# Que es React.

React es una librería UI en JavaScript creada por Jordan Walke, ingeniero de Facebook, en el 2011.
Fue construida con el fin de poder desarrollar grandes aplicaciones a nivel UI reutilizando componentes, optimizando los tiempos de desarrollo y sacando un mejor partido de la capacidad de render del navegador.

React es declarativo, permitiendo de una forma simple crear vistas para cada uno de los estados de la aplicación, redibujando sólo aquellas vistas que son necesarias (gracias al virtual DOM).

En React, todo está basado en componente que encapsulan los estados, permitiendo crear complejas aplicaciones gracias a la implementación de estos componentes en un esquema de de árbol. Los estados lógicos de la aplicación se encuentran fuera del DOM.

Con React podemos reutilizar el código escrito para que sea portado a Mobile (gracias a React Native) o portado a aplicaciones desktop (usando Electron por ejemplo).

Actualmente React es usado por empresas del primer nivel como Facebook, Airbnb, Instagram, Mural, etc.

Ya que React es una librería en JavaScript, antes de poder empezar a desarrollar nuestras primeras aplicaciones, es necesario que repasemos algunos conceptos de ECMAScript v6.

# ES6.

En todos los navegadores contamos con un motor de JavaScript, por ejemplo en Google Chrome contamos con V8. Todos los motores están basados en un estándar, llamado ECMAScript (ES).
Actualmente estamos en la versión 6 de dicho estándar, pero aún los navegadores no se han adaptado al 100%. Debido a ésta situación es que nos vemos en la necesidad de usar un javascript transpilers (como Babel por ejemplo).

Cuando trabajemos con React usaremos solamente ES6, por lo cual deberemos aprender las diferencias que existen con ES5 y los nuevos features.

## Variables.

Uno de los cambios más importantes de ES6 a nivel variables es sin duda la aparición de let y const.

### Constantes.

Ahora gracias a la palabra reservada 'const' podemos definir constantes en nuestro código como lo hacemos en lenguaje como C++.
Las constantes necesitan de un valor al ser declaradas y el mismo no podrá cambiar a lo largo del runtime.

```
const MAX=100;
const MIN=0;
console.log(MAX, MIN);
```

### Variables en contexto.

Gracias a la aparición de 'let' ahora podemos declarar variables que sólo existan dentro del bloque en donde fueron declaradas, a diferencia de var que definen las variables en forma global.
Por ejemplo, usando var el siguiente código funciona perfectamente.

```
for(var x=0; x<10; x++){
   console.log(x);
}
```

console.log(x);

El último valor que se imprime es el '10'.

Sin embargo, si cambiamos 'var' por 'let', el último console.log disparará un error del tipo:
Uncaught ReferenceError: x is not defined

## Arrow functions.

Las arrows functions con una nueva forma declarativa de funciones (expresión corta) usadas principalmente en callbacks.
Las arrow functions siempre son anónimas, y no cuentan con this, arguments o super.

Ejemplo:

```
[1,2,3,4,5].forEach( (e, i)=>{
        console.log(e, i);
});
```

## Clases.

A partir de ES6 contamos con la posibilidad de crear clases.

```
class Animals{
   constructor(type){
        this.age=0;
  }
}

class Cats extends Animals{
        constructor(type){
        super(type);
        console.log(this.age);
  }
}

new Cats('lion');
```

En el ejemplo anterior creamos la clase Animals, con un constructor que espera como argumento el 'type'.
Mas abajo creamos la clase Cats, que extiende de animals. El constructor de la clase cats espera como argumento 'type'. Cuando se instancia el constructor de la clase Cat se ejecuta 'super'. Mas abajo se hace un console.log de la propiedad 'age'.

### Super.

La función 'super' ejecuta el constructor de la clase padre, y es gracias a ello que la clase Cats puede hacer uso de la propiedad 'age'.

## Templates String.

```
let fname = "Pepe";
let lname = "Luis";
console.log(`Mi nombre es ${fname}  ${lname`);
```

## Destructuring

```
let obj = { fname: 'pepe', lname: 'luis' };
let { a,b } = obj;
console.log(a, b);

let foo =() =>  ["175", "75"];
let [c, d] = foo();
console.log(c,d);
```

## Valores por defecto

```
function hi(name=‘pepe’){
        console.log(name);
}

hi();
```