

Programación Web

Introducción

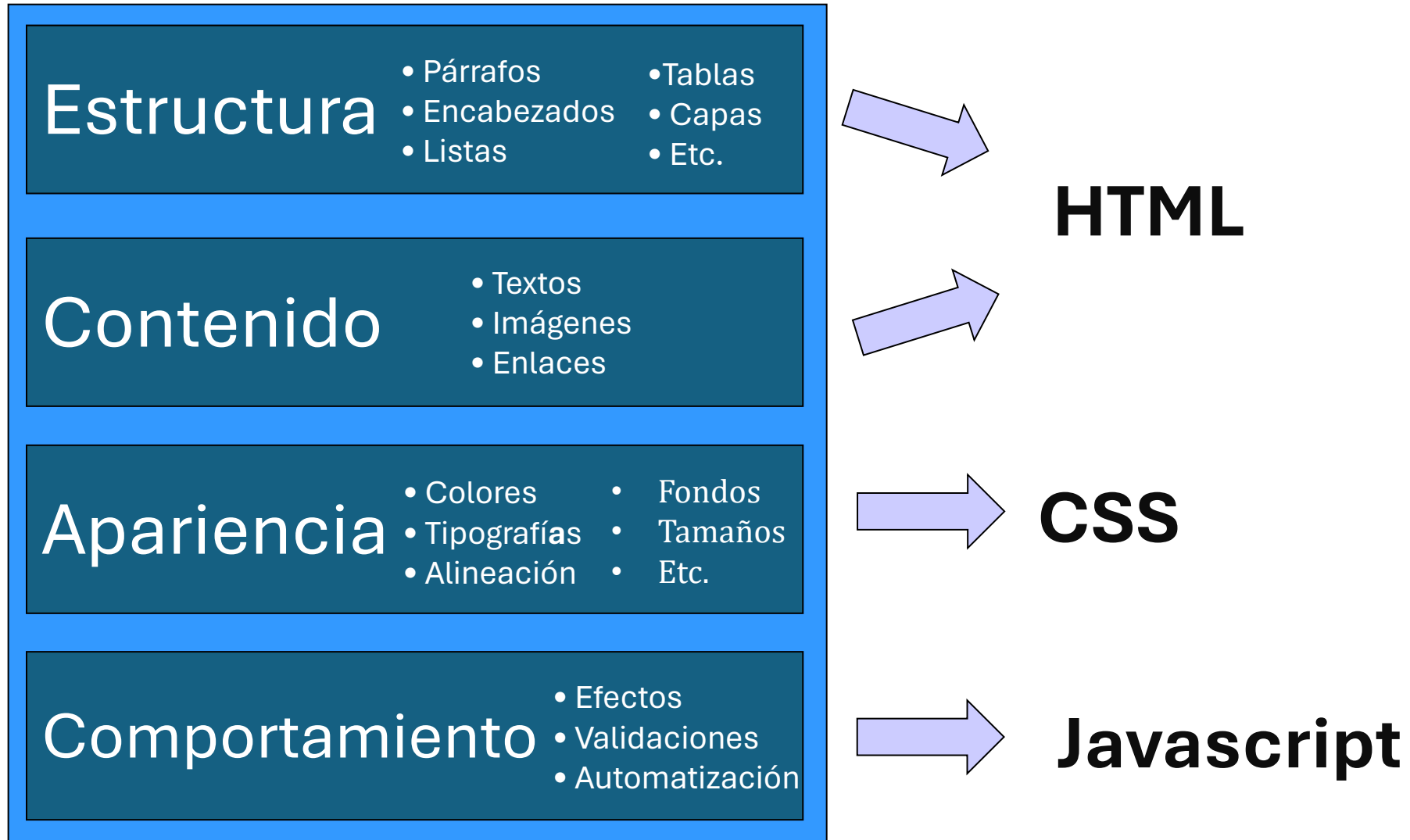
Facultad de Ciencias Naturales e Ingenierías
Tecnología en Desarrollo de Sistemas Informáticos

Docente: Ing. Jhon Pinto
Correo: jpintobarrera@gmail.com

¿Qué es HTML5?

- Es un lenguaje de marcas utilizado para el desarrollo de páginas web.
- Define la estructura y contenido que debe tener una web.
- No define el estilo visual que tendrá para eso se usará CSS.
- Ha sido establecido como estándar de diseño web por el W3C.
- Los navegadores deben saber interpretar este lenguaje de manera correcta (no siempre IE)
- Sobre HTML se desarrollan tecnologías para facilitar a los usuarios el diseño de una web.

Estructura de una página web



Elementos en HTML

- Los elementos son los componentes fundamentales del HTML
- Cuentan con 2 propiedades básicas:
 - Atributos
 - Contenido
- En general se conforman con una Etiqueta de Apertura y otra Cierre.
- Los atributos se colocan dentro la etiqueta de apertura, y el contenido se coloca entre la etiqueta de apertura y la de cierre.

