
UTE
EC
Xtratego
ACADEMY

UTE EC | Xtratego ACADEMY

CURSO
DESARROLLO FRONT END

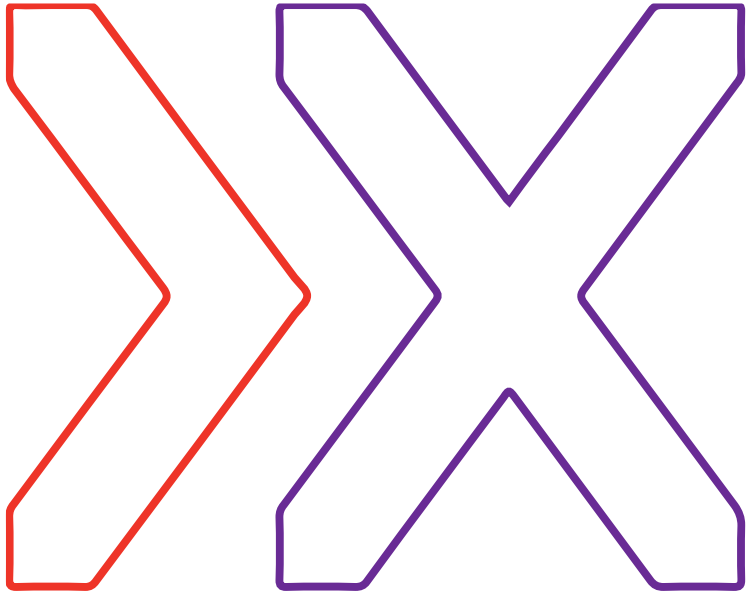
INTRO

¿Qué es el Frontend?

El **Frontend** es la parte del desarrollo web que se encarga de todo lo que el usuario ve y con lo que interactúa directamente en una página o aplicación web.

PARTE VISUAL

https://www.menti.com/alhqdgsrk6yr



Unidad 1

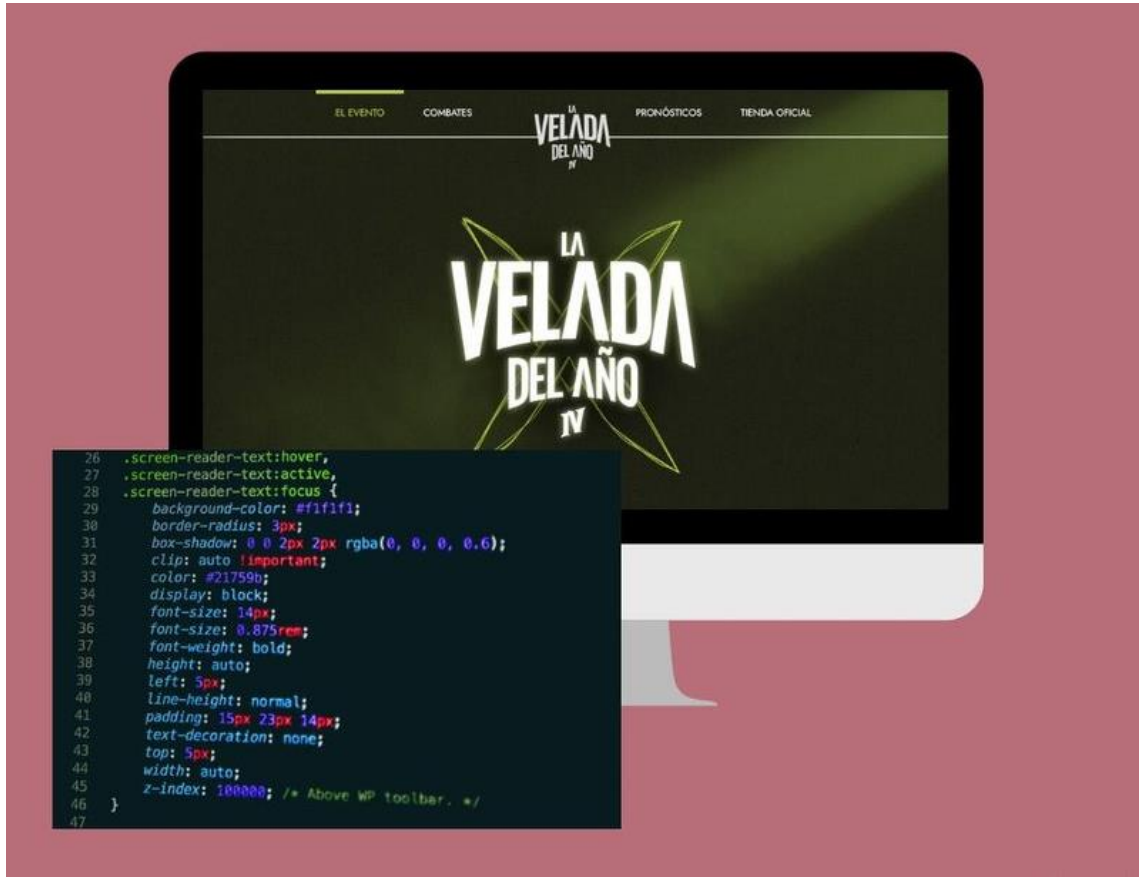
Fundamentos de JavaScript

Que es JavaScript?

Es un lenguaje de programación esencial para el desarrollo web, permitiendo la creación de contenido dinámico e interactivo en las páginas web

