strategy | consulting | digital | technology | operations

REACTJS – MODULE1

INTRODUCTION TO REACTJS

accenture

Module Objectives

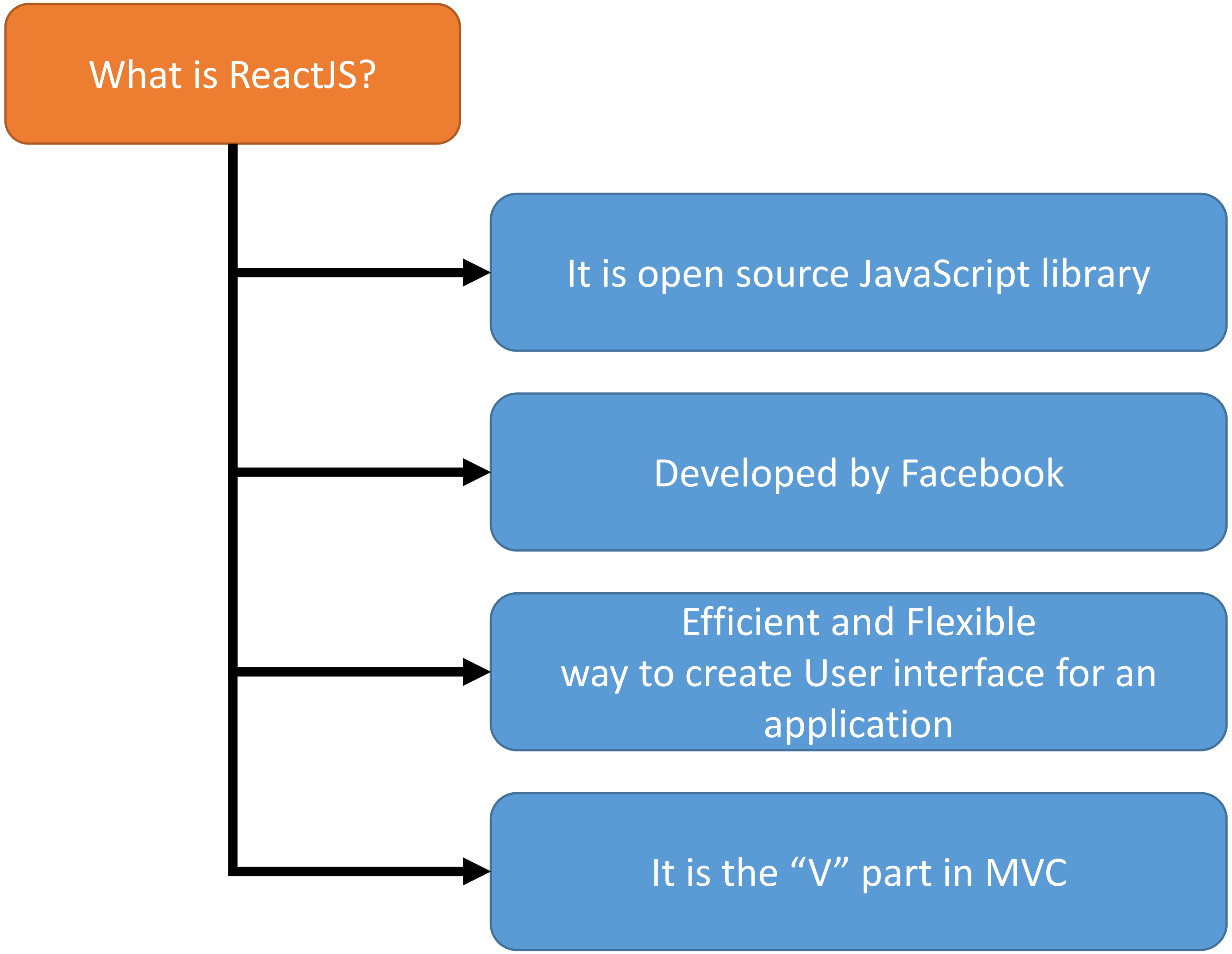
At the end of this module, you will be able to:

- Describe about React JS.
- Understand ES6 specification.
- Component structure of ReactJS.
- Create first react app

Agenda

- Introduction to ReactJS
- Features of ReactJS
- Component in ReactJS
- Create React app
- Understand project structure

Introduction to ReactJS



What is ReactJS?

