React JS

React

- The initial React release was 2013 by Facebook.
- React is a library made over javascript
- In recent years single page applications (SPA) have become popular.
- React is not an SPA framework but a “view” library.
- It is the V in the MVC (Model-View-Controller architectural pattern).

React

- Model-View-Controller (MVC) is an application model comprised of three dependent layers.
  - The model (data)
  - The view (user interface)
  - The controller (processes that handle input).
- React only enables you to render components as viewable elements in a browser.

React

- A Single Page Application (SPA) differs from a normal web application
- In SPA that you remain on the same URL and thereby the same page, hence the name.
- Traditionally in HTML we create multiple HTML files for multiple page website
- In SPA we create one HTML page and create Routes on them to show different pages

Why React

Why React

Because of :

- Its compositional model
- Its declarative rather than imperative
- Unidirectional Data flow/binding
- React is simply javascript

Compositional Model

Compositional model

- Composition is an act or mechanism to combine simple functions to build more complicated ones.
- Why Composition?
- Two things to remember:
  - Simple functions
  - Combination of simple functions to make another function

Compositional model

- Let's look at an a simple JS function example:

```
function getTwitterProfile (username) {
    return 'https://twitter.com/' + username
}
```

It’s a very simple function

Compositional model

- Similarly, the simple getTwitterUserPic function to return url for user twitter profile picture:

```
function getTwitterUserDP (username) {
   return 'https://avatars.io/twitter/' + username + '/medium'
}
```

Another very simple function

Compositional model

- Let's combine both functions

```
function getTwitterProfileData (username) {
    return {
        twitterProfile: getTwitterProfile(username),
        profilePic: getTwitterUserDP(username)
    }
}
```

That is composition!

QUESTION ARISES?

Why three function instead of one directly?

Compositional model

What we did

```
function getTwitterProfile (username) {
     return 'https://twitter.com ' + username
}
```

Function 1

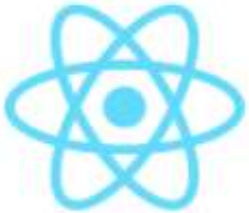
```
function getTwitterUserDP (username) {
   return 'https://avatars.io/twitter/' + username + '/medium'
}
```

Function 2

```
function getTwitterProfileData (username) {
   return {
       twitterProfile: getTwitterProfile(username),
       profilePic: getTwitterUserDP(username)
   }
}
```

Composite Function

Compositional model



How about

```
function getTwitterProfileData (username) {
   return {
     twitterProfile: 'https://twitter.com ' + username,

     profilePic: 'https://avatars.io/twitter/' + username + '/medium'
   }
}
```

Non composite way

```
function getTwitterProfileData (username) {
   return {
       twitterProfile: getTwitterProfile(username),
       profilePic: getTwitterUserDP(username)
   }
}
```

Composite Function

Compositional model

How about

```
function getTwitterProfileData (username) {
   return {
     twitterProfile: 'https://twitter.com ' + username,

     profilePic: 'https://avatars.io/twitter/' + username + '/medium'
   }
}
```

Non composite way

- Two separate functions with one composite function is better as it increase reusability.
- There is always a one rule for a good function, a rule of **"DOT"**, Do one thing!

React & Composition

- In React we rely on composition, heavily!
- In React we create Components to build different sections of a website
- Components are building blocks in React.
- For example following are three different components:
  - <LandingPage />
  - <AboutUs />
  - <ContactUs />
- Currently they are independent component

React & Composition

- In React, we can have a composite component as simple as

```
<LandingPage>
    <AboutUs />
    <ContactUs />
</LandingPage>
```

- <LandingPage> become parent component and other become child component
- This way we can use individual component blocks to build a big website.

