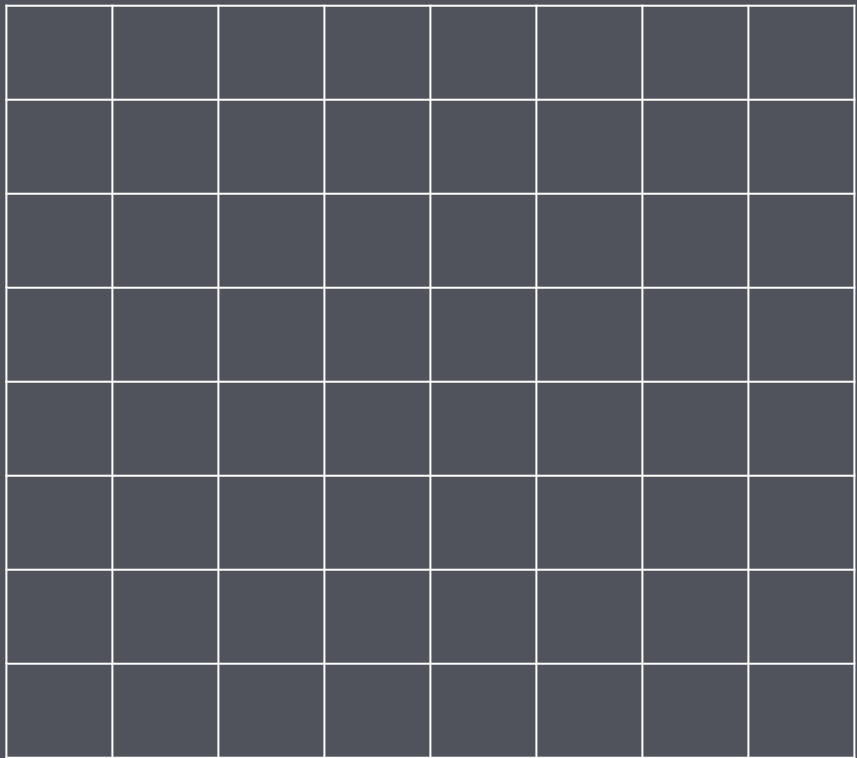
---

Programa “Introducción a la Programación Web”



# Introducción a Javascript

- Arrays en Javascript
- DOM



# Arrays

Un **Array** es una variable especial que permite almacenar **más de un valor al mismo tiempo**. En Javascript, tanto la **longitud** como el **tipo** de los elementos de un *array* son variables. Podemos agregar o quitar elementos de diferentes tipos de datos.

Importante: Recordar que los arrays se inician en la posición 0.

```
let autos = ["Civic", "Etios", 308]
```

Si queremos saber la cantidad de elementos en un array lo podemos hacer con:  
*array.length*

# Operaciones con Arrays

Añadir un elemento al final del array:

```
array.push("elemento")
```

Eliminar el último elemento de un array:

```
array.pop()
```

Añadir un elemento al comienzo del array:

```
array.unshift("elemento")
```

Eliminar el primer elemento de un array:

```
array.shift()
```

Más info sobre operaciones con arrays en [este link](#).

# Cómo recorrer un array

## Utilizando un bucle **for**:

```
var frutas = ["banana", "manzana", "tomate", "naranja"];

for(i = 0; i < frutas.length; i++){
    console.log(frutas[i]);
}
```

## ¿Y usando un **while**?

# ForEach

```
const array1 = ['a', 'b', 'c'];

array1.forEach(function(value, index) {
  console.log(value);
  console.log(index);
})
```

```
> var array1 = ['a', 'b', 'c'];
    array1.forEach(function(value, index){
      console.log(value + " " + index);
    })
a 0
b 1
c 2
```

# ¿Cómo definimos un objeto en JS?

```
var persona = {};
```

```
var persona = {  
  nombres: ['Rodrigo', 'Juan'],  
  edad: 32,  
  intereses: ['música', 'esquí']  
}
```

Para acceder a una propiedad específica lo hacemos utilizando el `.`  
Por ejemplo: ***persona.edad*** nos devolverá la edad almacenada en dicha variable.