It is open source JavaScript library

Developed by Facebook

Efficient and Flexible
way to create User interface for an application

It is the “V” part in MVC

React

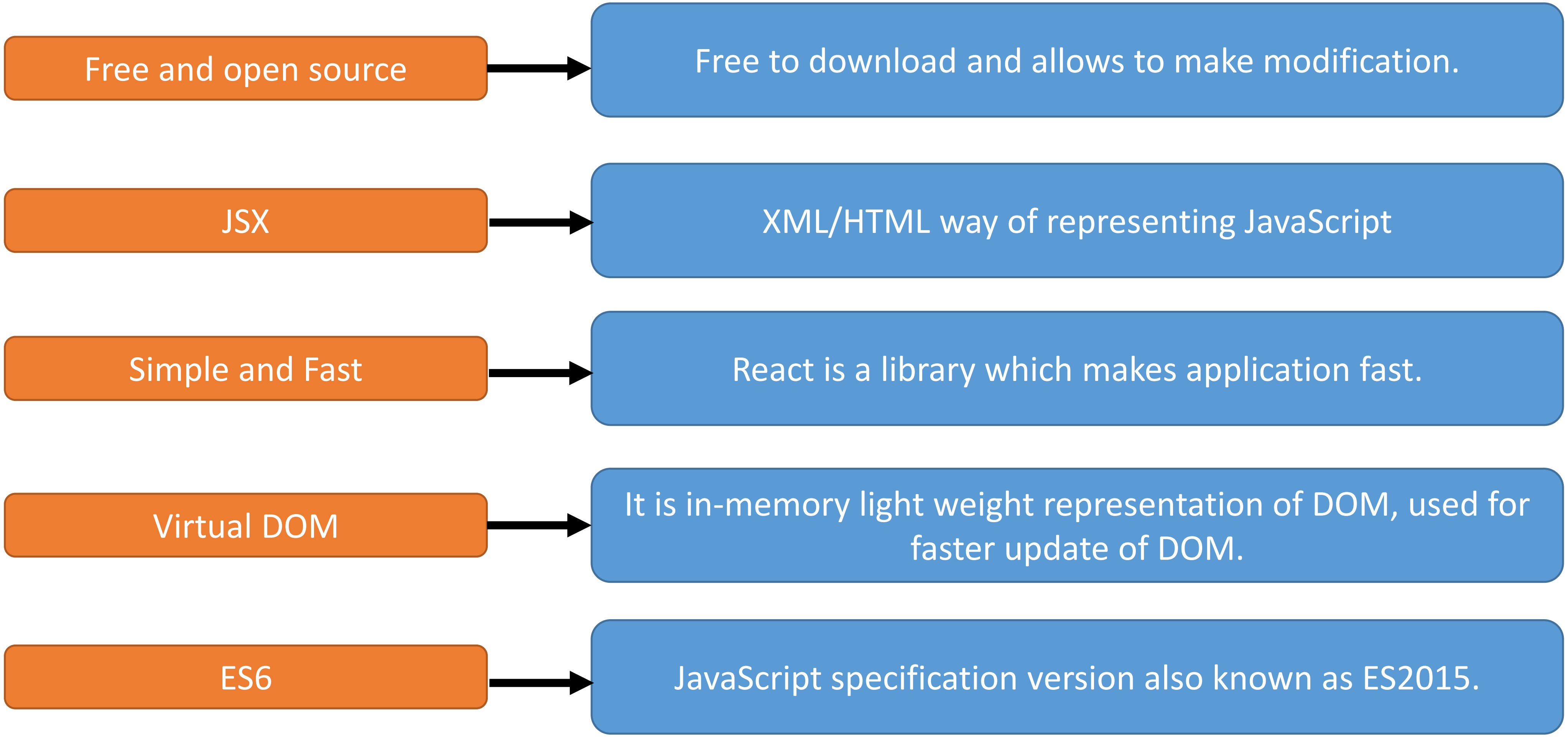
accenture

Introduction to ReactJS

React is popular and used by following

Flipboard
NETFLIX
Adobe
YAHOO!
Dropbox
PayPal
The New York Times
airbnb
reddit
BBC
Pinterest

ReactJS Features



Free and open source

Free to download and allows to make modification.

JSX

XML/HTML way of representing JavaScript

Simple and Fast

React is a library which makes application fast.

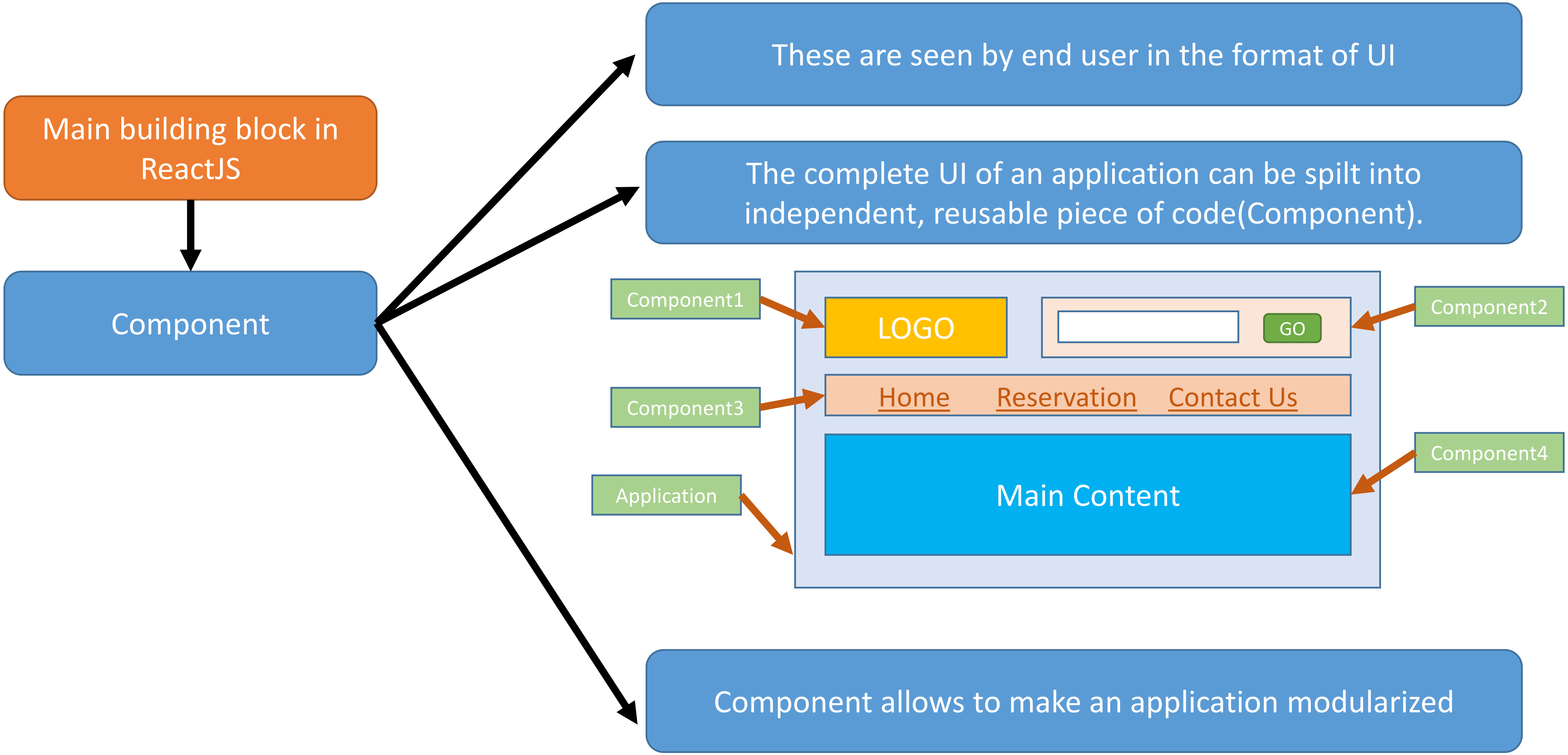
Virtual DOM

It is in-memory light weight representation of DOM, used for faster update of DOM.

ES6

JavaScript specification version also known as ES2015.

