

CLASE 3

PROFUNDIZACIÓN EN JAVASCRIPT: DOM, ESTRUCTURAS Y DATOS

AGENDA

1. Repaso Clase Anterior
2. Formularios Y Elementos Html
3. Dom - Document Object Model
4. Repaso De Javascript
5. Estructuras De Control
6. Objetos
7. Funciones

REPASO CLASE ANTERIOR

HTML

- Estructura del documento (en forma de árbol)
- Etiquetas con apertura y cierre
- Atributos en los elementos
- Permite anidamiento de elementos
- Metadatos

CSS

- Estilos en cascada
- Estilos en línea, en el documento y en archivos separados
- Propiedades de estilo
- Selección de elementos

EJEMPLO DE HTML Y CSS

HTML

```
<!DOCTYPE html>
<html lang="es">
  <head>
    <meta charset="UTF-8">
    <title>Clase 3 JavaScript</title>
    <link
      rel="stylesheet"
      href="estilos.css" />
  </head>
  <body>
    <h1>Título principal</h1>
    <h2 id="titulo">Título sec 1</h2>
    <p class="parrafo">Curso de JS</p>
    <a href="https://w3c.org">Link al W3C</a>
    <span><b>Curso:</b> JavaScript</span>
  </body>
</html>
```

CSS

```
h1 {
  color: #FF0000;
  font-family: 'Courier New';
}
#titulo {
  color: #0F0;
  font-size: 20px;
}
.parrafo {
  background-color: #00F;
}
```

ATRIBUTOS DE CLASE Y DE ID

CLASES

- Permite especificar una clase para varios elementos HTML
- Múltiples elementos HTML pueden compartir una clase

```
<p class="parrafo">Curso de JS</p>
```

```
.parrafo {  
  background-color: "#00F";  
}
```

IDS

- Permite especificar un ID único para un elemento HTML
- Un solo elemento HTML puede tener un ID

```
<h2 id="titulo">Título sec 1</h2>
```

```
#titulo {  
  color: "#0F0";  
  font-size: 20px;  
}
```

FORMULARIOS Y ELEMENTOS HTML

Un formulario es una colección de campos de entrada del usuario.

El elemento que define un formulario es `<form>`
`</form>`.

Los elementos de un formulario son los campos de entrada, como `<input />`, `<textarea>` o `<select>`.

Estos elementos se envían a un servidor para su procesamiento.

```
<form>  
  <!-- Elementos del formulario -->  
</form>
```

ELEMENTOS DE LOS FORMS

Hay varios elementos que se pueden usar en un formulario, y dependiendo de sus atributos, mostrarse con distintas formas y funcionalidades.

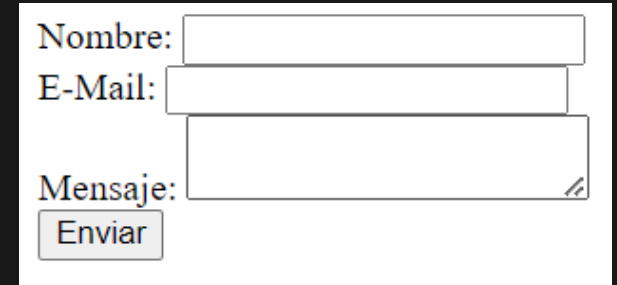
```
<input>  
<textarea></textarea>  
<select></select>  
<button></button>  
<label></label>
```

EJEMPLO DE UN FORMULARIO

HTML

```
<form>
  <label for="name">Nombre: </label>
  <input type="text" name="name" />
  <br />
  <label for="email">E-Mail: </label>
  <input type="email" name="email" />
  <br />
  <label for="message">Mensaje: </label>
  <textarea name="message"></textarea>
  <br />
  <button type="submit">Enviar</button>
</form>
```

NAVEGADOR



Visual representation of the HTML form in a browser. It shows three input fields: a text box for 'Nombre:', an email box for 'E-Mail:', and a text area for 'Mensaje:'. Below the text area is a button labeled 'Enviar'.

