Javascript

Conceptos Fundamentales

Temas

- Introducción al lenguaje
- Variables, tipos de datos, y operadores en JavaScript.
- Estructuras de control de flujo: if, else, switch, while, for.
- Funciones y alcance de variables.

¿Qué es JavaScript?

- Es un lenguaje interpretado, es decir, no requiere compilación.
- Funciona en el lado del cliente y del servidor (Node.js).
- Es un lenguaje orientado a eventos. Mediante JavaScript se desarrollan scripts que ejecutan acciones en respuesta a eventos
- Es un lenguaje basado en prototipos. Utiliza el concepto de prototipos para implementar o simular aspectos de la Orientación a Objetos.

Historia de JavaScript

JavaScript fue creado por **Brendan Eich en 1995** mientras trabajaba en Netscape Communications Corporation. Inicialmente, se llamó **Mocha**, el cual fue renombrado posteriormente a **LiveScript**, para finalmente quedar como **JavaScript**, cuando se estableció una colaboración con Sun Microsystems, que había desarrollado Java.

A lo largo de los años, JavaScript ha evolucionado significativamente. Se han desarrollado estándares como **ECMAScript** para definir su sintaxis y características, y hoy en día, JavaScript se utiliza no solo en el desarrollo web, sino también en el desarrollo de videojuegos, aplicaciones móviles, de escritorio y de servidor, gracias a plataformas como Node.js.

¿Cómo introducir JavaScript en la página?

Es posible implementar JavaScript con tres técnicas principales:

1. En línea.
2. Incorporación en el documento mediante la etiqueta <script>
3. Carga desde un archivo externo.

1. JavaScript en línea

JavaScript se puede agregar directamente en las propiedades de las etiquetas HTML.

```
<!DOCTYPE html>
<html>
<head>
    <title>Ejemplo de JavaScript</title>
</head>
<body>
    <h2>Ejemplo de "Hola Mundo"</h2>
    <button onclick="alert('Hola, mundo')">Haz clic
aquí</button>
</body>
</html>
```

2. JavaScript incrustado en el Head, mediante la etiqueta <script>

Si queremos que el script se ejecute sobre algún evento entonces colocamos el script en <HEAD> dentro de una función.

```
<html>
    <head>
      <script type="text/JavaScript">
        <!-- Esconde el script para otros navegadores
          function popup() {
                alert("Hello World")

          }
          // -->
      </script>
    </head>
    <body>
        <input type="button" onclick="popup()" value="popup">
    </body>
</html>
```

2. JavaScript incrustado en el Body, mediante la etiqueta <script>

Si queremos que el script se ejecute cuando se carga la página colocamos el script en el <BODY>

```
<html>
    <head>
    </head>
    <body>
        <script type="text/JavaScript">
        <!–
            alert("Hello World") //no lleva
            función
        // -->
        </script>
    </body>
</html>
```

3. Javascript en archivo separado

- Promueve la **reutilización** del código al permitir que múltiples páginas web accedan al mismo archivo JavaScript.
- Facilita **mejorar la organización y la mantenibilidad** del proyecto.
- Permite el **almacenamiento en caché d**el archivo JavaScript, lo que puede mejorar el rendimiento y reducir el tiempo de carga para los usuarios.
- Permite aplicar **técnicas de optimización** específicas, como la minificación y la concatenación, que pueden reducir el tamaño del archivo y mejorar el rendimiento de la página.

```
<!DOCTYPE html>
<html>
<head>
  <title>Ejemplo de JavaScript externo</title>
</head>
<body>
  <h2>JavaScript desde archivo externo</h2>
    <script src="script.js"></script>
</body>
</html>
```

Sintaxis

Uso de “;” como separador de sentencias. (NO Obligatorio)

document.write("Hello"); alert("Good bye")

Es lo mismo que:

document.write("Hello");
alert("Good bye");

O también

document.write("Hello")
alert("Good bye")

Sintaxis - sensitive case - comentarios

- **JavaScript es “Sensitive Case”**

document.write() es válida

document.Write() No válida

- **Comentarios**

// comentario de una línea

/* comentarios de varias
Líneas */

Variables y tipos de datos

Javascript es un lenguaje débilmente tipado. El tipo de dato de las variables se asigna automáticamente en base al contexto que se usa.