ReactJS Architecture



Main building block in ReactJS

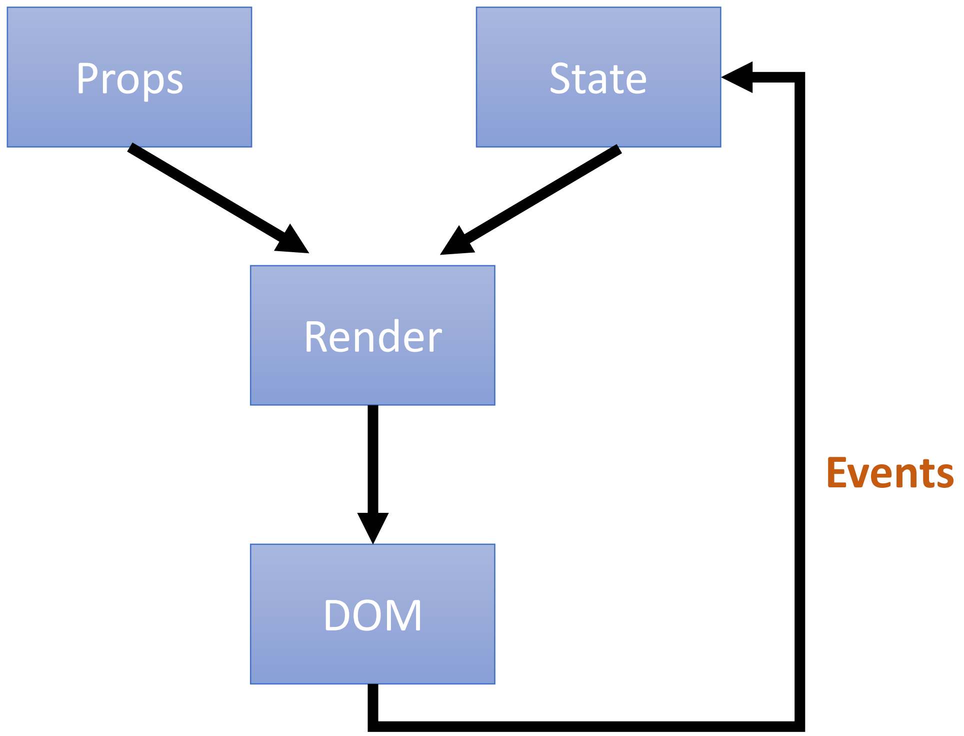
Component

These are seen by end user in the format of UI

The complete UI of an application can be spilt into independent, reusable piece of code(Component).

Component allows to make an application modularized

ReactJS Architecture



Component + Data = DOM

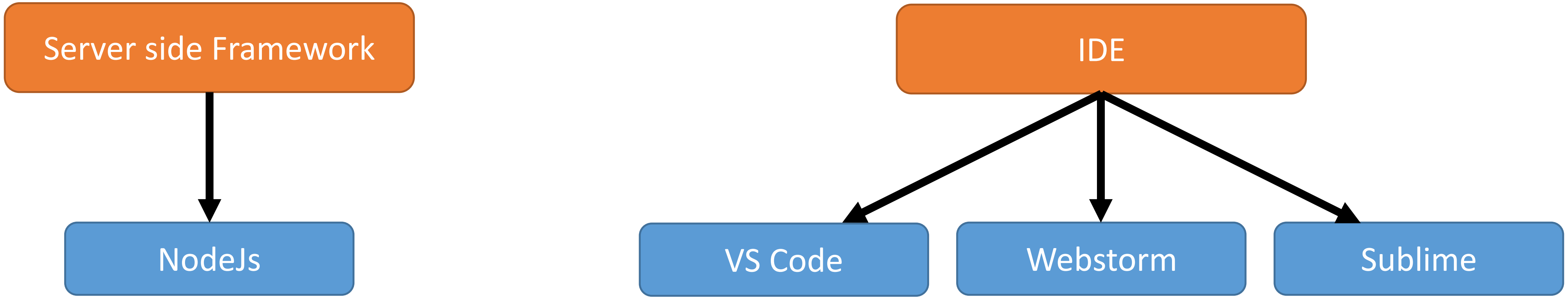
Props and State are used to represent data of a component.

Props are used to represent fixed data , however States are used to represent changing data

Render is a method, which can capture props and states and return a single DOM element

Component along with data is represented using DOM

Software Requirements



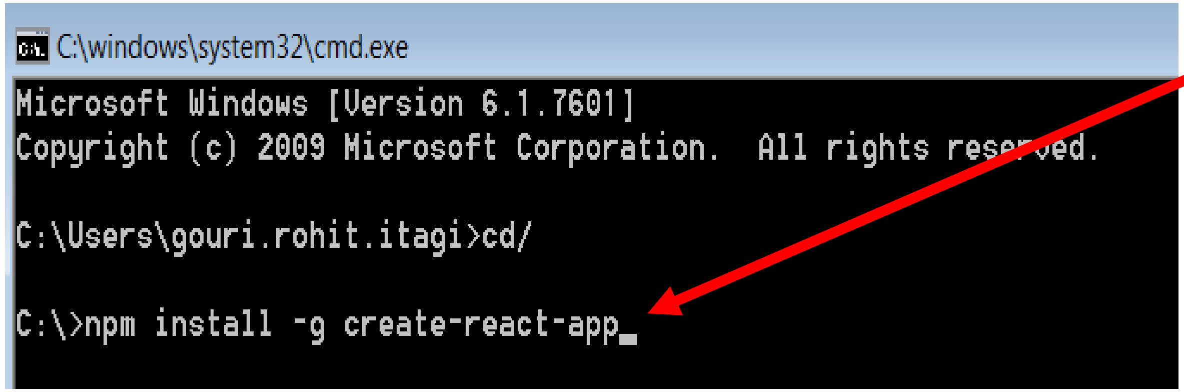
Note :

NodeJs version must be >= v6

Create First React app

Step 1

Open command prompt and go to C: drive and Install create-react-app globally



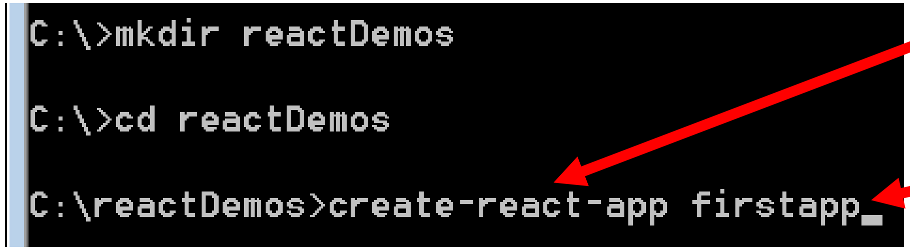
C:\windows\system32\cmd.exe

```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\gouri.rohit.itagi>cd/

C:\>npm install -g create-react-app_
```

Is the package and using this one can create react application

Step 2

Create new directory in C: drive, go to that specific directory and create application using “create-react-app”



```
C:\>mkdir reactDemos

C:\>cd reactDemos

C:\reactDemos>create-react-app firstapp_
```

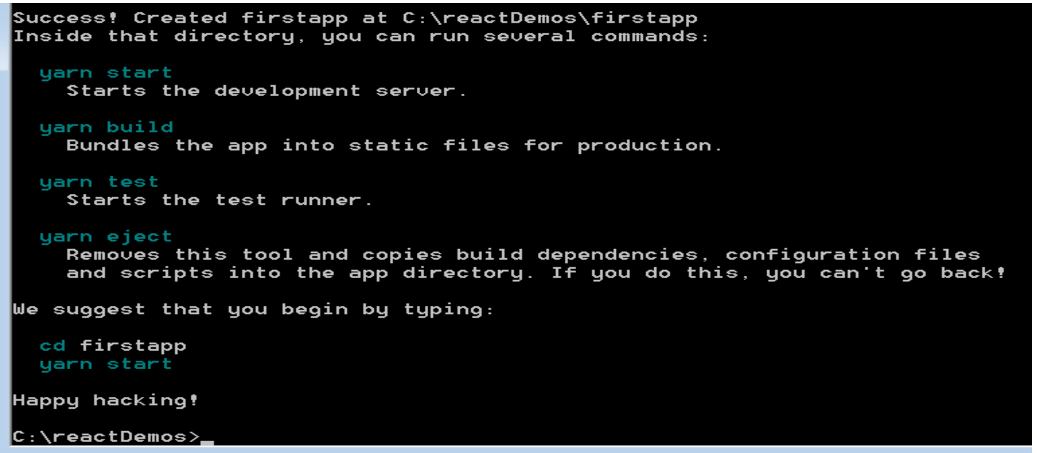
This command is used to create react application

Name of the application

accenture

Create First React app

Below screen will display after the successful react application creation.

