**What is React and how it works?**

* React is a library.
* Client Side have language, libraries and frameworks.
* **Client Side Language**
  + Javascript
  + Typescript
* **Client-Side Libraries**
  + jQuery
  + RxJS
  + jQlite
  + React JS
* **Client-Side Framework**
  + Angular JS
  + Knockout JS
  + Backbone JS
  + Angular

**What is difference between Language and Library?**

* Language requires functions defined explicitly to handle various interactions.
* Language requires lot of references.
* Library is a set of factories.
* Factory is a set of pre-defined functions.
* You can implement the existing function and define the functionality.
* Library reduces the compatibility issues.
* React JS is a library.

**What is difference between Library and Framework?**

* Library can build application.
* Library is passive[not active]
* Library cannot control the application flow.
* It requires browser events to make library active.
* Framework is a software architectural pattern that provides set of libraries to build application and can also control the application flow.

**What is React JS?**

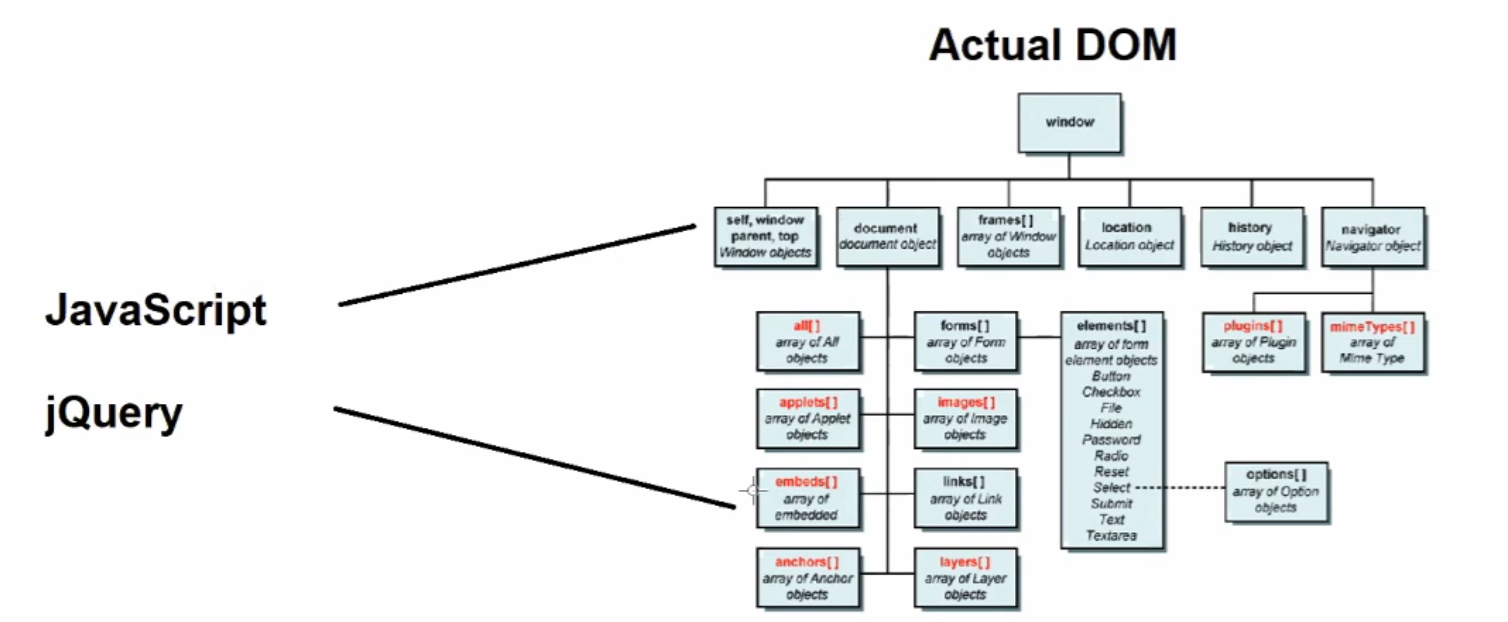
* React JS is a javascript library.
* Building user interface and handling user interactions client side.
* Facebook- Jordan Walke – For building SPA.
* 2013 initial release.
* 2020 – 17.0.1 latest stable version.

