**Intro**

* React is a java script library not any independent framework or language
* React's usage is dependent on different components.
* React is a library which helps to create a user interface using components.
* React helps us using same component again and again with different content
* Everything happens instantly

**Why React?**

* Reacts helps us with the problem to target a bigger JAVA Script application
* UI state becomes difficult to manage while using JAVA Script, so react helps us creating the same applications with less complexity
* It's efficient and fast too
* We can write a better code which must be easy to handle in future...

**Alternatives**

* Angular
* Vue
* Java Script

**Two types of applications**

* Single Page Applications
* Multi Page Applications

**JAVA Script refresher**

Some of the core features of JAVA Script are as below:

* Let and Const Different ways of creating variables.... We use it on the place of var, but it is recommended to use **let**at the place of var for creating **variables;**and **const**for creating a **constant**
* Functions & Arrow Functions  The Arrow functions just have a bit of **shorter syntax** than functions.

Function:

func printMyName(name){

Console.log(name)

}

Arrow function:

Const printMyName = (name) => {

Console.log(name)

}

// Soo if we call the function like below

printMyName(name)

We'll be getting same output from both of these types.

If you have only a single statement in the function, you can use the arrow functions as below:

Const multiplyBy2 = (number) => number\*2

This above statement is a complete arrow function... And this is very short version of writing a function....

* Import and Export  We have default exports and we need to move with the imports very carefully if we don't have default exports from the files

You can move for **as**keyword to import it with some other naming conventions. OR you can import everything as a bundle.

You choose the name as per your needs.

* Classes and Objects As we say, classes are essentially blueprints for objects (java script objects indeed)….

A class can have both properties and methods, methods are the simple function attached to classes and properties are different variables attached to classes.

Classes as in other programing languages, do also possess the feature of inheritance here too and works with the keyword **extends.**

You need to call a super() function to initialize the parent class and its constructor.

Example:

class Human{

constructor(){

this.gender = 'female'

}

printGender(){

console.log(this.gender);

}

}

class person extends human{

constructor(){

super();

this.gender = 'male';

this.name = 'kk';

}

printMyName(){

console.log(this.name);

}

}

const me =new person();

me.printMyname();

me.printGender();

So, with the above example you must understand that the classes are just the blueprints and the inheritance works with prototype things.

This was the basic java Script code..  But what the next generation JAVA Script code but what we use now a days is the next generation java script code, have a look at it below:

Example:

class Human{

gender = 'female'

printGender(){

console.log(this.gender);

}

}

class person extends human{

gender = 'male'

name = 'kk';

printMyName(){

Console.log(this.name);

}

}

const me =new person();

me.printMyname();

me.printGender();

The code above is the next generation java script code and we actually use the above for the implementations; This is in accordance with ES6/Babel.

Note: ES6 is the standard governing JavaScript whereas JavaScript is the programming language. ES6 is next gen JavaScript syntax and nothing else.

And Babel is a compiler that will convert your development code (written in ES6 with all of its goodies) to code that you will run on your production site, often bundled and minified with Webpack as well.

Spread and Rest operators  You must be wondering what are those operators... Let me take you through his complete area of operators.

**…** This is a spread operator, now you'll be thinking how will this operator be used. Consider the following example for the same.

Example:

const oldArray = [1,2,3,4,5];

const newArray = [...oldArray,6,7]

This is the spread operator, if we will print the values of the newArray, I will be – 1,2,3,4,5,6,7.

The rest and spread functions don't have a huge differnce. You can work with the rest operator using the spread operator only. We use a Rest operator in a function, pass some of the arguments in a function, say as per in the example below:

Example:

const filter = (...args) => {

return args.filter(el =>  el === 1);

}

console.log(filter(1,2,3,4));

So, the above performs a rest operator working. This gives us an output of :

[1]

So this is how a rest operator works...

De-Structuring  This is something which helps to differentiate between the parts and different elements of an array or object like things. Say if you assign some values to any of the array and give a particular name to it, and you access those elements independently is called de-structuring.

Example:

const numbers = [1,2,3,4];

[num1, num2, num3] = numbers;

Console.log(num1, num2, num3, num 4);

This will give you the following output:

1,2,3,<undefined>

However if we have the following piece of code then we'll get different output.

const numbers = [1,2,3,4];

[num1, num2, ,num3] = numbers;

Console.log(num1, num2, num 4);

Output -> 1,2,4