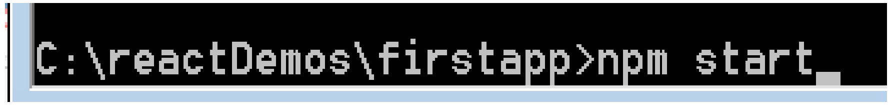


```
Success! Created firstapp at C:\reactDemos\firstapp
Inside that directory, you can run several commands:

  yarn start
    Starts the development server.

  yarn build
    Bundles the app into static files for production.

  yarn test
    Starts the test runner.

  yarn eject
    Removes this tool and copies build dependencies, configuration files
    and scripts into the app directory. If you do this, you can't go back!

We suggest that you begin by typing:

  cd firstapp
  yarn start

Happy hacking!

C:\reactDemos>_
```

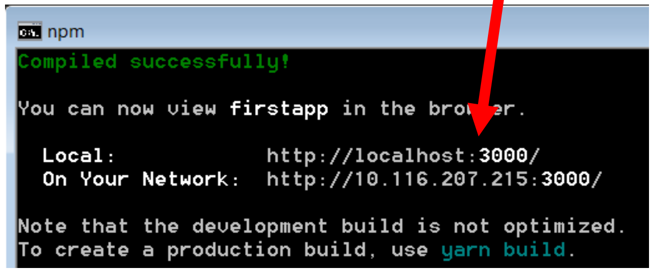
Step 3

To execute the application execute “npm start”

The application will run on localhost with portNo 3000



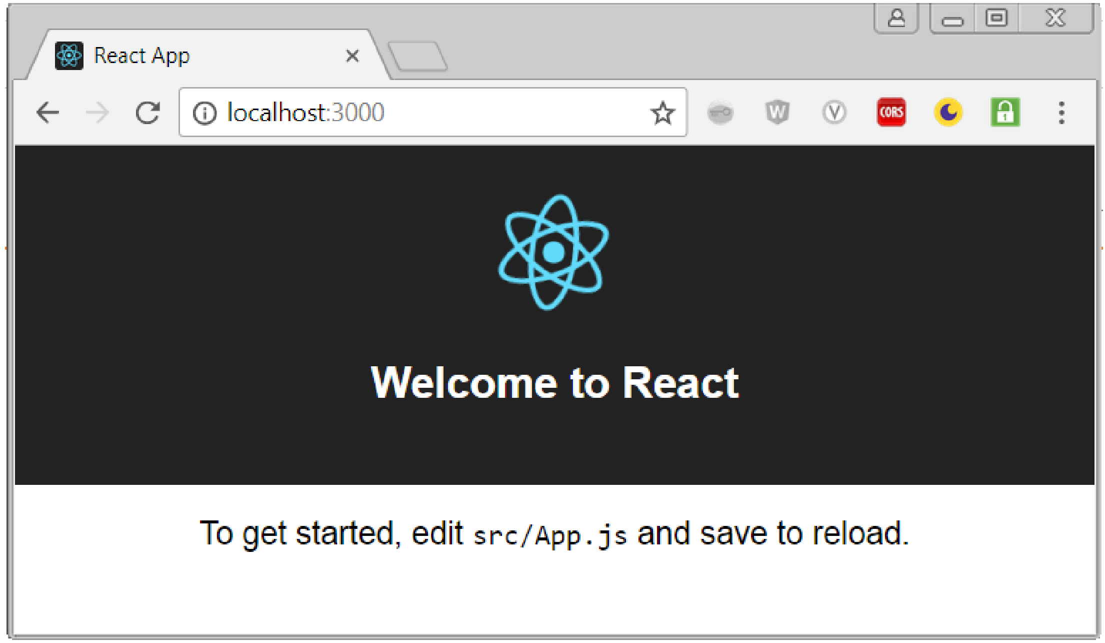
```
C:\reactDemos\firstapp>npm start_
```



```
Compiled successfully!

You can now view firstapp in the browser.

  Local:            http://localhost:3000/
  On Your Network:  http://10.116.207.215:3000/

Note that the development build is not optimized.
To create a production build, use yarn build.
```