Elementos en HTML

Elemento		
Etiqueta de Apertura	Contenido	Etiqu. de Cierre

<p > Curso programación Web </p>

Tipos de elementos en HTML

Estructurales:

- Describen el propósito del texto y no denotan ningún formato específico.
`<h1>Curso Programación Web</h1>`

Presentación:

- Describen la apariencia del texto, independientemente de su función.
` Curso Programación Web `
- Los elementos de presentación se encuentran obsoletos desde la aparición del CSS.

HiperTexto:

- Relaciona una parte del documento a otros documentos.
`Google`

Sección HEAD

- Contiene metainformación de la página.
- Establece el título y palabras clave para los buscadores.
- Incluye las hojas de estilo (CSS) a utilizar en la página
- Incluye el código JavaScript a usar en nuestra página.

Sección BODY

- Define el "contenido" real de la página.
- Establece cómo se visualizan los elementos.
- Hace uso de los scripts y hojas de estilo definidos en la sección HEAD.
- En este punto tenemos a nuestra disposición de todos los tags disponibles para maquetar nuestra página.

Estructura base de un documento HTML5

introduccion.html > ...

```
1  <!DOCTYPE html>
2  <html lang="en">
3      <head>
4          <meta charset="UTF-8" />
5          <meta name="viewport" content="width=device-width, initial-scale=1.0" />
6          <title>Document</title>
7      </head>
8      <body></body>
9  </html>
10
```

- El lenguaje HTML está limitado a la hora de aplicarle forma a un documento.
- Sus estructuras tienen poca flexibilidad a la hora de dar forma al contenido mostrado.
- HTML se creó originariamente para uso científico y posteriormente fue adoptado para el desarrollo web.
- Para "maquetar" se utilizan elementos HTML en un uso diferente de su objetivo (tablas por ejemplo).
- Todos estos problemas dieron lugar al origen de CSS.

¿Qué es CSS?

- Son hojas de estilo en cascada (**CSS**, cascading style sheets) que permiten crear páginas web atractivas
- Permite definir el estilo de cada elemento HTML de manera exacta.
- Permite escalar tamaños en función del tamaño de la pantalla.
- Permite crear plantillas de estilos que pueden importarse en otros HTML.

CSS

Los estilos CSS deben darse de alta en un fichero acorde para ello, no obstante:

- Pueden declararse dentro de un HTML mediante la etiqueta `<style>`
- Pueden aplicarse directamente sobre un elemento concreto en la propiedad "style".

Lo correcto es llevar los estilos a un o unos ficheros CSS, pero se permite añadir pequeños retoques directamente sobre el HTML.

Definición estilo CSS

- Un selector es un identificador para saber sobre qué elemento HTML debemos aplicar el estilo.
- Existen diferentes tipos de selectores.
- Se pueden crear jerarquías de estilos.
- Podemos definir estilos por defecto para los elementos HTML estándar.
- Los estilos pueden sobrescribir a otro, el orden de sobrescritura es el mismo en el que se cargan los ficheros css o se lee el fichero.

Tipos de selectores:

- De elemento HTML:

h1, table, div...

- De identificador

Todos los elementos HTML cuya propiedad "id" tenga un determinado valor, tendrán ese estilo.

- De clase

Todos los elementos HTML cuya propiedad "class" tenga un determinado valor tendrán ese estilo.

Como se define un estilo CSS

Declaración:

```
Selector {  
  
propiedad: valor;  
...  
}
```

Ejemplo:

```
H1 {  
  color: #CC9900;  
  ...  
}  
  
#id{  
  color: #CC9900;  
  ...  
}  
  
.class{  
  color: #CC9900;  
  ...  
}
```

- Normalmente se le aplica un estilo por defecto a los elementos HTML para conformar una plantilla.
- La personalización definitiva se suele realizar haciendo uso de la propiedad "class".
- Cuando incluyamos plantillas en un fichero HTML hay que estar seguros que no incorporan estilos CSS que sobrescriban los nuestros.
- En caso de conflicto, debemos asegurar que nuestros estilos quedan situados por encima de los otros (aunque no siempre es posible)

Taller de HTML y CSS

- Crear página web básica con html5, debe contener:
- Etiqueta de títulos
- Párrafo
- Imagen
- Lista ordenada con al menos 3 elementos
- Tabla con nombres, apellidos y grupos agregar bordes y colores
- Enlace a Wikipedia.
- Crear fichero .css donde cambie el fondo de la página, el tipo de letra, tamaño y color.
- Enviar pantallazo al correo **jpintobarrera@gmail.com**

Estructura Correo

- **Asunto:** Taller HTML y CSS -JhonPinto-B191
- **Cuerpo:**

Buenos días,

Envío el taller de HTML y CSS.

Gracias.