- Tipos de datos que pueden contener valores.
  - Number, BigInt
  - Boolean
  - Función
  - Objet (3 tipos Object, Array y Date)
  - String
  - Symbol
- Tipos de datos que NO pueden contener valores.
  - Nulo
  - Indefinido

Operador typeof

Devuelve una cadena de texto que indica el tipo de dato.

```
typeof  "John"                 // Returns string
typeof  3.14                   // Returns number
typeof  NaN                    // Returns number
typeof  false                  // Returns boolean
typeof  [1,2,3,4]              // Returns object
typeof  {name:'John', age:34}  // Returns object
typeof   new Date()            // Returns object
typeof   function () {}        // Returns function
typeof   myCar                 // Returns undefined (if myCar is not declared)
typeof   null                  // Returns object
```

No se puede usar typeof para determinar si una variable es Array (o Date)

Declaración de variables - identificadores

- Los nombres de las variables pueden contener letras, dígitos, guiones bajos (_) y signos de dólar ($).
- El nombre de la variable debe comenzar con una letra, un guión bajo (_) o un signo de dólar ($).
- JavaScript distingue entre mayúsculas y minúsculas en los nombres de las variables.
- No se pueden utilizar palabras clave reservadas de JavaScript como nombres de variables.

Ejemplos:

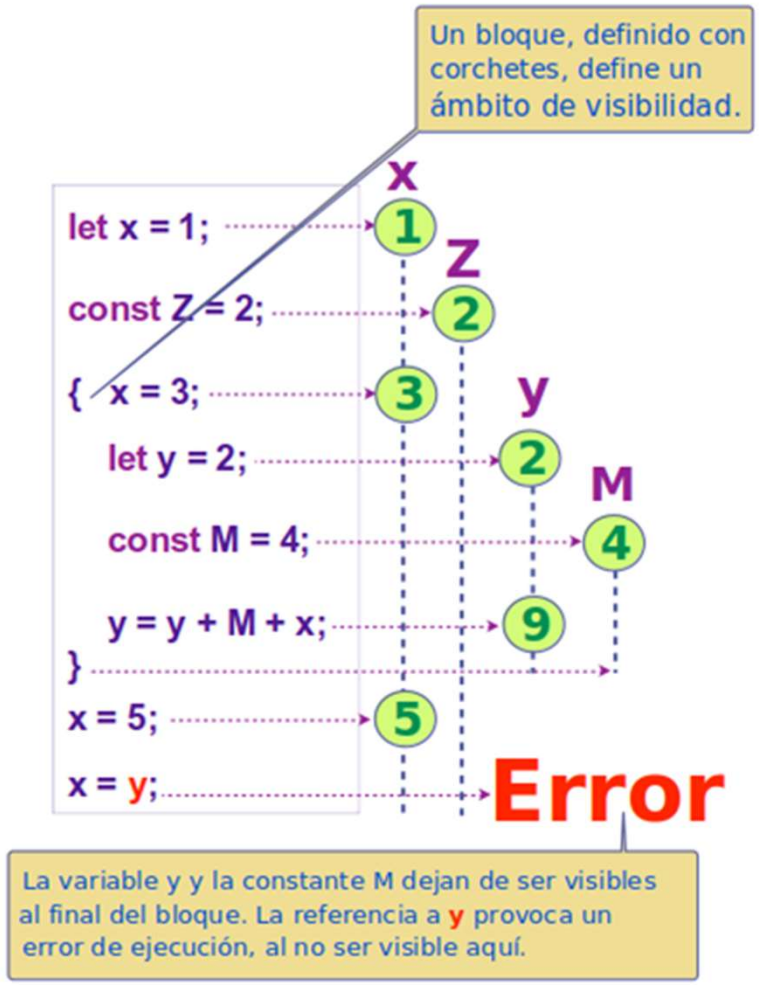
```
let edad = 25;
let nombreCompleto = "Juan Pérez";
let _precioProducto = 50.99;
let $descuento = 10;
```

Declaración de variables

- **let**: (ES6), permite declarar variables de ámbito de bloque. Esto significa que su alcance está limitado al bloque en el que se declaran.
  Ej. `let y = 10;`
- **const**: (ES6) se utiliza para declarar variables de sólo lectura.
  Ej. `const PI = 3.14;`
- **var**: (ES5) Se puede utilizar para declarar variables tanto locales como globales. (No se recomienda por ser poco seguro)
  Ej. `var x = 5;`

Alcance de las variables

El alcance de una variable se refiere a la parte del código donde la variable es accesible. Las variables declaradas con `var` tienen un alcance de función, mientras que las declaradas con `let` y `const` tienen un alcance de bloque.



