

Notas JavaScript

Var: genera variables globales que generan un gran lío por lo que

Se implementó **Let** para no romper con la lógica de la programación

No tiene un fuerte tipado, puedes guardar un entero donde tengas un **String** y mas (tipado débil)

Los datos numéricos son sólo **numbers** sin distinción de entero o flotante.

Booleano: **boolean**

Constantes: **const** Usar **Snake uppercase** (EJEMPLO_DE_SNAKE)

null: alguna variable que tiene un valor que no quieras que interprete

Undefined: es el valor de una variable que no se le asignado un valor aún.

Tipos de datos (Básicos)

```
//String es una cadena de texto siempre  
//debe de ir entre comillas dobles  
var vaso = "leche"  
  
//number, No lleva comillas y se refiere  
//al numero con el cual podemos hacer  
//operaciones matemáticas  
  
var edadUsuario =10;  
  
//boolean es un valor con solo 2 salidas  
true o false  
var usuarioPremium = false;  
  
//undefined variables sin inicializar,  
//existe pero no está definido  
var proximaCita= undefined;  
  
// null, es un valor que aún no tenemos  
//definido  
  
var asistenciaInvitado= null;
```

```
/*Tipos de datos primitivos y no primitivos  
string  
number  
boolean  
undefined  
null  
symbol  
object (No es primitivo )
```

Se recomienda **Lower camel case** para nombrar variables

EjemploLowerCamelCase

EjemploUpperCamelCase

procura que el nombre sea específico

Declarando listas

Let **myList**=["Hola", "adios", "pepe", 34] Debido al tipado débil
esta estructura la trabaja como un object se pueden guardar distintos tipos de datos

Existe otro dato llamado **set** el cuales una lista que no admite valores repetidos: Let **mySet**= new set (["Hola", "adios", "pepe", 25])

Si a esta estructura le ingreso un valor repetido va a seguir diciendo que tiene 4 elementos

Map es una estructura también conocida como diccionario que está organizado a manera

Clave valor el mapa no permite claves repetidas

Se declara: Let **myMap** = new myMap ([[pepe, 13], [jose, 25], [Angel, 30]])

para agregar un dato **myMap.set("clave", valor)** **myMap.get("clave")** arroja el valor

para agregar un dato a una lista se usa el método **add()** **mySet.add("pepe")**

Bucles

Declaración de bucles

```

for (const value of myList) {
    console.log(value)
}

let myCounter = 0

while (myCounter < myList.length) {
    console.log(myList[myCounter])
    myCounter++
}

```

de esta manera daría un undefined por el ==

Clases

tipo de objeto concreto

```

// Clases

class MyClass {
    constructor(name, age) {
        this.name = name
        this.age = age
    }
}

let myClass = new MyClass("Brais", 36)
console.log(myClass)
console.log(myClass.name)

```

Sí no le das valores igual se crea con valor indefinido
muestra ambos datos

Enum

```

const MyEnum = [
    DART: "dart",
    PYTHON: "python",
    SWIFT: "swift",
    JAVA: "java",
    KOTLIN: "kotlin",
    JAVASCRIPT: "javascript"
]

const myEnum = MyEnum.JAVASCRIPT
console.log(myEnum)

```

Similar a un mapa le asigna valores a cada clave
es similar al enum de C

Trabajando con JavaScript en nuestro código

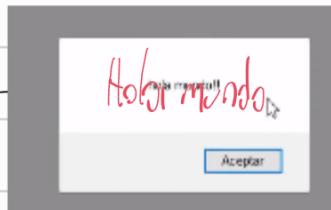
Podemos agregar código utilizando la etiqueta `<script>` en el HTML

```

ReposoJS
1  <!DOCTYPE html>
2  <html lang="es">
3      <head>
4          <meta charset="utf-8" />
5          <title>Reposo JavaScript</title>
6          <script src="main.js" type="text/javascript"></script>
7      </head>
8      <body>
9          <h1>
10             Aprendiendo JavaScript rápido
11         </h1>
12         <p>Hola soy Víctor Robles WEB</p>
13     </body>
14 </html>

```

O sea que carga un script
Alerta usando alert



EXPLORER

- OPEN EDITORS
- MASTER-FRAMEWORKS-JS
 - ReposoJS
 - index.html
 - main.js

Esta es la carpeta del ...

Reposo JavaScript

Victor Robles190

Aprendiendo JavaScript rápido

Hola soy Victor Robles WEB

```

<!DOCTYPE html>
<html lang="es">
  <head>
    <meta charset="utf-8" />
    <title>Reposo JavaScript</title>
  </head>
  <body>
    <h1>Aprendiendo JavaScript rápido</h1>
    <p>Hola soy Victor Robles WEB</p>
    <div id="datos"></div>
  </body>
</html>

```

```

index.html   JS main.js x
ReposoJS > JS main.js > ...
1
2 var nombre = "Victor Robles";
3 var altura = 190;
4
5 var concatenacion = nombre + " " + altura;
6
7 var datos = document.getElementById("datos");
8 datos.innerHTML = concatenacion;
9

```

Movemos script al final del body para que cuando se ejecute ya exista el div con el id "datos". Así podemos controlar donde se muestran los datos

```

var datos = document.getElementById("datos");
datos.innerHTML =
  <h1>Soy la caja de datos</h1>
  <h2>Mi nombre es: ${nombre}</h2>
  <h3>Mido: ${altura}</h3>
;

```

puedo agregar líneas de HTML en el script utilizando comillas simples y el método innerHTML

Aprendiendo JavaScript rápido

Hola soy Victor Robles WEB

Soy la caja de datos

Mi nombre es: Victor Robles

Mido: 190

```

if(altura >= 190){
  datos.innerHTML += '<h1>Eres una persona ALTA</h1>';
} else{
  datos.innerHTML += '<h1>Eres una persona BAJITA</h1>';
}

```

Aprendiendo JavaScript rápido

Hola soy Victor Robles WEB

Soy la caja de datos

Mi nombre es: Victor Robles

Mido: 189 cm

Eres una persona BAJITA

Si no usamos

+ = y solo usamos

= reescribiríamos el valor "HTML" que hay en datos

Otra manera de hacer
el bucle Por

```

for(var i = 0; i<=2020; i++){
  // bloque de instrucciones
  datos.innerHTML += "<h2>Estamos en el año: "+i;
}

```

Llamadas

```

function MuestraMiNombre(nombre, altura){
  var datos = document.getElementById("datos");
  datos.innerHTML =
    <h1>Soy la caja de datos</h1>
    <h2>Mi nombre es: ${nombre}</h2>
    <h3>Mido: ${altura} cm</h3>
  ;
}

MuestraMiNombre("Victor Robles WEB", 190);

```

Document.write
para mostrar
en pantalla

```

function MuestraMiNombre(nombre, altura){
    var misDatos = '';
    <h1>Soy la caja de datos</h1>
    <h2>Mi nombre es: ${nombre}</h2>
    <h3>Mido: ${altura} cm</h3>
}

return misDatos;
}

function imprimir(){
    var datos = document.getElementById("datos");
    datos.innerHTML = MuestraMiNombre("Victor Robles WEB", 190);
}

imprimir();

```

Similar a C usa el return

La función tiene un valor de salida
Puntaje (10) = 9

JavaScript en Close



Introducción a javascript

QUE ES JAVASRIPT? PARA QUE UTILIZARLO?
DIFERENCIAS ENTRE JAVASCRIPT Y JAVA
COMO FUNCIONA HTML+CSS+JAVASCRIPT
FORMAS DE UTILIZAR JAVASCRIPT EN NUESTRA PAGINA WEB
BUENAS PRACTICAS UTILIZADAS EN JAVASCRIPT

podemos usar JS para Front y para Back

Con esto podemos verificar los datos y validarlos

Ejemplo: validar que un hotmail tenga un @

Importante: usar bien nuestras variables

-Ser claros, declarar la variable con el tipo de dato con el que lo vamos a procesar.