Create First React app

Output

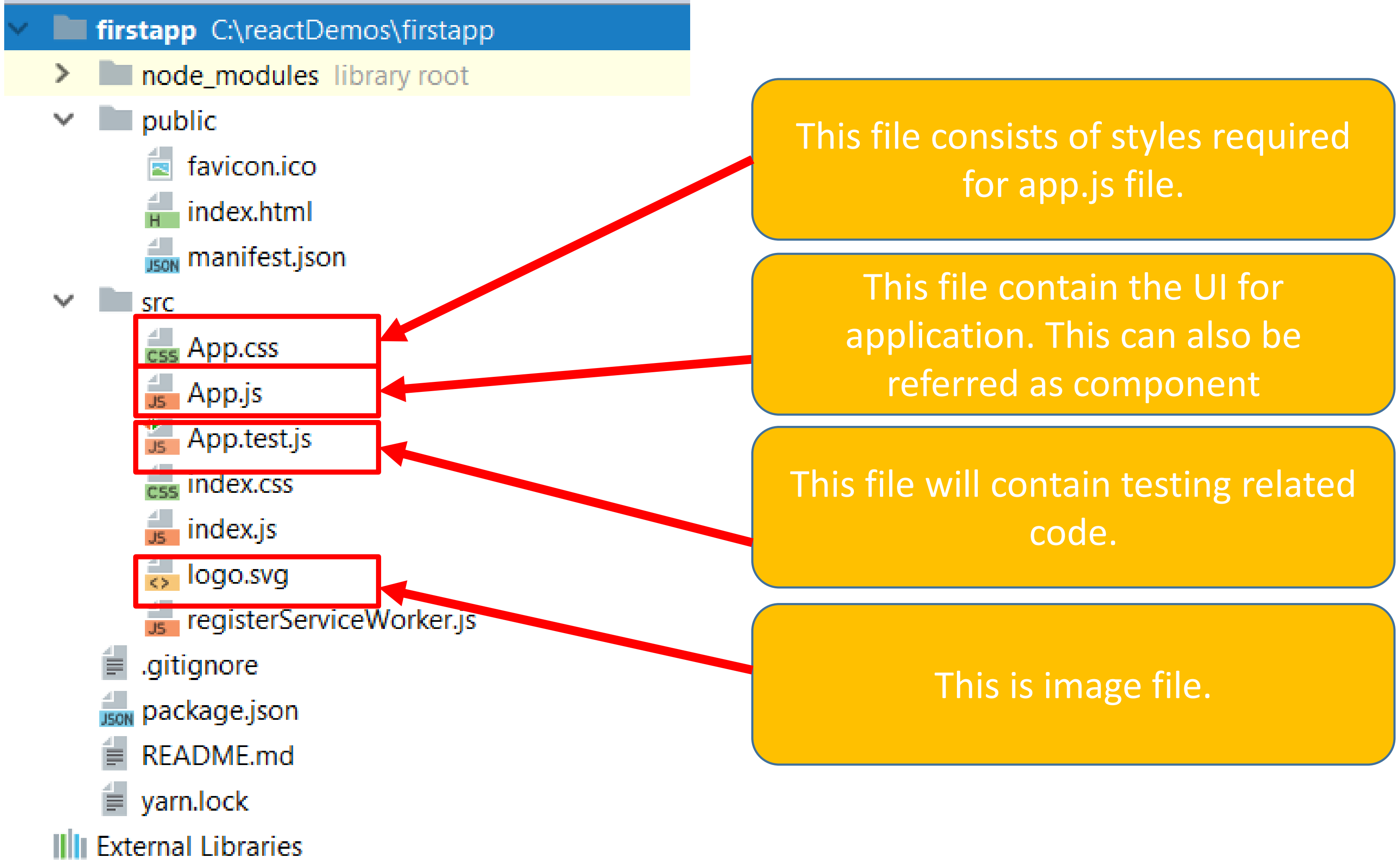


Welcome to React

To get started, edit `src/App.js` and save to reload.

Project Structure

Open the react application “firstapp” using IDE.



```
firstapp  C:\reactDemos\firstapp
  node_modules  library root
  public
    favicon.ico
    index.html
    manifest.json
  src
    App.css
    App.js
    App.test.js
    index.css
    index.js
    logo.svg
    registerServiceWorker.js
  .gitignore
  package.json
  README.md
  yarn.lock
External Libraries
```

This file consists of styles required for app.js file.

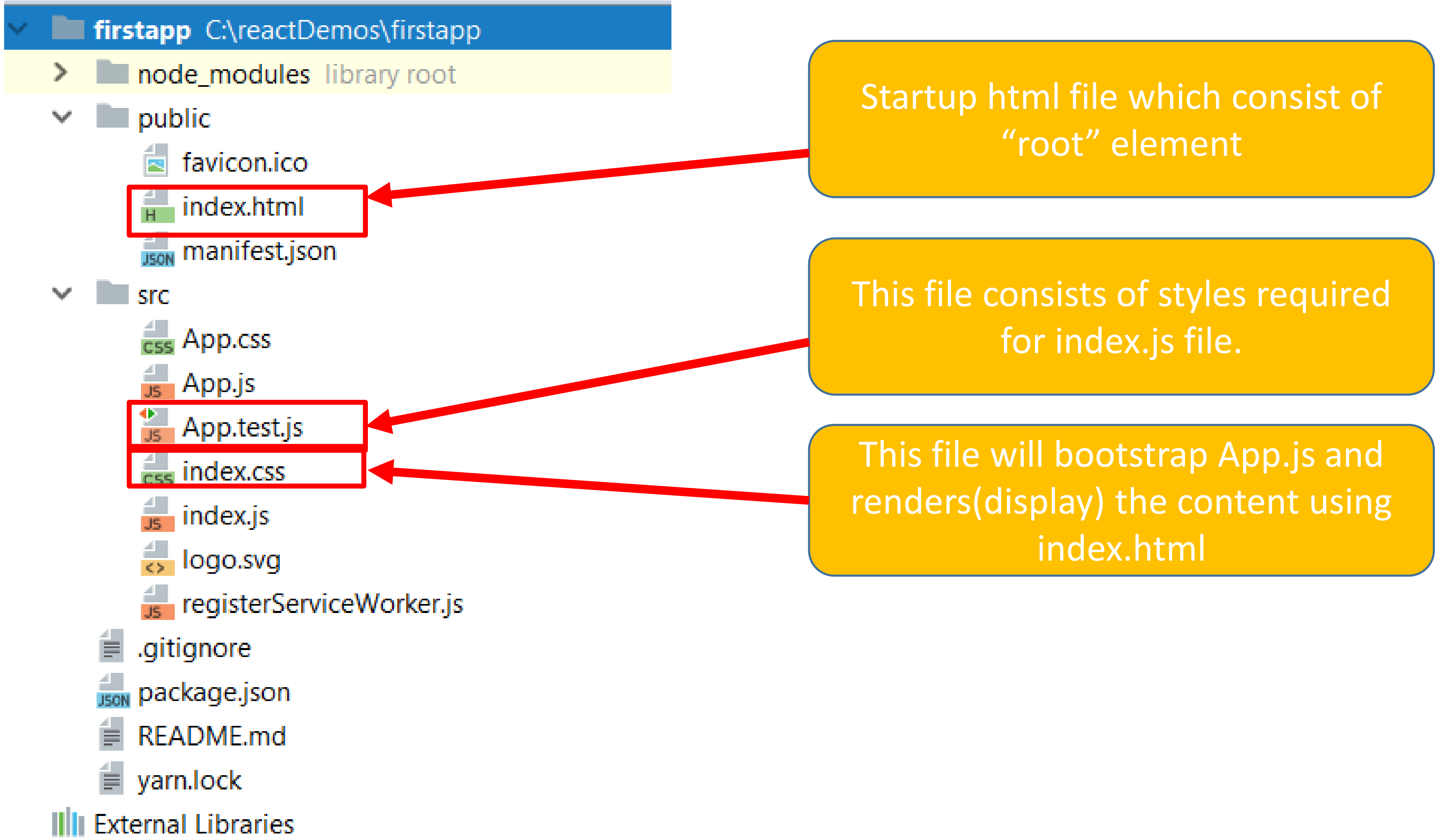
This file contain the UI for application. This can also be referred as component

This file will contain testing related code.

This is image file.

