React JS

React.js is a JavaScript library that was created by Facebook. is a declarative, component-based JavaScript library used for creating user interfaces.

In other words, ReactJS is an open-source, component based front end library responsible only for the view layer of the application.

Developed at Facebook and released to the world in 2013, it drives some of the most widely used apps, powering Facebook and Instagram among countless other applications.

It is often thought of as the "view" in a model-view-controller (MVC) user interface. This makes sense when you consider the fact that the only function that must be implemented in React is the "render" function. The render function provides the output that the user sees (the "view").

Today, there are over 220,000 live websites using React.

Not only that, but industry giants like Apple, Netflix, Paypal, and many others have also already started using React JS in their software productions.

Key benefits of React JS:

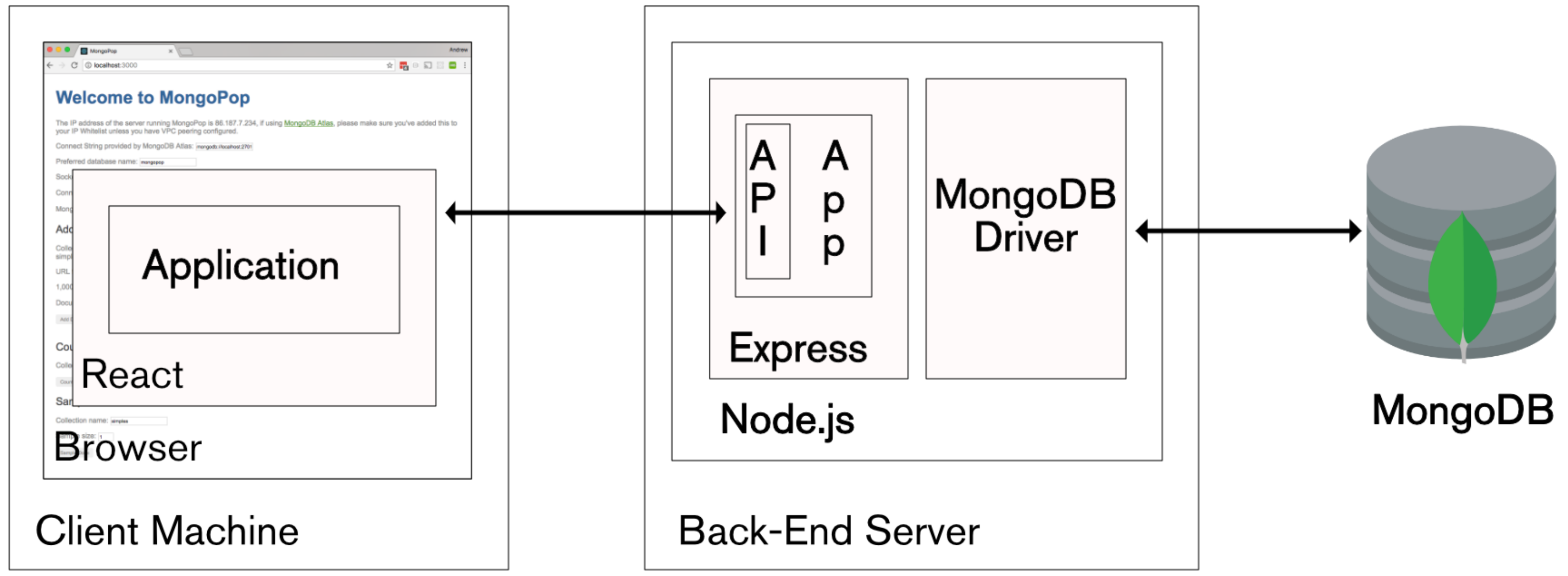
Speed: it allows developers to utilize individual parts of their application on both client-side and the server-side, which ultimately boosts the speed of the development process.

In simple terms, different developers can write individual parts and all changes made won’t cause the logic of the application.