Si declaramos un valor como cadena de texto, ya no vamos a poder aplicarle operaciones matemáticas

Más p:ky

Lenguaje compilado, si tienes un error en la linea 16 no corre las 15 anteriores

Amigo cool, jala a todos lados

Lenguaje interpretado

Si tienes un error en la linea 16 sí las 15 lineas anteriores

Solo es back

Se usa para proyectos grandes con demasiada información



VS



Difference Between Java and JavaScript

Back y Front

Compilado: traducción del lenguaje natural al lenguaje máquina

JavaScript nos ayudará a extraer la información de los formularios

Java: nos servirá para manipular estos datos

- La versión que más se usa es la 8 por que los cambios no han sido significativos

- Vamos a usar la versión 17 Java Enterprise JE

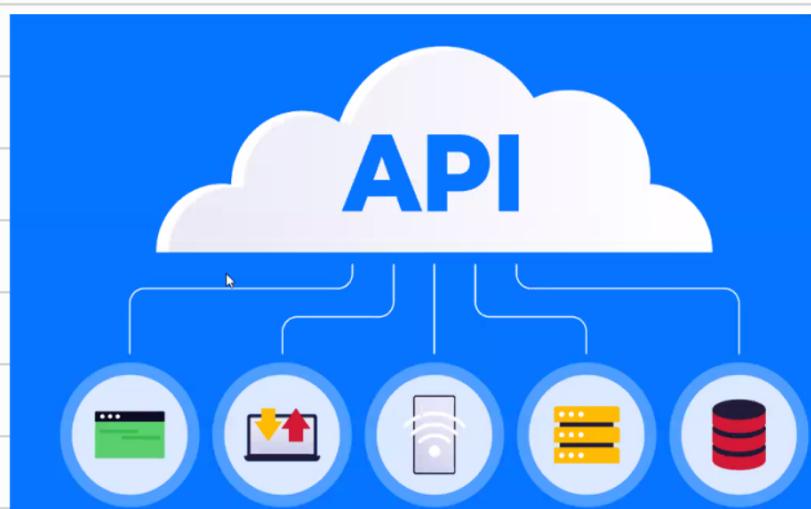
JS puede usarse en
muchos lados

Java Enterprise edition

Versión
para proy
grandes

podemos agregar animaciones con JavaScript

<https://webdesignerwall.com/trends/30-truly-interactive-websites-built-css-javascript>



API: Delimita la interacción

que puede tener el
JSvar 'o.

Hay APIs para todo

<https://fakestoreapi.com/>
Web de APIs

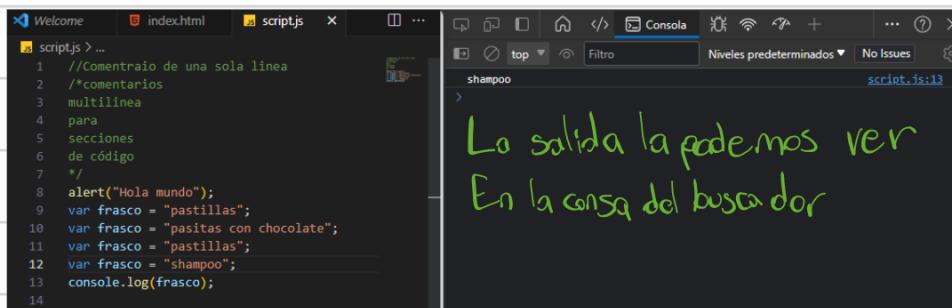
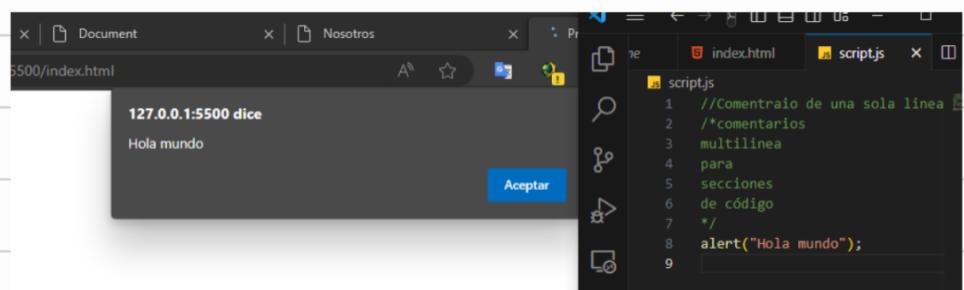


Ejemplos programando

```
index.html > html > head
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>Practicando con JavaScript</title>
7  </head>
8  <body>
9
10
11      <script src="./script.js"></script>
12  </body>
13  </html>
```

La etiqueta script nos permite agregar líneas de Código o llamar un archivo JS

Hola mundo con JS



```
var peso=80;
console.log(typeof(peso));
var peso=peso.toString();
console.log(typeof(peso));
console.log((peso.toString()));
```

Cambiando variable
de tipo de dato number → String

```
console.log(typeof(edad));
var edad=parseInt(edad);
console.log(typeof(edad));
```

Cambiando el tipo de dato ↗
de una variable String → number
Ejercicio

```
console.log("Cambiando de string a
number");
console.log(typeof(numeroTelefono));
var numeroTelefonoNumber= parseInt
(numeroTelefono);
console.log(typeof(
numeroTelefonoNumber));

console.log("Cambiando de number a string
a
string");
console.log(typeof(precioExtracción));
var precioExtracciónString =
precioExtracción.toString();
console.log(typeof(
precioExtracciónString));
```

funciones en JS

¿QUÉ ES UNA FUNCIÓN?

Una función es un trozo de código reutilizable en el que hay un conjunto de instrucciones, este código solo se ejecuta cuando se llama a dicha función.



```

Palabra reservada      Nombre de la función      parámetros      Cuerpo de la función
function miFuncion(p1, p2){
    console.log(p1 + p2);
}
  
```

Terminos

Scope

es el alcance de una variable en nuestro código global: Se puede utilizar una variable dentro de cualquier parte del código

Hoisting: Es cuando puedes invocar alguna función antes de declararla anticipa variables con var y las funciones

Ventajas de las funciones

Nos permiten a ahorrar líneas de código al reutilizar la sección una y otra vez

También nos permite darle abstracción al código para mantener seguridad

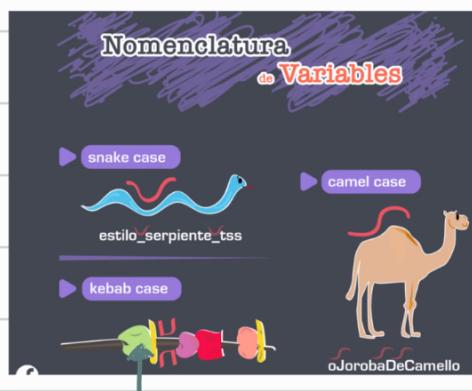
Declarar e invocar funciones

```

//Repaso de funciones
/*
una función en un bloque de
código que realiza una tarea
específica cuando se llama
* cada función de JavaScript es
almacenado como un object function
*/
function saludar(){
    console.log("Hola soy una
    función ");
}
saludar();
/**/
  
```

Recordatorio:

Una función se almacena como object Function.