Project Structure

Open the react application “firstapp” using IDE.



```
firstapp  C:\reactDemos\firstapp
  node_modules  library root
  public
    favicon.ico
    index.html
    manifest.json
  src
    App.css
    App.js
    App.test.js
    index.css
    index.js
    logo.svg
    registerServiceWorker.js
  .gitignore
  package.json
  README.md
  yarn.lock
External Libraries
```

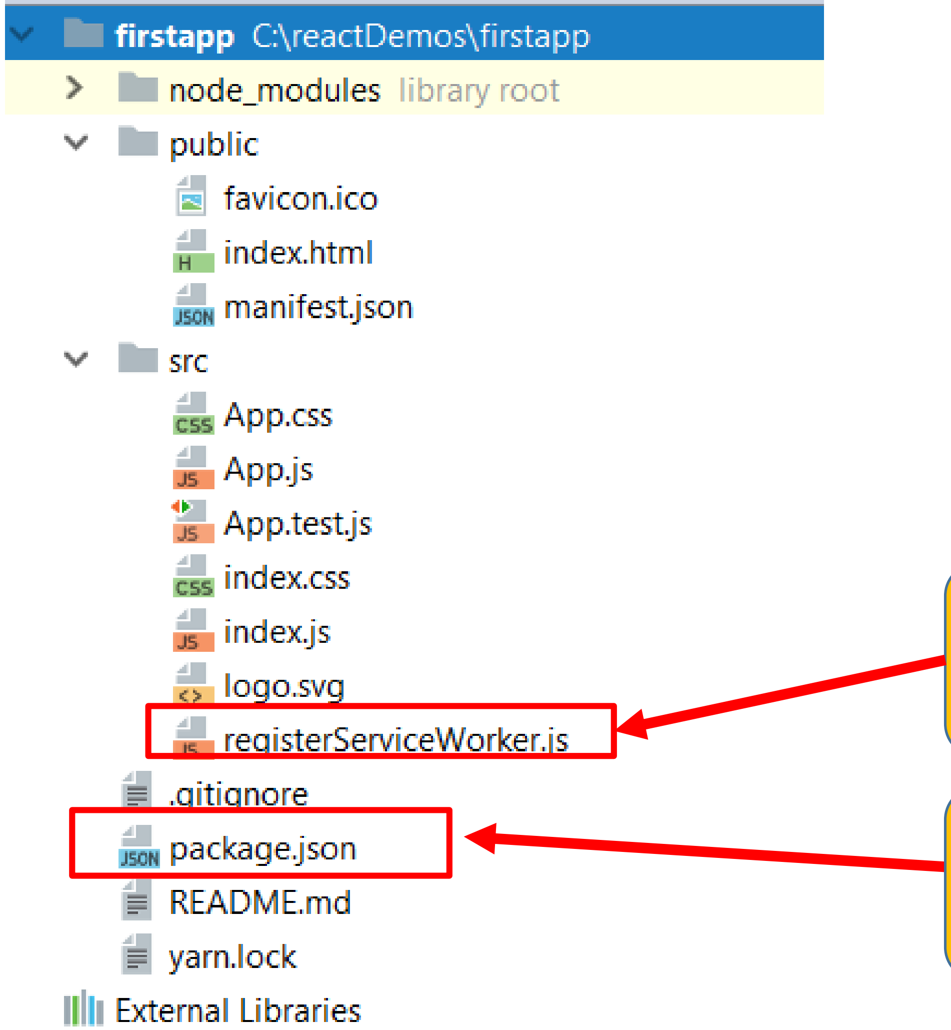
Startup html file which consist of “root” element

This file consists of styles required for index.js file.

This file will bootstrap App.js and renders(display) the content using index.html

Project Structure

Open the react application “firstapp” using IDE.



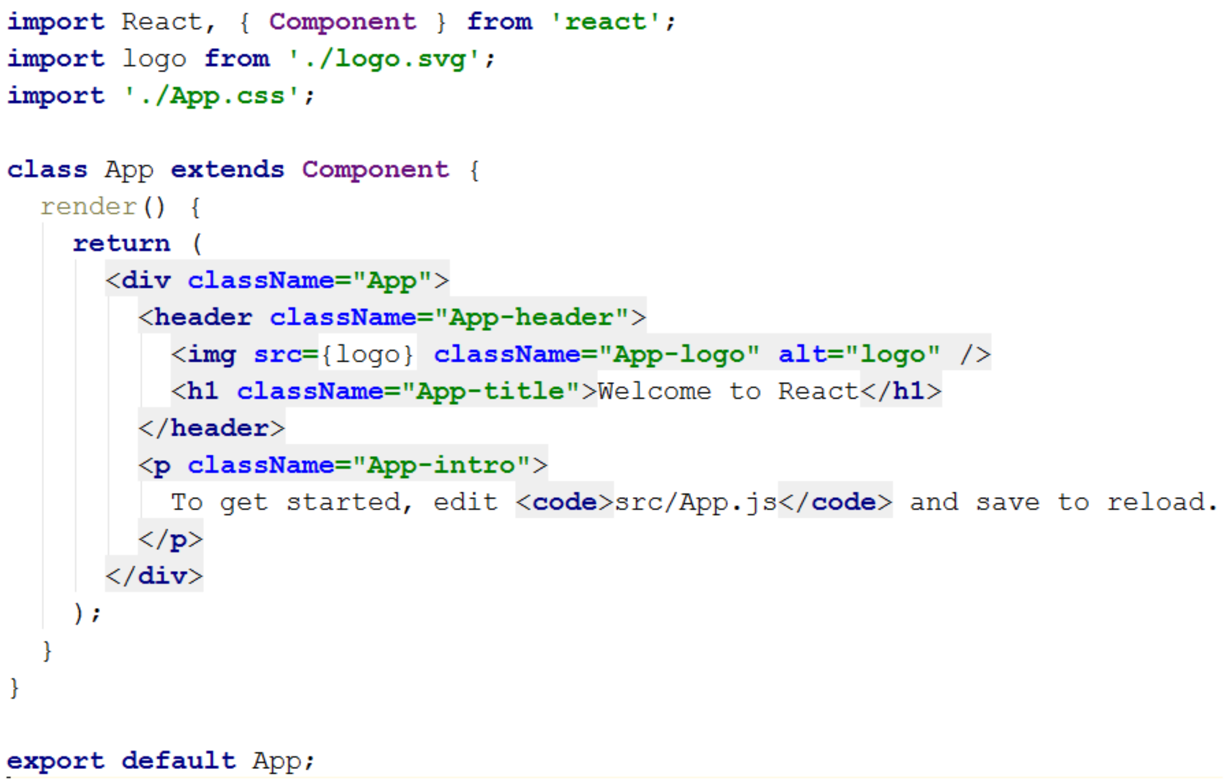
```
firstapp  C:\reactDemos\firstapp
  node_modules  library root
  public
    favicon.ico
    index.html
    manifest.json
  src
    App.css
    App.js
    App.test.js
    index.css
    index.js
    logo.svg
    registerServiceWorker.js
  .gitignore
  package.json
  README.md
  yarn.lock
External Libraries
```

This file is used by browser

Stores all dependency information required for application

Project Structure

App.js.



```
import React, { Component } from 'react';
import logo from './logo.svg';
import './App.css';

class App extends Component {
  render() {
    return (
      <div className="App">
        <header className="App-header">
          <img src={logo} className="App-logo" alt="logo" />
          <h1 className="App-title">Welcome to React</h1>
        </header>
        <p className="App-intro">
          To get started, edit <code>src/App.js</code> and save to reload.
        </p>
      </div>
    );
  }
}

export default App;
```

In ReactJS component will be created as javascript file with .js extension.

