CHECKPOINT7

1.¿Qué diferencia a Javascript de cualquier otro lenguaje de programación?

Lo que hace tan único a JavaScript es que JavaScript es el único lenguaje de programación que un navegador web puede entender.

Hace más de 25 años, algunos desarrolladores decidieron desarrollar un lenguaje de programación que un navegador pudiera entender. Cualquiera de los otros lenguajes de programación que existen como Java , Ruby , Python … todos deben estar en un servidor.

Deben estar en el servidor, y si estás creando un sitio web, ese servidor tiene que desarrollar todos los procesos y empaquetar ese código de una manera que el navegador realmente pueda interpretarlo.

Más información :

https://kinsta.com/es/base-de-conocimiento/que-es-javascript/

2.¿Cuáles son algunos tipos de datos JS?

En JavaScript podemos decir que admite ocho tipos de datos: número, BigInt, booleano, string o cadena, valor nulo, indefinido, símbolo y objeto.

Más información y ejemplos:

https://ifgeekthen.nttdata.com/es/tipos-de-datos-y-operadores-en-javascript#:~:text=En%20JavaScript%20podemos%20decir%20que,%2C%20indefinido%2C%20s%C3%ADmbolo%20y%20objeto.

https://developer.mozilla.org/es/docs/Web/JavaScript/Data_structures

Estos serían algunos ejemplos:

```
// x es un string o cadena
let x = "Hola Mundo";
// x es un número
x = 100;
// x es un booleano
x = true;
```

3.¿Cuáles son las tres funciones de String en JS?

JavaScript ofrece numerosas funciones predefinidas que facilitan el trabajo con cadenas de texto. Entre las posibilidades que ofrecen estas funciones tenemos el extraer un carácter, extraer un fragmento de cadena, separar una cadena en múltiples cadenas indicando un separador, etc.

En las siguientes tablas resumimos las principales funciones disponibles. Las hemos agrupado según el uso más habitual que se hace de ellas en funciones que se usan habitualmente para modificar las cadenas, funciones que se usan habitualmente para extraer subcadenas o caracteres y funciones que se usan para determinar el índice de posición de un carácter bajo ciertas condiciones.

FUNCIONES MODIFICADORAS

Función	Tarea y comentarios	Ejemplo
toUpperCase()	Transforma la cadena a mayúsculas	textoUsuario.toUpperCase()
toLowerCase()	Transforma la cadena a minúsculas	textoUsuario.toLowerCase()
replace('carácterA', 'caracterB')	Reemplaza la primera aparición de carácterA por carácterB en la cadena.	textoUsuario.replace('e', 'E');
replace (/carácterA/g, 'carácterB')	Reemplaza todas las apariciones de carácterA por carácterB en la cadena. Tener en cuenta que el primer parámetro no va entrecomillado.	textoUsuario.replace (/e/g, 'E') Reemplaza todas las e minúsculas por E mayúsculas. Si queremos reemplazar los espacios escribiremos: textoUsuario.replace (/ /g, 'E') dejando un espacio entre las barras. Es la forma de expresar el replaceAll de otros lenguajes.
replace (/cadenaA/g, 'cadenaB')	Reemplaza todas las apariciones de la subcadena cadenaA por cadenaB. Tener en cuenta que el primer parámetro no va entrecomillado.	textoUsuario.replace (/plo/g, 'XX') Reemplaza todas las apariciones de plo sustituyéndolas por XX

Más información en :

https://developer.mozilla.org/es/docs/Web/JavaScript/Reference/Global_Objects/String

4.¿Qué es un condicional?

Los condicionales son estructuras que permiten elegir entre la ejecución de una acción u otra. Son una condición, como bien indica su nombre, así que podemos pensar en ellos como si fueran el “si” condicional que usamos dentro de una frase.

El condicional if () javascript, si en español, nos permite introducir una situación que debe ser verdadera para que una acción suceda. Si es verdadera se ejecuta y sino no. También tenemos el condicional else que se añade como condición de que si el if no es verdadero y no se ejecuta se ejecuta otra segunda acción alternativa.

