* 3 tipos de datos, strings, números y booleanos
* Arreglos, matrices, Arrays, colección de datos que se pueden ordenar, utilizando métodos: .length (largo), contiene posiciones arrancando desde cero
* \*Null / undefined

METODO LENGTH

const nombres = ['Octavio', 'Osvaldo'];

console.log(nombres[0]);

traer el ultimo valor del array: nombres[nombres.length -1]

**ULTIMA POSICION DEL ARREGLO**

METODO PUSH > ingresar al final del arreglo un nuevo valor

nombres.push('Carlos');,

['Octavio', 'Osvaldo', 'Carlos']

METODO POP> elimina la última posición

nombres.pop('Giovanni');

['Giovanni', 'Octavio', 'Osvaldo', 'Carlos']

**PRIMERA POSICION DEL ARREGLO**

METODO .UNSHIFT > incorpora un Nuevo elemento en la primera posición

nombres.unshift('Giovanni');

['Giovanni', 'Octavio', 'Osvaldo', 'Carlos']

METODO .SHIFT >elimina de la primera posición

nombre.shift();

['Octavio', 'Osvaldo', 'Carlos']

RECORRER ARREGLOS

for(let i = 0; i < nombres.legth; i++) {

console.log(nombres[i]);

}

Map > patrón iterador

06-10-2021

Objetos son también colecciones de datos/clave valor

Los array son colecciones contienen datos que están relacionados

Clave valor> no están relacionados

El objeto esta inicializado con llaves const nuevoObjeto = { }, clave valor

const usuario = {

username: 'jj',

numeroFavorito: 100,

}

ACCEDER A LOS DATOS DE UN OBJETO

* Podemos acceder a un dato del clave valor objeto usuario

usuario.username

* Notación de corchete

Con notación de corchete: usuario['username']

INCORPORAR NUEVO CAMPO AL OBJETO

usuario.nombre = ‘Octi’ o usuario['username'] = ‘Octi’

ELIMINAR CAMPO

delete usuario.numeroFavorito

METODOS PARA TRABAJAR CON LOS OBJETOS

Nombre del objeto . metodo

const usuario = {sername: 'jj',

numeroFavorito: 100,

saludar: () => console.log('hola' + username);

}

REACT

Librería declarativa, eficiente, reactiva  
Función principal construcción de interfaces de usuario, como objetivo pretende brindar funciones de ayuda, colecciones de definiciones de clase para la reutilización de código.

Pensar en componentes, Component driver Development (CDD)

Estructura Básica>  
Importe  
Estilos  
Propiedades  
Componentes

Virtual DOM, renderizar paginas más rápidamente, manteniendo el código entendible y fácil de manejar, porque lo manejas por DOM.

* Los datos son levantados
* Cuáles son los elementos del DOM que necesitan ser actualizados

HOOKS: impiden que cambios sean actualizados por atrás que no sean visibles por mi al comienzo.

Observación de cambio de datos:  
- Virtual DOM: Crea representación del DOM en la memoria

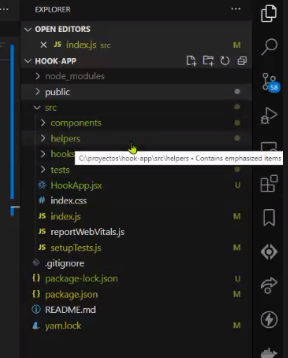
Metodologia CDD: (Metodologia TDD)

Respecto a los COMPONENTES:  
Single Responsability Principle (principio de responsabilidad Unico)   
La responsabilidad del componente es única

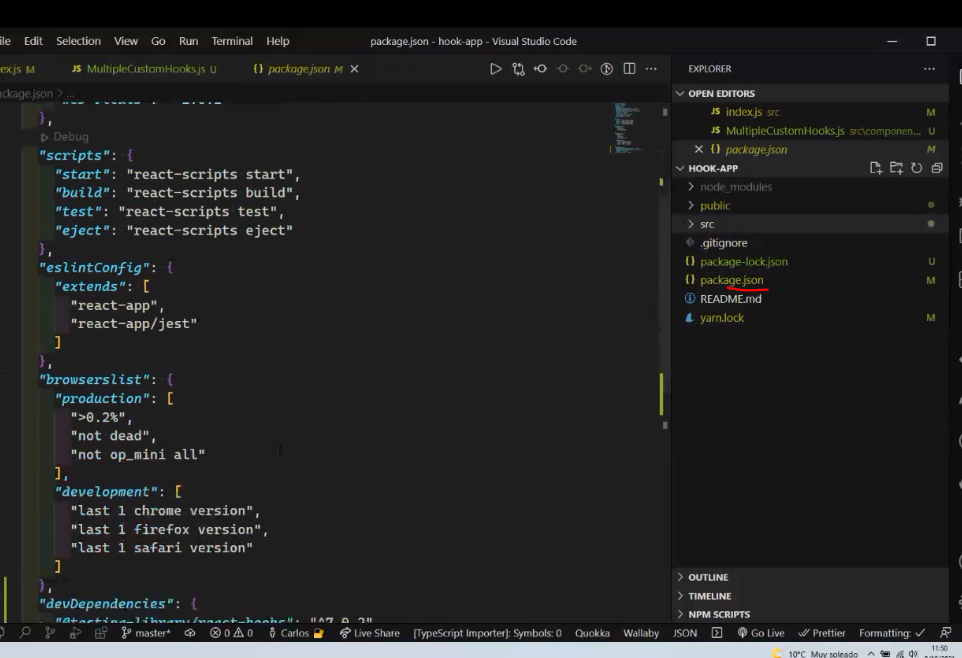
* Componentes grandes a chiquitos, todos subdividos en funciones

JS es un lenguaje, agrupa librerías a través de WebPack  
Babel, librería, transforma JSX y transpila a javascript (en uso de typescript usa TSX)

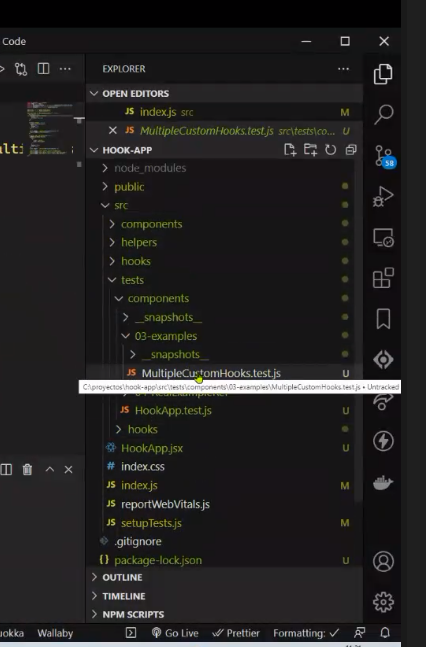
Compilar, Convierte lenguaje de alta a lenguaje bajo para que lo lea la maquina  
Transcripción, convierte un lenguaje a otro



Dependencias



NPM INSTALL, va a pakage.json contiene todo lo necesario para correrlo y



Callbacks hell

Promesas solucionan los callbacks  
.then  
.catch

Javascript siempre es Sincronico

Asincronico: assync-away