Comparación Páginas Web

Estáticas



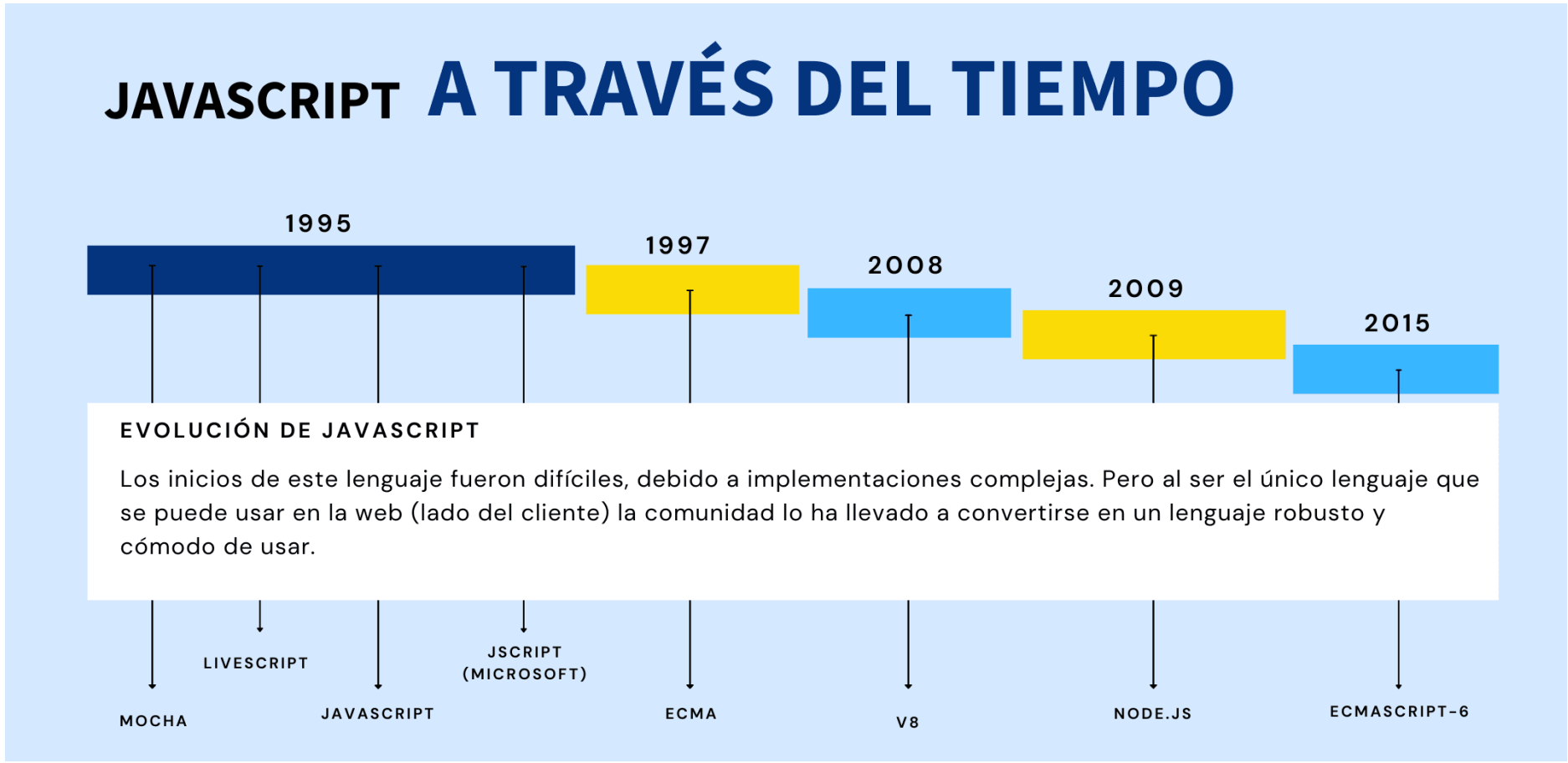
Dinámicas

Origen y Evolución de JavaScript

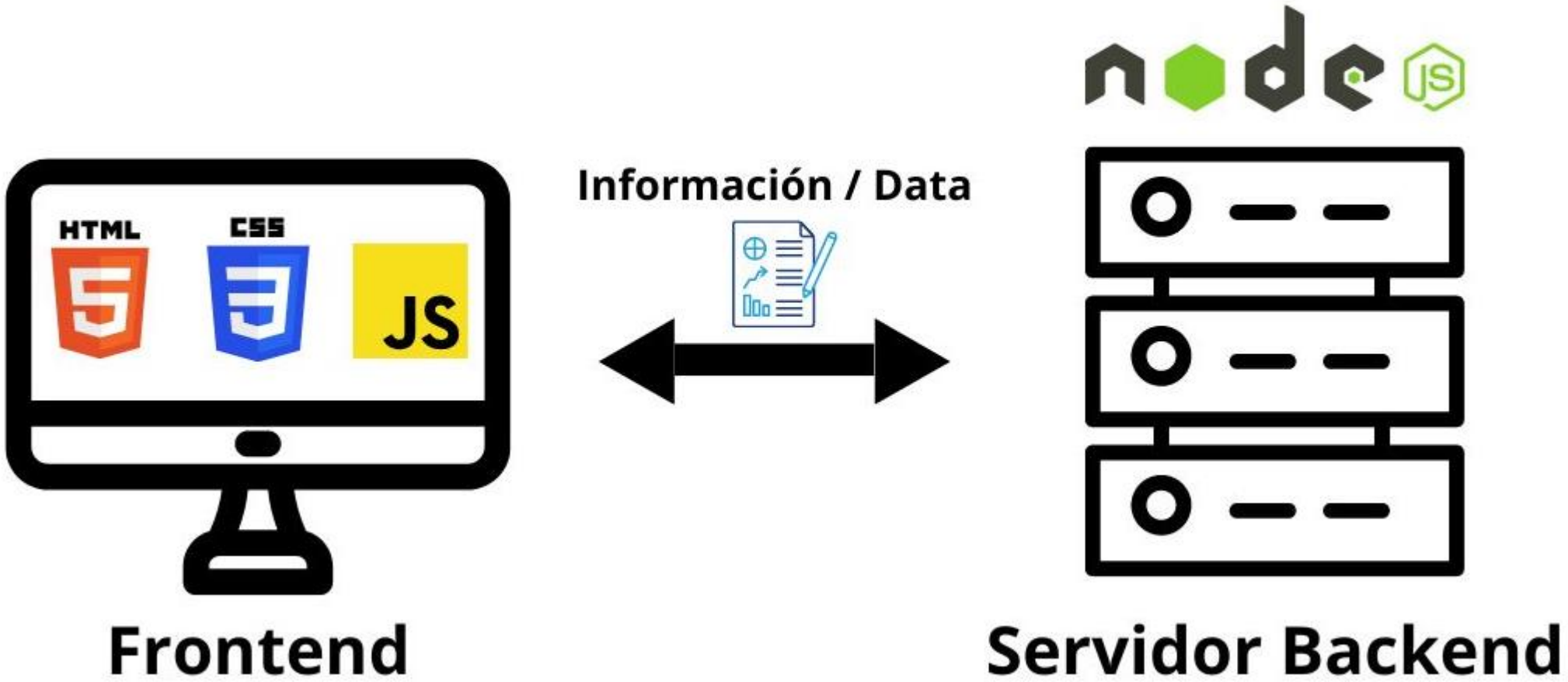
JavaScript fue creado por Brendan Eich, inspirado por otros lenguajes como Java, Scheme y Self. El propósito de JavaScript fue dar solución a la problemática de las páginas estáticas.

HolaMundo

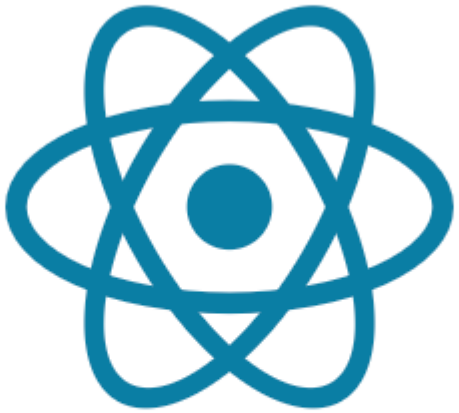
Evolución de JavaScript



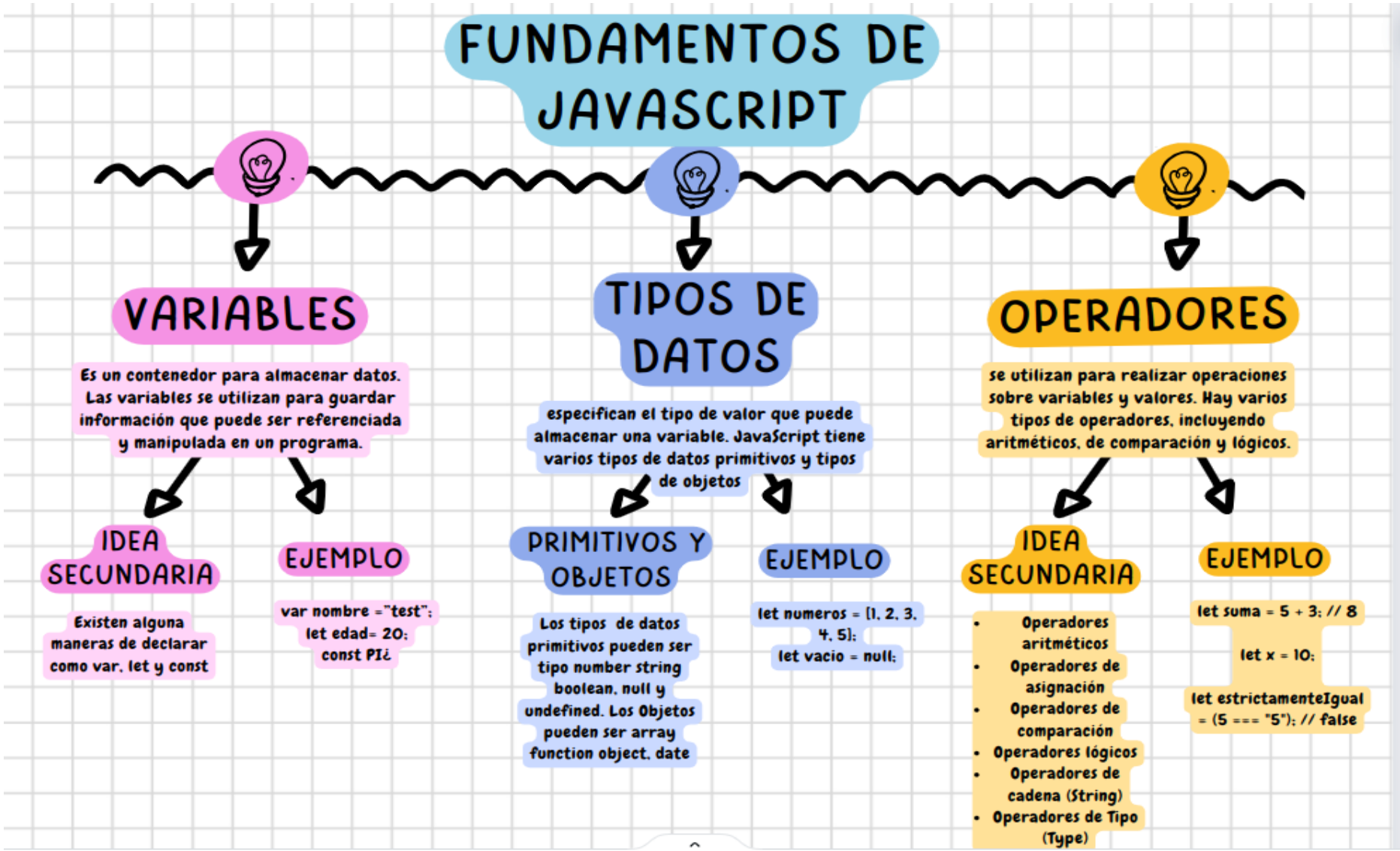
JavaScript en la actualidad



JavaScript en la actualidad



React


FUNDAMENTOS DE JAVASCRIPT
VARIABLES
Es un contenedor para almacenar datos. Las variables se utilizan para guardar información que puede ser referenciada y manipulada en un programa.
IDEA SECUNDARIA
Existen alguna maneras de declarar como var, let y const
EJEMPLO
var nombre ="test";
let edad= 20;
const PI¿
TIPOS DE DATOS
especifican el tipo de valor que puede almacenar una variable. JavaScript tiene varios tipos de datos primitivos y tipos de objetos
PRIMITIVOS Y OBJETOS
Los tipos de datos primitivos pueden ser tipo number string boolean, null y undefined. Los Objetos pueden ser array function object, date
EJEMPLO
let numeros = [1, 2, 3, 4, 5];
let vacio = null;
OPERADORES
se utilizan para realizar operaciones sobre variables y valores. Hay varios tipos de operadores, incluyendo aritméticos, de comparación y lógicos.
IDEA SECUNDARIA
Operadores aritméticos
Operadores de asignación
Operadores de comparación
Operadores lógicos
Operadores de cadena (String)
Operadores de Tipo (Type)
EJEMPLO
let suma = 5 + 3; // 8
let x = 10;
let estrictamenteIgual = (5 === "5"); // false