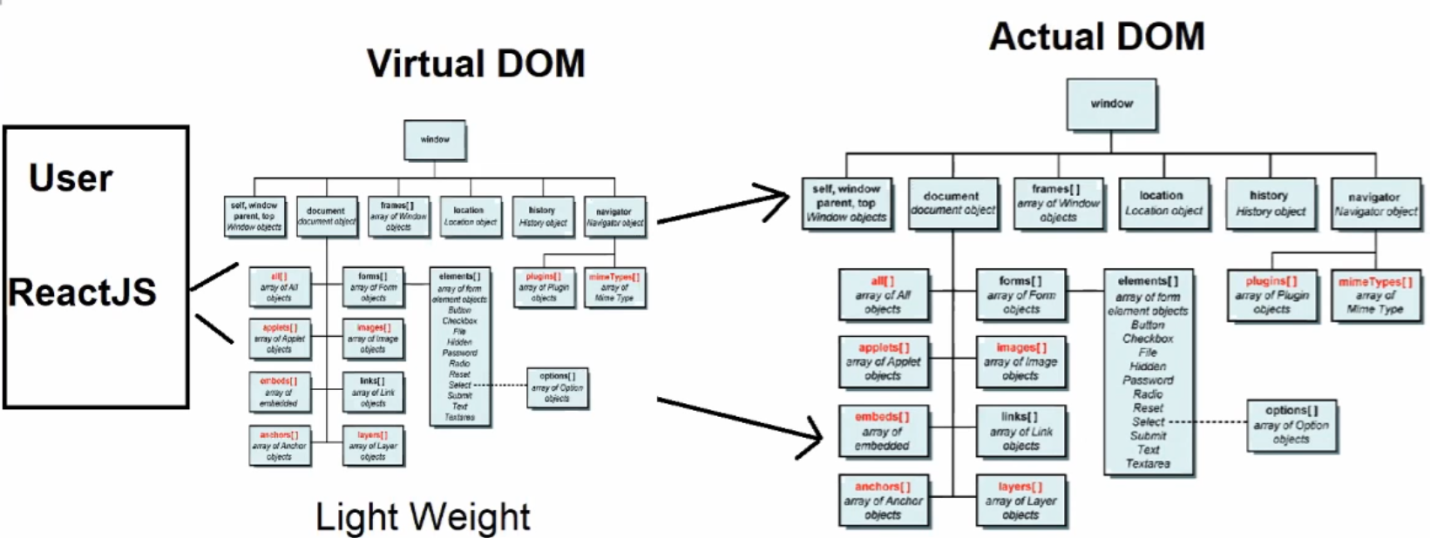
**It is not designed for what you are using, hence limited library.**

**React JS library requires several external libraries to integrate and handle functionality.**

**Features of ReactJS**

* ReactJS uses virtual DOM that makes the user experience better.
* Faster Interaction





* Faster Interaction as virtual dom is light weight
* React interacts with the virtual dom and the virtual dom interacts with the actual dom in the backend.
* **React uses JSX, which is easy to handle the manipulation on DOM elements**.
  + JSX is faster that JavaScript DOM manipulations.
  + Logic and mark-up can be defined in a single file.
  + It is easy to create template.
  + Template comprises of presentation and logic.
* **React uses ”One Way Data Binding”**
  + Data Binding is a technique used to update the component data to the view[UI]
  + Any change in component data will update to UI[HTML]
  + Changes made in UI are not directly updated back to component; they are handled on virtual DOM.
* **Component based architecture.** 
  + It is an alternative for legacy type library
  + It is more asynchronous
  + It loads only what is required for the situation.
  + It improves the performance of application.
  + It makes your library modular

**Concerns with ReactJS**

* Rapid changes in development
* Problem for developers to catch and update the pace of developing features
* Poor Documentation due to rapid changes.
* Poor SEO
* JSX [Presentation with Logic] – Insecure content are ignored due to jsx
  + Increases risk of XSS attacks[cross site scripting attacks]
* Need a lot of external libraries to handle various interactions.

**Setup Environment for ReactJS**

* Install Node JS
  + Node JS provides a package manager NPM
  + Package Mangaer is a tool used to install and configure library required for our application.
  + There are several package manager like: YUarn, NPM, Bower, Ruby Gems, NuGet etc.
  + Installing Node JS on your PC will get you NPM  
    <https://nodejs.org/en/download/> [make sure that node js verson 10+]
  + Ager installing Node JS check the version from command prompt
    - Node –v
    - Npm –v

**Install visual studio code [IDE- Integrated Development Environment]**

* IDE provides environment to build, debug, test and deploy application.
* You can use various editors: “editorconfig.org”
* Download and install VS code:  
  <https://code.visualstudio.com>
* Go to customize and install support for JS
* Go to extensions and install the following:
  + Live server: It is required to preview our application
  + Vscode-icons: It gives suitable icons for specific file extensions and names, so developer can easily identify the file type.
  + Intellisense for CSS class names: It provides suggestions for css

**Creating a new Project**

* Create a new folder for project on your PC
* Open the folder in VS code
* Open terminal: go to terminal menu and select “New Terminal” or ctrl+`
* Run the command
  + npm init