Flexibility: Compared to other frontend frameworks, the React code is easier to maintain and is flexible due to its modular structure. This flexibility, in turn, saves huge amount of time and cost to businesses.

React JS allows us to create reusable UI components.
It is more flexible, efficient and declarative Java script Library for building UI components.
It provides a one way data flow and is meant for Single Page Applications (SPAs)

React JS architecture



ADVANTAGES OF REACTJS

Intuitive: ReactJS is extremely intuitive to work with and provides interactivity to the layout of any UI. It means that it provides ease of use and easy to understand. Moreover it enables fast and quality assured application development that in turn saves tome for both - clients and developers.

Declarative: ReactJS enables significant data changes that result in automatic alteration in the selected parts of user interfaces. Owing to this progressive functionality, there is no additional function that you need to perform to update your user interface.

Provides Reusable Components: ReactJS provides reusable components that developers have the authority to reuse and create a new application. Thereby, reducing the development effort and ensuring a flawless performance.

JavaScript library: A strong blend of JavaScript and HTML syntax is always used, which automatically simplifies the entire process of writing code for the planned project. The JS library consists several functions including one that converts the HTML components into required functions and transforms the entire project so that it is easy to understand.

Components Support: ReactJS is a perfect combination of JavaScript and HTML tags. The usage of the HTML tags and JS codes, make it easy to deal with a vast set of data containing the document object model.

SEO-friendly: React JS was introduced after immense research and improvements by Facebook. Naturally, it stands out from the crowd and allows developers to build amazing, SEO-friendly user interfaces across browsers and engines.

Proficient Data Binding: ReactJS trails one-way data binding. This means that absolutely anyone can track all the changes made to any particular segment of the data. This is a symbol of its simplicity.

Some Additional Advantages Of REACT JS:

- Makes JavaScript coding easier
- Extremely competent
- Excellent cross-platform support
- Handles dependencies
- Template designing made easy
- Provides amazing developer tools
- UI focused designs
- Easy to adopt

React is based on components and states. This is what makes React such a popular library.

When you want to create an application, you usually break it into simpler parts. When programming with React, you will want to break your interface into its most basic parts, and those will be your React components.

React in itself has a very small API, and you basically need to understand 4 concepts to get started:

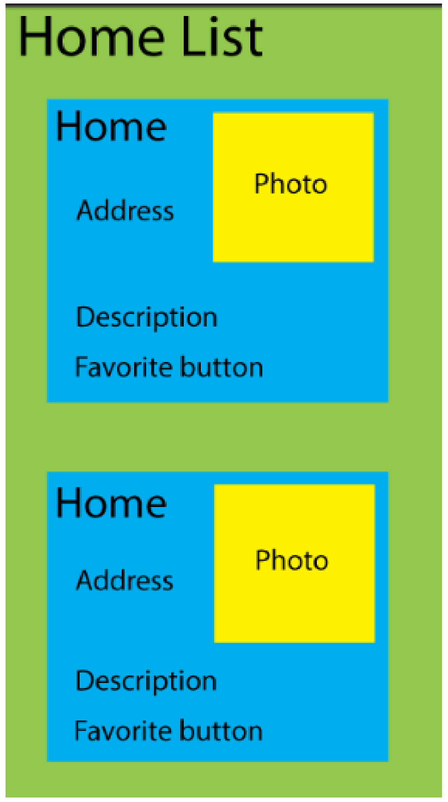
- Components
- JSX
- State
- Props

Components are wonderful because they are modular and reusable.

You can take a basic component used in one area of an application and reuse it in others without having to duplicate code. This helps to speed up development.

Components can be nested, so that the most basic components can be grouped into a parent component.

For example, if you were to create a home listing interface with React, the top level component would be the home list itself. Within the list, you would have a description of a single home.