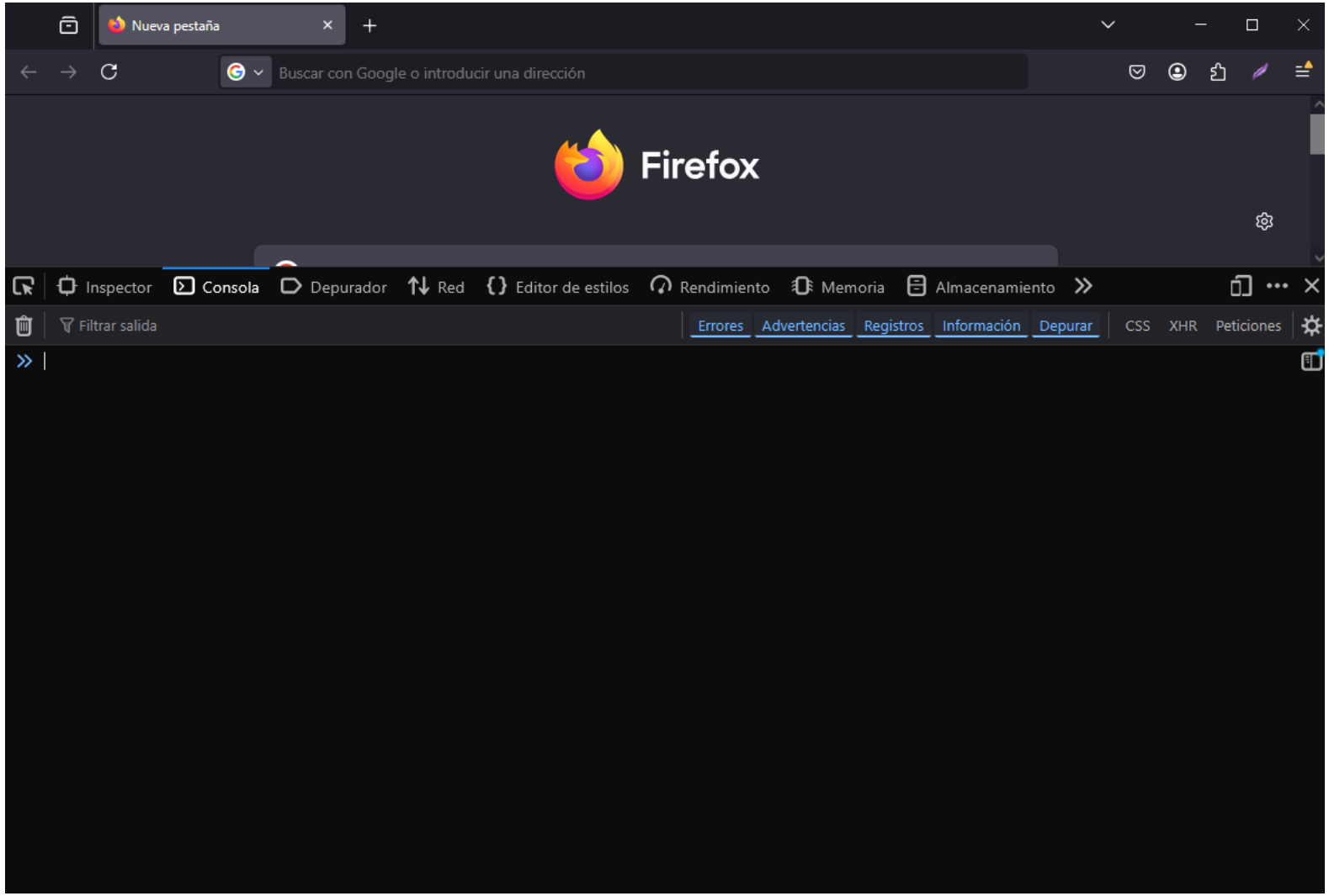
Variables, tipos de datos y operadores

Una variable es un contenedor que se utiliza para almacenar datos que pueden ser referenciados y manipulados a lo largo del programa. Las variables permiten a los desarrolladores guardar valores y reutilizarlos, lo que hace que el código sea más flexible y fácil de mantener.

Ambiente JavaScript en Navegador

Navegador
Mozilla Firefox

Abrir Consola
F12



Declarar una variable en JavaScript

Se utilizan las palabras clave **var, let, o const.**

Ejemplo

```
var nombre = "Juan";
console.log(nombre); // Output: Juan
```

Ejemplo

```
let edad = 30;
console.log(edad); // Output: 30
```

Ejemplo

```
const PI = 3.1416;
console.log(PI); // Output: 3.1416
```

Declarar una variable en JavaScript

Se utilizan las palabras clave **var, let, o const.**

Ejemplo

```
let sinValor;
console.log(sinValor); // Output: undefined
```

Ejemplo

```
let saludo = "Hola, mundo";
console.log(saludo); // Output: Hola, mundo
```

Tipos de Datos en JavaScript

Tipo de Dato	Descripción	Ejemplos
Number	Representa números enteros y de punto flotante.	`let entero = 42;` `let decimal = 3.14;`
String	Representa cadenas de texto.	`let entero = 42;` `let decimal = 3.14;`
Boolean	Representa valores verdaderos o falsos.	`let esVerdadero = true;` `let esFalso = false;`
Null	Representa un valor único e inmutable.	`let vacio = null;`
Undefined	Representa un valor único e inmutable.	`let indefinido;` `console.log(indefinido); // undefined`
Symbol	Representa un valor único e inmutable.	`let simbolo = Symbol('descripcion');`

Tipos de Datos de Objeto en JavaScript

Tipo de Dato	Descripción	Ejemplos
Object	Representa una colección de propiedades clave-valor.	`let persona = { nombre: "Ana", edad: 25 };`
Array	Representa una lista ordenada de valores.	`let numeros = [1, 2, 3, 4, 5];`

Function	Representa una lista ordenada de valores.	`function saludar() {`
		`  console.log("Hola");`
		`}`

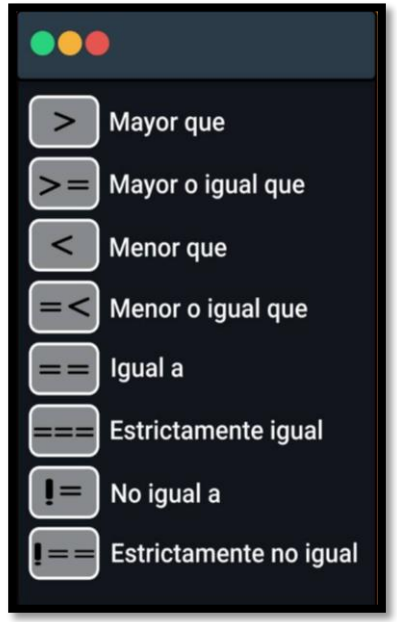
Date	Representa una fecha y hora.	`let hoy = new Date();`
RegExp	Representa una expresión regular.	`let expresion = /ab+c/;`
Map	Representa una expresión regular.	`let mapa = new Map();` `mapa.set('clave', 'valor');`
Set	Representa una expresión regular.	`let conjunto = new Set();` `conjunto.add(1);`

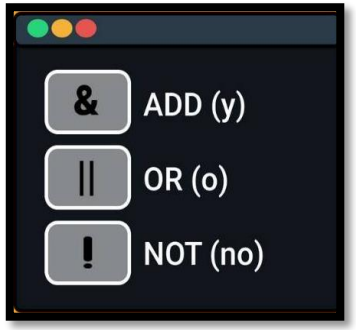
Operadores

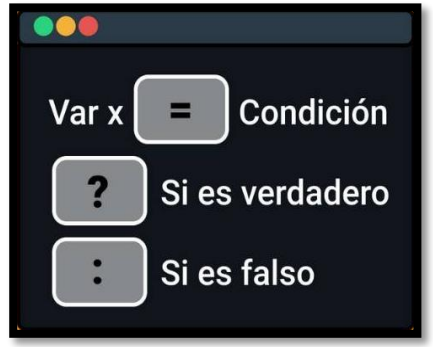
Los operadores en JavaScript son herramientas esenciales que permiten manipular datos, realizar comparaciones, ejecutar operaciones lógicas y trabajar con cadenas de texto y tipos de datos.

Clasificación de los operadores

- Operadores aritméticos
- Operadores de asignación
- Operadores de comparación
- Operadores lógicos
- Operadores de cadena
- Operadores de Tipo







Operadores Aritméticos

Ejemplo

```
let suma = 5 + 3; // 8
```

```
let resta = 5 - 3; // 2
```

```
let multiplicacion = 5 * 3; // 15
```

```
let division = 6 / 3; // 2
```

```
let modulo = 5 % 2; // 1
```

Operadores Aritméticos

Ejemplo

```
let incremento = 5;
incremento++; // 6
```

```
let decremento = 5;
 decremento--; // 4
```

Operadores de asignación

Ejemplo

```
let x = 10;
```

```
let y = 5;
y += 3; // y = y + 3; y ahora es 8
```

```
let z = 5;
z -= 2; // z = z - 2; z ahora es 3
```

```
let a = 5;
a *= 2; // a = a * 2; a ahora es 10
```

```
let b = 10;
b /= 2; // b = b / 2; b ahora es 5
```

```
let c = 10;
c %= 3; // c = c % 3; c ahora es 1
```

Operadores de Comparación

Ejemplo

```
let igual = (5 == "5"); // true
```

```
let estrictamenteIgual = (5 === "5"); // false
```

```
let desigual = (5 != "5"); // false
```

```
let estrictamenteDesigual = (5 !== "5"); // true
```

Operadores de Comparación

Ejemplo

```
let mayor = (5 > 3); // true
```

```
let mayorOIgual = (5 >= 5); // true
```

```
let menor = (3 < 5); // true
```

```
let menorOIgual = (3 <= 5); // true
```

Operadores Lógicos

Ejemplo

```
let and = (true && false); // false
```

```
let or = (true || false); // true
```

```
let not = (!true); // false
```

```
let ternario = 5 > 9 ? 5 : 0; // condición es falsa entonces ternario = 0
```

Operadores de Cadena

Ejemplo

```
let saludo = "Hola" + " " + "mundo"; // "Hola mundo"
```

```
let texto = "Hola";
texto += " mundo"; // "Hola mundo"
```