Declarative Nature

Declarative nature

- Most of JavaScript is imperative code.
- We spoon feed each and every step to make javascript aware of how to get desired result.
- Let's take a example water tank level
  - Manual - (Imperative)
  - Auto - (Declarative)

Declarative nature

Imperative code!

```
const teachers = ['Zia', 'Irfan', 'Muneeb', 'Aamir']
const titles = []

for (let i = 0; i < teachers.length; i++) {
     titles[i] = 'Mr. ' + teachers[i]
}

console.log(titles)
```

RESULT: **['Mr. Zia', 'Mr. Irfan', 'Mr. Muneeb', 'Mr. Aamir']**

Declarative nature

Declarative code!

```
const teachers = ['Zia', 'Irfan', 'Muneeb', 'Aamir']

Const titles = teachers.map(name => 'Mr. ' + name )

console.log(titles)
```

RESULT: ['Mr. Zia', 'Mr. Irfan', 'Mr. Muneeb', 'Mr. Aamir']

React - Declarative Nature

In React we will soon going to see declarative code like following

```
<PrintOnBrowser name='Aamir' />

<PrintOnBrowser name='Aamir Pinger' />
```

RESULT on Browser:

Aamir

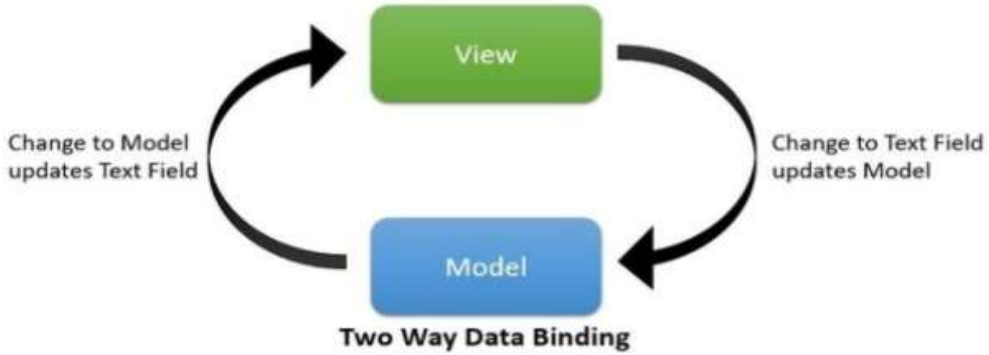
Aamir Pinger

Imperative/ Declarative

1. **Imperative** : changing the DOM by your self (individual part you directly change by your self.)
2. **Declarative** : we only tell to react what to change in DOM, and react do it by itself

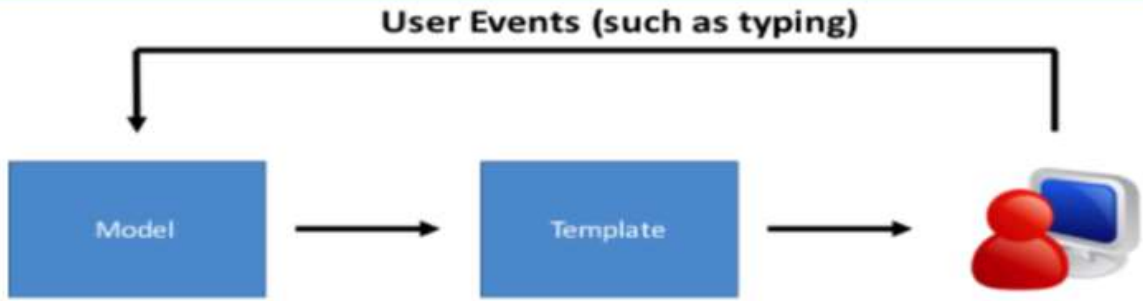
Unidirectional Data Binding

Two way data binding



- Many front-end framework like Angular uses two way data binding.
- Two-way data binding look really great, but when application grows it is hard to determine where the data is actually being updated.

Unidirectional Data Binding



- One-way data binding only propagates changes from the model to the UI, not vice versa.
- It is known as **"single source of truth"**.

