|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| REACT COMPLETE DEVELOPER RESUMY | | | | | | | |
| **CHAPITER ONE** | | | | | | | **REACT KEY CONCEPTS** |
| **BEFORE REACT (The birth of React) ?** | | | | | | | |
| Using HTML, CSS & JavaScript to manipulate web pages.  The problem is, all these websites run on different browsers which they worked by different group of developers. So, each one of these browsers is implemented differently. We have to insure that JS code work on all of these variety of browsers. | | | | | | | |
| **JQUERY** | | | | Technology (library) which allow developers to easily interact with the **DOM**(Document Oriented Object) across all these browsers. | | | |
| Developers started to build bigger & bigger applications instead of just something small.(Like Facebook, Airbnb …) | | | | | | | |
| **BACKBONE.JS** | | | | | | Library that organize JavaScript file. | |
| **SPA** | The birth of Single page applications because it becomes easier to work with the DOM. | | | | | | |
| **AJAX** | | | It stands for **A**synchronous **J**avaScript **A**nd **X**ML. It’s not a programming language. It’s a combination of:   * A browser built-in XML Http Request object(to request data from a web server). * JavaScript and HTML DOM (to display or use the data). | | | | |
| Application acts like Desktop application by staying on the same page the entire time and JavaScript file simply changes or updates the HTML file or the DOM. So, we can sign-in & interact with that application without ever communicating with the server. | | | | | | | |
| **ANGULARJS** | | | | | Created by Google in 2010, it became the standard way of building applications. It allowed developers to build these large scale applications by forming these containers that well wrapped your project, and because of Google, it had a lot of power. | | |
| Things started to get more & more complex. It becomes harder and harder to find bugs in the code and understand how each part of the app is affecting the other. | | | | | | | |
| **REACT** | | Facebook developers see the need of thinking about how they should organize code, how they should manipulate data, and how the data flows through the application in general. Facebook teams release the first version of react in 2013 to their developers teams. | | | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **REACT CONCEPTS ?** | | | |
| 1 | **Don’t touch the DOM. I’ll do it** | | |
| DOM is what the browsers uses to display a website or web application, and JavaScript is simply manipulating this DOM. It’s just a tree representation of the page. | | |
| **Imperative programming** | | |
| It’s an imperative paradigm you directly change individual pars of your app in response to various user events. | | |
| The problem with Imperative programming, it becomes difficult to see the relationships between events and all these edge cases. | | |
| DOM manipulation is one of the biggest performance bottlenecks (expensive operation). It takes a long time for DOM changes to happen. Browsers has to do really intensive operations which affect performance. | | |
| **Declarative programming** | | |
| React is taking care of DOM changing operations. We give the state of our application & React automatically do the changes for us. The major thing here is, that all different states are accounted for in one place(One single source of truth). One single JavaScript file that describes how our app should look. These kind of programming provides:  Less code complexity, better code quality, faster developers times. | | |
| **2** | **Build websites like Lego blocks** | | |
| React is designed around the concept of reusable components. Each of these individual pieces create the whole application. People & developers can now share these components across different websites. | | |
| **Components** | | These components are created based on the state of the application. Components are simply just JavaScript functions that receive some sort of input or attributes which we call props, and it returns this kind of looks HTML. |
| **JSX** | A syntax extension to JavaScript. It is recommended to use it with react to describe what the UI should look like. | |
| **3** | **Unidirectional Data flow** | | |
| **VITUAL DOM** | | This approach enables the declarative API of React. We tell React what state we want the UI to be in, and it makes sure the DOM matches that state. (JavaScript Object that describe our web page) |
| Any time, we want the DOM change, data has to change in our app. App state moves in only one direction from the top to the bottom. By adding these restriction, it becomes easy to debug code. | | |

|  |  |
| --- | --- |
| **4** | **UI, The rest is up to you** |
| React works only with view (UI). Provides the variety of cross-platform (Desktop: electron, mobile: React Native, VR: React VR) |
| **THE JOB OF A REACT DEVELOPER** | |
| **1** | Decide on Components |
| **2** | Decide the state and where it lives |
| **3** | What changes when state changes |