Separados por guion medio

* Las funciones son simplemente objetos especiales con la capacidad adicional de ser invocados (llamados como funciones). Esto significa que puedes asignar funciones a variables, pasárselas como argumentos a otras funciones y devolver funciones desde otras funciones. Esto facilita la creación de funciones de orden superior, como funciones de devolución de llamada y funciones que generan otras funciones.

```

function suma(a,b){
    return a+b;
}

console.log(suma);
var resultado=suma(5,3);
console.log(resultado);

```

global: Se pone utilizar una variable dentro de cualquier parte del código

- 👉 **Var:** no es tan recomendable por el comportamiento de la variable y la poca adaptabilidad
- 👉 **let** preferible y más seguro de utilizar porque además de funcionar de manera global también puede ser utilizado de manera local
- 👉 **Const:** su valor no puede ser reasignado.

Funciones anónimas

Funciones declaradas sin un nombre y luego asignadas a variables o pasarse como parámetro a otra función

```

//funciones anónimas/
/*
Estas funciones pueden ser declaradas sin un nombre y luego asignadas a variables o pasarse como parámetros en otras funciones la diferencia principal de estas funciones es que declaran sin un nombre
*/
const saludo= function(){
    console.log("Hola vida");
}
saludo();

```

```

const resta= (a,b)=> a-b;
console.log(resta(5,3));
/* esta función flecha es una forma más corta de:
const resta= function(a,b){
    return a-b;
}

```

/*//Funciones flecha//
son una forma más concisa de definir funciones anónimas. se le llama flecha debido a su sintaxis que utiliza ($=>$)
*Básicamente, se trata de reemplazar la palabra function y añadir $=>$ antes de abrir nuestras llaves

Un **bloque** se define como lo que declaras entre {}

Tipos de Función:

Tradicionales

Anónimas

Función Flecha

Control de Flujo

Poder manipular el orden

Cuando hablamos del orden natural en el que se ejecutan las instrucciones de nuestro programa(arriba/abajo, izq/der) estas instrucciones pueden ser una función un console.log, un ciclo, etc.

la importancia que tiene el control de flujo radica en la posibilidad de poder ejecutar cosas instrucciones dependiendo de una condicional (un elemento o instrucción detonante)

Condicionales

son palabras reservadas que nos permiten evaluar si una condición se cumple o no, bajos ciertos criterios. las respuestas o salidas esperadas siempre partiran de un true o false

If

nos sirve para validar si una condición se cumple o no. Si se cumple, entonces ejecutamos un bloque de código que está dentro del if. si no se cumple, entonces no hacemos nada.

Nota: es la condicional más sencilla, pero la más limitante porque solo nos da una opción.

```
if(condicion){  
    //bloque de código que se ejecuta si la condición es verdadera()  
    //ejemplo: Felipe va a ir una fiesta y necesita tener su camisa favorita limpia para poder ir  
}  
/*
```

```
if (camisaLimpia=true){  
    console.log("Felipe va a la fiesta");  
}  
/*
```

else (de otro modo, de lo contrario, si no)

```
if(condición detonante){  
    //bloque de código que se ejecuta si la condición es verdadera
```

```
}else{//bloque de código que se ejecuta si la condición es falsa
```

```
}
```

```
*/  
  
if (camisaLimpia=true){  
    console.log("Felipe va a la fiesta");  
}else{ console.log("Felipe se queda en su casa a hacer lloración")}
```

```

}
/*
else if
Esta condición nos ayuda a jugar con más opciones y más detonantes, para tener muchas
posibilidades en el mismo código. para lograr esto vamos a unir el else con el if
if(condición detonante){
//bloque de código si la condición es verdadera
}else if(condición detonante 2){
//bloque de código si la condición 1 es falsa y la condición 2 es verdadera
}else{
//coloque de código si las dos condiciones son falsas
}
*/
//ejemplo2: Felipe tiene hambre, y quiere comer chilaquiles verdes
if (hambre = true){
console.log("Felipe va a comer");
}else if(chiaquiles= verdes){
console.log("Felipe come chilaquiles")
}else{
console.log("Felipe no come y se pone triste");
}

```

Prompt: es la espera de un valor de entrada como un **Getch**
 O como la versión que es más sencilla como **Readline**
El código se detiene hasta que le ingresemos el valor El dato que ingreso
lo mete como cadena
de texto

- Ingresar especialidad
- Validar usuarios registrados
- Verificar cita
- Generar cita
- Registrar pacientes
- * Buscar información

CUNCIIONES QUE
VAMOS A REALIZAR

Operadores lógicos y Condicionales

Objetivos

Al final de esta sesión seremos capaces de:

- Describir qué son los operadores
- Utilizar los principales operadores de JS: asignación, cadena, lógicos, aritméticos, de comparación e igualdad estricta.
- Definir y leer expresiones.
- Usar operadores para combinar expresiones lógicas.

```
Operador
Un elemento que nos permite realizar una operación entre dos o mas elementos

tipos:
-operadores aritméticos (+, -, *, /)
- operadores de asignación (=, ==, ===)
-operadores de cadenas(toLowerCase,toUpperCase,trim,toString, concat) *tambien conocidos como métodos
operadores lógicos (&&, || , !)
operadores de comparación (<, >, <=, >=, ==, ===)

operadores aritméticos
Son operadores aritméticos que nos permiten tomar valores numéricos como sus operandos y retornar un valor numérico único.
corresponden a operaciones matemáticas

+ suma +
- resta -
* multiplicación
/ división /
% módulo % (a%b te arroja el residuo de la división de a/b )
++ incremento ++ (variable++ incremento en uno en la variable llamada variable)
-- decremento -- (variable-- decremento en uno en la variable llamada variable)

diferencias operador
= igual (asignación de valor a una variable)
== igual (comparación )
=== los números deben de ser totalmente iguales ya que java al comparar 2 variables de tipo de dato diferente hace una conversión

por ejemplo
5=="5" sería igual a true aunque uno sea un número y el otro sea una cadena de texto
5==="5" //false todos deben de ser el mismo tipo y del mismo valor, en ocasiones es más recomendado este para tener un cuidado más estricto de nuestros tipos de datos.

*/
//función para convertir de grados celsius a fahrenheit

function CtoF(gradosC){
    return gradosC*1.8+32;
}
let gradosF=CtoF(12);
console.log("el total de la conversión es: "+ gradosF+ "grados farenheit");
```