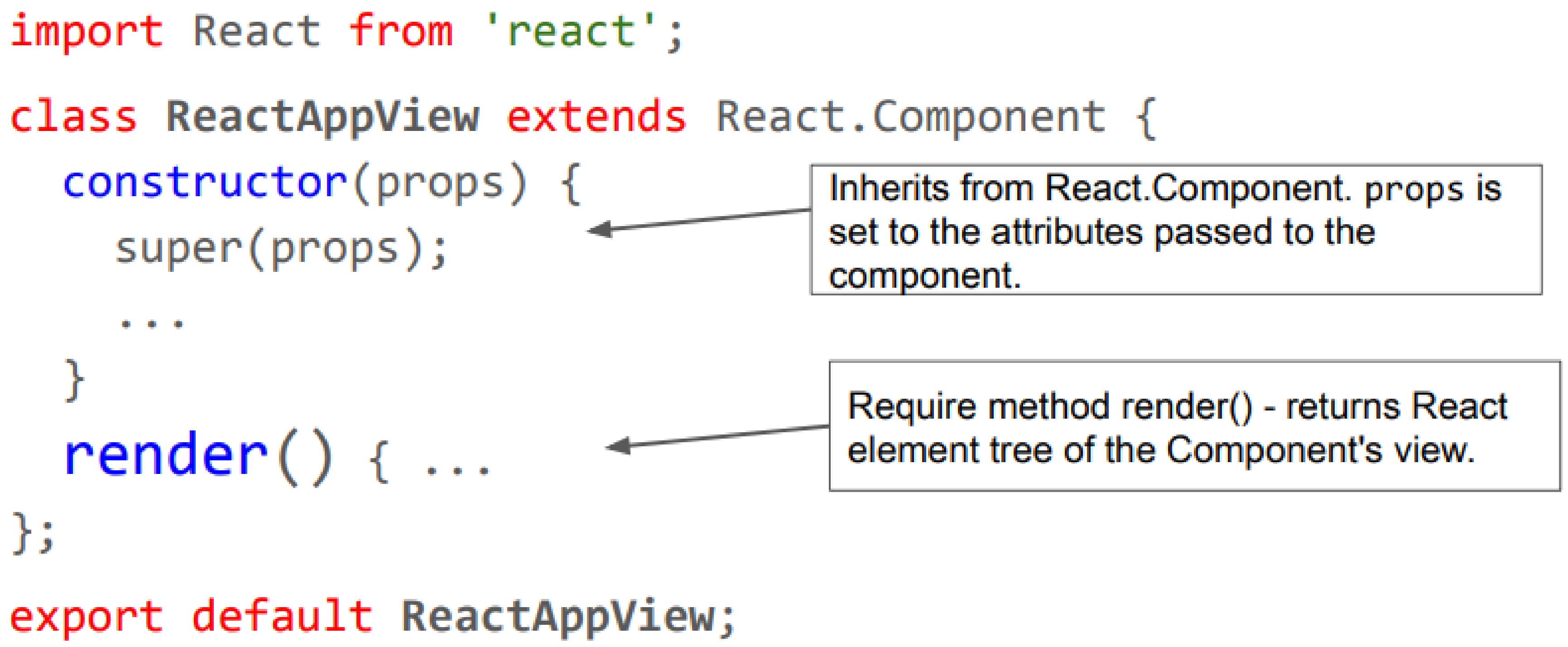


Within the home component, you would have the address of the home, as well as other small components such as a photo, perhaps a favorite or save button, and a link to view details and a map.

A React component can be defined as an ES6 class that extends the base *React.Component* class.

In its minimal form, a component must define a render method that specifies how the component renders to the DOM.

The render method returns React nodes, which can be defined using JSX syntax as HTML-like tags.



```
import React from 'react';

class ReactAppView extends React.Component {
  constructor(props) {
    super(props);
    ...
  }
  render() { ...
};

export default ReactAppView;
```

Inherits from React.Component. props is set to the attributes passed to the component.