App.js is a JavaScript file, and it uses keywords like “class”, ”extends” and “exports”

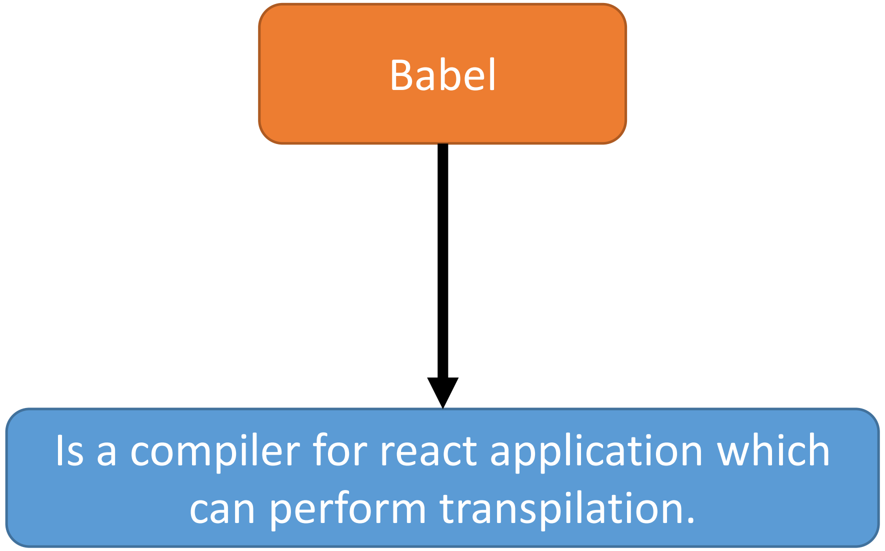
These new features are supported by ECMA Script 2015 or ES6

Transpilation

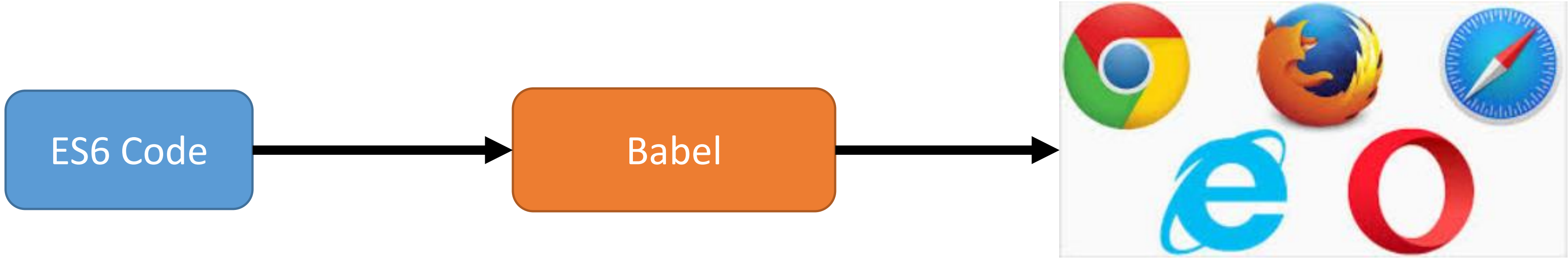
Browser cannot understand ES6 code, however it can understand only plain vanilla javascript code.

The process which converts ES6 code to plain javascript code is known as “Transpilation”

The compiler which would perform transpilation is known “Transpiler”


Babel

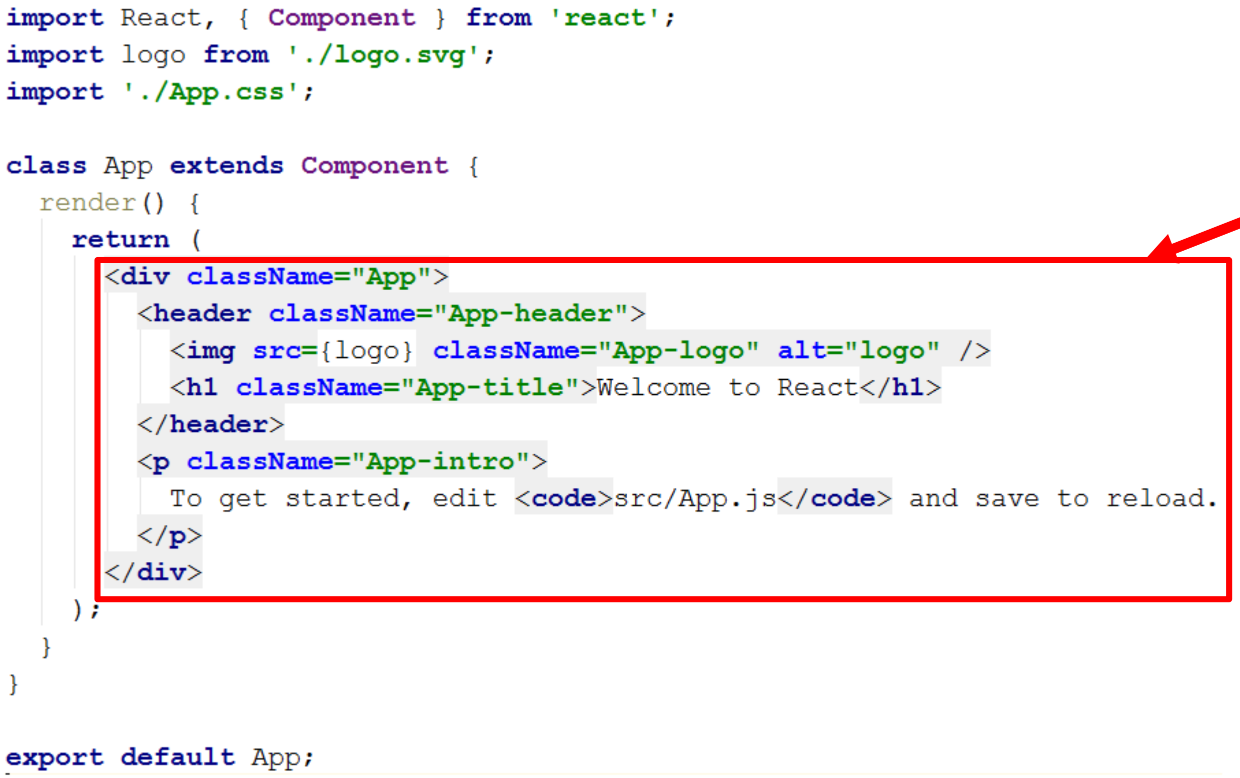
Is a compiler for react application which can perform transpilation.