El condicional *else if* en javascript nos permite plantear una situación extra que debe ser verdadera para que otra acción se ejecute. Este condicional solo tendrá sentido si *if,* el primer condicional, es falso.

EJEMPLO:

```
updateTeams (result) {

if (result.homeGoals > result.awayGoals) {

// gana el local

} else if (result.homeGoals < result.awayGoals) {

// gana el visitante

} else {

// empatan
```

Como ves, utilizamos primero la palabra clave *if* para definir la primera situación verdadera, luego la palabra clave *else if* para la segunda situación y finalmente la palabra clave *else* en javascript como última opción.

Más info y ejemplos :

https://makeitrealcamp.gitbook.io/javascript-book/condicionales

5.¿Qué es un operador ternario?

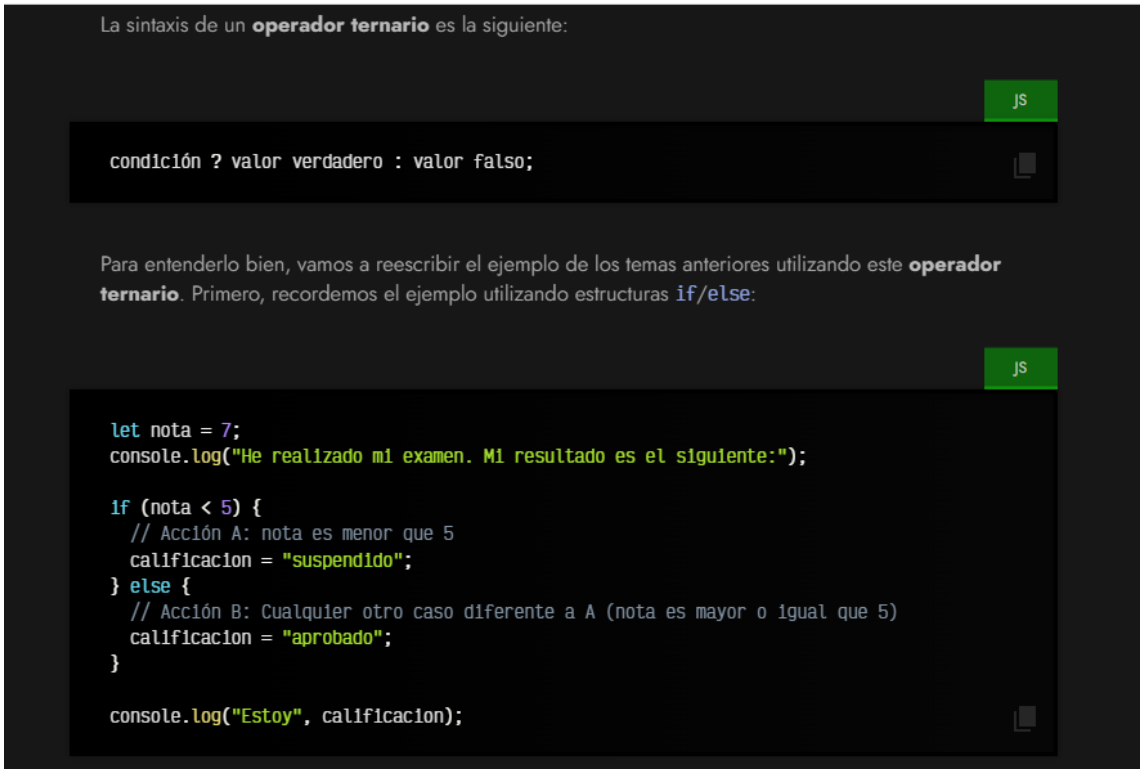
El operador ternario es una alternativa al condicional if/else de una forma mucho más compacta y breve, que en muchos casos resulta más legible. Sin embargo, hay que tener cuidado, porque su sobreutilización puede ser contraproducente y producir un código más difícil de leer.

El operador condicional (ternario) es el único operador en JavaScript que tiene tres operandos. Este operador se usa con frecuencia como atajo para la instrucción if.