Nombre del archivo: TallerHTMLyCSS-JhonPinto-B191.jpg

Enviar a: jpintobarrera@gmail.com

Examen - Corte 2

- Lunes 29 Abril 2024

Taller tienda Online

- Crear página web utilizando **HTML5 (Utilizar el header, nav, aside, section y footer) y CSS** sobre una tienda online, puede seleccionar vender cualquier tipo de producto (Ropa, Zapatos, Videojuegos, Motos, Carros...)
- Debe contener una página de “**Mi perfil**” con nombre, apellidos, correo, otra página de “**Productos**” donde se va mostrar todos los productos con el nombre del producto, descripción, precio, dos imagenes del producto, material del producto, beneficios del producto y un botón para agregar la compra a un carrito de compras, una página de “**Nosotros**”, donde se va a mostrar la visión, misión e historia de la tienda online y por último, una página de “**Contáctenos**”, donde van a crear un formulario web donde el usuario ingrese el nombre, correo y el mensaje. Por favor, ser creativos. Utilizar el CSS para que la tienda se vea con buena apariencia.
- Se pueden guiar de la tienda de Nike, Adidas, Mercado libre
- Enviar el jueves 11 de abril, todo el proyecto comprimido al correo **jpintobarrera@gmail.com**

JavaScript

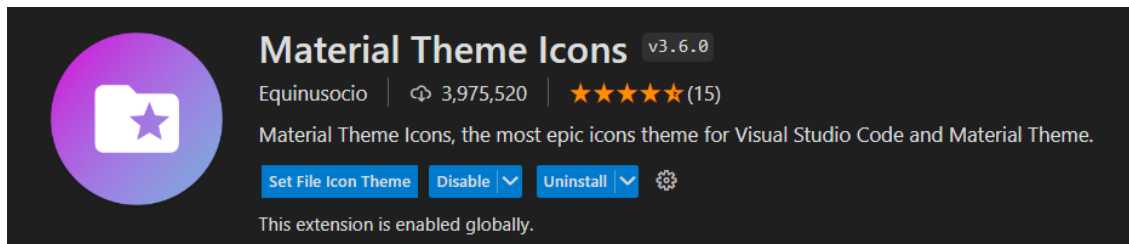
Facultad de Ciencias Naturales e Ingenierías
Tecnología en Desarrollo de Sistemas Informáticos

Docente: Ing. Jhon Pinto
Correo: jpintobarrera@gmail.com

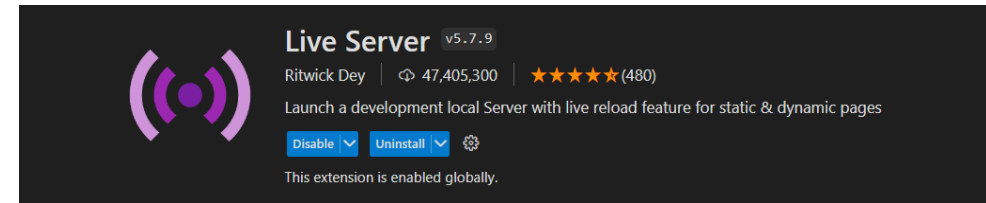
The logo consists of a solid yellow square. Inside the square, the letters 'JS' are written in a bold, black, sans-serif font. The 'J' and 'S' are slightly stylized, with the 'J' having a small hook and the 'S' being a simple, thick stroke.

Extensiones Visual Studio Code

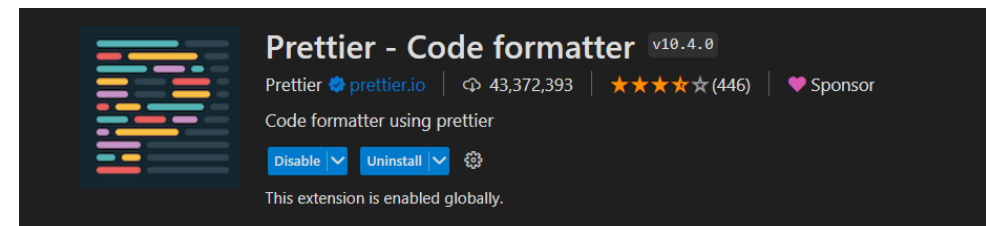
- Live server
- Prettier – Code formatter
- JavaScript code Snippets
- Bracket pair color DLW
- Material Theme icons



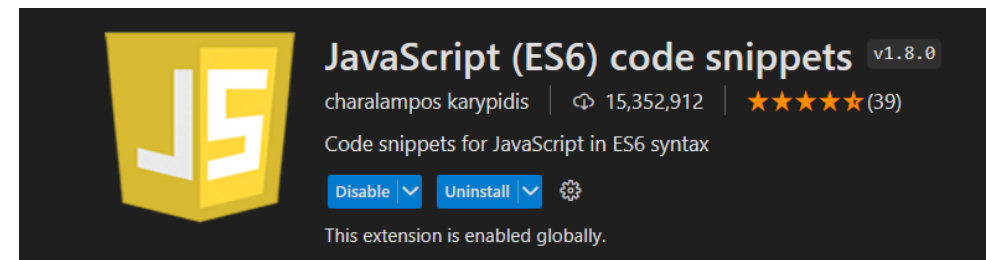
Material Theme Icons v3.6.0
Equinusocio | 3,975,520 | ★★★★★ (15)
Material Theme Icons, the most epic icons theme for Visual Studio Code and Material Theme.
Set File Icon Theme | Disable | Uninstall | ⚙️
This extension is enabled globally.