Source: https://www.accelebrate.com/blog/two-way-data-binding-angular-2-and-React/

REACT CONCEPTS

1. Don't touch the DOM. I'll do it
2. Build websites like lego blocks
3. Unidirectional data flow

React is simply Javascript

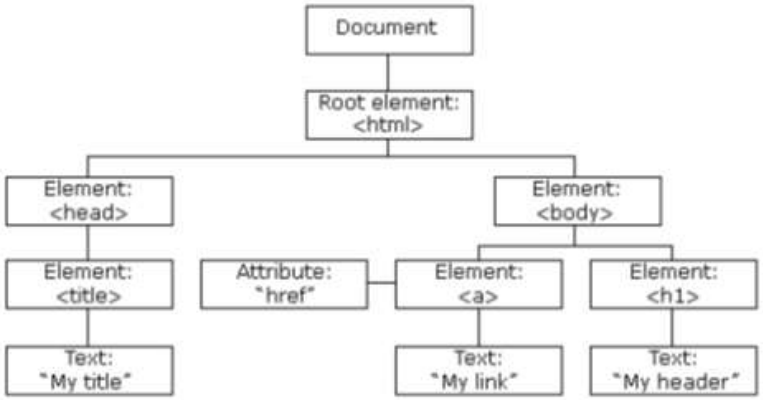
React is simply Javascript

- Uptil now we haven't seen any code other then javascript code
- React is small library based on Javascript
- Even components in React are JavaScript class or function
- Arrow functions, .map() and .filter() will be seen used extensively in any React code

DOM vs Virtual DOM

DOM

- DOM is Document Object Model
- It's a programming interface for HTML and XML documents
- When a web page is loaded, the browser creates a Document Object Model of the page.

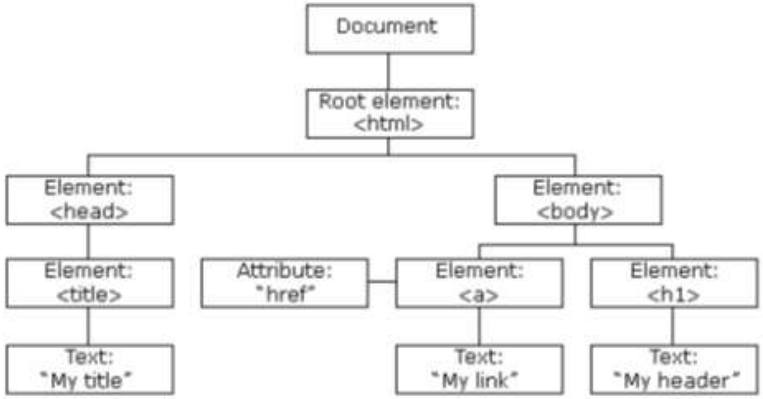


DOM Tree

https://www.w3schools.com/js/js_htmldom.asp

DOM

- The DOM is an object-oriented representation of the web page.
- Scripting language such as JavaScript can modify the document structure, style, and content.



DOM Tree

https://www.w3schools.com/js/js_htmldom.asp

Virtual DOM

- React introduced Virtual DOM (VDOM)
- The VDOM is a programming concept where a virtual representation of a UI is kept in memory
- It's is a tree based on JavaScript objects created with React that resembles a DOM tree
- A process called Reconciliation is used to sync Real DOM with VDOM
- React uses ReactDOM Library updates VDOM and render it on actual DOM

QUESTION ARISES?

Why Virtual DOM is needed?