ES6 Code

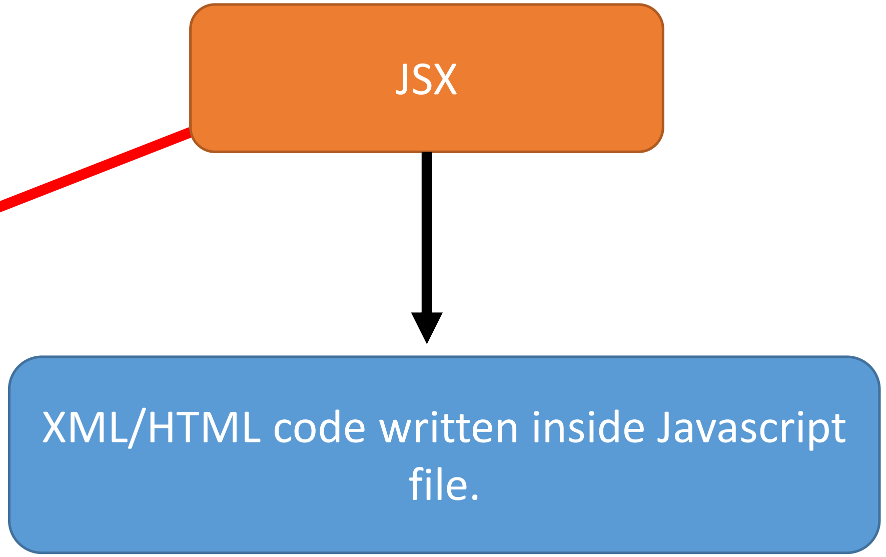
Babel

Project Structure

App.js.



```
import React, { Component } from 'react';
import logo from './logo.svg';
import './App.css';

class App extends Component {
  render() {
    return (
      <div className="App">
        <header className="App-header">
          <img src={logo} className="App-logo" alt="logo" />
          <h1 className="App-title">Welcome to React</h1>
        </header>
        <p className="App-intro">
          To get started, edit <code>src/App.js</code> and save to reload.
        </p>
      </div>
    );
  }
}

export default App;
```



JSX

XML/HTML code written inside Javascript file.

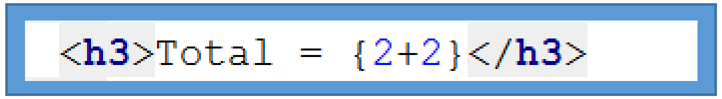
JSX

Stands for “Javascript XML”

It is an extension to javascript using XML/HTML

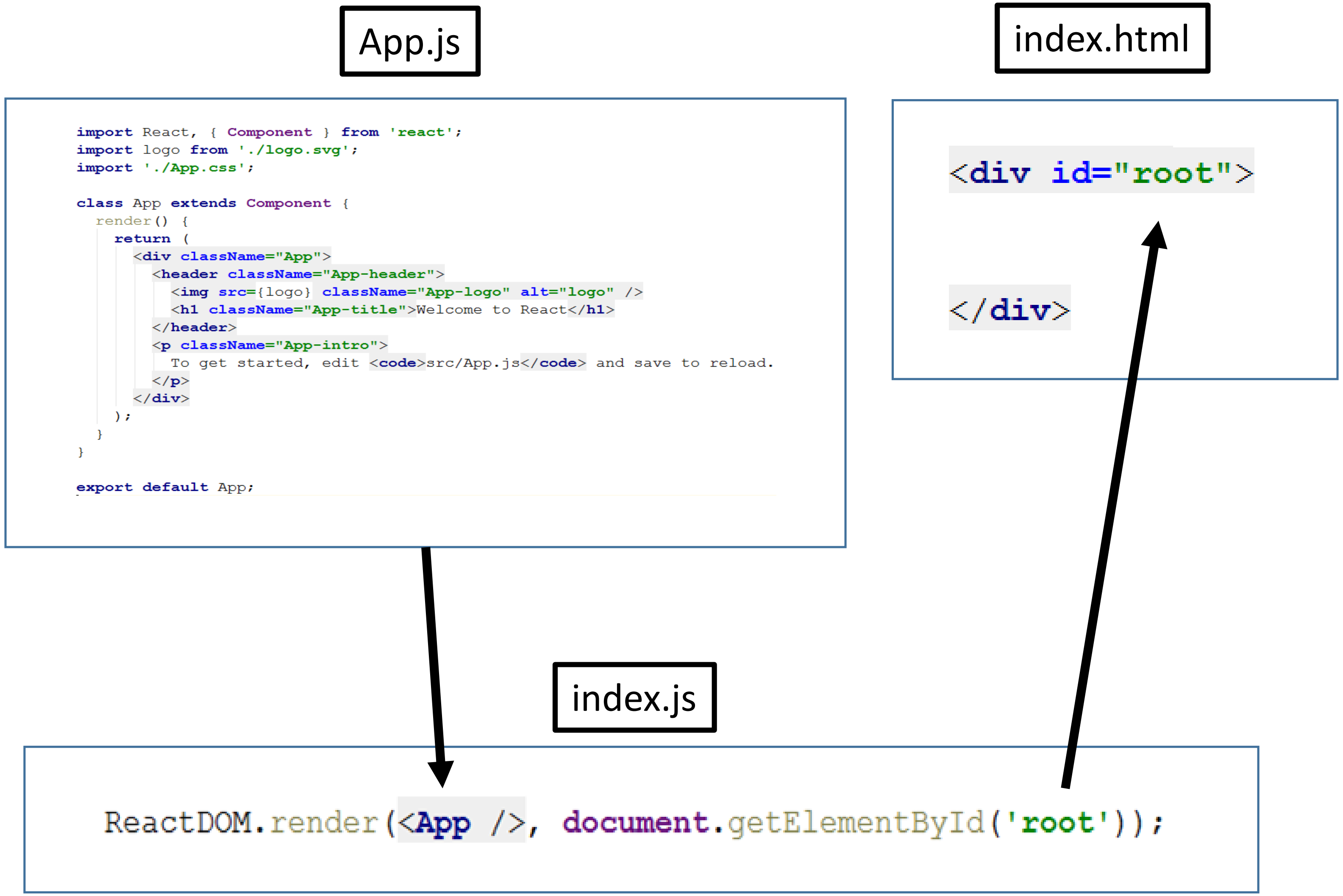
Easy to understand

Allows to combine expressions, calculations inside markup



```
<h3>Total = {2+2}</h3>
```

Understand how code works



App.js

```
import React, { Component } from 'react';
import logo from './logo.svg';
import './App.css';

class App extends Component {
  render() {
    return (
      <div className="App">
        <header className="App-header">
          <img src={logo} className="App-logo" alt="logo" />
          <h1 className="App-title">Welcome to React</h1>
        </header>
        <p className="App-intro">
          To get started, edit <code>src/App.js</code> and save to reload.
        </p>
      </div>
    );
  }
}

export default App;
```

index.html

```
<div id="root">


</div>
```

index.js

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

App.js will have user interface which we want to display

Index.html will have <div> section with the id=root

Index.js is responsible to capture the UI from App.js and renders(display) this UI in div section of index.html

accenture

MODULE SUMMARY

accenture

Module Summary

- What is ReactJS
- Features of ReactJS
- Architecture of ReactJS
- Project structure
- Why Babel
- What is JSX
- Understand how code works

THANK YOU

accenture