Realizamos una conversión de grados C° a F°

add event listener para agregar llamadas a una función a partir de una acción algo así como un callback

```
abrir.addEventListener("click")//define una función que va a ser llamada cada cuando el evento especificado es entregado/le ocurre al objetivo  
los objetivos comunes son elementos o sus hijos, documentos o la ventana, pero el objetivo debe de ser un objeto que soporte la acción  
/*syntax  
addEventListener(type, listener)  
addEventListener(type, listener, options)  
addEventListener(type, listener, useCapture)  
type: un string representando el caso sensible a escuchar (me imagino que si no escucha no actúa)  
listener: el objeto que recibe una notificación (objeto que realiza la interfaz de eventos ) cuando el evento del tipo especificado ocurre  
options (opcional):un objeto que especifica características del oyente del evento las opciones disponibles son  
-capture: un booleano indicando que eventos de este tipo van a ser enviado al oyente registrado antes de ser enviado al Objetivo del evento  
debajo del arbol del DOM. Si no se especifica el valor predeterminado es falso  
-once: un valor booleano que indique que el oyente debe ser invocado como maximo una vez después de ser agregado. si es true, el oyente  
sería removido cuando es invocado si no se especifica es falso  
-passive: Un valor booleano que, si es verdadero, indica que la función especificada por el oyente nunca llamará a preventDefault(). Si un  
oyente pasivo llama a preventDefault(), el agente de usuario no hará nada más que generar una advertencia en la consola.  
-signal: Una señal de aborto. El oyente se eliminará cuando se llame al método abort() del objeto AbortSignal dado. Si no se especifica, no  
se asocia ningún AbortSignal con el oyente  
useCapture: Un valor booleano que indica si los eventos de este tipo se enviarán al oyente registrado antes de enviarse a cualquier EventTarget  
debajo de él en el árbol DOM. Los eventos que se propagan hacia arriba a través del árbol no activarán un oyente designado para utilizar la  
captura. La propagación y captura de eventos son dos formas de propagar eventos que ocurren en un elemento anidado dentro de otro elemento,  
cuando ambos elementos han registrado un identificador para ese evento. El modo de propagación de eventos determina el orden en que los  
elementos reciben el evento. Consulte Eventos de nivel 3 de DOM y Orden de eventos de JavaScript para obtener una explicación detallada.
```

```
outer.addEventListener("click", onceHandler, once);  
outer.addEventListener("click", noneOnceHandler, noneOnce);  
middle.addEventListener("click", captureHandler, capture);  
middle.addEventListener("click", noneCaptureHandler, noneCapture);  
inner1.addEventListener("click", passiveHandler, passive);  
inner2.addEventListener("click", nonePassiveHandler, nonePassive);
```

ejemplos de uso de la sintaxis para la negación de passive o capture o once solo se agrega el none

al exponenciación se realiza utilizando **
primero las variables después las funciones y al ultimo
las líneas de código general como recomendación

Estructuras de control) Ciclos, arreglos y matrices

Objetivos

- Definir y usar matrices
- Acceder a elementos de una matriz en una posición determinada
- Agregar un nuevo elemento a una matriz
- Eliminar un elemento de una matriz
- Usar los diferentes bucles de JavaScript (While y For)
- Explicar las principales diferencias entre While y For.

//Expresiones

//ciclos matrices y arreglos

/*

condiciones: alterar el orden natural del código

estructura: Controlan la ejecución del código

-ciclos o bucles (loops)

También conocidos como bucles, son estructuras de control que nos van a permitir ejecutar un bloque de código de forma repetida, hasta que su cumpla una condición específica

es importante usar ciclos para automatizar tareas repetitivas/

ciclos:

-while

-do while

-for

iterar: proceso de repetir una secuencia de instrucciones estructura de control: herramienta especial para crear algoritmos o programas más dinámicos y flexibles

-while: ejecuta un bloque de código mientras se cumpla una condición específica este ciclo se repite mientras las condiciones sea verdadera, y el bloque de código se ejecuta siempre y cuando la condición se mantenga. la condición se evalúa antes de cada interacción

```
while(condición){  
    //código que se ejecuta  
}
```

```
*/  
while (miCruhMeQuiere){  
    console.log("seré feliz")
```

```
//NOTA esto significa peligro  
while(true){  
}
```

Es un ciclo sin fin ya que siempre es verdadero.

```
while(pacientesEnConsultorio<capacidadPacientes)
  console.log("seguimos dando servicio")
  pacientesEnConsultorio++;
}
```

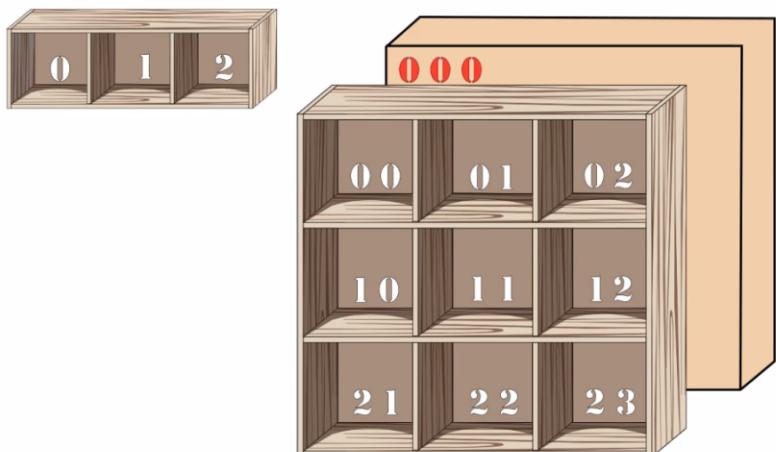
Seguimos dando servicio

Resultado del bucle

muestra el número de veces que se mostró el mensaje

Arreglos

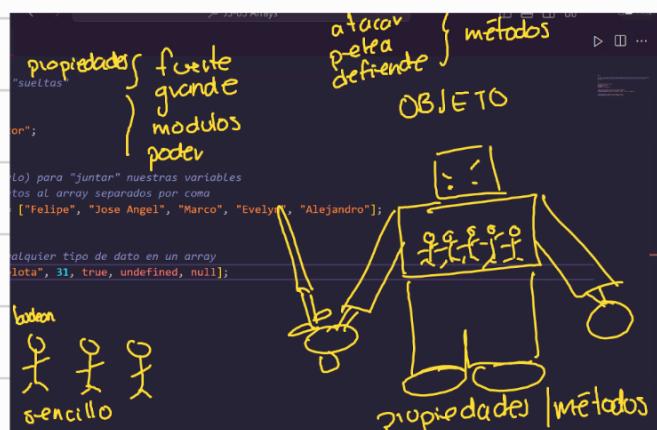
Arreglos



Los elementos de un arreglo

- tienen algo en común
- objeto
- función

* tener un array es sistematizar su almacenamiento y ordenamiento

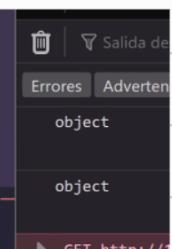


Propiedades: como se ve
Métodos: lo que pue hacer

Cuando los power rangers(datos primitivos/solitos) se ven superados se unen en algo más grande lo cual les da poderes habilidad (métodos)

```
//Usar un array (arreglo) para "juntar" nuestras variables
//Agregamos los elementos al array separados por coma
let personasDeLaCH31 = ["Felipe", "Jose Angel", "Marco",
"Evelyn", "Alejandro"];

//Podemos almacenar cualquier tipo de dato en un array
let cosasRandom = ["Pelota", 31, true, undefined, null];
console.log(typeof(cosasRandom)); //object
console.log(typeof(personasDeLaCH31));
```



