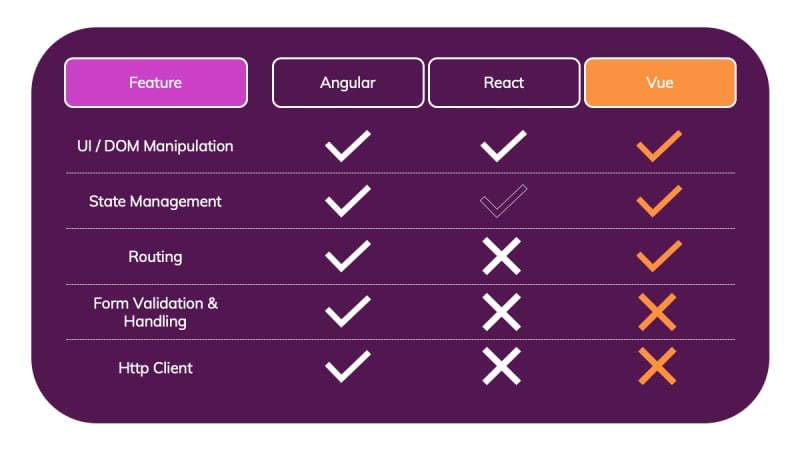
**What is ReactJS**

ReactJS is a declarative, efficient, and flexible JavaScript library for building reusable UI components. It is an open-source, component-based front end library which is responsible only for the view layer of the application. It was initially developed and maintained by Facebook and later used in its products like WhatsApp & Instagram.

**Understand the difference between some alternatives like Angular, vue.js**

**Angular** is a framework developed by Google: Google also uses Angular internally, hence we'll not see Angular disappear over night. It will be maintained and continuously improved

**React** is a library built by Facebook: Facebook also uses React internally, hence we'll not see React disappear over night. It will be maintained and continuously improved

**Vue** is a "standalone" project that is not built inside of any company. It used to be a one-man show (Evan You, its founder) but those days are long gone. Nowadays, it has a dedicated team of core contributors that work on Vue. 

**Angular**

Angular definitely is the "biggest" framework of the three. It's sometimes even called a "platform" rather than a framework

Because Angular out of the box includes support for a lot of things. It helps you (= the developer) with controlling the UI, reacting to user input, validating user input in forms, routing, state management sending Ajax Http requests, providing offline support & PWA capabilities, testing, building your application, managing multiple applications and connecting them and much, much more!

All frameworks have the goal of making it easier for you, as a developer, to build reactive, complex user interfaces. But Angular gives you the full set of tools for that. It doesn't stop after DOM manipulation support - it adds the above-mentioned features to help you with any aspect you could require in apps you're working on.

In addition, there's an official CLI which helps you with creating and managing Angular projects, with keeping them up-to-date, with adding dependencies and even with deployment!

In its core, Angular is all about building re-usable user interface components which you then control with Angular and which you can combine with other components to build an entire user interface from those Angular-controlled components.

**React (or React.js)**

React is totally different!

Where Angular gives you everything, React gives you only one thing: A library for rendering content to the DOM and controlling it efficiently thereafter. It's also all about components and all about building user interfaces from components. It also gives you all the "tools" you need to define what should be rendered in which way under which circumstances.

But it does not include built-in form validation support. It does not include a router (for rendering different components based on URL changes) and it does not ship its own Http client. It has some state management support built-in but not for all scenarios. It also does not come with any other special features and it's definitely "slimmer" than Angular in that regard.

**Vue (or Vue.js)**

Vue is a framework which kind of sits between React and Angular. It's not as "big" as Angular but it definitely includes more features than React does. Vue does give you built-in state management and it also ships with a built-in router. It does, however, not include form validation or Http client functionalities.

Just like Angular and React, the core of Vue is all about building user interfaces by combining re-usable components. But beyond that, it gives you a bit more than React and a bit less than Angular.

**How does React Work?**

React creates a VIRTUAL DOM in memory.

* Instead of manipulating the browser's DOM directly, React creates a virtual DOM in memory, where it does all the necessary manipulating, before making the changes in the browser DOM.
* React only changes what needs to be changed!
* React finds out what changes have been made, and changes only what needs to be changed.

**What is ES6?**

ES6 stands for ECMAScript 6.

ECMAScript was created to standardize JavaScript, and ES6 is the 6th version of ECMAScript, it was published in 2015, and is also known as ECMAScript 2015.

**Classes**

ES6 introduced classes.

A class is a type of function, but instead of using the keyword function to initiate it, we use the keyword class, and the properties are assigned inside a constructor() method.

