Contenido:

JavaScript necesario antes de comenzar con REACT

1. Template Literals EMACScript 6:

Consiste en utilizar ``` para poder ejecutar código JavaScript .

```
const nombre = "Isaías";
const apellido = "FL";
console.log(nombre + " " + apellido);
//Ahora con template Literals
console.log(`${nombre} ${apellido}`);
```

2. Short and Property names (acortadores de nombres):

```
const nombre = "María";
const apellidos = "Salmerón";
const edad = 48;

const persona = {
  nombre: nombre,
  apellidos: apellidos,
  edad: edad,
};
//Esto se sustituye por:
const persona = {
  nombre,
  apellidos,
  edad,
};
```

3. Arrow Function:

```
function nombre() {
  return "hola";
}

// con Arrow sería:
const nombre = () => "hola";
// más Arrow function😉
```

```
const suma= (a=0,b=0) => a+b;

const factorial = numero => {
    if (numero <= 1) {
        return 1;
    }
    return numero * factorial (numero -1);
}
```

4. Destructuring de Objetos y Arrays:

```
const persona = {
  nombre: "Isaías",
  apellidos: "FL",
  cursos: ["Java", "JavaScript", "React", "Node"],
};
// Destructuring🕗
const fullName = ({ nombre, apellidos }) => `${nombre} ${apellidos}`;
console.log(fullName(persona));              // Obtenemos 'Isaías FL'
// Destructuring dentro del propio objeto🕗
const nuevaPersona = {
    ...persona,
    nombre: "Lucas",
};          // creamos un nuevo objeto, sobreescribiendo la clave nombre.
// también podemos extraer y renombrar a la vez🕗
const { nombre: nuevoNombre } = nuevaPersona;
console.log(nuevoNombre);                    //Obtenemos "Lucas"
```

También se podría hacer la destructuración con Arrays.

```
// destructurin con arrays.
const [primerCurso, ...otrosCursos] = persona.cursos;

// primerCuso almacena "Java"
// otrosCursos almacena un array con el resto de cursos.

const nuevoObjeto = { nuevoNombre, primerCurso };
// aquí creamos un objeto con laves nuevoNombre que contiene el valor de
este
// y otra clave primerCurso que contiene el valor de esa misma.
// hay un acortador en la declaración del objeto.
```

```
// también podemos hacer una destructuración de los parámetros
// en una función:

const sumar = (...numeros) => numeros.reduce((total, num) => total + num
,0);

console.log(sumar(1,3,5,6));  //Obtendría 15 que es el resultado de sumar
esos números.
// lo interesante es que le puedo pasar cuantos números quiera y
funcionaría.
```

5. Parámetros por defecto:

```
function suma(a, b) {
  return a + b;
}
// daría error con suma(3)
//habría que:
function suma(a, b) {
  if (a === undefined || b === undefined) {
    console.log("hay que poner 2 parámetros");
    return;
  }
  return a + b;
}

// los parámetros por defecto serían:

function suma(a = 0, b = 0) {
  return a + b;
}
```

6. Spread y Rest Operator (sintaxis extendida):

Ya la hemos visto en el punto anterior de pasada, pero de todas formas la recalco por la importancia que entraña en React y en JavaScript en general:

```
const array = [1, 2, 3, 4, 5];
const otroArray = [6, 7, 8, 9, 10];
//antes:
const nuevoArray = array.concat(otroArray);
//ahora:
```

```
const nuevoArray = [...array, ...otroArray];

// con objetos sería:
const obj1 = {
  a: "a",
  b: "b",
};
const obj2 = {
  c: "c",
  d: "d",
};
// antes:
const nuevoObjeto = Object.assign({}, obj1, obj2);
//ahora
const nuevoObjeto = { ...obj1, ...obj2 };
```

7. Módulos en EMACScript 6:

Los módulos nos permiten crear diferentes ficheros para separar el código y de esta forma con sólo importarlo donde deseemos podamos reutilizar código y componentes.

```
// un fichero llamado suma.js:(exportación por defecto)
export default function suma(a,b){
    ....
}
// otro fichero llamado resta.js: (exportación nombrada)
export function resta(a,b) {
    ...
}

// Fichero donde voy a usar las funciones creadas en sus respectivos
ficheros y que han sido exportadas:
import suma from "./suma" //(si el fichero se llama suma.js)
import {resta, otra_u_otras_funcions} from "./resta"

function calculadora(a,b) {
    suma(a,b);
    resta(a,b);
}
```

En React lo veremos casi siempre así:

```
import React, {useState, useEffect, Fragment } from 'react'; //(no tengo
que poner ./ porque es un módulo instalado con npm)
```

8. Operador condicional ternario:

```
const coche = {
    marca: "Peugeot",
    model: "207",
}
let modelo;
if (coche.model) {
  modelo = coche.model;
} else {
  modelo = "X";
}
console.log(modelo);

// con operaciones ternarias:
let modeloNew = coche.model ? coche.model : "X";
console.log(modeloNew);
```

9. Promise / Async Await:

Visto en temas anteriores de JavaScript.

10. Nullish coalescing operator (??):

El operador de fusión nula (??) es un operador lógico que devuelve su operando del lado derecho cuando su operando del lado izquierdo es null o undefined, y de lo contrario devuelve su operando del lado izquierdo. Mejor a través de un ejemplo:

```
function suma(a,b) {
    a = a == null ? 0 : a;
    b = b == null ? 0 : b;
    return a + b;
```

```
}
// esto se puede mejorar con ??:

function suma(a,b) {
    a = a ?? 0 ; //(esto se leería a es igual a "a" y si no 0 , dando
igual si a es cero o undefined.
    b = b ?? 0 ;
    return a + b;
}
suma(2);
```

Más información en el siguiente enlace:

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/Nullish_coalescing_operator

11. Optional chaining:

Una de las características que trae la especificación ECMAScript 2020 es un nuevo operador: el ***optional chaining***, presente en otros lenguajes como Swift y que llega a Javascript para hacernos más sencillo trabajar con objetos cuyas claves no estamos seguros de que existan.

```
const persona = {
    name: 'Gerardo',
    email: 'mail@mail.com'
};
  //Si tratamos de hacer lo siguiente:
const titulo = persona.trabajoActual.titulo;  // Error
//el intérprete de Javascript nos arrojará un error al tratar de acceder
// a una propiedad ( TrabajoActual ) cuyo valor es undefined .
//Gracias al operador optional chaining podemos prevenir este error del
//siguiente modo:


const titulo = person?.trabajoActual?.titulo;
console.log(titulo);
//De modo que, si o bien persona o bien su propiedad trabajoActual son
undefined ,
//el valor que obtendrá la variable titulo será undefined .
```

12. Métodos importantes para los ARRAYS que se deben dominar:

Existen numerosos métodos que poseen los arrays que se van a utilizar de forma continuada con React. Es recomendable practicarlos. Destacamos:

a. find() y findIndex():
b. some():
c. includes():
d. every():
e. map(): *****
f. filter(): ****
g. reduce(): ****
h. forEach():
i. flat() y flatMap():