Require method render() - returns React element tree of the Component's view.

The following example shows how to define a minimal Component:

Ex:

```
Import React from ‘react’;

class HelloWorld extends React.Component {

        render() {

                return <h1> Hello, world! </h1>
        }
}

export default HelloWorld;
```

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.

React allows us to write components using a domain-specific language called JSX. React will internally convert this into a virtual DOM, and will ultimately output our HTML for us.

Most React implementations make use of JSX, which allows you to put XML-like syntax right within JavaScript. Since React displays output as its main function, we will be using HTML in just about every component.

JSX simplifies the code that would normally have to be written in JavaScript, which makes your code much more readable and simplified.

JSX is not required, but consider the difference between two very simple statements.

```
const element = <h1>Hello, world!</h1>;
```

This funny tag syntax is neither a string nor HTML.

- It is called JSX, and it is a syntax extension to JavaScript.

- One can recommend using it with React to describe what the UI should look like.

- JSX may remind you of a template language, but it comes with the full power of JavaScript.

- JSX produces React “elements”.

JSX converts HTML tags into react elements.

Note: You are not required to use JSX, but JSX makes it easier to write React applications.

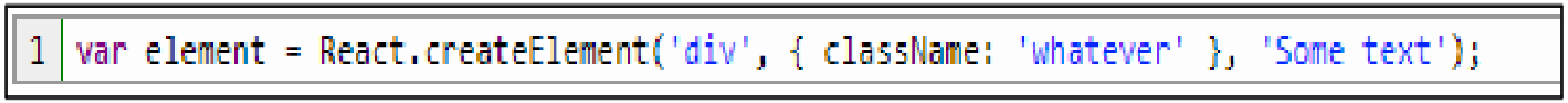
Expressions in JSX:

With JSX you can write expressions inside curly braces { }.

The expression can be a React variable, or property, or any other valid JavaScript expression. JSX will execute the expression and return the result:

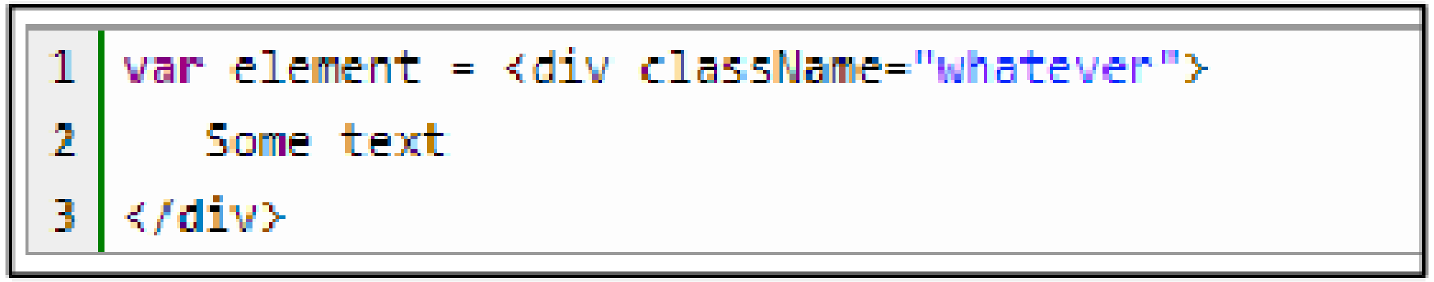
```
import React from 'react’;
import ReactDOM from 'react-dom/client’;

const myElement = <h1>React is {5 + 5} times better with JSX</h1>;

const root = ReactDOM.createRoot(document.getElementById('root'));

root.render(myElement);
```

The following statement is created without JSX:



```
var element = React.createElement('div', { className: 'whatever' }, 'Some text');
```

The following is with JSX:



```
var element = <div className="whatever">
    Some text
</div>
```

```
const name = 'Josh Perez’;
const element = <h1>Hello, {name}</h1>;
```