# ¿Cómo definimos un objeto en JS?

```
var persona = {  
  nombres: ['Rodrigo', Juan],  
  edad: 32,  
  intereses: ['música', 'esquí'],  
  
  saludo: function() {  
    alert('Hola, cómo estar?');  
  }  
};
```

De la misma forma en que nosotros definimos un objeto con **propiedades y métodos**, Javascript entiende los elementos de nuestro .html de la misma manera.



# Arrays de Objetos

Como en Javascript generalmente tenemos que manipular objetos, es necesario entender que muchas veces podemos estar trabajando con Arrays en cuyas posiciones tenemos un Objeto.

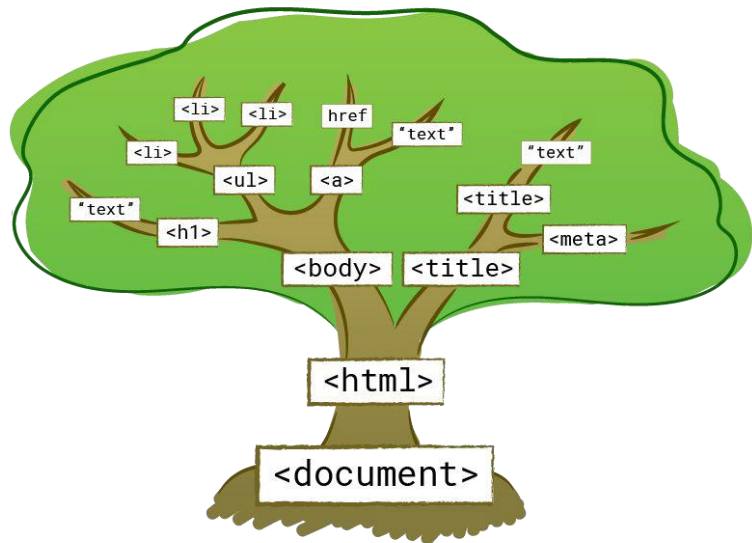
Se definen de la siguiente forma:

```
let autos = [  
  {  
    'marca': 'Honda',  
    'modelo': 'Civic',  
    'color': 'Negro',  
    'anio': 2010  
  },  
  {  
    'marca': 'VolksWagen',  
    'modelo': 'Golf',  
    'color': 'Blanco',  
    'anio': 2015  
  }  
]
```

***¿Cómo obtenemos la marca y el modelo del auto en la primera posición?***

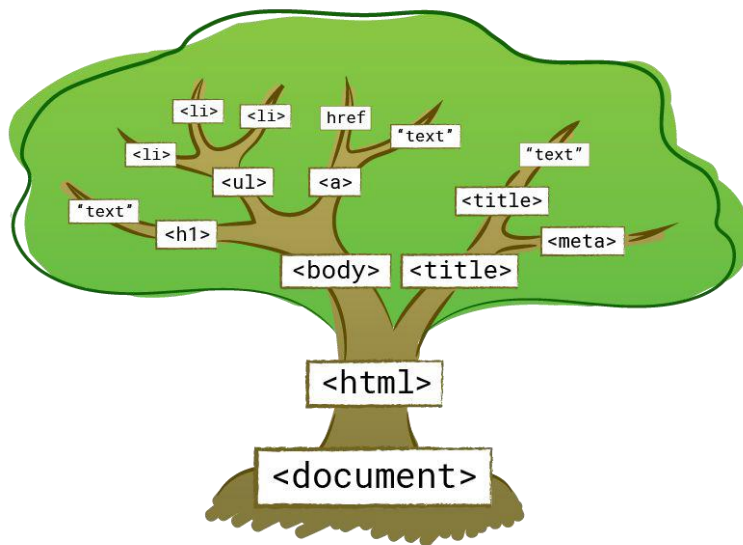
# Javascript y HTML

Javascript entiende nuestra página web como una **estructura de árbol**, donde cada elemento de nuestro .html es un **objeto** o **nodo** con determinadas **características** y **funciones asociadas**.