ELEMENTO INPUT ¹

El `<input>` puede tomar una variedad de tipos, especificados mediante el atributo `type`, que determinan la forma en que el usuario puede interactuar con él y el tipo de datos que el usuario puede ingresar.

Además provee validación del dato ingresado dependiendo de su tipo.

```
<input type="text" id="nombre" name="nombre">
```

ELEMENTO INPUT ²

Algunos de los tipos más comunes incluyen:

- **text**: Un campo de texto simple para ingresar texto.
- **password**: Un campo de texto que oculta los caracteres ingresados.
- **checkbox**: Una casilla que el usuario puede marcar o desmarcar.
- **radio**: Un botón de radio que permite seleccionar una sola opción de un conjunto.
- **submit**: Un botón que envía el formulario.
- **email**: Un campo de texto optimizado para direcciones de correo electrónico.
- **number**: Un campo para ingresar números, que incluye controles para aumentar o disminuir el valor.

ELEMENTO INPUT ³

Con la introducción de HTML5 y sus sucesivas actualizaciones, se han añadido varios nuevos tipos de input para formularios, ampliando significativamente las capacidades de interacción y entrada de datos en las páginas web.

La compatibilidad varía entre navegadores; es esencial probar, utilizar validadores y, si es necesario, implementar soluciones alternativas o polyfills.

ELEMENTO INPUT ⁴

Algunos de los tipos más avanzados incluyen:

- `color`: Seleccionar un color de una paleta.
- `date`: Selección de fechas.
- `range`: Valor numérico dentro de un rango.
- `tel`: Ingresar teléfono con formato válido.
- `time`: Seleccionar una hora.
- `url`: Ingresar una URL con formato válido.

DOM - DOCUMENT OBJECT MODEL

Es un estándar de W3C.

Representa la estructura del documento HTML (página Web) en memoria.

Funciona de interfaz para documentos HTML y XML.

De esta manera es posible manipularla a través de scripts.

DOM: ESTRUCTURA

Es una estructura jerárquica de nodos en forma de árbol.

Cada rama termina en nodos que contienen los objetos.

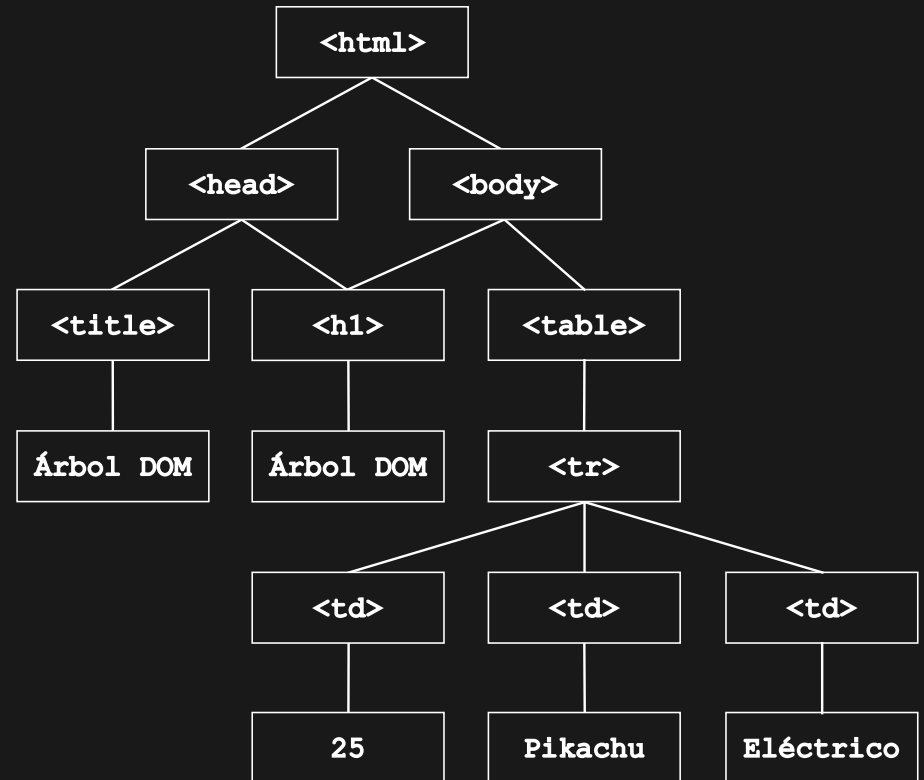
Los nodos pueden poseer eventos que al suceder es atendido por el manejador de eventos.