It will generate package.json that comprises of meta data

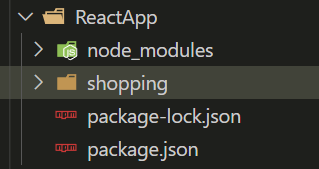
* **Install React Modules into Project:**
  + npm install --save react
  + npm install –save react-dom
* Installed Library will be maintained in **“node\_modules”** folder

**Create a React Application**

* Command:
  + npm install –g create-react-app
  + create-react-app shopping
  + npm start [Start the react Application]

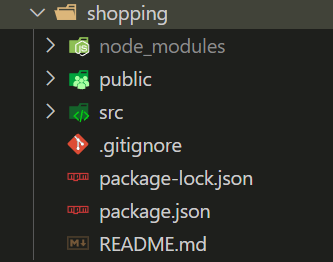
**React Workspace**

* It comprises of shared libraries for all your React JS projects.
* In workspace you can maintain multiple projects
* Workspace comprises of



|  |  |
| --- | --- |
| **File/Folder** | **Description** |
| node\_modules | * It comprises of project “dependencies” and “devDependencies” * The library required for your react application is stored in “node\_modules” * These dependencies can be shared to all projects in workspace. |
| package.json | * It comprises of information about the packages installed in your workspace |
| packagelock.json | * It comprises of packages meta data. |
| Shopping | * It is your react application in the workspace |

The files and folder of React application



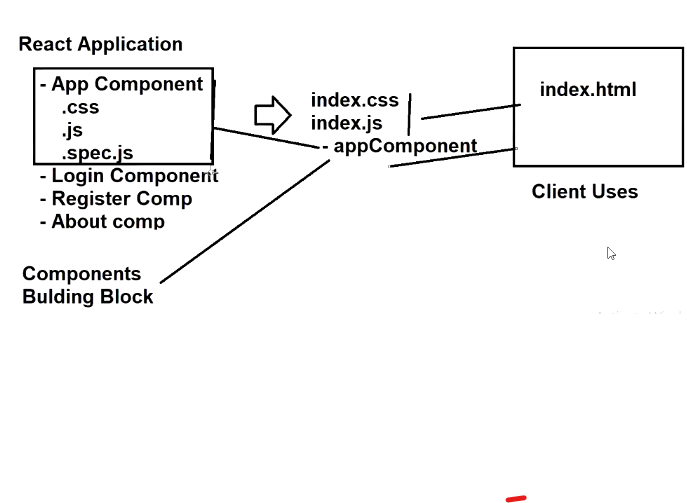
|  |  |
| --- | --- |
| **File/Folder** | **Description** |
| node\_modules | It contains local dependencies. |
| public | It comprises of all our project static files |
| src | It comprises of our project dynamic files. eg: CSS, js,etc.[LASS, SASS files] |
| .eslintcache | It is used to track and identity the issues in your javascript code. It can configure the rules for javascript. It ensures that proper javascript coding standard is followed. |
| .gitignore | It specifies the dependence to ignore while deploying on GIT |
| packagelock.json | Local dependencies meta data |
| package.json | Local dependencies information |
| README.md | Help Document |

**Files in Public Folder**

|  |  |
| --- | --- |
| Favicon.ico | Shortcut icon used for bookmarking |
| Index.html | It is the start up page. |
| manifest.json | It comprises of metadata. Information about your react application. Browsers can understand your application by accessing details from manifest. |
| Robots.txt | It comprises of User Agent details, which is information about the browsers supported. |

**Files in SRC Folder**

|  |  |
| --- | --- |
| App.css | It comprises of global styles used for application. |
| App.js | It comprises the logic required for default component ie. “app” component. Every project contains set of several components. React application by default comes with “app” component.  Application starts with “app” component.  Every component comprises of 3 files:   * .css : Styles * .js : Logic (JSX- HTML, JS) * .spec.js : It is test file |
| App.test.js | It is the test file for app component |
| Index.css | Styles for index page |
| Index.js | Contains jsx for index page |
| Setuptests.js | It configures the testing environment for application testing. |
| reportWebVitals.js | Error Reporting |



**What is Toolchain for React?**

* Toolchain is a set of programming tools that are used in performing a complex software development task or to create a software product.
* In general, “toolchain” is a set of tools used for building/developing an application.  
  Ex:  
  Language: used for writing the logic. Eg: javascript  
  Compiler: for translating in machine format. Eg: v8, babel, jit  
  Debugger: to identity and fix the bugs. [error/issues]  
  Testing: a framework for testing  
  Deployment: tool for deploying application on production.

**What is recommended toolchain for ReactJS application?**

* Package Manage [NPM, Yarn]
* Bundler [Webpack, Parcel]
* Compiler [Babel-differential loading]

|  |  |
| --- | --- |
| Package Manager | Installing and Maintaining dependencies.  Eg: npm, yarn |
| Bundler | Writing modular code and bundle it together for application.  Eg: Webpack, Parcel |
| Compiler | It allows handling differential loading.  Library will be classified according to browser. [Legacy, Modern] |

**What are popular toolchains used for Reactjs?**

* Neutrino
* Nx
* Parcel
* Razzle

**When do we need the individual toolchain?**

* When we are trying to integrate ReactJS into existing application.