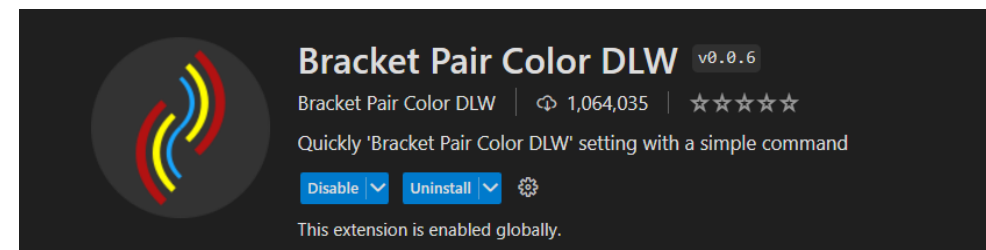
Live Server v5.7.9
Ritwick Dey | 47,405,300 | ★★★★★ (480)
Launch a development local Server with live reload feature for static & dynamic pages
Disable | Uninstall | ⚙️
This extension is enabled globally.



Prettier - Code formatter v10.4.0
Prettier | prettier.io | 43,372,393 | ★★★★★ (446) | ❤️ Sponsor
Code formatter using prettier
Disable | Uninstall | ⚙️
This extension is enabled globally.



JavaScript (ES6) code snippets v1.8.0
charalampos karypidis | 15,352,912 | ★★★★★ (39)
Code snippets for JavaScript in ES6 syntax
Disable | Uninstall | ⚙️
This extension is enabled globally.



Bracket Pair Color DLW v0.0.6
Bracket Pair Color DLW | 1,064,035 | ★★★★★
Quickly 'Bracket Pair Color DLW' setting with a simple command
Disable | Uninstall | ⚙️
This extension is enabled globally.

¿Qué es JavaScript?

- JavaScript es el lenguaje de programación que debes usar para añadir características interactivas a tu sitio web.
- JavaScript es un robusto lenguaje de programación que se puede aplicar a un documento HTML y usarse para crear interactividad dinámica en los sitios web.
- Puedes hacer casi cualquier cosa con JavaScript. Desde crear cosas sencillas como carruseles de imágenes o respuestas a eventos, hasta crear aplicaciones basadas en bases de datos.
- Además, con JavaScript podremos modificar nuestras páginas web, tanto el código HTML como CSS, creando mayor dinamismo a la hora de representar el contenido.
- Podemos también realizar conexiones y coger datos de recursos externos, como pueden ser servidores o APIs.
- JavaScript es uno de los lenguajes más potentes, y el más usado en desarrollo web.

Insertar JavaScript en nuestro proyecto

Puede ser añadido de dos formas:

- **Interna:** Mediante la etiqueta `<script>`

```
<script>
  function onClick(){
    document.getElementById("example").innerHTML = "Example function";
  }
</script>
```

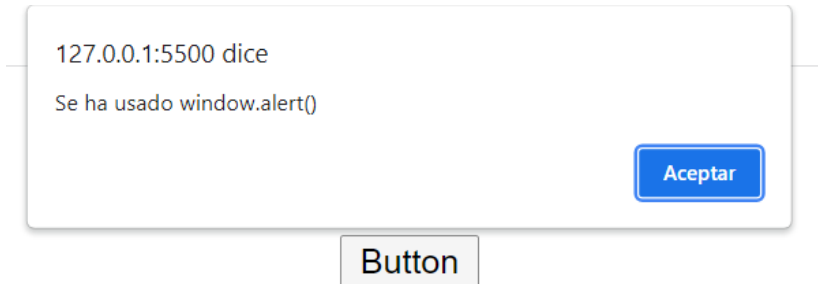
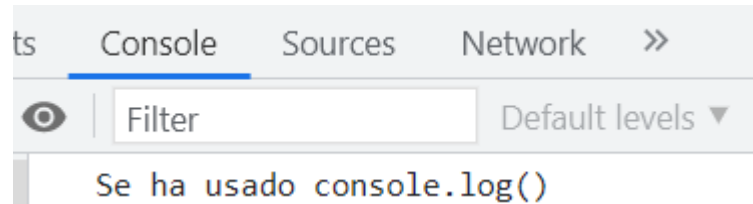
- **Externa:** Mediante la etiqueta `script` llamando a un archivo externo especificando su ruta con el atributo `src`. Tiene la ventaja de que separa el HTML del JS, haciendo que sean más fáciles de leer y mantener.

```
<script src="main.js"></script>
```


Opciones para la salida de datos en JS

JavaScript puede mostrar datos de formas diferentes:

- Mediante la consola del navegador utilizando `console.log()`.
- Escribiendo en una caja de alertas, usando `window.alert()`.
- Escribiendo en un elemento HTML, utilizando `innerHTML`. Para acceder a un elemento podemos usar el método `element = document.getElementById(id)`.
- Escribiendo en la salida estándar de HTML, mediante `document.write(“ ”)`. Esto es solo recomendable en fases de prueba.