ÁRBOL DOM ¹

Cuando se carga una página Web, el navegador crea un Document Object Model de la página, en forma de árbol.

ÁRBOL DOM ²

```
<html>
  <head>
    <title>Árbol DOM</title>
  </head>
  <body>
    <h1>Árbol DOM</h1>
    <table>
      <tr>
        <td>25</td>
        <td>Pikachu</td>
        <td>Eléctrico</td>
      </tr>
    </table>
  </body>
</html>
```



COMPONENTES DEL DOM

- **Objetos:** los elementos HTML.
- **Propiedades:** atributos de los elementos.
- **Métodos:** acciones que se pueden realizar sobre los elementos.
- **Eventos:** reacciones a acciones específicas dentro de la página.

COMPONENTES DEL DOM: EJEMPLO

```
<!DOCTYPE html>
<html>
  <head>
    <title>Ejemplos de DOM</title>
  </head>
  <body>
    <h1>DOM</h1>
    <p id="parrafo">Este es un párrafo</p>
    <script>
      const parrafo = document.getElementById('parrafo') // Método
      console.log(parrafo) // Se imprime el objeto
      parrafo.innerHTML = 'Este es un nuevo párrafo' // Propiedad
      parrafo.addEventListener('click', () => {} }) // Evento
    </script>
  </body>
</html>
```

OBJETOS DEL DOM ¹

Se pueden seleccionar y manipular por propiedades, métodos y eventos.

Se implementa a través de APIs que funcionan, como `document`, `element`, `node`, `event`, entre otros.

Es independiente al lenguaje de programación.

OBJETOS DEL DOM ²

// JavaScript

```
const parrafos = document.getElementsByTagName('p')
const primerParrafo = parrafos[0]
console.log(primerParrafo.nodeName)
primerParrafo.nodeValue = 'Nuevo Contenido'
```

Python

```
from xml.dom import minidom
doc = minidom.parse('page.html')

parrafos = doc.getElementsByTagName('p')
primer_parrafo = parrafos[0]
print(primer_parrafo.nodeName)
primer_parrafo.nodeValue = 'Nuevo Contenido'
```

REPASO DE JAVASCRIPT

- Fue creado por Brendan Eich, cofundador del proyecto Mozilla.
- Javascript es un lenguaje de interpretado o con compilación just-in-time.
- Es dinámico, que permite la construcción de objetos a partir de prototipos.
- Es Multiparadigma: objetos, imperativa y declarativa.
- Su estándar es ECMAScript.

VARIABLES

- Son case sensitive y soportan Unicode.
- Tipado dinámico.
- `var`: variables locales y globales. Reemplazada por `let` y `const`.
- `let`: variables con un alcance limitado al bloque, declaración o expresión donde se usa.
- `const`: similar a `let`, pero para declarar variables cuyo valor no se pretende cambiar (constantes).

TIPOS DE DATOS

- **Números:** Para representar tanto enteros como flotantes.
- **Strings:** Para representar texto.
- **Booleanos:** true o false.
- **Objetos:** Para agrupar datos y funciones relacionadas.
- **Arreglos:** Para representar listados.
- **Undefined y null:** Para variables sin valor o con un valor nulo.