//todos los arreglos son objetos, sin importar si albergan datos del mismo tipo o no

// no es lo mismo el index y la cantidad de elementos, ya que el indice inicia desde el cero ya que para la computadora el 0 es muy importante
//hay 5 elementos y su indice de posición van del 0 al 4

```
let pacientes=[];
let dentista=["Dr. Smith","Dra. Garcia", "dr.Hous
dentista[3]="similares";
console.log(dentista[3]);
dentista[8]="Dr. strange";
console.log(dentista);
```



se guardan como vacío igual como si quiero mostrar un elemento de un índice mayor al

Si intentamos asignar el valor de una variable que excede el límite de los índices si se asigna y los demás espaciios

numero de elementos puede mostrar un undefined
Momentos donde no combiene esta caracteristica

```
71 dentistas[3] = "Similares";
72 dentistas[8] = "Dr. Strange"; //undefined por Dr.Strange
73
74
75 console.log(typeof(dentistas[7]));
```

let productos = [0, 1, 3, 4, 5]

NOS MOSTRARIA UN ESPACIO
VACIO EN NUESTRA PÁGINA

```
//metodos de array
let listaDelSuper = ["gansitos", "Cocas", "Arroz", "atún",
"verduritas"];

//metodo length para conocer la longitud del array
console.log("La cantidad de elementos de mi array es de ", +
listaDelSuper.length);

//Método push() para poner elementos al final de mi array
listaDelSuper.push("jabón para ropa", "jabón para trastes");

console.log(listaDelSuper);
//metodo pop() para eliminar elemento del final del array
listaDelSuper.pop(); //siempre debe de llevar parentesis ya que
es una función
console.log(listaDelSuper);
```



Método push() para agregar
elementos al fin
y pop para eliminar el último

```
La cantidad de elementos de mi array es de 5
arreglosYBucles.js:56:9

▶ Array(7) [ "gansitos", "Cocas", "Arroz", "atún", "verduritas",
"jabón para ropa", "jabón para trastes" ]
arreglosYBucles.js:61:9

▶ Array(6) [ "gansitos", "Cocas", "Arroz", "atún", "verduritas",
"jabón para ropa" ]
arreglosYBucles.js:64:9
```

Unshift, shift, IndexOf

```
//tenemos métodos para agregar valores al final al inicio o en medio,
pero si agregamos algo en medio o al final todos los indices cambian
para JS(se ejecuta desde el lado del cliente) no le importa los
recursos que eso necesite pero Java no deja hacer eso
// osea que el arreglo inconsistente

//Agregar un elemento al principio del array con método unshift()
listaDelSuper.unshift("sabritones");
console.log(listaDelSuper);

//Eliminar el primer elemento del array shift elimina el elemento al
principio del array
listaDelSuper.shift();
console.log(listaDelSuper);

//Saber la posición de las verduritas
console.log(listaDelSuper.indexOf("verduritas"));
// no es lo mismo Verduritas que verduritas estos métodos son case
sensitive
```



```
▶ Array(7) [ "sabritones", "gansitos", "Cocas",
"Arroz", "atún", "verduritas", "jabón para ropa" ]
arreglosYBucles.js:70:9

▶ Array(6) [ "gansitos", "Cocas", "Arroz", "atún",
"verduritas", "jabón para ropa" ]
arreglosYBucles.js:74:9

4
arreglosYBucles.js:78:9
```

```

//método splice para agregar, eliminar o reemplazar elementos
//nombreArreglo.splice(inicio, cantidadElementosAEliminar,
elementoAInsertar1, elementoInsertar2)
/*
inicio: a partir de donde va a iniciar a sobre escribir los
elementos
cantidadDeElementosAEliminar: cuantos elementos se elimina

*/
console.log(listaDelSuper);
listaDelSuper.splice(2,0,"Cacahuates", "Manzana");//no borra
nada y solo agrega valores

console.log(listaDelSuper);
//eliminamos y modificamos

listaDelSuper.splice(5,2, "platano");
console.log(listaDelSuper);

```

Salida

```

▶ Array(6) [ "gansitos", "Cocas", "Arroz", "atún",
"verduritas", "jabón para ropa" ]
arreglosYBucles.js:89:9

▶ Array(8) [ "gansitos", "Cocas", "Cacahuates",
"Manzana", "Arroz", "atún", "verduritas", "jabón para
ropa" ]
arreglosYBucles.js:92:9

▶ Array(7) [ "gansitos", "Cocas", "Cacahuates",
"Manzana", "Arroz", "platano", "jabón para ropa" ]
arreglosYBucles.js:96:9

```

```

98 //método .map()
99 let numeros= [1,2,3,4,5];
100
101 //crear otro arreglo con numeros multiplicados x2 [2,4,6,8,10]
102
103 //toma cada valor del arreglo y lo mete en la función
104
105 let numerosPorDos=numeros.map(function(numero){
106   return numero*2;
107 })
108 console.log(numeros);
109
110 console.log(numerosPorDos);

```

Salida

```

▶ Array(5) [ 1, 2, 3, 4, 5 ]
▶ Array(5) [ 2, 4, 6, 8, 10 ]

```

```

let pacientesConsultorio=[{
  nombre:"Felipe",
  edad: 31,
},
{
  nombre: "Fatima",
  edad: 26,
},
{
  nombre: "jesus",
  edad: 51,}];

//Función para agregar el texto "Necesita atención especial"
function agregarEstadoSalud(paciente){
  let estado="Saludable";

  //si el paciente tiene más de 40 años
  if(paciente.edad>40){
    estado="Necesita atención especial";
  }

  return{
    ...paciente,//Copia del paciente
    estadoSalud:estado,//para agregar un nuevo campo o variable (como la edad o el nombre)
  }
}

//Vamos a aplicar la función en cada elemento del arreglo con el map
let pacientesConsultorioConEstado=pacientesConsultorio.map(agregarEstadoSalud);//cuando la mandamos
a llamar a partir del .map no necesitamos poner los parentesis

console.log(pacientesConsultorio);
console.log(pacientesConsultorioConEstado);

```

Salida

```

▼ Array(3) [ ..., ..., ... ]
  ▶ 0: Object { nombre: "Felipe", edad: 31 }
  ▶ 1: Object { nombre: "Fatima", edad: 26 }
  ▶ 2: Object { nombre: "jesus", edad: 51 }
  ▶ <prototype>: Array []
arreglosYBucles.js:147

▼ Array(3) [ ..., ..., ... ]
  ▶ 0: Object { nombre: "Felipe", edad: 31, estadoSalud: "Saludable" }
  ▶ 1: Object { nombre: "Fatima", edad: 26, estadoSalud: "Saludable" }
  ▶ 2: Object { nombre: "jesus", edad: 51, estadoSalud: "Necesita atención especial" }
  ▶ length: 3
  ▶ <prototype>: Array []
arreglosYBucles.js:147

```

Ejercicio Individual - Carreritas

En una carrera se tienen los siguientes corredores con su respectiva posición:

- Primer lugar: Roberto
- Segundo lugar: Andrea
- Tercer lugar: Jorge
- Cuarto lugar: Ramiro
- Quinto lugar: Sofía

Después de 3 vueltas, Jorge adelanta a Roberto, Ramiro adelanta a Jorge y Roberto se lesiona y sale de la carrera. Además, Andrea baja una posición, Ramiro mantiene su lugar y el quinto lugar es reemplazado por José.