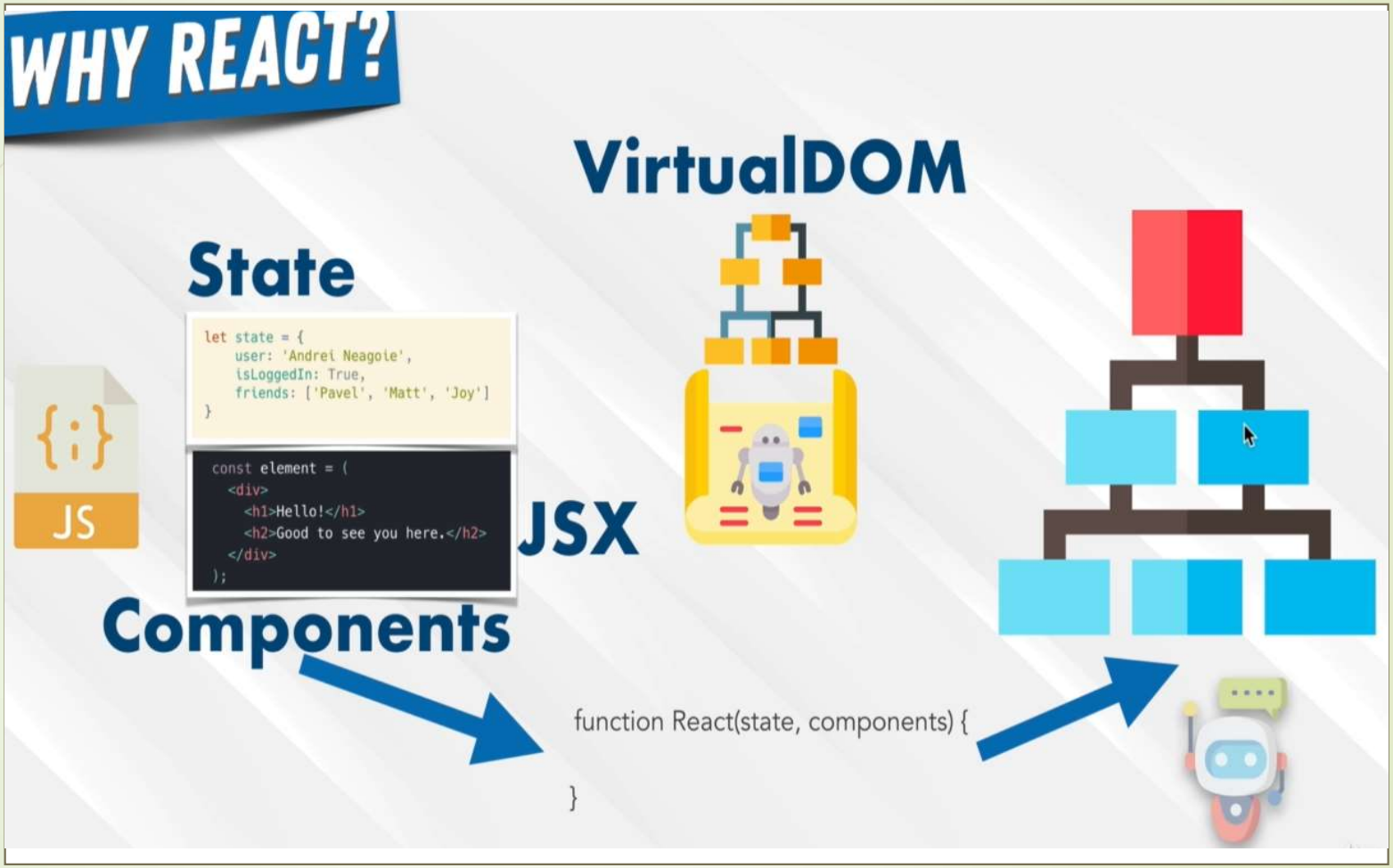
Why Virtual DOM is needed?

- Making changes in memory (VDOM) is quite faster than updating a complete browser screen (Real DOM)

- React creates first VDOM when application launches and then put everything on browser