Convertir números a string

Usar comillas para convertir un valor numérico en cadena o la suma de un número con un espacio;

```
Prueba="59";   o
Prueba = Prueba + "";
```

Método *String()* para convertir en cadena

```
micadena = String(587);  o
(587).toString()
```

Booleanos a string

```
String(false)       // returns "false"
String(true)        // returns "true"
false.toString()    // returns "false"
```

Date a string

```
String(Date())      // returns Thu  Apr 21 2024 16:38:19 GMT+0200 (W. Europe Daylight Time)
```

Convertir string a números

El método global Number() puede convertir string a números.

```
Number("3.14")                    // returns 3.14
Number("    ")                    // returns 0
Number("")                        // returns 0
Number("99 88")   // returns NaN
```

Las funciones *eval(unaExpresion)*, *parseInt(unaExpresion)* y *parseFloat(unaExpresion)* para convertir un valor string en número;

Booleanos a números

```
Number(false)     // returns 0
Number(true)      // returns 1
```

Date a número

```
d = new Date();
Number(d)         // retorna timestamp
1404568027739
```

Conversiones

isNaN(unaExpresion) Indica si un valor es NaN (Not a Number), es decir si el valor no es un número.

```
<script language="Javascript">
if (isNaN(parseInt(cNumber))) {
      alert("El dato ingresado no es numérico.") ;
}
</script>
```

- *toPrecision()* retorna un número con determinada precisión

```
var x = 9.656;
x.toPrecision();       // returns 9.656
x.toPrecision(2);      // returns 9.7
x.toPrecision(4);      // returns 9.656
x.toPrecision(6);      // returns 9.65600
```

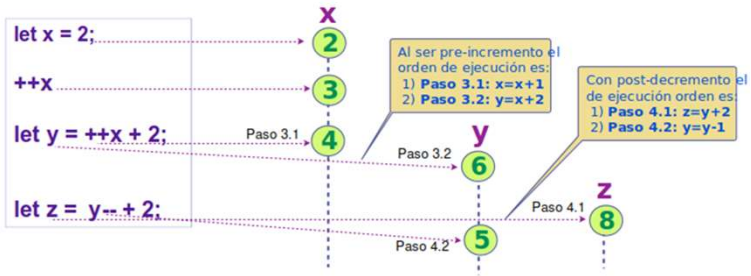
Operadores básicos

Los operadores aritméticos se utilizan para realizar operaciones matemáticas básicas. Los más comunes son:

Operador	Descripción	Ejemplo	Resultado
+	Suma	`var resultado = 5 + 3;`	8
–	Resta	`var resultado = 5 - 3;`	2
*	Multiplicación	`var resultado = 5 * 3;`	15
/	División	`var resultado = 6 / 3;`	2
%	Módulo (Resto de División)	`var resultado = 7 % 3;`	1

Operadores que simplifican la ejecución de operaciones

Operador	Descripción	Ejemplo	Resultado
++	Incremento (añade 1 al valor)	`var numero = 5;` `numero++;`	`numero` es `6`
–	Decremento (resta 1 al valor)	`var numero = 5;` `numero--;`	`numero` es `4`
+=	Incremento por un valor específico	`var numero = 5;` `numero += 3;`	`numero` es `8`
-=	Decremento por un valor específico	`var numero = 5;` `numero -= 2;`	`numero` es `3`
*=	Multiplicación por un valor específico	`var numero = 5;` `numero *= 2;`	`numero` es `10`
/=	División por un valor específico	`var numero = 10;` `numero /= 2;`	`numero` es `5`
%=	Módulo (Resto de División)	`var numero = 7;` `numero %= 3;`	`numero` es `1`