¿Cómo quedan las posiciones después de esas 3 vueltas?

Pistas

```
//Jorge adelanta a Roberto (cambiamos las posiciones de Jorge y Roberto)
//Ramiro adelanta a Jorge (cambiamos las posiciones de Ramiro y Jorge)
//Roberto se lesionó y sale de la carrera (eliminamos el primer elemento)
//Andrea baja una posición (intercambiamos las posiciones de Andrea y del último corredor)
//El quinto corredor es reemplazado por José (cambiamos las posiciones del último corredor con José)
//Imprimimos arreglo final
```

Tarea

71/Septiembre/2023

Método forEach

Por Each() ejecuta la función callback una vez por cada elemento del array en orden ascendente. No es invocada para índices que hayan sido eliminados o no inicializado.

Cuando forEach y cuando map?

Aunque parezcan muy similares, fueron creados para distintas cosas el método map se utiliza para hacerle una transformación o cambio. genera una copia de elemento para hacerlo

(*ForEach se suele utilizar para hacer impresiones del arreglo*)

Método sort

El método sort es una de las funciones globales que nos permiten manipular los elementos de un array. Sirve para ordenar los elementos según un criterio, criterio que se puede definir con el uso opcional de un callback. Si usamos el método sort sin el uso de un callback este ordenará los elementos en orden ascendente por defecto.

```

14 //la variable paciente está declarada e inicializada (tiene un nombre, y tiene un valor a pesar de que su arreglo esté vacío)
15
16 //Los índices del arreglo están undefined porque no hay elementos pero si hay espacios disponibles
17 let pacientes=[];
18 let citas=[];
19
20
21 function registrarPaciente(nombre,edad){
22     //Crear un array dentro de otro array
23
24     let paciente = {
25         nombre,
26         edad,
27         citas: []
28         //el uso de ":" es porque ya estamos utilizando un = dentro de la misma sentencia (let paciente) se llama escape de caracteres
29     };
30 }
31

```

Lo veremos mejor
Con objetos y
objetos JSON.

```

//Crear un array dentro de otro array
let paciente = {
    nombre,
    edad,
    citas : [
        citasManiana = [
            citasManianaAntesDeLas9 = [],
        ],
        citasTarde = [],
    ]
};

}

```

Via vez que ya usamos
podemos usar cuantos
quieramos
||, + =

```

registroCitasYPacientes.js > registrarPaciente
20
21 //Los índices del arreglo están undefined, porque no hay elementos pero si hay espacios disponibles
22 let pacientes = []; //pacientes = definido
23
24
25 // Función para registrar un nuevo paciente
26 function registrarPaciente(nombre, edad){
27
28     //Crear un array dentro de otro array
29     let paciente = {
30         nombre,
31         edad,
32         citas : [] vacío
33     };
34
35     //Agregar paciente al array
36     pacientes.push(paciente);
37
38
39 }
40
41 console.log("Felipe el 'amante numero 1' de los chilaquiles verdes");

```

- ① Pedir datos (invocación)
- nombre
- edad
- ② Almacenar en variables { }
- ③ Par de datos se guarde en el arreglo

```

// Función para registrar un nuevo paciente
// Paso 1. Ejecución de mi función. Pido dos datos
function registrarPaciente(nombre, edad){

    //Crear un array dentro de otro array
    //Paso 2. Guardo cada par de datos (nombre y la edad en una variable llamada paciente)
    let paciente = {
        nombre,
        edad,
        citas : []
    };

    //Agregar paciente al array
    //Paso 3. Ya que tengo el par de datos, Le hago un push a mi array para guardar al paciente
    pacientes.push(paciente);
    return paciente;
}

```

no estamos
mandando la
variable solo le
mandamos un dato al ultimo

(let i actúa como global)
(al no estar dentro
de algún bloque de
código)

```

        console.log("Edad del paciente: " , paciente.edad); //31
        console.log("Citas registradas: ");
    }

//Registrar un paciente desde la invocacion de la funcion
registrarPaciente
let paciente1 = registrarPaciente("Felipe", 31);
let paciente2 = registrarPaciente("Naruto", 20);
let paciente3 = registrarPaciente("Dr. Simi", 50);

//Registrar La cita de un paciente desde la invocacion de la
funcion registrarCita
registrarCita(paciente1, "2023-09-12", "1:00 PM");

//Mostrar informacion del paciente
mostrarInfoPaciente(paciente1);
mostrarInfoPaciente(paciente2);
mostrarInfoPaciente(paciente3);

```

Citas registradas:
Nombre del paciente: Naruto
Edad del paciente: 20
Citas registradas:
Nombre del paciente: Dr. Simi
Edad del paciente: 50
Citas registradas:

▶ GET http://127.0.0.1:55... [HTTP/1.1 404 Not Found 0ms]

```

60
61 // Funcion para mostrar La informacion del paciente con su cita
62 function mostrarInfoPaciente(paciente){
63     console.log("Nombre del paciente: " + paciente.nombre);
64     console.log("Edad del paciente: " , paciente.edad); //31
65     console.log("Citas registradas: ");
66
67     //forEach para imprimir citas (parameter) indice: any
68     paciente.citas.forEach((cita, indice) =>{
69         console.log("Indice: " + indice + " Fecha: " + cita.fecha +
70         "Hora: " + cita.hora)
71     });
72
73
74 //Registrar un paciente desde La invocacion de la funcion
75 registrarPaciente
76 let paciente1 = registrarPaciente("Felipe", 31);
77 let paciente2 = registrarPaciente("Naruto", 20);
78 let paciente3 = registrarPaciente("Dr. Simi", 50);
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
279
280
281
282
283
284
285
286
287
288
289
289
290
291
292
293
294
295
296
297
298
299
299
300
301
302
303
304
305
306
307
308
309
309
310
311
312
313
314
315
316
317
318
319
319
320
321
322
323
324
325
326
327
328
329
329
330
331
332
333
334
335
336
337
338
339
339
340
341
342
343
344
345
346
347
348
349
349
350
351
352
353
354
355
356
357
358
359
359
360
361
362
363
364
365
366
367
368
369
369
370
371
372
373
374
375
376
377
378
379
379
380
381
382
383
384
385
386
387
388
389
389
390
391
392
393
394
395
396
397
398
399
399
400
401
402
403
404
405
406
407
408
409
409
410
411
412
413
414
415
416
417
418
419
419
420
421
422
423
424
425
426
427
428
429
429
430
431
432
433
434
435
436
437
438
439
439
440
441
442
443
444
445
446
447
448
449
449
450
451
452
453
454
455
456
457
458
459
459
460
461
462
463
464
465
466
467
468
469
469
470
471
472
473
474
475
476
477
478
479
479
480
481
482
483
484
485
486
487
488
489
489
490
491
492
493
494
495
496
497
498
499
499
500
501
502
503
504
505
506
507
508
509
509
510
511
512
513
514
515
516
517
518
519
519
520
521
522
523
524
525
526
527
528
529
529
530
531
532
533
534
535
536
537
538
539
539
540
541
542
543
544
545
546
547
548
549
549
550
551
552
553
554
555
556
557
558
559
559
560
561
562
563
564
565
566
567
568
569
569
570
571
572
573
574
575
576
577
578
579
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
598
599
599
600
601
602
603
604
605
606
607
608
609
609
610
611
612
613
614
615
616
617
618
619
619
620
621
622
623
624
625
626
627
628
629
629
630
631
632
633
634
635
636
637
638
639
639
640
641
642
643
644
645
646
647
648
649
649
650
651
652
653
654
655
656
657
658
659
659
660
661
662
663
664
665
666
667
668
669
669
670
671
672
673
674
675
676
677
678
679
679
680
681
682
683
684
685
686
687
688
689
689
690
691
692
693
694
695
696
697
697
698
699
700
701
702
703
704
705
706
707
708
709
709
710
711
712
713
714
715
716
717
718
719
719
720
721
722
723
724
725
726
727
728
729
729
730
731
732
733
734
735
736
737
738
739
739
740
741
742
743
744
745
746
747
748
749
749
750
751
752
753
754
755
756
757
758
759
759
760
761
762
763
764
765
766
767
768
769
769
770
771
772
773
774
775
776
777
778
778
779
779
780
781
782
783
784
785
786
787
788
788
789
789
790
791
792
793
794
795
796
797
797
798
799
800
801
802
803
804
805
806
807
808
809
809
810
811
812
813
814
815
816
817
818
819
819
820
821
822
823
824
825
826
827
828
829
829
830
831
832
833
834
835
836
837
838
839
839
840
841
842
843
844
845
846
847
848
849
849
850
851
852
853
854
855
856
857
858
859
859
860
861
862
863
864
865
866
867
868
869
869
870
871
872
873
874
875
876
877
878
878
879
879
880
881
882
883
884
885
886
887
888
888
889
889
890
891
892
893
894
895
896
897
897
898
899
900
901
902
903
904
905
906
907
908
909
909
910
911
912
913
914
915
916
917
918
919
919
920
921
922
923
924
925
926
927
928
929
929
930
931
932
933
934
935
936
937
938
939
939
940
941
942
943
944
945
946
947
948
948
949
949
950
951
952
953
954
955
956
957
958
959
959
960
961
962
963
964
965
966
967
968
969
969
970
971
972
973
974
975
976
977
978
978
979
979
980
981
982
983
984
985
986
987
987
988
988
989
989
990
991
992
993
994
995
996
997
997
998
999
999
1000
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1048
1049
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1078
1079
1079
1080
1081
1082
1083
1084
1085
1086
1087
1087
1088
1088
1089
1089
1090
1091
1092
1093
1094
1095
1095
1096
1096
1097
1097
1098
1099
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1148
1149
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1178
1179
1179
1180
1181
1182
1183
1184
1185
1186
1187
1187
1188
1188
1189
1189
1190
1191
1192
1193
1194
1195
1195
1196
1196
1197
1197
1198
1199
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1218
1219
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1228
1229
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1238
1239
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1248
1249
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1268
1269
1269
1270
1271
1272
1273
1274
1275
1276
1277
1277
1278
1278
1279
1279
1280
1281
1282
1283
1284
1285
1286
1287
1287
1288
1288
1289
1289
1290
1291
1292
1293
1294
1295
1296
1296
1297
1297
1298
1298
1299
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1309
1310
1311
1312
1313
1314
1315
1316
1317
1317
1318
1318
1319
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1328
1329
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1338
1339
1339
1340
1341
1342
1343
1344
1345
1346
1347
1347
1348
1348
1349
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1358
1359
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1368
1369
1369
1370
1371
1372
1373
1374
1375
1376
1377
1377
1378
1378
1379
1379
1380
1381
1382
1383
1384
1385
1386
1387
1387
1388
1388
1389
1389
1390
1391
1392
1393
1394
1395
1396
1396
1397
1397
1398
1398
1399
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1409
1410
1411
1412
1413
1414
1415
1416
1417
1417
1418
1418
1419
1419
1420
1421
1422
1423
1424
1425
1426
1427
1427
1428
1428
1429
1429
1430
1431
1432
1433
1434
1435
1436
1437
1437
1438
1438
1439
1439
1440
1441
1442
1443
1444
1445
1446
1447
1447
1448
1448
1449
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1458
1459
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1468
1469
1469
1470
1471
1472
1473
1474
1475
1476
1477
1477
1478
1478
1479
1479
1480
1481
1482
1483
1484
1485
1486
1487
1487
1488
1488
1489
1489
1490
1491
1492
1493
1494
1495
1496
1496
1497
1497
1498
1498
1499
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1509
1510
1511
1512
1513
1514
1515
1516
1517
1517
1518
1518
1519
1519
1520
1521
1522
1523
1524
1525
1526
1527
1527
1528
1528
1529
1529
1530
1531
1532
1533
1534
1535
1536
1537
1537
1538
1538
1539
1539
1540
1541
1542
1543
1544
1545
1546
1547
1547
1548
1548
1549
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1558
1559
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1568
1569
1569
1570
1571
1572
1573
1574
1575
1576
1577
1577
1578
1578
1579
1579
1580
1581
1582
1583
1584
1585
1586
1587
1587
1588
1588
1589
1589
1590
1591
1592
1593
1594
1595
1596
1597
1597
1598
1598
1599
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1609
1610
1611
1612
1613
1614
1615
1616
1617
1617
1618
1618
1619
1619
1620
1621
1622
1623
1624
1625
1626
1627
1627
1628
1628
1629
1629
1630
1631
1632
1633
1634
1635
1636
1637
1637
1638
1638
1639
1639
1640
1641
1642
1643
1644
1645
1646
1647
1647
1648
1648
1649
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1658
1659
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1668
1669
1669
1670
1671
1672
1673
1674
1675
1676
1677
1677
1678
1678
1679
1679
1680
1681
1682
1683
1684
1685
1686
1687
1687
1688
1688
1689
1689
1690
1691
1692
1693
1694
1695
1696
1697
1697
1698
1698
1699
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1709
1710
1711
1712
1713
1714
1715
1716
1717
1717
1718
1718
1719
1719
1720
1721
1722
1723
1724
1725
1726
1727
1727
1728
1728
1729
1729
1730
1731
1732
1733
1734
1735
1736
1737
1737
1738
1738
1739
1739
1740
1741
1742
1743
1744
1745
1746
1747
1747
1748
1748
1749
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1758
1759
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1768
1769
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1778
1779
1779
1780
1781
1782
1783
1784
1785
1786
1787
1787
1788
1788
1789
1789
1790
1791
1792
1793
1794
1795
1796
1797
1797
1798
1798
1799
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1809
1810
1811
1812
1813
1814
1815
1816
1817
1817
1818
1818
1819
1819
1820
1821
1822
1823
1824
1825
1826
1827
1827
1828
1828
1829
1829
1830
1831
1832
1833
1834
1835
1836
1837
1837
1838
1838
1839
1839
1840
1841
1842
1843
1844
1845
1846
1847
1847
1848
1848
1849
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1858
1859
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1868
1869
1869
1870
1871
1872
1873
1874
1875
1876
1877
1877
1878
1878
1879
1879
1880
1881
1882
1883
1884
1885
1886
1887
1887
1888
1888
1889
1889
1890
1891
1892
1893
1894
1895
1896
1897
1897
1898
1898
1899
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1909
1910
1911
1912
1913
1914
1915
1916
1917
1917
1918
1918
1919
1919
1920
1921
1922
1923
1924
1925
1926
1927
1927
1928
1928
1929
1929
1930
1931
1932
1933
1934
1935
1936
1937
1937
1938
1938
1939
1939
1940
1941
1942
1943
1944
1945
1946
1947
1947
1948
1948
1949
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1958
1959
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1968
1969
1969
1970
1971
1972
1973
1974
1975
1976
1977
1977
1978
1978
1979
1979
1980
1981
1982
1983
1984
1985
1986
1987
1987
1988
1988
1989
1989
1990
1991
1992
1993
1994
1995
1996
1997
1997
1998
1998
1999
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2009
2010
2011
2012
2013
2014
2015
2016
2017
2017
2018
2018
2019
2019
2020
2021
2022
2023
2024
2025
2026
2027
2027
2028
2028
2029
2029
2030
2031
2032
2033
2034
2035
2036
2037
2037
2038
2038
2039
2039
2040
2041
2042
2043
2044
2045
2046
2047
2047
2048
2048
2049
2049
2050
2051
2052
2053
2054