Operadores de Tipo de Dato

Ejemplo

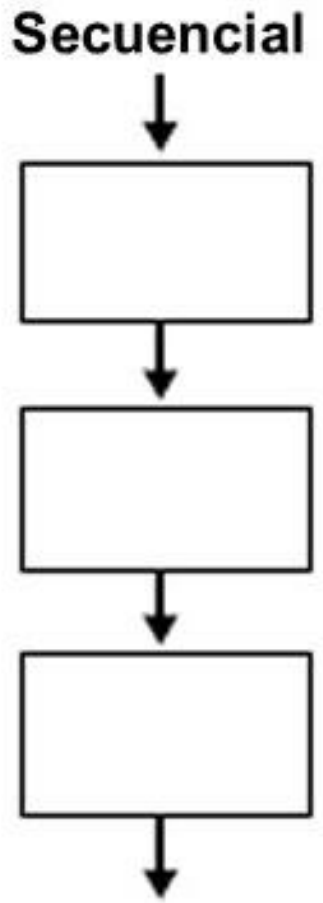
```
let tipo = typeof 5; // "number"
```

```
let fecha = new Date();
let esFecha = fecha instanceof Date; // true
```

```
let dias = new Map();
let esMapaDias = dias instanceof Date; // false
```

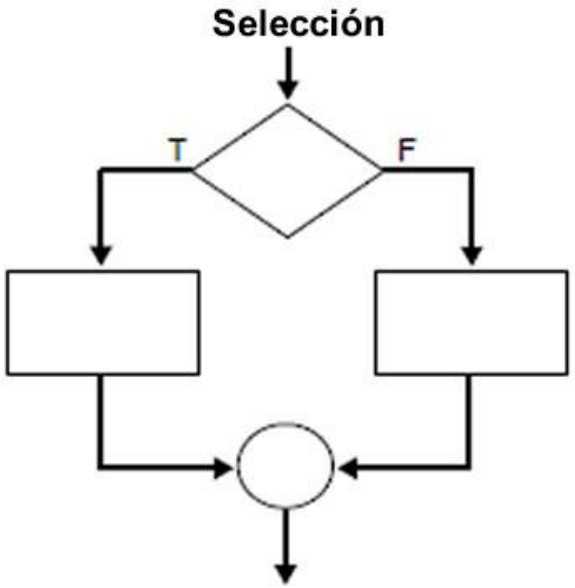
Estructuras de control

Las estructuras de control son esenciales para dirigir el flujo de ejecución del código. Permiten tomar decisiones, ejecutar bloques de código repetidamente y manejar la ejecución condicional



Estructuras Condicionales

if / else if / else




```
if (condicion) {
  // Código a ejecutar si la condición es verdadera
} else if (otraCondicion) {
  // Código a ejecutar si otra condición es verdadera
} else {
  // Código a ejecutar si ninguna de las condiciones anteriores es verdadera
}
```

Estructuras Condicionales
if / else if / else

Ejercicio

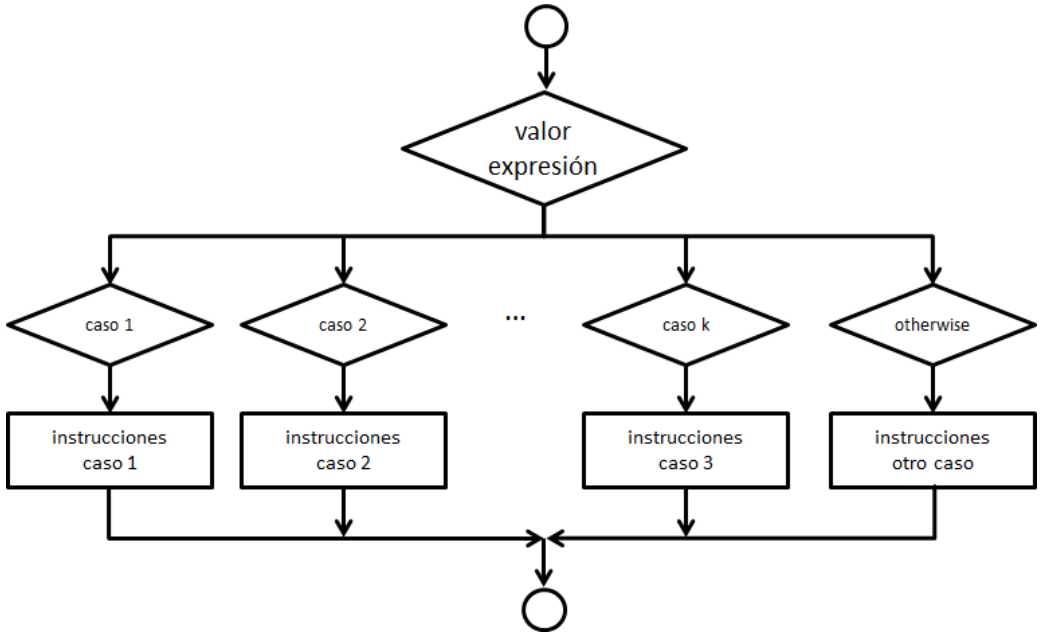
Un cine ofrece distintos descuentos según la edad del cliente y el día de la semana. El sistema debe calcular el precio del boleto según las siguientes condiciones:
El precio base del boleto es de $10.

1. Si el cliente tiene menos de 12 años:
    Si es lunes o martes, paga $4.
    En cualquier otro día, paga $5.
2. Si el cliente tiene entre 12 y 64 años:
    Si es miércoles, tiene un descuento del 20%.
    Si es estudiante y es jueves, paga $6.
    En cualquier otro caso, paga el precio completo ($10).
3. Si el cliente tiene 65 años o más:
    Siempre paga $5, sin importar el día.

https://www.programiz.com/online-compiler/0hIMdb4ZmqZ0f

Estructuras Condicionales

switch




```
let dia = "lunes";
switch (dia) {
  case "lunes":
    console.log("Hoy es lunes");
    break;
  case "martes":
    console.log("Hoy es martes");
    break;
  default:
    console.log("No es lunes ni martes");
}
```

Estructuras Condicionales switch

Ejercicio

Una máquina expendedora ofrece diferentes productos numerados del 1 al 5. Cada número representa un producto distinto. El programa debe mostrar el nombre del producto seleccionado y su precio. Si el usuario ingresa un número fuera del rango, se le debe indicar que la opción no es válida.
Los productos son:

1. Agua - $1.00
2. Gaseosa - $1.50
3. Jugo - $2.00
4. Café - $2.50
5. Té - $2.00

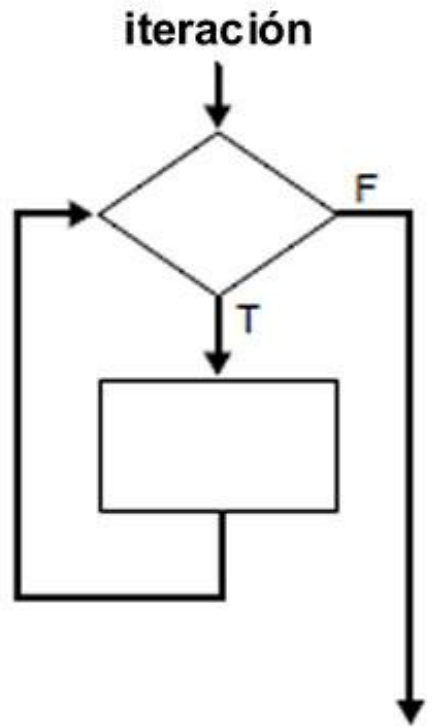
https://www.programiz.com/online-compiler/6EGco9G2nkL5P

Estructuras de Bucle

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

```
let i = 0;
while (i < 5) {
  console.log("Iteración número: " + i);
  i++;
}
```

```
let i = 0;
do {
  console.log("Iteración número: " + i);
  i++;
} while (i < 5);
```



Estructuras de Bucle for

Ejercicio

Se necesita un programa que imprima la tabla de multiplicar de un número ingresado por el usuario, y el limite de hasta cuando se debe de generar la Tabla.

Ejemplo

5 x 1 = 5
5 x 2 = 10
5 x 3 = 15
5 x 4 = 20
5 x 5 = 25

https://www.programiz.com/online-compiler/3TjAwDHIOWKQp

Estructuras de Bucle while

Ejercicio

Se quiere simular un sistema de login en consola que pida al usuario una contraseña. El sistema debe seguir pidiendo la contraseña infinitamente hasta que el usuario ingrese la correcta.
La contraseña correcta es **"secreto123"**

https://www.programiz.com/online-compiler/3rlox8ObQ4RY8

Estructuras de Control de Salto

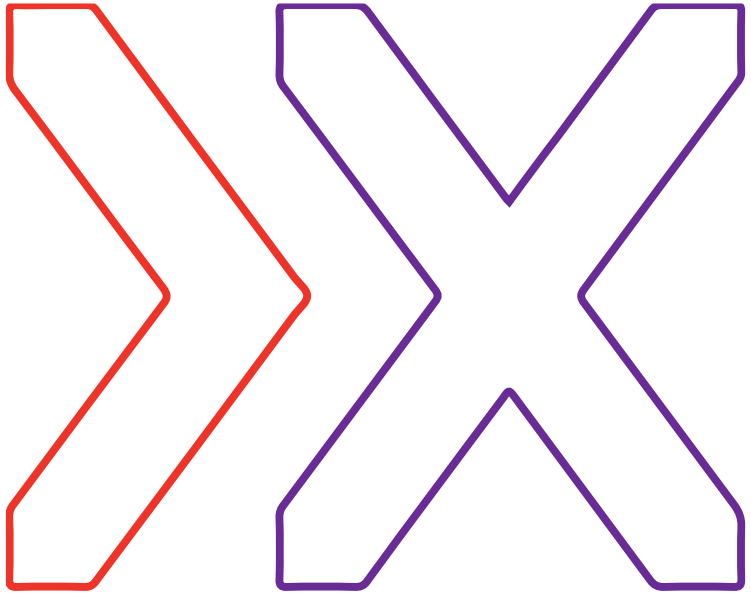
break

```
for (let i = 0; i < 10; i++) {
  if (i === 5) {
    break; // Sale del bucle cuando i es igual a 5
  }
  console.log(i);
}
```

continue

```
for (let i = 0; i < 10; i++) {
  if (i === 5) {
    continue; // Salta la iteración cuando i es igual a 5
  }
  console.log(i);
}
```