**What is NPX?**

* It is a tool that comes with NPM.
* It can directly create react application.
* We don’t require “create-react-app” to install if we are using “npx”.
* npx create-react-app yourAppName

**How to add “ReactJS” into an existing application?**

* Create a new folder for project “C:\\HTMLwithReact
* Open your folder in “Visual Studio Code”
* Open Terminal [Ctrl+`] or Open folder location in command prompt
* Run the Command  
  > npm init

{

  "name": "htmlreact",

  "version": "1.0.0",

  "description": "",

  "main": "index.js",

  "scripts": {

    "test": "echo \"Error: no test specified\" && exit 1"

  },

  "keywords": [

    "HTML",

    "Application",

    "React",

    "Application"

  ],

  "author": "Sumeet",

  "license": "MIT"

}

* “package.json” file is created
* Add a new file into public folder “index.html”
* You can install dependencies for project using “npm”  
  Eg:  
  > npm install bootstrap
* It will generate “node\_modules” and “package-lock.json”
* To integrate “ReactJS” into your page you need
  + React.developement.js
  + React-dom.development.js
* You have to get these files from official ReactJS website CDN links
* Add these scripts into head section:

<script crossorigin src="https://unpkg.com/react@18/umd/react.development.js"></script>

<script crossorigin src="https://unpkg.com/react-dom@18/umd/react-dom.development.js"></script>

* Add babel cdn to your page: it will act as the compiler

    <script crossorigin src="https://unpkg.com/@babel/standalone/babel.min.js"></script>

**Create React JS Component**

* Components are building blocks for creating applications.
* Technically component is a template.
* It comprises of design and functionality, which you can inject and use in any page.
* Components will provide re-usable templates for ReactJS application.
* Components are classified into 2 major types:
  + Functional Component
  + Class Component
* Component is designed by using JSX[Javascript Extension]
* ReactJS component can be embedded into Page or can be defined in module.
  + Embedded Technique
    - Component is designed and used in the same page.
  + Module Technique
    - Component is designed in a separate JavaScript [“.js”]
    - Component is linked to any page.

Eg: Component using Embedded Technique

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>HTML React App</title>

    <script crossorigin src="https://unpkg.com/react@18/umd/react.development.js"></script>

    <script crossorigin src="https://unpkg.com/react-dom@18/umd/react-dom.development.js"></script>

    <script crossorigin src="https://unpkg.com/@babel/standalone/babel.min.js"></script>

    <!-- <script src="../src/hello.js"></script> -->

    <script type="text/babel">

        ReactDOM.render(

            <p>Welcome to React JS</p>,

            document.getElementById('reactContainer')

        )

    </script>

</head>

<body>

    <h2>HTML React App</h2>

    <div id="reactContainer">

    </div>

</body>

</html>

**Note: The MIME type of React js “JSX” script must be “text/jsx” or “text/babel”**

Syntax:

<script type="text/jsx">  
<script type="text/babel">

**Embedded Component:**

* In embedded technique the component is designed in the page.
* It is faster in access.
* It reduces the load time.
* It reduces the number of requests made to page.
* Disadvantage:
  + Reusability issues
  + Extensibility issues
  + Maintainability and testability issues.

**Module Technique:**

* Component is designed in a separate Javascript file “.js”
* Javascript file is considered as a Module.
* Module comprises of classes and functions
* It is easy to extend
* Easy to re-use
* Easy to test
* Disadvantage:
  + Using an external file will always increase the number of requests made to page and also the page load time.

Eg:

* Go to “src” folder and add a new file “hello.js”
* Add the following code into “hello.js”

ReactDOM.render(

    <p>Welcome to External React JS component</p>,

    document.getElementById("reactContainer")

);

* Link to your HTML page:

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>HTML React App</title>

    <script crossorigin src="https://unpkg.com/react@18/umd/react.development.js"></script>

    <script crossorigin src="https://unpkg.com/react-dom@18/umd/react-dom.development.js"></script>

    <script crossorigin src="https://unpkg.com/@babel/standalone/babel.min.js"></script>

    <script type="text/jsx" src="../src/hello.js"></script>

    <script type="text/babel">

        ReactDOM.render(

            <p>Welcome to React JS</p>,

            document.getElementById('root')

        )

    </script>

    <script type="text/javascript">

        function bodyload(){

            alert("ReactJS Loaded...");

        }

    </script>

</head>

<body onload="bodyload()">

    <h2>HTML React App</h2>

    <div id="root">

    </div>

    <div id="reactContainer">

    </div>

</body>

</html>