Operadores de comparación

Operador	Significado	Ejemplo	Resultado
==	Igual a	`5 == 5` `5 == "5"`	true true
===	Igual a (estricto)	`5 === '5'` `5 === 5`	false true
!=	Diferente de	`5 != 3`	true
!==	Diferente de (estricto)	`5 !== '5'`	true
>	Mayor que	`5 > 3`	true
<	Menor que	`3 < 5`	true
>=	Mayor o igual que	`5 >= 5`	true
<=	Menor o igual que	`3 <= 5`	true

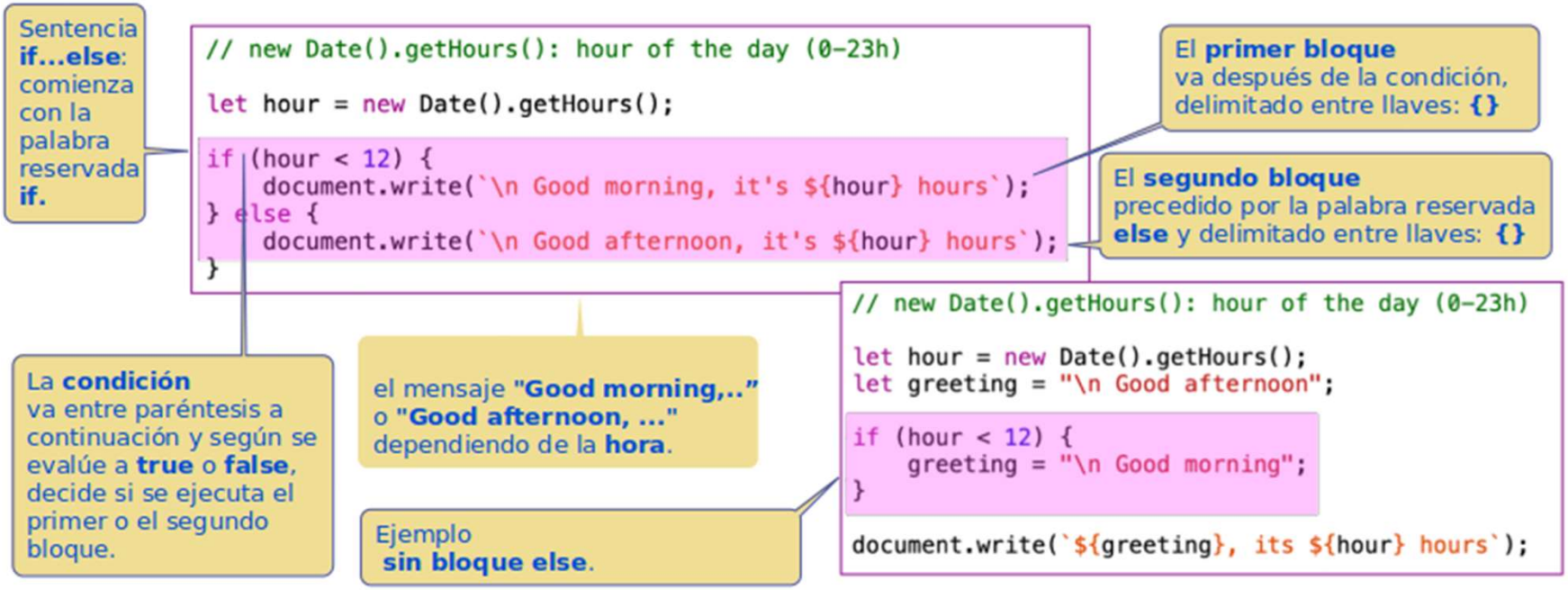
Operador	Uso	Descripción
&&	`expr1 && expr2`	Devuelve verdadero si ambos operandos son verdaderos
||	`expr1 || expr2`	Devuelve verdadero si uno de los operandos es verdadero
!	`!expr`	Devuelve verdadero si el operando es falso. Si el operando es verdadero devuelve falso

Estructuras de Control de Flujo

En JavaScript, la ejecución se da a partir de una sucesión ordenada de comandos que se ejecutan uno detrás de otro. Sin embargo, con frecuencia es necesario recurrir a comandos especiales que alteran o controlan el orden en el que se ejecutan las acciones.