OPERADORES DE COMPARACIÓN

- `==`: Igualdad
- `===`: Igualdad estricta
- `!=`: Desigualdad
- `!==`: Desigualdad estricta
- `<`: Menor que
- `<=`: Menor o igual que
- `>`: Mayor que
- `>=`: Mayor o igual que

COMENTARIOS

```
// Comentario de línea
```

```
/*  
 * Comentario de bloque  
 * No es necesario el asterisco, mejoran la legibilidad  
 */
```

ESTRUCTURAS DE CONTROL

- Estructuras condicionales: `if`, `if/else`, operador ternario, `switch`.
- Estructuras iterativas: `while`, `for`, `do..while`, `for .. in`, `for .. of`.

ESTRUCTURAS CONDICIONALES ¹

- **if**: Condicional simple.
- **if/else**: Condicional con alternativa.
- **operador ternario**: Condicional en una sola línea.
- **switch**: Múltiples condicionales.

ESTRUCTURAS CONDICIONALES ²

```
// if and if/else
if (name === "Palpatine") {
  console.log("Did your ever hear the tragedy of Darth Plagueis the Wise")
} else if (name === "Obi-Wan Kenobi") {
  console.log("You were the chosen one")
} else {
  console.log("Let the force be with you")
}

// Operador ternario
const bladeColor = (name === "Palpatine") ? "red" : "blue/green/yellow"
```

ESTRUCTURAS CONDICIONALES ³

```
let bladeColor;
switch (character) {
  case 'Luke Skywalker':
    bladeColor = 'green'
    break
  case 'Obi-Wan Kenobi':
    bladeColor = 'blue'
    break
  case 'Mace Windu':
    bladeColor = 'purple'
    break
  default:
    bladeColor = 'unknown'
    break
}
```

ESTRUCTURAS ITERATIVAS ¹

- **while**: Bucle con condición de fin.
- **for**: Bucle iterativo.
- **do .. while**: Bucle con condición de continuación.
- **for .. in**: Iterar sobre elementos.
- **for .. of**: Iterar sobre valores de un objeto.

ESTRUCTURAS ITERATIVAS ²

```
let n = 0
while (n < 5) {
  console.log(n)
  n++
}
```

```
for (let i = 0; i < 5; i++) {
  console.log(i)
}
```

```
let j = 0
do {
  console.log(j)
  j++
} while (j < 5)
```


ESTRUCTURAS ITERATIVAS ³

```
const objeto = {a: 1, b: 2, c: 3}
for (let clave in objeto) {
  console.log(`clave: ${clave}, valor: ${objeto[clave]}`)
}

const array = ['a', 'b', 'c']
for (let valor of array) {
  console.log(valor)
}
```

OBJETOS

Un objeto es una tipo de dato que dentro una variable puede tener otras variables y funciones.

Es el tipo de datos `object`.

```
const objeto = new Object() // constructor new
const objeto = {}           // Esto es un objeto vacío.
const miObjeto = {
  nombre: 'Leia Organa',
  role: 'general',
  color: function() { return "blue" },
}
```

CARACTERÍSTICAS

Posee propiedades, que son las variables almacenadas dentro.

Incluye una colección de funciones relacionadas, denominadas métodos.

Todos los objetos heredan métodos de `Object.prototype`, como `toString()`, y tipos específicos como los números también tienen métodos propios como `toFixed()` en `Number.prototype`.

MÉTODOS HEREDADOS

```
const longitud = 13.5;  
longitud.toString();           // Devuelve "13.5" (Método Object)  
longitud.toLocaleString();     // Devuelve "13,5" (Método Object)  
longitud.toFixed(3);           // Devuelve "13.500" (Método Number)
```

FUNCIONES

Son bloques de código reutilizable.

Existen varias maneras de definir funciones, sin embargo ahora vamos a ver una forma simple.

Puede recibir parámetros y retornar un valor.

FUNCIONES: EJEMPLO

```
function miFuncion() {  
    alert('Hola mundo')  
}  
  
function suma(n1, n2) {  
    return n1 + n2  
}  
  
miFuncion()  
const valor = suma(1, 2)
```

FIN DE LA CLASE