Más información y ejemplos en:

https://lenguajejs.com/fundamentos/estructuras-de-control/operador-ternario/

https://developer.mozilla.org/es/docs/Web/JavaScript/Reference/Operators/Conditional_operator



La sintaxis de un **operador ternario** es la siguiente:

JS

```
condición ? valor verdadero : valor falso;
```

Para entenderlo bien, vamos a reescribir el ejemplo de los temas anteriores utilizando este **operador ternario**. Primero, recordemos el ejemplo utilizando estructuras `if/else`:

JS

```
let nota = 7;
console.log("He realizado mi examen. Mi resultado es el siguiente:");

if (nota < 5) {
  // Acción A: nota es menor que 5
  calificacion = "suspendido";
} else {
  // Acción B: Cualquier otro caso diferente a A (nota es mayor o igual que 5)
  calificacion = "aprobado";
}

console.log("Estoy", calificacion);
```

También puede realizar más de una operación por caso, separándolas con una coma:

JS

```
var stop = false,
  age = 23;

age > 18
  ? (alert("OK, puedes continuar."), location.assign("continue.html"))
  : ((stop = true), alert("Disculpa, eres menor de edad!"));
```

6.¿Cuál es la diferencia entre una declaración de función y una expresión de función?

Si la función se declara como una declaración separada en el flujo del código principal, eso se llama “Declaración de función”. Si la función se crea como parte de una expresión, se llama “Expresión de función”. Las Declaraciones de Funciones se procesan antes de ejecutar el bloque de código.

Las diferencias de sintaxis son claras, las expresiones de funciones principalmente están a la DERECHA del símbolo igual, y las declaraciones no lo tienen.

```
// declaraciones de función

// Función con nombre
function suma1(a, b) {
  return a + b
}

// ========================

// expresiones de función

// Función anónima
var suma2 = function(a, b) {
  return a + b
}

// Función con nombre
var suma2 = function suma2(a, b) {
  return a + b
}

// Función de flecha (anónima por defecto)
var suma2 = (a, b) => {
  return a + b
}
```

Más información y ejemplos en:

https://blog.koalite.com/2011/10/javascript-diferencias-entre-declaracion-de-funcion-y-expresion-con-funcion/

7.¿Qué es la palabra clave "this" en JS?

En JavaScript existe una palabra clave muy importante que se llama «this». «this» hace referencia a un objeto en memoria. El objeto que haga referencia dependerá del lugar donde la palabra se llame.

Cuando se usa en una función, this simplemente apunta a un objeto al que está vinculado. Responde a la pregunta "de dónde debería obtener algún valor o datos":

This en JavaScript es una palabra clave muy utilizada dentro de funciones y clases, pues tiene un valor flexible. This significa esto en español y, como su nombre indica, hace referencia al objeto en cuestión. Es decir, si estamos creando cualquier función, la palabra clave this se usará para representar o llamar al objeto que dicha función está modificando.

A continuación, te ponemos un ejemplo con una variable llamada *ejemplo*:

```
const ejemplo = {
    color: azul,
func: function ( ) {
 return this.color;
    },
    };
```

Esta variable tiene dos propiedades, *function* y *color*. Aquí, *this* hace referencia a la constante *ejemplo*, pues es el objeto al que pertenece en este contexto. Entonces, **al hacer que la función nos devuelva la propiedad *this.color*, el programa entiende que lo que hará realmente es devolvernos la propiedad *ejemplo.color*.**

Más información y ejemplos :

https://www.freecodecamp.org/espanol/news/como-usar-la-palabra-clave-this-en-javascript/

EJERCICIO(RESUELTO EN EL ARCHIVO DE JS)

-Cree una función JS que acepte 4 argumentos. Suma los dos primeros argumentos, luego los dos segundos y multiplícalos. Si el número creado es mayor que 50, la consola registra "¡El número es mayor que 50!". Si es más pequeño, la consola registra "¡El número es menor que 50!"

```
function ejercicio() {

var num1= 5;

var num2= 2;

var num3= 8;

var num4= 7;

var resultado =(num1+num2)*(num3+num4);

if(resultado>50){

console.log ( "¡El número es mayor que 50!");

}

else{

console.log ( "¡El número es menor que 50!");

}

return resultado;

}

console.log(ejercicio());
```