Llamamos estructuras de control del flujo de las acciones al conjunto de reglas que permiten controlar el flujo de las acciones de un algoritmo o programa. Las mismas pueden clasificarse en secuenciales, condicionales e iterativas.

Condicional if- else



Condicional - switch


```
// Math.round(Math.random()*10):
// entero aleatorio entre 0 y 9

let result = Math.round(Math.random()*10);

switch (result) {
  case 9:
    document.write("\n You win the first prize!");
    break;
  case 8:
  case 7:
    document.write("\n You win the second prize!");
    break;
  default:
    document.write("\n Sorry, no prize!");
}
```

La sentencia **break** finaliza de la sentencia **switch-case**. Es

Se pueden agrupar varios **case** es solo un punto de comienzo de ejecución.

Cuando la expresión no coincide con ningún **case**, se pasa a ejecutar las sentencias **default:** (es opcional incluirlo).

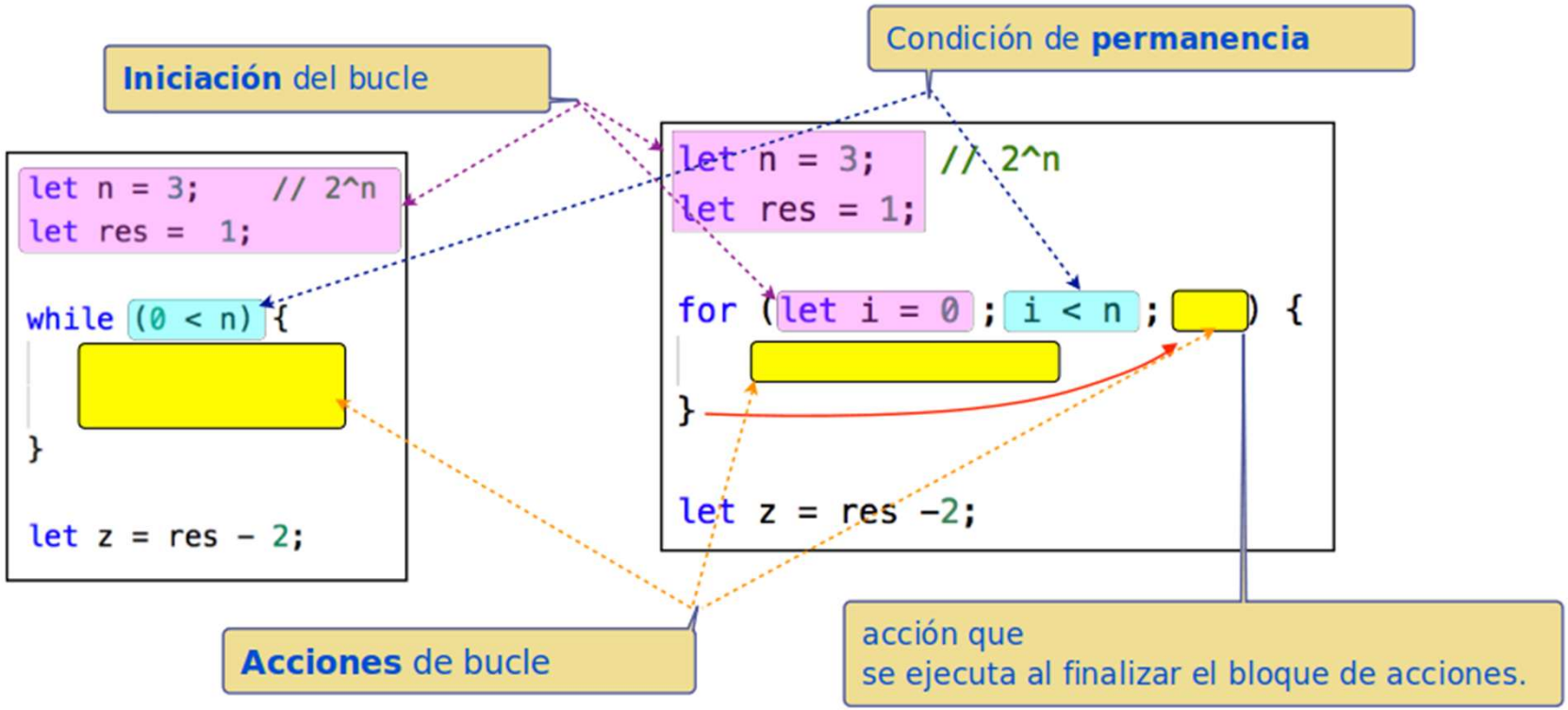
Bucles - while - do while - for


```
for (inicialización; condición; actualización) {
  // Bloque de código a repetir
}
```

```
do {
  // Bloque de código a repetir
} while (condición);
```

```
while (condición) {
  // Bloque de código a repetir
}
```