**Example(Method in Class)**

class Car {

constructor(name) {

this.brand = name;

}

present() {

return 'I have a ' + this.brand;

}

}

const mycar = new Car("Ford");

mycar.present();

**Class Inheritance:**

Create a class named "Model" which will inherit the methods from the "Car" class:

class Car {

constructor(name) {

this.brand = name;

}

present() {

return 'I have a ' + this.brand;

}

}

class Model extends Car {

constructor(name, mod) {

super(name);

this.model = mod;

}

show() {

return this.present() + ', it is a ' + this.model

}

}

const mycar = new Model("Ford", "Mustang");

mycar.show();

The super() method refers to the parent class.

By calling the super() method in the constructor method, we call the parent's constructor method and gets access to the parent's properties and methods.

**Arrow Function**

Arrow functions allow us to write shorter function syntax:

**With Arrow Function:**

hello = () => {

return "Hello World!";

}

**Destructuring**

To illustrate destructuring, we'll make a sandwich. Do you take everything out of the refrigerator to make your sandwich? No, you only take out the items you would like to use on your sandwich.

Destructuring is exactly the same. We may have an array or object that we are working with, but we only need some of the items contained in these.

Destructuring makes it easy to extract only what is needed.

**Before:**

const vehicles = ['mustang', 'f-150', 'expedition'];

// old way

const car = vehicles[0];

const truck = vehicles[1];

const suv = vehicles[2];

**With destructuring:**

const vehicles = ['mustang', 'f-150', 'expedition'];

const [car, truck, suv] = vehicles;

**The Render Function**

The ReactDOM.render() function takes two arguments, HTML code and an HTML element.

The purpose of the function is to display the specified HTML code inside the specified HTML element.

**But render where?**

There is another folder in the root directory of your React project, named "public". In this folder, there is an index.html file.

You'll notice a single <div> in the body of this file. This is where our React application will be rendered.

**Example:**

The root node can be called whatever you like:

<body>

<header id="sandy"></header>

</body>

**Display the result in the <header id="sandy"> element:**

ReactDOM.render(<p>Hallo</p>, document.getElementById('sandy'));

**What is JSX?**

* JSX stands for JavaScript XML.
* JSX allows us to write HTML in React.
* JSX makes it easier to write and add HTML in React.

**Coding JSX**

JSX allows us to write HTML elements in JavaScript and place them in the DOM without any createElement() and/or appendChild() methods.

JSX converts HTML tags into react elements.

Example 1

**JSX:**

const myelement = <h1>I Love JSX!</h1>;

ReactDOM.render(myelement, document.getElementById('root'));

Example 2

**Without JSX:**

const myelement = React.createElement('h1', {}, 'I do not use JSX!');

ReactDOM.render(myelement, document.getElementById('root'));

**Attribute class = className**

The class attribute is a much used attribute in HTML, but since JSX is rendered as JavaScript, and the class keyword is a reserved word in JavaScript, you are not allowed to use it in JSX.

**Use attribute className instead.**

JSX solved this by using className instead. When JSX is rendered, it translates className attributes into class attributes.

**Example**

Use attribute className instead of class in JSX:

const myelement = <h1 className="myclass">Hello World</h1>;

**Use ternary expressions instead:**

Example

Write "Hello" if x is less than 10, otherwise "Goodbye":

const x = 5;

const myelement = <h1>{(x) < 10 ? "Hello" : "Goodbye"}</h1>;

**React Components**

Components are independent and reusable bits of code. They serve the same purpose as JavaScript functions, but work in isolation and return HTML.

Components come in two types, Class components and Function components,

**Class Component**

A class component must include the extends React.Component statement. This statement creates an inheritance to React.Component, and gives your component access to React.Component's functions.

The component also requires a render() method, this method returns HTML.

Example

Create a Class component called Car

class Car extends React.Component {

render() {

return <h2>Hi, I am a Car!</h2>;

}

}

**Function Component**

Here is the same example as above, but created using a Function component instead.

A Function component also returns HTML, and behaves much the same way as a Class component, but Function components can be written using much less code, are easier to understand, and will be preferred in this tutorial.

**Example**

Create a Function component called Car

function Car() {

return <h2>Hi, I am a Car!</h2>;

}

**Rendering a Component**

Now your React application has a component called Car, which returns an <h2> element.

To use this component in your application, use similar syntax as normal HTML: <Car />

**Example**

Display the Car component in the "root" element:

ReactDOM.render(<Car />, document.getElementById('root'));

**Components in Files**

React is all about re-using code, and it is recommended to split your components into separate files.

To do that, create a new file with a .js file extension and put the code inside it:

**Note that the filename must start with an uppercase character.**

**Example**

This is the new file, we named it "Car.js":

function Car() {

return <h2>Hi, I am a Car!</h2>;

}

export default Car;

To be able to use the Car component, you have to import the file in your application.

**Example**

Now we import the "Car.js" file in the application, and we can use the Car component as if it was created here.

import React from 'react';

import ReactDOM from 'react-dom';

import Car from './Car.js';

ReactDOM.render(<Car />, document.getElementById('root'));