```

Manejando el DOM

12 de septiembre 2023

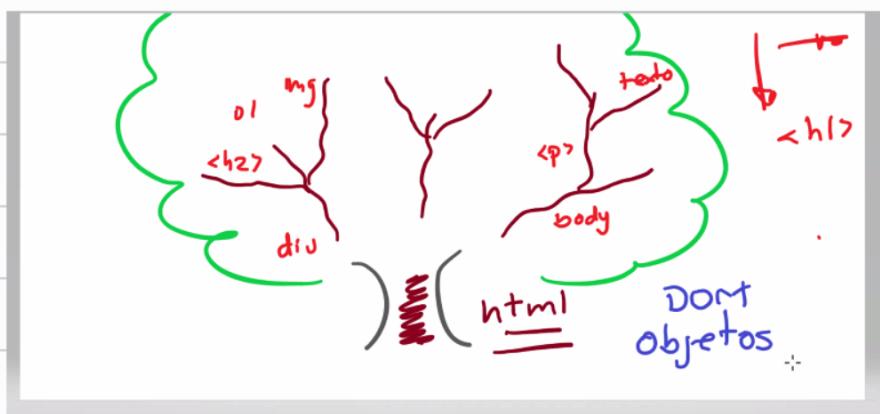
Objetivos

Al final de la sesión, seré capaz de:

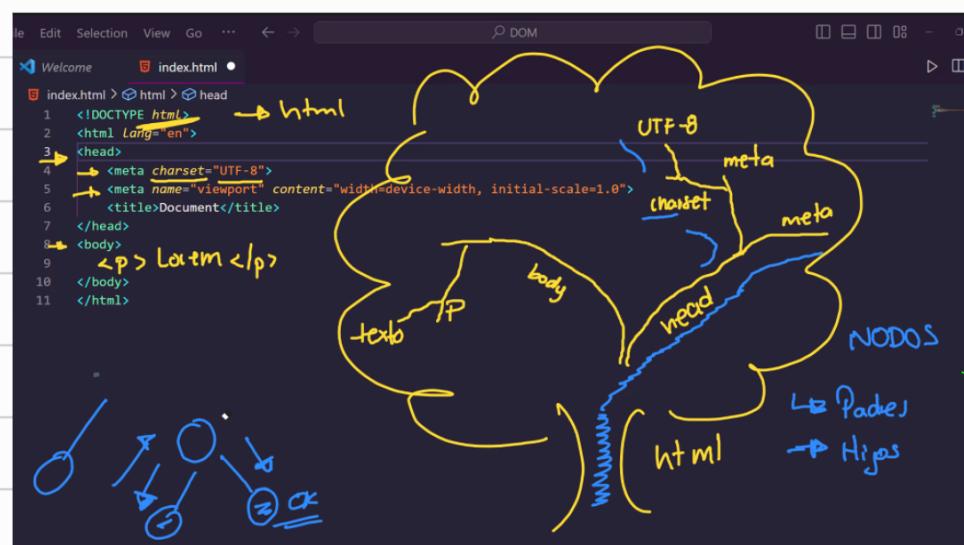
- Explicar el **DOM**
- Usar la **API DOM** para acceder a elementos HTML (`document.getElementById`)
- **Acceder** y **modificar** una clase y atributos de objeto DOM
- **Ocultar** y **mostrar** elementos HTML usando la propiedad de visualización
- **Bonus: Leer** y **validar** datos de **formulario** usando JavaScript

(Como en el video de youtube
de hace tu propio menú de
hamburguesa)

Document
Object
Model

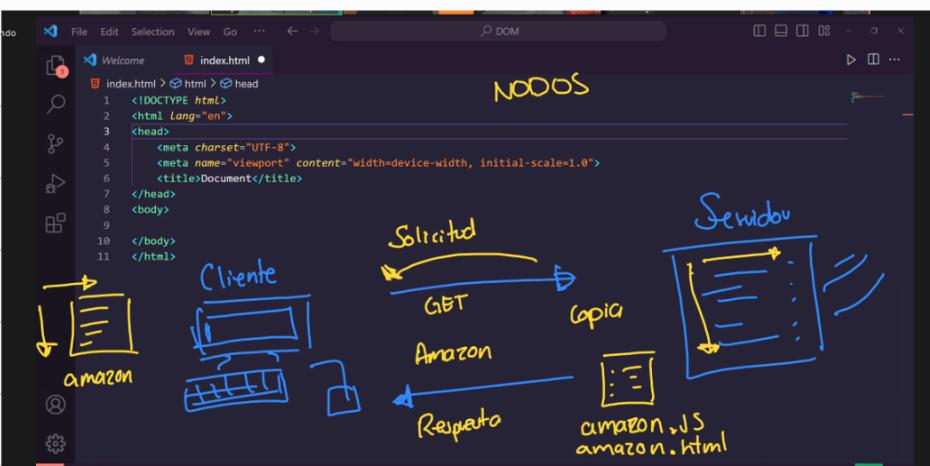


Un modelo
donde cada ele-
mento del HTML (La
Base de todo) es visto
como un objeto



después podemos hacer
objetos a partir de
alguno de sus objetos
padre.

Tenemos un objeto que alberga
Otros objetos



En un e-commerce el DOM ayuda a solo tener una plantilla de productos, de imagen de un artista.