Ejemplo: https://www.programiz.com/online-compiler/3Hd8uB51nL2aB



Unidad 2

Funciones en JavaScript

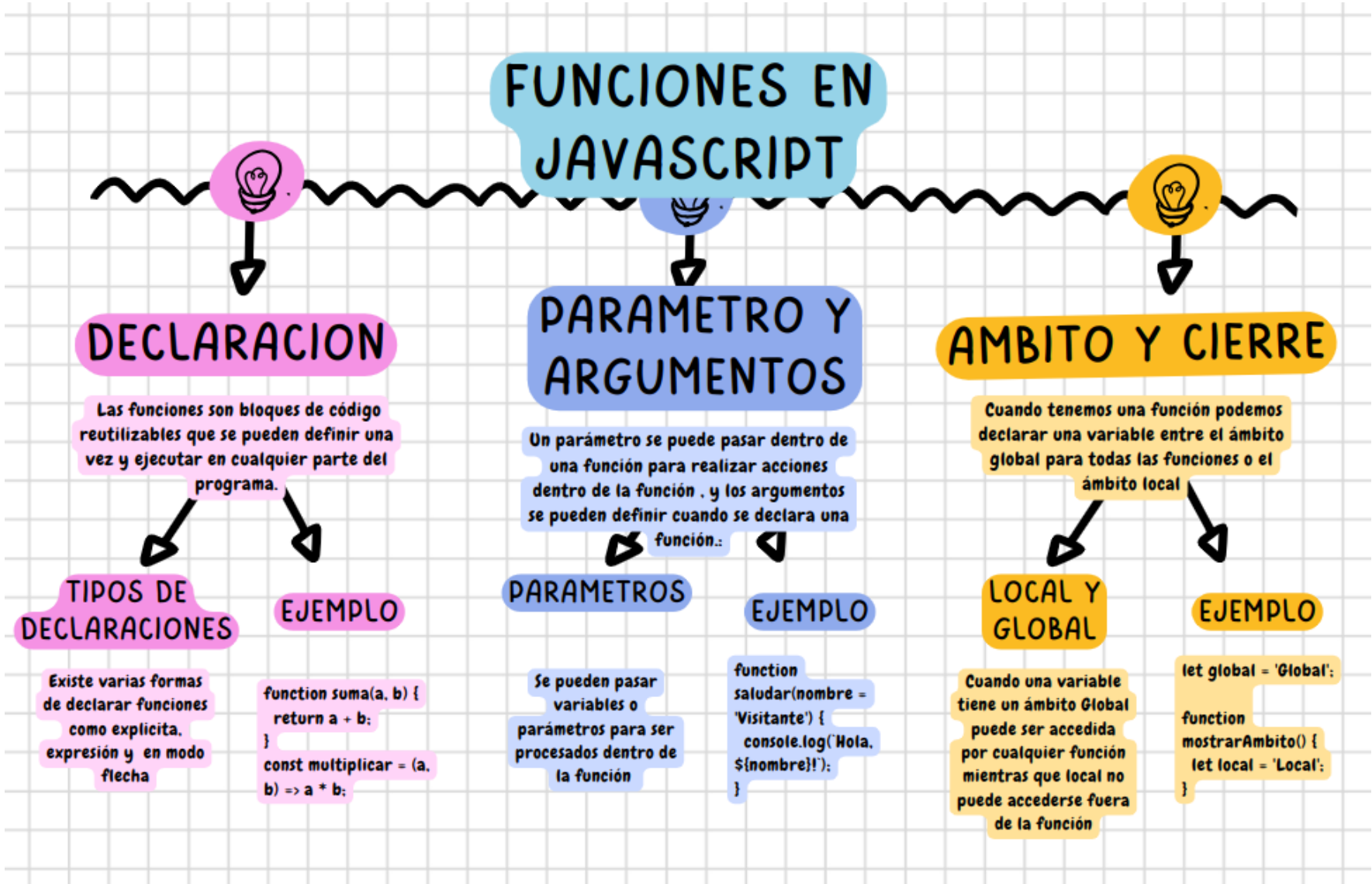
Funciones

Las funciones en JavaScript permiten agrupar y reutilizar bloques de código, facilitando la modularidad y el mantenimiento de las aplicaciones

Los eventos del **DOM** permiten responder a las acciones del usuario, como clics y movimientos del mouse, haciendo las aplicaciones web interactivas y dinámicas

Funciones

CLICK


FUNCIONES EN JAVASCRIPT
DECLARACION
Las funciones son bloques de código reutilizables que se pueden definir una vez y ejecutar en cualquier parte del programa.
TIPOS DE DECLARACIONES
Existe varias formas de declarar funciones como explicita, expresión y en modo flecha
EJEMPLO
function suma(a, b) { return a + b; }
const multiplicar = (a, b) => a * b;
PARAMETRO Y ARGUMENTOS
Un parámetro se puede pasar dentro de una función para realizar acciones dentro de la función , y los argumentos se pueden definir cuando se declara una función.:
PARAMETROS
Se pueden pasar variables o parámetros para ser procesados dentro de la función
EJEMPLO
function saludar(nombre = 'Visitante') { console.log(`Hola, ${nombre}!`); }
AMBITO Y CIERRE
Cuando tenemos una función podemos declarar una variable entre el ámbito global para todas las funciones o el ámbito local
LOCAL Y GLOBAL
Cuando una variable tiene un ámbito Global puede ser accedida por cualquier función mientras que local no puede accederse fuera de la función
EJEMPLO
let global = 'Global'; function mostrarAmbito() { let local = 'Local'; }

Funciones

Las funciones son bloques de código reutilizables que se pueden definir una vez y ejecutar en cualquier parte del programa

Ejemplo

```
function nombreFuncion() {
  // código a ejecutar
}
```

Declaración de Funciones

Ejemplo

```
// Declaración de función
function suma(a, b) {
  return a + b;
}

console.log(suma(2, 3)); // Imprime 5

// Expresión de función
const resta = function(a, b) {
  return a - b;
};

console.log(resta(5, 2)); // Imprime 3
```

https://www.programiz.com/online-compiler/9dsty3bmwhZRL

Funciones flecha

Ejemplo

```
const multiplicar = (a, b) => a * b;
console.log(multiplicar(3, 4)); // Imprime 12
```

```
function saludar(nombre) {
  console.log(`Hola, ${nombre}!`);
}
saludar('Carlos'); // Imprime 'Hola, Carlos!'
```

https://www.programiz.com/online-compiler/6oVZ6Qkh4Erwz

Estructuras de Bucle

Ejercicio

Crea un programa interactivo que simule una calculadora de operaciones matemáticas básicas (suma, resta, multiplicación y división), debe ingresar dos números, el programa debe mostrar un menú con las opciones:

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

- El programa debe repetir el menú de opciones hasta que el usuario seleccione la opción de salida (opción 5).
- Realizar la operación correspondiente y mostrar el resultado en la consola.
- Si el usuario ingresa una opción no válida (fuera del rango 1-5), el programa debe indicar que la opción seleccionada no es válida y volver a mostrar el menú.
- El programa debe continuar ejecutándose en un bucle hasta que el usuario seleccione la opción 5 para salir.

https://www.programiz.com/online-compiler/8sFHADwSoIwb2

Parámetros Predeterminados

Se puede asignar valores predeterminados a un parámetro, si no se pasa un argumento, toma el valor por defecto.

Ejemplo

```
function saludar(nombre = 'Visitante') {
  console.log(`Hola, ${nombre}!`);
}
saludar(); // Imprime 'Hola, Visitante!'
saludar('Ana'); // Imprime 'Hola, Ana!'
```

Ver ejemplo

Parámetros rest

Capturar múltiples argumentos en una función

Ejemplo

```
function sumarTodos(...numeros) {
  return numeros.reduce((acc, num) => acc + num, 0);
}
console.log(sumarTodos(1, 2, 3, 4)); // Imprime 10
```

Ver ejemplo

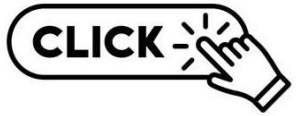
Ámbito de las variables

Cuando tenemos una función podemos declarar una variable entre el ámbito global para todas las funciones o el ámbito local

Ejemplo

```
let global = 'Global';
function mostrarAmbito() {
  let local = 'Local';
  console.log(global); // Imprime 'Global'
  console.log(local); // Imprime 'Local'
}
mostrarAmbito();
console.log(local); // Error: local no está definida
```

Ver ejemplo



Closures (Cierres)

Detallan las funciones que recuerdan el ámbito en el que fueron creadas

Ejemplo

```
function crearContador() {
  let contador = 0;
  return function() {
    contador++;
    return contador;
  };
}
const contador1 = crearContador();
console.log(contador1()); // Imprime 1
console.log(contador1()); // Imprime 2
```

Ver ejemplo

Funciones de orden superior

Funciones que toman otras funciones como argumentos o las devuelven

Ejemplo

```
function aplicarOperacion(a, b, operacion) {
  return operacion(a, b);
}
const suma = (a, b) => a + b;
const producto = (a, b) => a * b;

console.log(aplicarOperacion(5, 3, suma)); // Imprime 8
console.log(aplicarOperacion(5, 3, producto)); // Imprime 15
```

Ver ejemplo

Callbacks

Estas son funciones que se pasan como argumentos para ser ejecutadas después

Ejemplo

```
function obtenerDatos(callback) {
  setTimeout(() => {
    const datos = { nombre: 'Juan', edad: 30 };
    callback(datos);
  }, 1000);
}

function mostrarDatos(datos) {
  console.log(`Nombre: ${datos.nombre}, Edad: ${datos.edad}`);
}
obtenerDatos(mostrarDatos); // Imprime 'Nombre: Juan, Edad: 30' después de 1
segundo
```