Sintaxis de JavaScript

La sintaxis de JavaScript define como el código de los programas está construido. Nos vamos a encontrar lo siguiente:

- **Valores:** Pueden ser literales, constantes o variables, que almacenan los valores de los datos.
- **Operadores:** Hay aritméticos, de comparación y lógicos.
- **Expresiones:** Combinan operadores, valores literales y variables; que dan como resultado un valor.
- **Palabras clave:** Identifican una acción a realizar, como puede ser crear una variable.
- **Comentarios:** Parte del código que no debe ser ejecutado y sirven para la comprensión del código.
- **Identificadores:** Son nombres que sirven para identificar variables, funciones o palabras clave.

```
// Existen 2 tipos de comentarios, de una linea o de múltiples lineas
```

```
/*
```

```
Este es un
```

```
comentario
```

```
de
```

```
múltiples
```

```
lineas
```

```
*/
```

Variables y Constantes

- Las variables son contenedores para valores que se asignan a los datos, y que pueden cambiar durante la ejecución del programa.
- Las constantes son contenedores de valores que no se modifican en ningún momento.
- Estos dos tipos de contenedores deben ser identificados con nombres únicos. Se les asigna un valor mediante el signo “=”, pudiendo ser igualado su valor a otra variable, a un valor o a una expresión.
- Las variables se declaran mediante las palabras **var** (se puede declarar más de una vez) y **let** (si solo se va a declarar una vez).
- Las constantes son declaradas mediante la palabra **const**, y su valor no debe ser reasignado una vez se declara.

Reglas para nombres de Variables

```
// Reglas de las variables:

// Pueden tener: letras, numeros, _
// No pueden iniciar con numero
var 99dias;
var dias99;

var _01;
var 01;

// Estilos para nombrar variables con más de una palabra

// más de una palabra.
var nombreProducto = 'Monitor 30 Pulgadas'; // CamelCase
var nombre_producto = 'Monitor 30 Pulgadas'; //underscore
var NombreProducto = 'Monitor 30 Pulgadas'; // pascal case
var nombreproducto = 'Monitor 30 Pulgadas';
```

Diferencias Let y Var

```
js > JS fundamentos.js > ...
1  //-----var -----
2  // Ambito global
3
4  var equipoFutbol = "Real Madrid";
5  console.log(equipoFutbol);
6
7  var equipoFutbol = "Barcelona";
8  console.log(equipoFutbol);
9
10 //let equipoFutbol = "Bucaramanga";
11 //console.log(equipoFutbol);
12
13 //-----let -----
14 //Ambito de bloque
15 {
16     let equipoFutbol = "Juventus";
17     console.log(equipoFutbol);
18
19     let equipoFutbol2 = "PSG";
20     console.log(equipoFutbol2);
21 }
```

Tipos de datos

El concepto de tipo de datos es muy importante en programación. Nos dan información respecto a los valores de las variables que estamos utilizando, y dictaminan de qué manera podemos utilizarlos.

Las variables en JS pueden cambiar de tipo de forma dinámica, es decir, se puede reutilizar la misma variable para alojar otro tipo de datos. Los más importantes son:

- **Strings:** Contienen cadenas de caracteres entre comillas.
- **Numbers:** Contienen números, tanto enteros como decimales.
- **Booleanos:** Pueden tomar dos valores, true o false.
- **Arrays:** Contienen diferentes elementos, cada uno con un índice comenzando por el 0.
- **Objetos:** Tienen distintas propiedades que se representan mediante pares nombre-valor, separadas por comas.
- **Undefined:** Representa un dato sin valor. Es el que toman las variables al ser declaradas.
- Podemos obtener el tipo de un dato mediante el operador **typeof**.