Bucles - while - for



Bucles

While

```
var lista = new Array(10);
var ind=0;
while (ind < 10)
{
  lista[ind] = '0';
  ind++;
}
```

Do .. While

```
var lista = new Array(10);
var ind=0;
do{
   lista[ind] = '0';
   ind++;
} while (ind < 10);
```

For

```
var lista = new Array(10);
var ind;
for (ind=0; ind < 10; ind++)
{
  lista[ind] = '0';
}
```

Break y Continue

-break: Utilizada para romper el bucle y salir en cualquier momento de la ejecución.
-continue: Utilizada para romper el bucle, pero continúa con la ejecución.

```
var lista = new Array(10);
var ind;
for (ind=0; ind < 10; ind++)
    {
    if(ind==5) continue;
    If(ind==8) break;
    lista[ind] = '1';
    }
```

Métodos en JavaScript

Conversión de Datos	`parseInt()`	Convierte una cadena en un número entero.
	`parseFloat()`	Convierte una cadena en un número decimal.
	`isNaN()`	Determina si un valor es NaN (Not a Number).
Formato de Números	`toFixed()`	Formatea un número utilizando notación de punto fijo con un número específico de dígitos a la derecha del punto decimal.
Interacción con el Usuario	`prompt()`	Muestra un cuadro de diálogo con un mensaje para que el usuario ingrese datos.
	`alert()`	Muestra un cuadro de diálogo con un mensaje de advertencia.
Registro y Depuración	`console.log()`	Muestra un mensaje en la consola del navegador o en la consola del sistema.

Métodos en JavaScript: manipulación de texto ejemplos

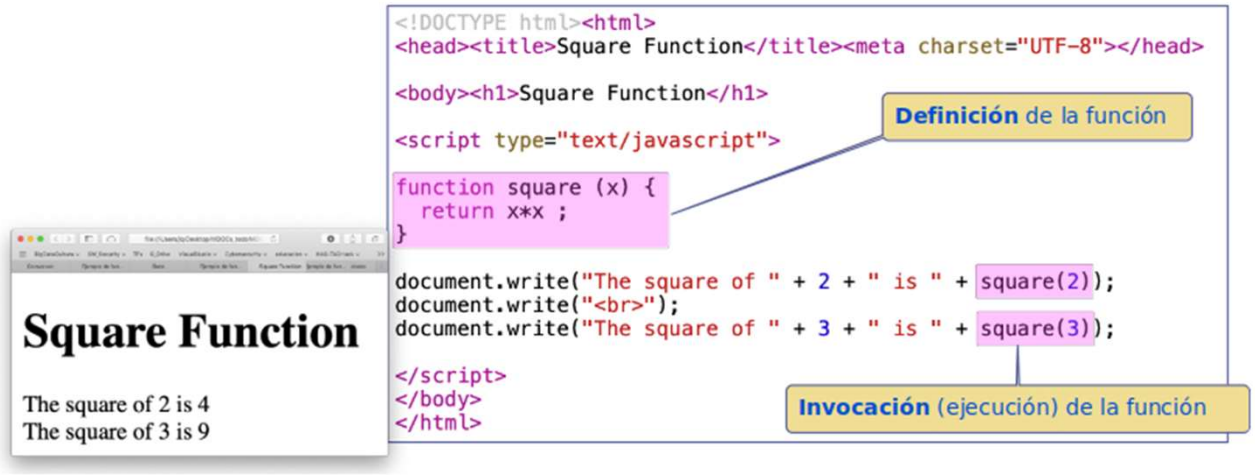
Manipulación de Cadenas de Texto	`toUpperCase()`	Devuelve la cadena de texto en mayúsculas.
	`toLowerCase()`	Devuelve la cadena de texto en minúsculas.
	`indexOf()`	Devuelve la posición de la primera ocurrencia de un valor especificado en una cadena.
	`lastIndexOf()`	Devuelve la posición de la última ocurrencia de un valor especificado en una cadena.
	`slice()`	Extrae una sección de una cadena y devuelve una nueva cadena.
	`replace()`	Reemplaza una parte de la cadena con otra.
	`split()`	Divide una cadena en un array de subcadenas.
	`trim()`	Elimina los espacios en blanco del principio y del final de una cadena.