Ver ejemplo

Función Recursiva

Estas son funciones que se pasan como argumentos para ser ejecutadas después

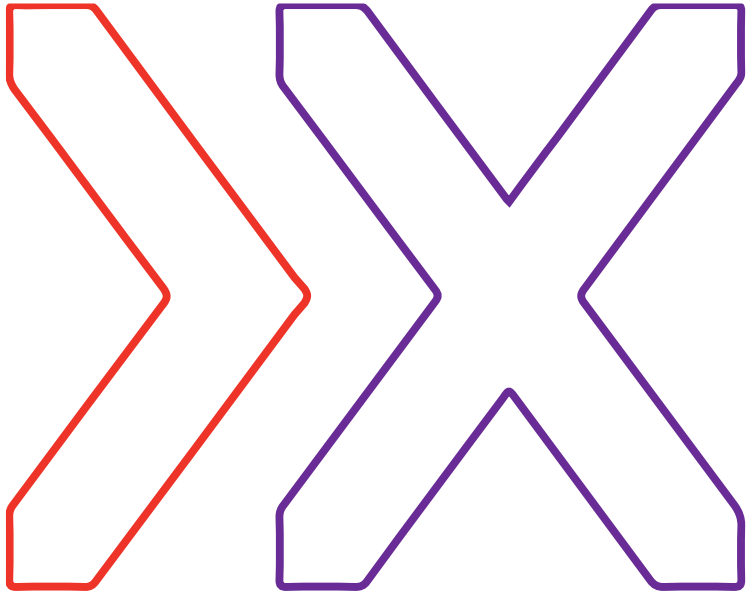
Ejemplo

$$n! = n \times (n-1) \times (n-2) \times ... \times 1$$

$$1! = 1$$
$$2! = 2 \times 1 = 2$$
$$3! = 3 \times 2 \times 1 = 6$$
$$4! = 4 \times 3 \times 2 \times 1 = 24$$
$$5! = 5 \times 4 \times 3 \times 2 \times 1 = 120$$

```
function factorial(n) {
  if (n === 0) {
    return 1;
  }
  return n * factorial(n - 1);
}
console.log(factorial(5)); // Imprime 120
```

Ver ejemplo



Unidad 3

Eventos y manejo del DOM en JavaScript

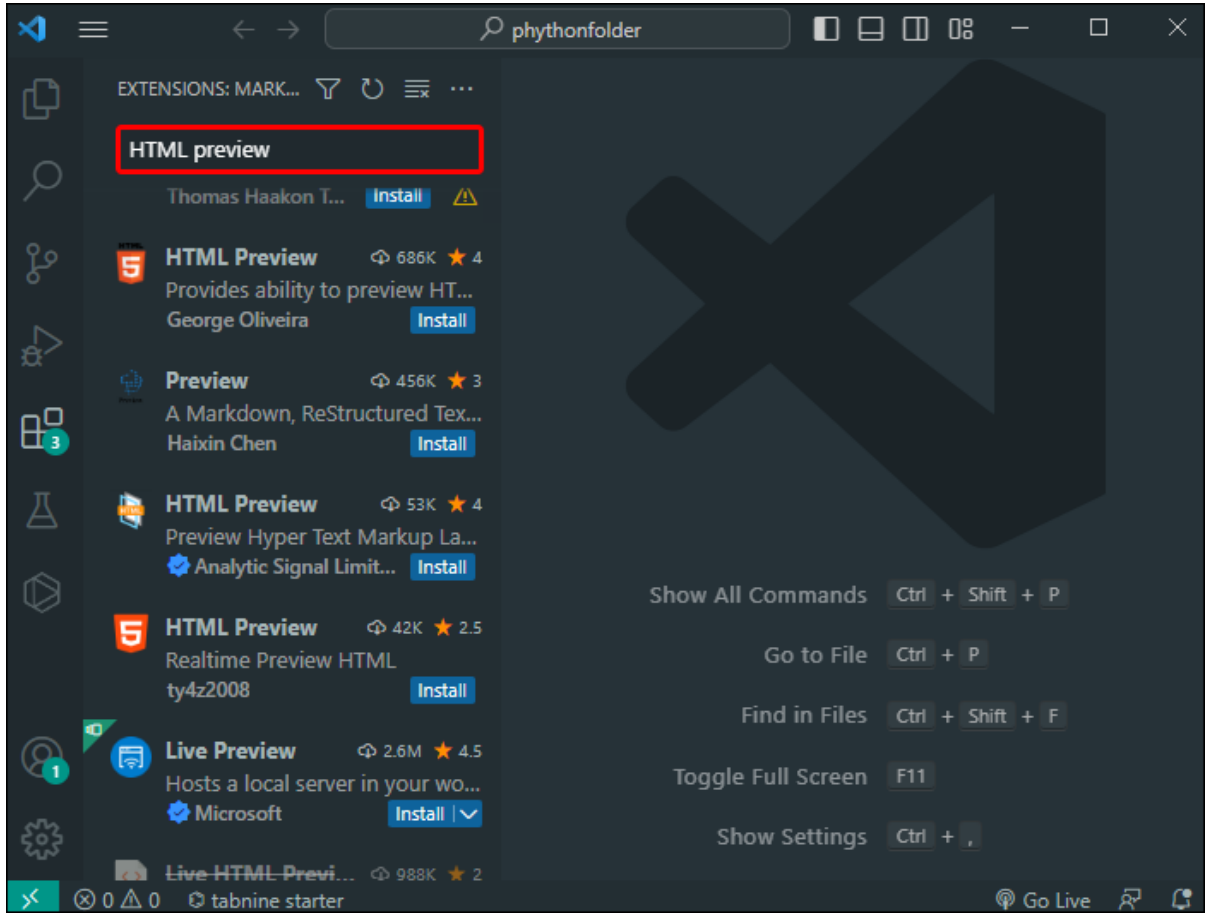
Uso de Visual Studio Code

Download

https://code.visualstudio.com

Comando CMD

```
code .
```



Uso de Git y Github

Download

https://git-scm.com/

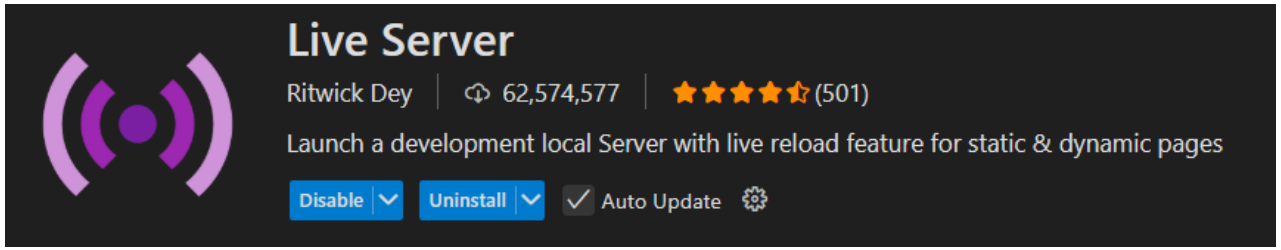
Comando CMD

git

https://github.com

Uso de Git y Github

git clone




MANEJO DE EVENTOS DEL DOM
MANEJO
Para acceder al DOM desde javascript tenemos algunos elementos en javascript, la idea es cambiar el titulo, algunos estilos de componentes del html con una función al hacer click en un boton
MODIFICAR DOM
Hay varias maneras de seleccionar y modificar el DOM, con ello podemos cambiar, adicionar o editar elmentos
EJEMPLO
const titulo = document.getElementById('titulo');
titulo.textContent = 'Nuevo Titulo';
TIPO DE EVENTOS
Dependiendo del tipo de objetos tenemos diferente tipo de eventos si hacemos un click, nos movemos encima con el mouse, modificamos una tecla
TIPOS
Entre los tipos podemos encontrar
Eventos de Mouse
Eventos de Teclado
Eventos de formulario
Eventos de documento
EJEMPLO
boton.addEventListener('click', saludar);
LISTENERS
Los listeners son escuchas para realizar acciones cuando ocurre un evento
CUANDO ACCIONAR
Cuando tenemos un tipo de evento un listener se encarga de escuchar cuando exista un cambio, dependiendo del tipo de evento
EJEMPLO
boton.addEventListener('click', saludar);

