REACTJS COURSE (FERNANDO HERRERA)

¿QUÉ ES REACT?

Es una librería que nos permite hacer aplicaciones web. Está hecho para trabajar con aplicaciones de toda magnitud. Es declarativa y eficiente. Unicamente hace el cambio en el elemento que ha modificado su valor. Trabaja con componentes (pequeñas piezas encapsuladas) sencillas y fáciles de mantener.

JSX = JS + XML

```
<div id="root">

</div>

<script type="text/babel">
  const divRoot = document.querySelector("#root");
  const nombre = "Goku";
  const h1Tag = <h1>Hola, soy { nombre }</h1>;
  ReactDOM.render(h1Tag, divRoot);
</script>
```

El Objeto **ReactDOM.render()** permite renderizar el objeto JSX dentro de una etiqueta seleccionada.

BABEL

Es un transpilador de Javascript moderno. Esta herramienta transforma el código hacia una versión compatible para todos los exploradores.

Nueva Sintaxis:

```
const restApi = {
  personajes: ["Goku", "Vegeta"],
};
console.log(restApi.personajes?.length);
```

Sintaxis Compatible:

```
var _restApi$personajes;

const restApi = {
  personajes: ["Goku", "Vegeta"],
};
console.log(
  (_restApi$personajes = restApi.personajes) === null ||
    _restApi$personajes === void 0
    ? void 0
```

```
    : _restApi$personajes.length
);
```

JAVASCRIPT MODERNO

Objetos:

Son elementos de memoria que pueden tener diferentes datos. Es necesario saber que inicialmente debe de crearse la estructura:

```
const persona = {
  nombre: "Tony",
  apellido: "Stark",
  edad: 45,
  direccion: {
    ciudad: "New York",
    zip: 55321321,
    lat: 14.3232,
    lng: 34.9233321,
  },
};
```

Si queremos crear otro objeto, con la referencia de memoria que ya poseemos, necesitamos saber que no podemos asignarlo directamente a otra variable, ya que estariamos cambiando la referencia inicial de nuestro objeto padre.

```
const persona2 = persona;
// FORMA INCORRECTA SI SE IMPRIME AQUÍ, VEREMOS UN FALSO POSITIVO
// SI LO IMPRIMIMOS ABAJO, VEREMOS QUE EL OBJETO PERSONA CAMBIÓ EL NOMBRE
DEBIDO A QUE SE LA ASIGNACIÓN
// NO COPIA EL OBJETO SINO LO MODIFICA
const persona2 = { ...persona };
persona2.nombre = "Peter";

console.log(persona);
console.log(persona2);
```

Arreglos:

Para adherir un nuevo elemento a un array, es recomendable hacerlo de la siguiente manera y no usando el método **PUSH** ya que el mismo, modifica el array y no siempre queremos eso.

```
const arreglo = [1, 2, 3, 4];
let arreglo2 = [...arreglo, 5];
const arreglo3 = arreglo2.map(function (numero) {
```

```
  return numero * 2;
});
```

Funciones:

Es recomendable usar la sintaxis con **CONST** para evitar que cambien los valores si se hace un cambio de valores por error:

```
const saludar2 = (nombre) => {
  return `Hola, ${nombre}`;
};

const saludar3 = (nombre) => `Hola, ${nombre}`;
const saludar4 = () => `Hola Mundo`;

console.log(saludar2("Vegeta"));
console.log(saludar3("Goku"));
console.log(saludar4());
// Hola, Vegeta
// Hola, Goku
// Hola Mundo
```

Para imprimir un elemento más complejo en un **RETURN** como un objeto en una **Función de Flecha**, es necesario agregar paréntesis y corchetes para hacerle entender a JS que estamos regresando un objeto.

Si declaramos una función así con un **return** implícito, tendremos un **error** ya que el JS no entenderá que lo que se está regresando es un objeto.

```
const getUsuarioActivo = (nombre) => {
  uid: "ABC567",
  username: nombre,
};
```

Para retornar un objeto de con un **return implícito**, se deberá de hacer de la siguiente manera:

```
const getUsuarioActivo = (nombre) => ({
  uid: "ABC567",
  username: nombre,
});

const usuarioActivo = getUsuarioActivo("Fernando");
console.log(usuarioActivo);
```