Operadores aritméticos

- Podemos realizar las operaciones clásicas mediante los signos *, +, -, **, /, %, combinándolas en expresiones con variables, constantes o números.
- Mediante el operador ++ incrementaremos en una unidad el valor de un dato, y haremos lo mismo para restar una unidad con --.
- Podemos ver en esta tabla resumen otros operadores disponibles:

Operador	Ejemplo	Equivalente a
=	x = y	x = y
+=	x += y	x = x + y
-=	x -= y	x = x - y
*=	x *= y	x = x * y
/=	x /= y	x = x / y
%=	x %= y	x = x % y
**=	x **= y	x = x ** y


```
let a = 3; // La variable a se declara con valor 3
let b; // Se declara la variable b
b = a + 4; // b = a + 4 = 7
b = b++; // b se incrementa en 1, vale 8
var c = b; // Se declara c con valor 8
c /= 2; // c = c/2 = 4
const d = 5 ** 2; // Se declara la constante d = 25
c %= a; // c = c mod a = 1
a -= 1; // a = a - 1 = 3
let m = (a + b + c + d)/4 // m es la media de todas, m = 8,625
```

Operadores de comparación

- Las comparaciones devuelven un valor booleano true or false.

Operador	Significado
==	equal to
===	equal value and equal type
!=	not equal
!==	not equal value or not equal type
>	greater than
<	less than
>=	greater than or equal to
<=	less than or equal to

Operadores lógicos

- las operaciones lógicas devuelven un valor booleano según si una condición se cumple o no.

Operador	Uso	Devuelve true si...
AND (&&)	expr1 && expr2	Las dos expresiones son ciertas
OR ()	expr1 expr2	Al menos una de las dos expresiones es cierta
NOT (!)	!expr	La expresión es falsa

String y sus métodos

Ya hemos visto qué son los **strings**, vamos a ver algunos de los métodos y propiedades que tienen:

- **length**: Propiedad para saber la longitud de un string.
- **substring (inicio, final)**: Método con el que podemos sacar una subcadena entre dos posiciones del string.
- **toUpperCase ()**: Para convertir sus caracteres a mayúsculas.
- **concat (string, string,...)**: Este método devuelve la unión de varios strings.
- **split (string)**: Método que nos permite convertir el string en un array con las subcadenas que quedan separadas por el carácter que le especifiquemos.
- **indexOf (string)**: podemos saber cual es la posición de un carácter o un string en la cadena.
- **slice (inicio, final)**: Para extraer la subcadena contenida entre dos posiciones.
- **replace (string, string)**: Permite remplazar una subcadena por otra.
- **includes (string)**: Devuelve un valor booleano según si nuestra cadena contiene la que le pasamos.

Examen 2 corte

- Lunes 29 abril 2024

Definición de números

```
// Crear Números

const numero1 = 30;
const numero2 = 20;
const numero3 = 20.20;
const numero4 = .10213;
const numero5 = -3;

// Otra forma de crear número

const numero8 = new Number(20);
console.log(numero8);
```

Definición operaciones

```
// Suma
resultado = numero1 + numero2;
// Resta
resultado = numero1 - numero2;
// Mult
resultado = numero1 * numero2;
// Division
resultado = numero1 / numero2;
// Modulo
resultado = numero1 % numero2;

console.log(resultado);
```

Incremento y decremento

```
let puntaje = 5;  
puntaje++;  
puntaje--;  
++puntaje;  
--puntaje;  
  
puntaje += 3;  
puntaje -= 3;  
  
console.log(puntaje);
```


Objetos en JS

Los objetos son tipos de datos que pueden contener varios atributos y métodos propios. Estos atributos o propiedades se escriben con pares nombre:valor, y pueden ser accedidos desde cualquier parte mediante `objectName.propertyName` o `objectName["propertyName"]`.

Los métodos son acciones que pueden ser realizadas en los objetos, son funciones propias del mismo. Podemos acceder a ellos de forma similar a los atributos con la forma `objectName.functionName()`.

Mediante la palabra clave **this**, podemos hacer referencia al objeto en cuestión dentro de sus propios métodos. Podemos definir sus propiedades tanto de forma conjunta como de una en una.

```
let car = {
  model: 'Ford Focus',
  km: 120000,
  registration: '0123ABC',
  getData: function getData(){ // Devuelve Ford Focus 120000km 0123ABC
    return this.model + ' ' + this.km + 'km ' + this.registration;
  }
}
```

Array y sus métodos

Ya hemos visto lo que son los **arrays**, vamos a ver ciertos métodos propios que tiene:

- **pop ()**: Borra el último elemento de un array.
- **push (elemento)**: Añade un nuevo elemento al final de un array,
- **sort ()**: Este método ordena un array de forma alfabética.
- **reverse ()**: Invierte el orden de los elementos de un array.
- **forEach (function)**: Llama a una función para cada elemento del array.
- **splice (índice, numero de elementos, elementos a insertar)**: Borra tantos elementos como le especifiquemos, además de ser capaz de añadir elementos nuevos en su lugar.

10	34	15	20
v[0]	v[1]	v[2]	v[3]

Métodos en los tipos de datos

En los números, podemos usar el método **toExponential()**, que lo convierte a notación exponencial; el método **toFixed(number)**, que nos formatea el número con la cantidad de dígitos decimales que le especifiquemos; y el método **toPrecision(number)**, que formatea el número con la longitud que le pasemos como parámetro.