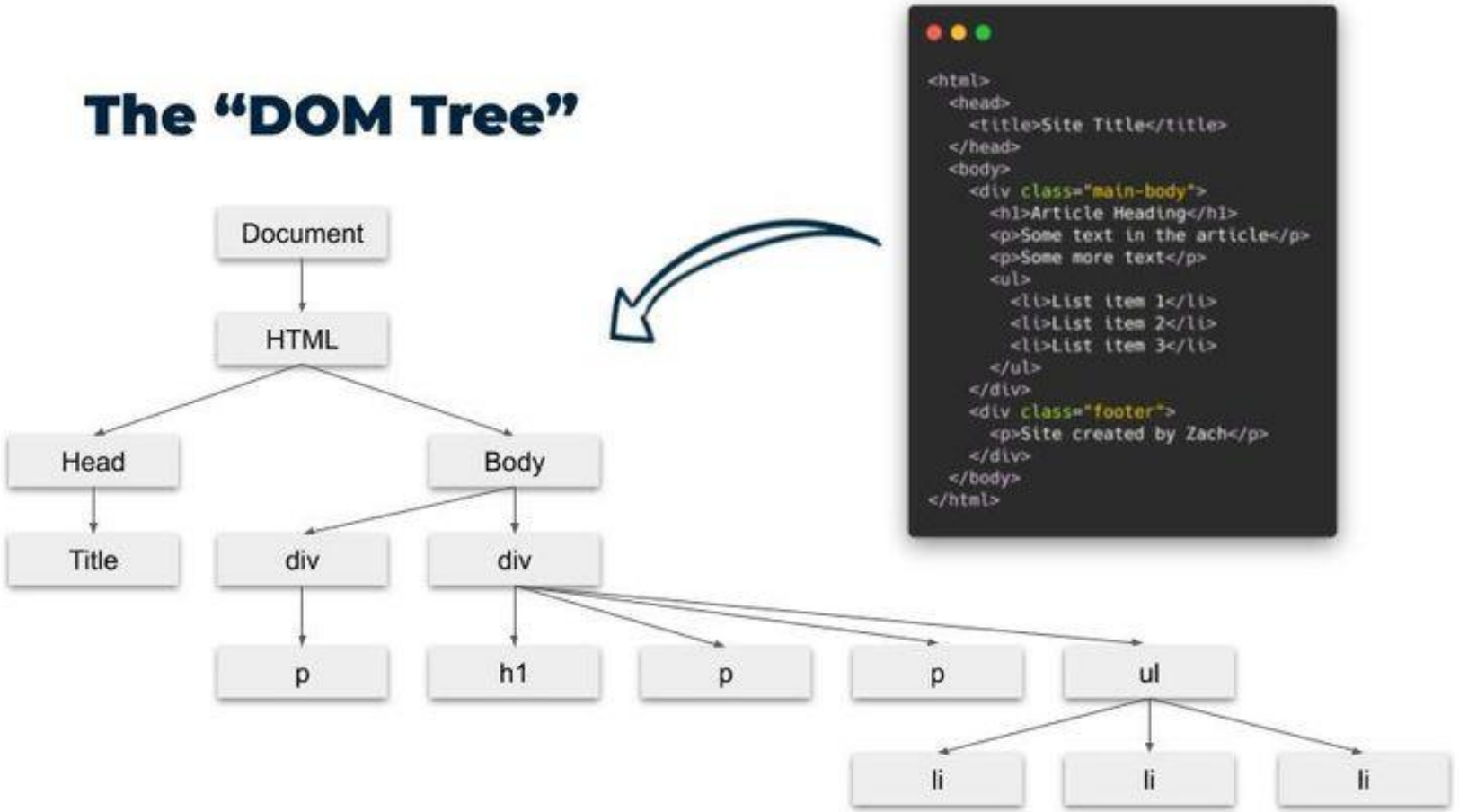
Introducción al DOM

HTML HyperText Markup Language

El **DOM (Document Object Model)** en HTML es una representación estructurada de un documento web que permite a los lenguajes de programación interactuar con el contenido, estructura y estilo de las páginas web.

Cada elemento del HTML, como etiquetas, atributos y texto, se convierte en un objeto en el DOM, lo que permite a los desarrolladores manipular dinámicamente el contenido, cambiar estilos y responder a eventos del usuario a través de JavaScript.

Introducción al DOM



Acceso y Selección de Elementos

Ejemplo **DOM-1**

```
// Seleccionar un elemento por su ID
const titulo = document.getElementById('titulo');
// Seleccionar elementos por su clase
const items = document.getElementsByClassName('item');
// Seleccionar elementos por su etiqueta
const parrafos = document.getElementsByTagName('p');
// Selección más flexible usando querySelector y querySelectorAll
const primerParrafo = document.querySelector('p');
const todosLosParrafos = document.querySelectorAll('p');
// Cambiar el texto de un elemento
titulo.textContent = 'Nuevo Título';
// Cambiar el HTML de un elemento
primerParrafo.innerHTML = '<strong>Párrafo modificado</strong>';
// Cambiar un atributo de un elemento
primerParrafo.setAttribute('class', 'nuevoEstilo');
```

Accediendo al DOM desde JavaScript

Ejemplo **DOM-2**

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>DOM Example</title>
</head>
<body>
    <h1 id="main-title">Hello, World!</h1>
    <p class="text">This is a paragraph.</p>
    <p class="text">This is another paragraph.</p>
    <button id="change-text">Change Text</button>
    <script src="script.js"></script>
</body>
</html>
```

Accediendo al DOM desde JavaScript

Ejemplo JavaScript **DOM-2**

```
// Acceso al DOM usando ID
const title = document.getElementById('main-title');
title.textContent = 'Hello, JavaScript!';
// Acceso al DOM usando clase
const paragraphs = document.getElementsByClassName('text');
for (let i = 0; i < paragraphs.length; i++) {
    paragraphs[i].style.color = 'blue';
}
// Manejo de eventos
const button = document.getElementById('change-text');
button.addEventListener('click', function() {
    title.textContent = 'Text Changed!';
});
```

Introducción a los eventos en JavaScript

Los eventos son acciones que ocurren en los componentes de una página web dentro del navegador, como clics, movimientos del mouse o la pulsación de teclas

Tipos de eventos

Mouse

click

dblclick

mouseover

mouseout

Teclado

keydown

keyup

keypress

Form

submit

change

Document

DOMContentLoaded

load

Modelo de eventos

El modelo de eventos del DOM depende de la selección y el objetivo que se destina podemos definir la propagación

Ejemplo **DOM-3**

Fase de Captura

El evento se propaga desde el elemento más externo hacia el objetivo del evento.

Fase de Objetivo

El evento alcanza el objetivo y se ejecutan los event listeners asociados a él

Fase de Burbuja

El evento se propaga de regreso desde el objetivo hacia el elemento más externo

Agregando listeners en JavaScript

Ejemplo

```
const boton = document.getElementById('miBoton');
function saludar() {
  alert('Hola!');
}
boton.addEventListener('click', saludar);
```

```
boton.removeEventListener('click', saludar);
```

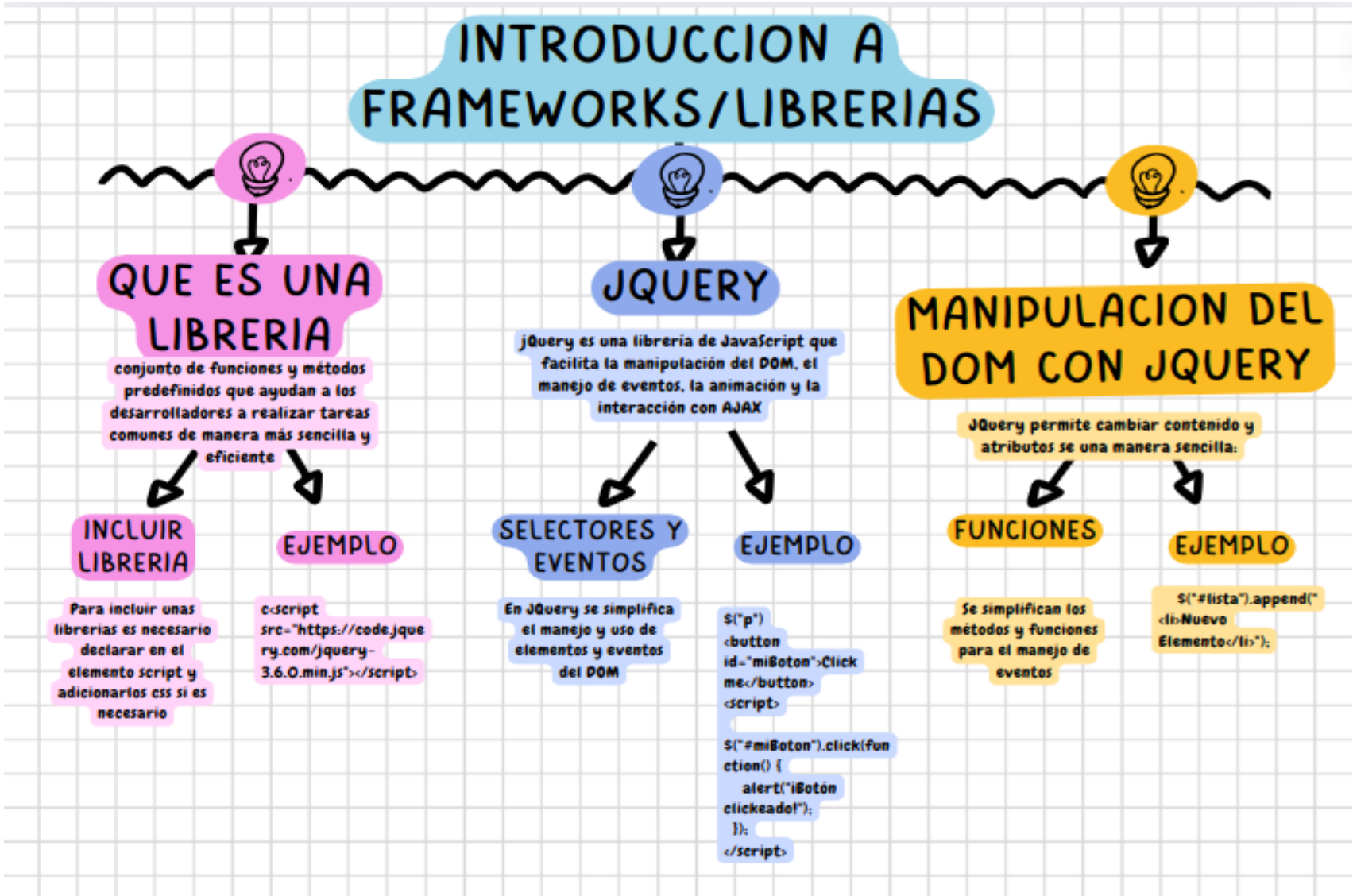
Propagación de eventos

Ejemplo **DOM-4**