# DOM

## Document Object Model



# DOM y Javascript

- El DOM es una representación orientada a objetos de la página web.
- Diagrama la página web como una **estructura de árbol**, donde cada elemento de nuestro .html es un **objeto** o **nodo** con determinadas **propiedades** y **funciones asociadas**.
- Javascript utiliza esta representación de la página de manera que podamos cambiar la estructura del documento, sus estilos y contenido.

# Propiedades y Métodos

Una **propiedad** es un valor que podés **obtener** o **definir**, como por ejemplo obtener y redefinir el contenido de un elemento HTML, el color de letra aplicado a un texto, etc.

Un **método** es una acción (función) que puedes realizar como *obtener*, *agregar* o *eliminar* un elemento HTML.

De esta forma, el DOM nos proporciona de métodos y propiedades asociados a cada elemento HTML que nos permitirán manipular el documento.

En [https:// developer.mozilla.org/](https://developer.mozilla.org/) podemos encontrar propiedades y métodos que nos sirven para manejar los elementos del documento.

# Algunas propiedades y métodos

## Propiedades

```
element.innerHTML  
element.classList  
element.children  
element.id  
element.style  
element.value  
...
```

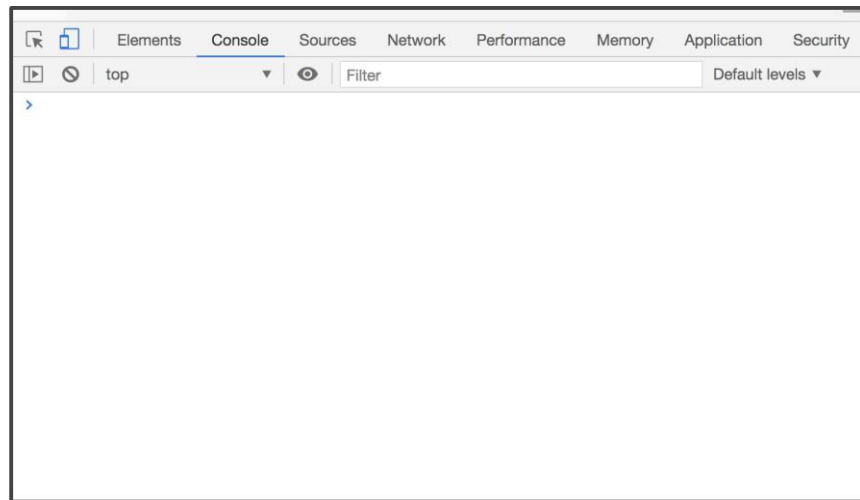
## Métodos

```
document.getElementById("")  
document.getElementsByClassName("")  
document.querySelectorAll("")  
document.getElementsByTagName("")  
element.insertCell("")  
element.appendChild()  
...
```

# Javascript en la consola del navegador

La consola permite interactuar con el contenido de una página web mediante la ejecución de sentencias de Javascript dentro del contexto de la página. En otras palabras, el navegador tiene un intérprete de JavaScript, visible a través de la consola, que permite ejecutar sentencias dentro del contexto de cada “ventana”.

Básicamente la consola brinda la capacidad de **escribir, administrar y monitorear** código JavaScript dentro del entorno de una página web.



# Practiquemos en la consola

- Abrir la consola en una página web.
- Obtener todos los elementos "<p>".
- Obtener todos los elementos "<h1>"
- Obtener un elemento por su Id. (Necesitamos conocer el id)
- Cambiarle el color de fondo a dicho elemento.
- *Obtener todos los elementos "<p>" y aplicarles un color de texto rojo.*



Como sabemos recorrer un array...

- Crear un documento html que contenga una lista `<ul>` con más de tres elementos `<li>`

Abrir el documento en el navegador. Abrir la consola y desde aquí

*Obtener todos los elementos "`<li>`" y aplicarles un color de texto rojo.*

## ¿Cómo se integra con HTML?

Aunque JavaScript se puede **incluir directamente** en un documento **HTML** (dentro de las etiquetas "script"), ocurre lo mismo que con las hojas de estilo (**CSS**): suele ser más útil como un archivo independiente vinculado a cada documento que necesite de los comportamientos definidos en él; así sólo hay que mantener unos pocos archivos .js para actualizar los comportamientos de todo un sitio.