Funciones

Son bloques de código reutilizables que realizan una tarea específica.

Se definen utilizando la palabra clave **function**, seguida del nombre de la función y un par de paréntesis () en los que se especifican los parámetros.

Los parámetros son variables que reciben los valores necesarios para que la función realice su tarea.



Parámetros y argumentos

-Los **parámetros** son variables utilizadas para almacenar valores que una función espera recibir cuando es invocada.

-Los **argumentos** son los valores reales pasados a la función cuando se llama. Puedes definir múltiples parámetros separándolos por comas.

Ej.

```
// Declaración de la función
function saludo(nombre, apellido) {
  console.log(`Hola, ${nombre}, ${apellido}!`);
}
// Llamada a la función
saludo("Andrea"," Gonzalez"); // Salida: Hola, Andrea Gonzalez!
```

Retorno de valores

Las funciones en JavaScript pueden devolver valores utilizando la palabra clave **return**. Esto permite que la función calcule un valor y lo pase de vuelta al código que la llamó.

Ej.

```
// Función que retorna el cuadrado de un numero
function cuadrado(numero) {
  return numero*numero;

}
// Probamos la función con un ejemplo
const numero = 6;
console.log("El cuadrado de " + numero + " es: "+ cuadrado(numero));
```

```
Archivo  Editar  Ver  Buscar  Terminal  Ayuda
mariel@X1:~/Escritorio/js$ node ej.js
El cuadrado de 6 es: 36
mariel@X1:~/Escritorio/js$
```

Cuadros de Diálogo

- **alert**: Muestra un mensaje simple de alerta.
- **confirm**: Pide al usuario confirmar una acción con un botón OK o Cancelar. Retorna true si el usuario hace clic en OK y false si hace clic en Cancelar.
- **prompt**: Solicita al usuario que ingrese un valor. Retorna el texto ingresado por el usuario o null si el usuario hace clic en Cancelar.

Cuadros de Diálogo: prompt, Confirm y alert


```
<body>
    <h2>Ejemplo de Cuadros de Diálogo prompt y confirm</h2>
    <button onclick="pedirNombre()">Por favor, introduce tu nombre</button>
    <script>
        // Función para pedir el nombre, confirmar si le gusta JavaScript y mostrar un mensaje
        function pedirNombre() {
            let nombre = prompt('Por favor, introduce tu nombre:');
            if (nombre) {
                let leGustaJS = confirm('¿Te gusta JavaScript?');
                if (leGustaJS) {
                    alert('Hola, ' + nombre + '! Me alegra saber que te gusta JavaScript.');
                } else {
                    alert('Hola, ' + nombre + '. Qué lástima que no te guste JavaScript.');
                }
            } else {
                alert('No introdujiste ningún nombre.');
            }
        }
    </script>
</body>
```

Referencias Útiles

- Mozilla Developer Network (MDN) - JavaScript
  - URL:https://developer.mozilla.org/en-US/docs/Web/JavaScript
- W3Schools - JavaScript
  - URL: W3Schools: JavaScript
- JavaScript.info
  - URL: JavaScript.info

Extensiones útiles en VS

- Babel javascript (sintaxis)
- Error lens (Corrector de sintaxis)
- Nightly (debugger)
- Prettier (formatear código)

Ejemplos

- Ejemplo cuadros de diálogo
- Ejemplo sobre declaración de variables - var-let-const
- Ejemplo sobre strings
- Ejemplo if-else
- Ejemplo for
- Ejemplo while
- Ejemplo do-while
- Ejemplo break
- Ejemplo continue