Tanto para arrays, booleanos, numbers y otros tipos de datos podemos utilizar el método **toString()**, que convierte nuestros valores a cadenas de caracteres.

El método **parseInt(string, base)** nos devuelve un entero en base 10 a partir de un número o string, definiendo nosotros la base en la que está el primer argumento.

Además de las que están ya implementadas en JS, seremos nosotros mismos los que aumentaremos este número de métodos mediante funciones, sobre todo en los objetos que creemos desde cero.

Funciones

- Las funciones son bloques de código que ejecutan una tarea determinada y que pueden ser ejecutados solo cuando se les llama mediante una parte del código en ejecución, por ejemplo, ante un evento.
- Las funciones se definen mediante la palabra clave **function**, seguida de un identificador, parámetros de entrada entre paréntesis y un bloque de código a ejecutar entre **{ }**.
- Al invocar una función, se le pasan como argumento valores que serán tratados como parámetros dentro de ella.
- Cuando queremos terminar la ejecución una función, usamos la palabra **return**, y volveremos a la parte del código donde fue invocada. Las funciones pueden devolver o no un valor.
- Utilizamos funciones para poder reutilizar código, se define una vez y se usa todas las que se necesite. También pueden ser utilizadas para asignar valores.

```
function sumar() {  
  console.log(2 + 2);  
}
```

// Las funciones deben llamarse, de otra forma en realidad no hacen mucho:

```
sumar(); // se manda llamar por su nombre seguido del parentesis()
```

// Expresión de función - Este tipo de funciones se asigna como si fuera una variable

```
const sumar2 = function() {  
  console.log(3 + 3);  
}
```

```
sumar2(); // se manda llamar de la misma forma
```

Diferencias entre método y función

- en realidad, terminan siendo prácticamente lo mismo, la forma en que se utilizan tiene que ver más que nada en el contexto que son utilizadas..

```
const numero1 = 20;  
const numero2 = "20";  
  
console.log( parseInt(numero2) ); // Esto es una función  
  
console.log( numero1.toString() ); // Esto es un método
```

Estructuras de control

Sentencia "If-else"

La sentencia **if** es una estructura de control condicional que se utiliza para determinar la realización de una acción si una cierta condición se cumple. Tiene la siguiente sintaxis:

- La palabra clave **if**, seguida de la condición y un bloque de código a ejecutar.
- La sentencia opcional **else** sirve para ejecutar una acción si la condición es falsa.
- La sentencia opcional **else if** se utiliza para especificar una nueva condición si la anterior ha sido falsa.

```
if (condicion) {  
    // bloque de código si se cumple la condición  
} else {  
    // bloque si no se cumple la condición  
}
```

```
if (condicion1) {  
    // bloque de código si se cumple la condición 1  
} else if (condicion2) {  
    // bloque si se cumple la condición 2  
}
```


Ejercicio

- Realice un programa(*Condicional*) que lea 3 números enteros (x, y, z) desde la consola y determina si los tres son iguales, si hay dos iguales decir cuales lo son o si todos son diferentes.

Solución

```
1  let x = 1;
2  let y = 2;
3  let z = 3;
4
5  if (x == y) {
6      if (y == z) {
7          console.log("x, y, z son iguales");
8      } else {
9          console.log("x, y son iguales");
10     }
11 } else {
12     if (x == z) {
13         console.log("x, z son iguales");
14     } else {
15         if (y == z) {
16             console.log("y, z son iguales");
17         } else {
18             console.log("x, y, z son diferentes");
19         }
20     }
21 }
22
```

Sentencia “switch-case”

La sentencia **switch** es una estructura de control que se usa para determinar una acción a realizar basada en las diferentes posibilidades que puede tomar una variable o expresión. Tiene la siguiente sintaxis:

- La palabra clave **case** define los valores que puede tomar la variable o expresión que decide qué caso se ejecuta.
- La palabra **break** detiene la ejecución del bloque de código de switch.
- Usamos **default** si no se cumple ninguna condición, y en ese caso ejecuta su código.

```
switch (variable) {  
    case value1:  
        // Bloque de código  
        break;  
    case value2:  
        // Bloque de código  
        break;  
    default:  
        // Bloque de código  
        break;  
}
```

¿Qué son los bucles?

Los bucles se utilizan para ejecutar un bloque de código un número determinado de veces. Pueden ser utilizados para recorrer el mismo código con distintos valores cada vez, para recorrer arrays y para otras muchas funciones.

Los bucles nos permiten ejecutar la misma tarea una y otra vez, lo que se denomina **iteración**. Un bucle normalmente tiene estas características:

- Un contador: Que se inicia a un determinado valor e irá cambiando durante las iteraciones.
- Una condición de salida: Es la que marca el final del bucle, puede ser que el contador llegue a determinado valor, la consecución de un valor que buscábamos,...
- Un iterador: Incrementa el valor del contador hasta alcanzar una condición de salida.