Para ello habría que incluir en el head del documento una línea como ésta:

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

# Formularios HTML

Los formularios HTML son elementos de las páginas web que nos sirven para recuperar información del usuario y enviarla al servidor para que pueda ser procesada. La sintaxis para definir un formulario:

**`<form> ... </form>`**

# Elementos de un Formulario HTML

## El elemento <input>

El elemento <input> es el más importante y según el type puede ser por ej: "text", "password", "checkbox", "radio", "reset", "file", "hidden", "image" y "button".

```
<input type="text" name="name"/>
```

## El elemento <select>

El elemento <select> define una lista seleccionable.  
El elemento <option> define los elementos dentro del <select>. El atributo **múltiple** nos permite seleccionar más de una opción

```
<select name="colores" size="3">  
  <option value="rojo">Rojo</option>  
  <option value="azul">Azul</option>  
  <option  
value="amarillo">Amarillo</option>  
  <option value="negro">Negro</option>  
</select>
```

# Elementos de un Formulario HTML

## El elemento <textarea>

Con el <textarea> tenemos un campo en el que podemos escribir varias líneas de texto.

```
<textarea name="message" rows="10"
cols="30">
Esto es un mensaje de prueba escrito en
un textarea
</textarea>
```

## El elemento <button>

El elemento <button> define un botón cliqueable. Es importante definir siempre el type, porque es posible que los navegadores tengan comportamientos diferentes.

```
<button type="button"
onclick="alert('¡Hola
programador!')">¡Haz click!</button>
```

# Elementos de un Formulario HTML

## El elemento <datalist>

El elemento <datalist> especifica una lista predefinida para un elemento <input>

```
<form action="/action_page.php"

  oninput="x.value=parseInt(a.value)+par
seInt(b.value)">
  0
  <input type="range" id="a" name="a"
value="50">
  100 +
  <input type="number" id="b" name="b"
value="50">
  =
  <output name="x" for="a b"></output>
  <br><br>
  <input type="submit">
</form>
```

```
<form action="/action_page.php">
  <input list="browsers">
  <datalist id="browsers">
    <option value="Internet Explorer">
    <option value="Firefox">
    <option value="Chrome">
    <option value="Opera">
    <option value="Safari">
  </datalist>
</form>
```

## El elemento <output>

El elemento <output> especifica el resultado de un cálculo

# Eventos: EventListeners

Podemos utilizar Javascript para que realice ciertas funciones respondiendo a eventos determinados.

Algunos de los eventos más utilizados son:

**onclick - onchange - onload - onkeyup - onkeydown - onkeypress**

## ¿Cómo usamos estos eventos?

Una de las formas es definir un atributo en nuestro elemento html que indica a qué evento queremos que javascript responda y qué función queremos que ejecute.

```
<button type="button" name="button" onclick="cargarDatosEnTabla()">  
  Cargar datos  
</button>
```



# ¡Practiquemos un poco!

## **Ejercicio 1:**

Crear una “caja” con borde negro y color de fondo blanco. Al hacer click en un botón el color de fondo debe cambiar a rojo.

## **Ejercicio 2:**

Utilizando la misma caja, pedirle al usuario que ingrese un color (usemos el formato hexadecimal o el nombre del color en inglés) para luego cambiar el color de fondo de la caja al elegido por el usuario.

## **Ejercicio 3:**

Crear una interfaz que pida al usuario el nombre y apellido y luego lo “salude” (usemos un alert) cuando hace click en el botón “saludar”.



# ¡Practiquemos otro poco (más difícil)!

Vamos a crear un formulario que solicite al usuario algunos datos personales (nombre, dni, sexo, número telefónico, etc) y al hacer click en un botón, estos datos sean tomados y mostrados en una tabla en la misma página.

Nombre	Apellido	Sexo	DNI	Celular
--------	----------	------	-----	---------

# Referencias

- [Arrays - Javascript](#)
- [Ejemplos y Ejercitación de Arrays - Javascript](#)
- [DOM - Javascript](#)
- [Introducción a DOM - Javascript](#)
- [ForEach - Javascript](#)

---

# ¿Preguntas?

---



**Argentina  
programa  
4.0**

# Gracias!

---