```
// Detener la propagación de eventos
boton.addEventListener('click', function(event) {
  alert('¡Clic en el botón!');
  event.stopPropagation();
});
// Manejador de evento en un elemento padre
document.body.addEventListener('click', function() {
  alert('¡Clic en el cuerpo del documento!');
});
```

Unidad 4

Introducción a frameworks / librerías JQuery


INTRODUCCION A FRAMEWORKS/LIBRERIAS
QUE ES UNA LIBRERIA
conjunto de funciones y métodos predefinidos que ayudan a los desarrolladores a realizar tareas comunes de manera más sencilla y eficiente
INCLUIR LIBRERIA
Para incluir unas librerias es necesario declarar en el elemento script y adicionarlos css si es necesario
EJEMPLO
c<script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
JQUERY
jQuery es una libreria de JavaScript que facilita la manipulación del DOM, el manejo de eventos, la animación y la interacción con AJAX
SELECTORES Y EVENTOS
En JQuery se simplifica el manejo y uso de elementos y eventos del DOM
EJEMPLO
$("p")
<button id="miBoton">Click me</button>
<script>
$("#miBoton").click(function() {
alert("¡Botón clickeado!");
});
</script>
MANIPULACION DEL DOM CON JQUERY
JQuery permite cambiar contenido y atributos se una manera sencilla:
FUNCIONES
Se simplifican los métodos y funciones para el manejo de eventos
EJEMPLO
$("#lista").append("
Nuevo Elemento
");

Introducción a las Librerías en JavaScript

Una librería es un conjunto de funciones y métodos predefinidos que ayudan a los desarrolladores a realizar tareas comunes de manera más sencilla y eficiente

Las librerías permiten reutilizar código, ahorrar tiempo y aprovechar el trabajo de una comunidad de desarrolladores.

Introducción a las Librerías en JavaScript

Ventajas

- Ahorro de tiempo en el desarrollo.
- Código reutilizable y mantenible.
- Amplio soporte y documentación.

Ejemplos de librerías populares de JavaScript

- **jQuery**: Simplifica el manejo del DOM y eventos. Enlace
- **Lodash**: Proporciona utilidades para manipular y trabajar con datos. Enlace
- **D3.js**: Facilita la creación de gráficos y visualizaciones de datos. Enlace

Introducción a jQuery

jQuery es una librería de JavaScript que facilita la manipulación del DOM, el manejo de eventos, la animación y la interacción con AJAX.

Fue creada en 2006 y se ha vuelto extremadamente popular debido a su simplicidad y poder.

Cómo incluir jQuery en un proyecto

Usando un CDN

```
<script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
```

Descarga e inclusión local

```
<script src="ruta/a/jquery.min.js"></script>
```

Sintaxis Básica de jQuery

jQuery utiliza selectores CSS para seleccionar elementos del DOM.
La sintaxis básica es $(selector)

Ejemplo

```
// Seleccionar todos los párrafos
$("p")
// Seleccionar un elemento por ID
$("#miElemento")
// Seleccionar elementos por clase
$(".miClase")
```

Eventos en jQuery

jQuery simplifica el manejo de eventos con métodos como click, hover, keydown, etc.

Ejemplo

```
<button id="miBoton">Click me</button>
<script>
  $("#miBoton").click(function() {
    alert("¡Botón clickeado!");
  });
</script>
```

Manipulación del DOM

Cambiar contenido

JQuery permite cambiar contenido y atributos se una manera sencilla:

text(): Obtiene o establece el texto de los elementos seleccionados.

html(): Obtiene o establece el HTML de los elementos seleccionados.

attr(): Obtiene o establece atributos de los elementos seleccionados

Ejemplo

```
<p id="miParrafo">Texto original</p>

<button id="cambiarTexto">Cambiar Texto</button>
<script>
  $("#cambiarTexto").click(function() {
    $("#miParrafo").text("Texto cambiado");
  });
</script>
```

Manipulación del DOM

Añadir y remover elementos

- **append():** Añade contenido al final de los elementos seleccionados**.**
- **prepend():** Añade contenido al inicio de los elementos seleccionados**.**
- **remove():** Elimina los elementos seleccionados del DOM**.**

Ejemplo

```
<ul id="lista">
  <li>Elemento 1</li>
</ul>
<button id="añadirElemento">Añadir Elemento</button>
<script>
  $("#añadirElemento").click(function() {
    $("#lista").append("<li>Nuevo Elemento</li>");
  });
</script>
```

Manipulación del DOM

Modificar CSS

- **css():** Obtiene o establece propiedades CSS de los elementos seleccionados**.**
- **addClass(), removeClass(), toggleClass():** Manipula clases CSS**.**

Ejemplo

JQUERY-1

```
<ul id="lista">
    <li>Elemento 1</li>
</ul>
<button id="añadirElemento">Añadir Elemento</button>
<script>
    $("#añadirElemento").click(function() {
      // Añadir un nuevo elemento
      $("#lista").append("<li>Nuevo Elemento</li>");
      // Cambiar el estilo CSS del último elemento añadido
      $("#lista li:last").css({
        "color": "blue",
        "font-size": "20px"
      });
    });
</script>
```

Ejemplos con Jquery

Alternar la visibilidad de un elemento

Ejemplo

JQUERY-2

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Ejemplo 2</title>
  <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
</head>
<body>
  <button id="alternarVisibilidad">Toggle Visibility</button>
  <div id="caja" style="width:100px;height:100px; background-
color:red;"></div>
  <script>
    $("#alternarVisibilidad").click(function() {
      $("#caja").toggle();
    });
  </script>
</body>
</html>
```

Ejemplos con Jquery

Crear un formulario simple y validar que todos los campos estén llenos antes de enviar

Ejemplo

JQUERY-3

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Ejercicio 1</title>
  <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
</head>
<body>
  <form id="miFormulario">
    <input type="text" id="nombre" placeholder="Nombre">
    <input type="email" id="correo" placeholder="Correo">
    <button type="submit">Enviar</button>
  </form>
  <script>
    $("#miFormulario").submit(function(event) {
      if ($("#nombre").val() === "" || $("#correo").val() === "") {
        alert("¡Todos los campos son obligatorios!");
        event.preventDefault();
      }
    });
  </script>
</body>
</html>
```

Ejemplos con Jquery

Crear una lista de tareas donde se puedan añadir y remover elementos

Ejemplo

JQUERY-4

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Ejercicio 2</title>
  <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
</head>
<body>
  <input type="text" id="tarea" placeholder="Nueva tarea">
  <button id="añadirTarea">Añadir Tarea</button>
  <ul id="listaTareas"></ul>
  <script>
    $("#añadirTarea").click(function() {
      var tarea = $("#tarea").val();
      if (tarea !== "") {
        $("#listaTareas").append("<li>"+tarea+"<button
class='remover'>Remover</button></li>");
        $("#tarea").val("");
      }
    });

    $("#listaTareas").on("click", ".remover", function() {
      $(this).parent().remove();
    });
  </script>
</body>
</html>
```

Evaluación final del Modulo 1

Descripción

En esta práctica, los estudiantes desarrollarán una aplicación web simple que permita al usuario gestionar una lista de tareas (to-do list). La aplicación debe permitir añadir nuevas tareas, marcar tareas como completadas y eliminar tareas. Se espera que los estudiantes apliquen los conceptos de variables, tipos de datos, operadores, estructuras de control, funciones, eventos del DOM y jQuery.

Requisitos de la Práctica

Evaluación final del Modulo 1

Requisitos de la Práctica

1. **Interfaz de Usuario:**
   - **Una entrada de texto para agregar nuevas tareas.**
   - **Un botón para añadir la tarea a la lista.**
   - **Una lista (ul) donde se mostrarán las tareas.**
   - **Cada tarea debe tener un botón para marcarla como completada y otro para eliminarla.**
2. **Funcionalidades:**
   - **Añadir nuevas tareas a la lista.**
   - **Marcar tareas como completadas (cambiar el estilo para indicar que está completada).**
   - **Eliminar tareas de la lista.**
3. **Requisitos Técnicos:**
   - **Uso de variables, tipos de datos y operadores para manejar las tareas.**
   - **Estructuras de control (condicionales y bucles) para procesar las tareas.**
   - **Definición y uso de funciones para modularizar el código.**
   - **Manejo de eventos del DOM para interactuar con el usuario.**
   - **Utilización de jQuery para simplificar la manipulación del DOM.**

Evaluación final del Modulo 1

Rúbrica de Evaluación

Criterio	Excelente (4)	Bueno (3)	Regular (2)	Insuficiente (1)
Interfaz de Usuario	Interfaz completa, bien diseñada y totalmente funcional.	Interfaz completa y funcional con algunos detalles menores.	Interfaz funcional pero con elementos faltantes o mal diseñados.	Interfaz incompleta o no funcional.
Funcionalidad	Todas las funcionalidades (añadir, completar, eliminar tareas) implementadas y funcionando perfectamente.	Funcionalidades principales implementadas y funcionando con pequeños errores.	Algunas funcionalidades implementadas, pero con varios errores o faltantes.	Funcionalidades principales no implementadas o no funcionan correctamente.
Uso de Variables y Estructuras de Control	Uso adecuado y eficiente de variables y estructuras de control (condicionales, bucles).	Uso adecuado de variables y estructuras de control con pequeños errores.	Uso ineficiente o incorrecto de variables y estructuras de control, pero funcional en su mayoría.	Uso incorrecto o ausencia de variables y estructuras de control, resultando en código no funcional.
Definición y Uso de Funciones	Funciones bien definidas, reutilizables y correctamente invocadas.	Funciones definidas y utilizadas correctamente con pequeños errores.	Funciones definidas pero con problemas de reutilización o invocación incorrecta.	Funciones mal definidas o ausencia de funciones, resultando en código no modular.
Manejo de Eventos del DOM	Manejo correcto y eficiente de eventos del DOM, con uso de delegación de eventos donde aplica.	Manejo adecuado de eventos del DOM, con algunos errores menores.	Manejo básico de eventos del DOM, con errores significativos o falta de uso de delegación de eventos.	Manejo incorrecto o ausencia de eventos del DOM, resultando en funcionalidad limitada o inexistente.
Uso de jQuery	Uso correcto y eficiente de jQuery para manipulación del DOM y manejo de eventos.	Uso adecuado de jQuery con pequeños errores o ineficiencias.	Uso básico de jQuery, con varios errores o uso incorrecto.	Ausencia de jQuery o uso incorrecto, resultando en funcionalidad limitada o inexistente.

GRACIAS