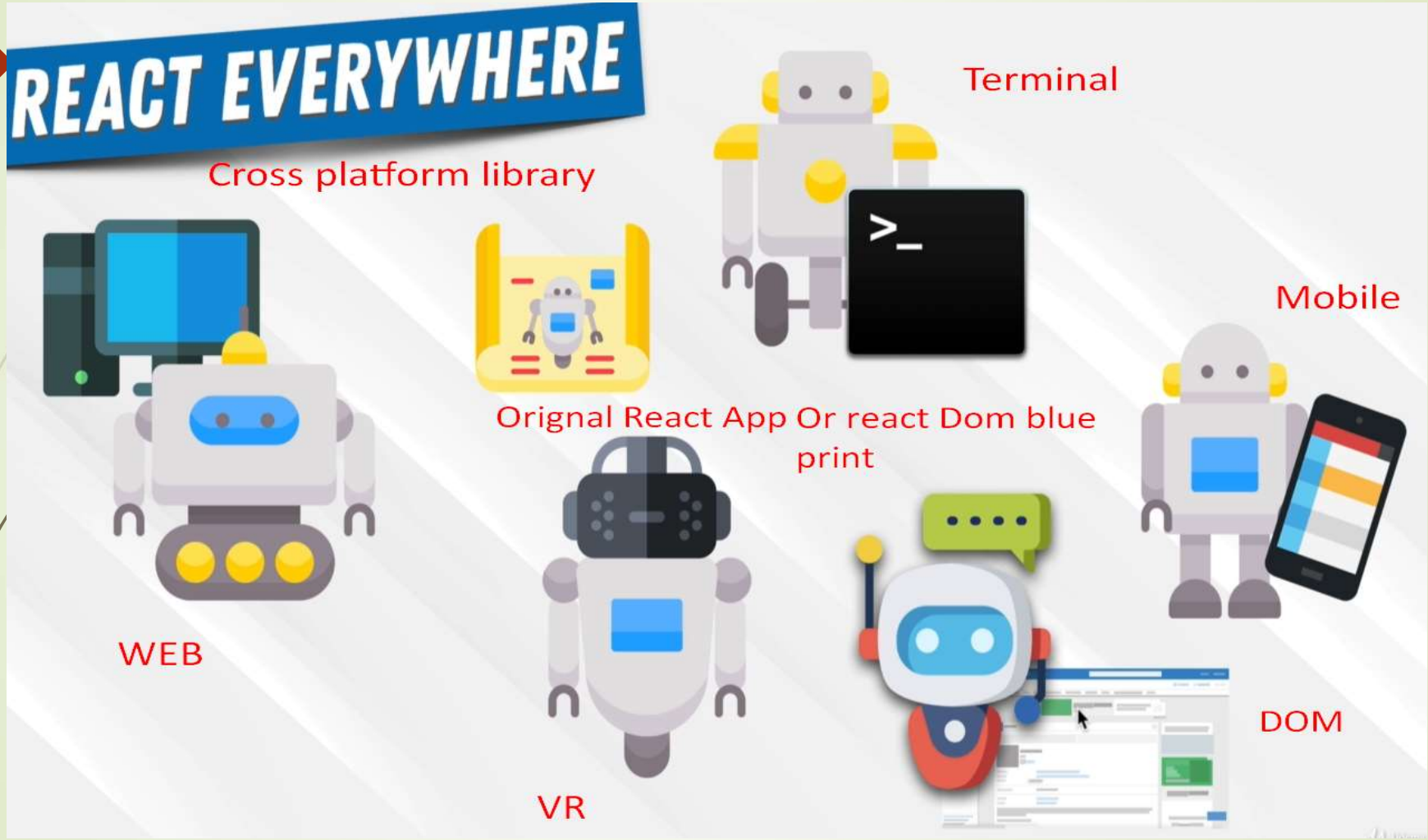
Why Virtual DOM is needed?

- Once app need to update browser screen, React creates new updated VDOM
- Through reconciliation process React find out difference between new and old VDOM
- Lastly only updates the difference to browser (Real DOM)


WHY REACT?
VirtualDOM
State
let state = {
user: 'Andrei Neagoie',
isLoggedIn: True,
friends: ['Pavel', 'Matt', 'Joy']
}
JS
const element = (

<h1>Hello!</h1>
<h2>Good to see you here.</h2>

);
JSX
Components
function React(state, components) {
}


REACT EVERYWHERE
Cross platform library
Terminal
>_
Mobile
Orignal React App Or react Dom blue print
WEB
VR
DOM

You can use same code and same technologies concepts/ rules for building any kind of apps

1. React 360
2. React blessed -> terminal
3. React desktop
4. React JS
5. React Native

Setting up React App

Three Methods for using React JS

- By using CDN LInks
  - https://www.w3schools.com/react/react_getstarted.asp
- Through Installation Book Page 21
- Sandbox

Setting up React

Setting up React

- Easiest way to setup React environment is by using **create-react-app**
- **Create-react-app** scaffold a basic React application with ease and will get you up and running in no time.

To install create-react-app

```
aamir@ap-linux:~$ npm install create-react-app -g
```

Setting up React

To scaffold React application

```
aamir@ap-linux:~$ npx create-react-app my-app

aamir@ap-linux:~$ cd my-app

aamir@ap-linux:~/my-app$ npm start
```

http://localhost:3000