Estos bucles van a ayudarnos de una forma sencilla a realizar tareas repetitivas en pocas líneas. Los más famosos son el bucle **for** y el bucle **while**.

Bucle "for"

El bucle **for** sirve para ejecutar un bloque de código un número determinado de veces. Este bucle tiene la siguiente sintaxis:

- La palabra clave **for**, tres declaraciones entre paréntesis y un bloque de código que se va a ejecutar una o más veces.
- La primera declaración es ejecutada antes de entrar en el bucle, la segunda define la condición para volver a ejecutar el bloque de código posterior y la tercera es ejecutada al terminar cada iteración.

```
for (statement1; statement2; statement3) {  
    //Bloque de código a ejecutar en cada iteración  
}
```

```
let sumaVector = 0;  
let v = [1,2,3,4]  
for (i = 0; i < v.length; i++) {  
    sumaVector += v[i];  
}
```

¿Qué imprime el siguiente código?

```
for (let i = 0; i < 5; ++i) {  
  Console.log(i);  
  if (i % 2 == 0) {  
    Console.log("p");  
  }  
  Console.log(" ");  
}
```

- A) 0p 1 2p 3p 4
- B) 0p 1 2p 3 4p
- C) 0p 1p 2 3p 4
- D) 0 1 2p 3 4p

Bucles “for..in” y “for..of”

El bucle **for..in** se usa para iterar sobre las propiedades de un objeto.

```
for (variable in objeto){  
    // Bloque a ejecutar  
}
```

```
let charIndex = {a: 1, b: 2, c: 3}  
for (x in charIndex){  
    console.log(x);  
    console.log(charIndex[x]);  
}
```

El bucle **for..of** se utiliza para iterar sobre un objeto iterable, como pueden ser arrays, strings, listas de nodos,...

```
for (variable of iterable){  
    // Bloque a ejecutar  
}
```

```
for (char of 'cadena'){  
    console.log(char)  
}
```

Bucle “while” y “do while”

Se utilizan para ejecutar un bloque de código una y otra vez mientras la condición del bucle sea cierta. Tienen la siguiente sintaxis:

- La palabra clave **while**, seguida de una condición que en caso de ser verdadera hace que se ejecute el bucle de nuevo. Puede ir seguida de un bloque de código entre { }. Debemos tener cuidado para no entrar en un bucle infinito.
- La palabra **do**, que especifica el bloque de código a ejecutar.

```
while(condicion){  
  // Bloque a ejecutar mientras se cumpla la condicion  
}
```

```
do {  
  // Bloque a ejecutar mientras se cumpla la condicion  
}  
while(condicion);
```

```
let i = 0;  
let sumaVector = 0;  
let v = [1,2,3,4]  
while(i < v.length){  
  sumaVector += v[i];  
  i++;  
}
```


Diferencias do-while y while

```
let i = 100;  
  
do {  
    Console.log("HoLa");  
} while (i < 5);
```

¿Cuál es la diferencia entre los dos bloques de código?

```
let i = 100;  
  
while (i < 5) {  
    Console.log("HoLa");  
}
```

Bucles y sentencias anidadas

Cuando implementamos código en JS, podemos anidar varias sentencias y bucles, lo que nos permitirá realizar estructuras de control más precisas, recorrer varios objetos a la vez, acceder a datos dentro de iterables con más de un índice, como pueden ser arrays dentro de otros arrays,... entre otras muchas posibilidades.

```
function printOdd(a){
  if (typeof(a) == "number"){
    if (a%2==0){
      console.log('Falso, la variable es par');
    }
    else{
      console.log('Verdadero, la variable es impar');
    }
  }
  else {
    console.log('La variable no es un número');
  }
}
```

```
let m = [[1,2],[2,3],[3,4]]
sumaTotal = 0;
for (let i = 0; i < m.length; i++) {
  for (let j = 0; j < m[i].length; j++){
    sumaTotal += m[i][j];
  }
}
console.log(sumaTotal);
```

Ejercicio FizzBuzz

- Realice un programa que imprima los números de 1 al 100. Para los múltiplos de 3 imprime la palabra "Fizz" en vez del número y para los múltiplos de 5 imprime "Buzz". Para los números que son múltiplos de 3 y de 5, imprime "FizzBuzz".

Solución

```
// Fizz Buzz...

// 3 6 9 12 15 ... FIZZ
// 5 10 15 20 25 ... BUZZ
// 15 30 45 ... Fizz Buzz

for(let i = 1; i < 100; i++ ) {
    if(i % 15 === 0) {
        console.log(`${i} FIZZ BUZZ`)
    } else if(i % 3 === 0) {
        console.log(`${i} fizz`);
    } else if ( i % 5 === 0 ) {
        console.log(`${i} buzz`)
    }
}
```