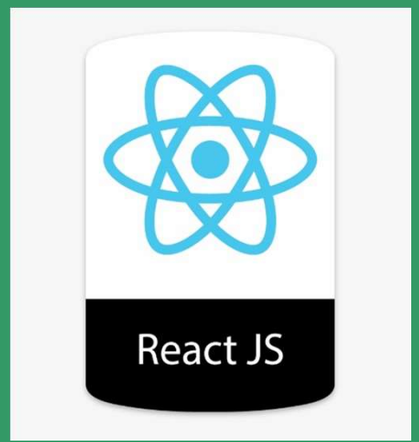
4

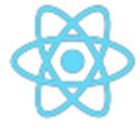
React JS



Deep Dive

React - Characteristics

- React apps are built with the latest version of ES.
- React makes use of reusable components.
  - A component is a function/class that returns a section of the interface.
- React is declarative
  - React allows to describe what the application interface should look like
- Unidirectional data flow
  - React applications are built as a combination of parent and child components.
    - Data always flows from parent to child and never in the other direction.
- Powerful type-checking using PropTypes



React – The Virtual DOM

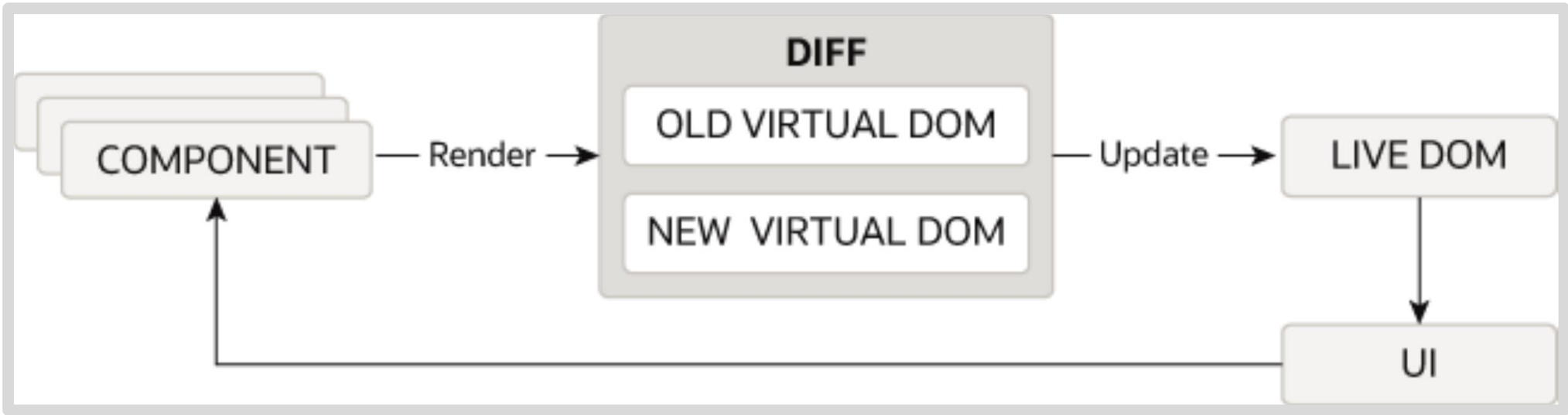
- DOM manipulation is the heart of most modern and interactive web applications.
- DOM manipulation is slow, moreover many JavaScript libraries and frameworks update the DOM more than needed.
- In React for every DOM object there is a corresponding Virtual DOM object.
- A Virtual DOM object is an in-memory representation of the real DOM object.
  - It has the same properties as the real DOM object.
- It lacks the power to directly change what is on the screen.

About Virtual DOM Architecture

- Virtual DOM architecture is a different way of building apps and web components from the Model-View-ViewModel (MVVM) architecture

- Virtual DOM architecture is a programming pattern where a virtual representation of the DOM is kept in memory and synchronized with the live DOM by a JavaScript library.
- As a pattern, it has gained popularity for its ability to efficiently update the browser's DOM (the live DOM).
- React is the JavaScript library uses to synchronize changes in the virtual DOM to the live DOM.

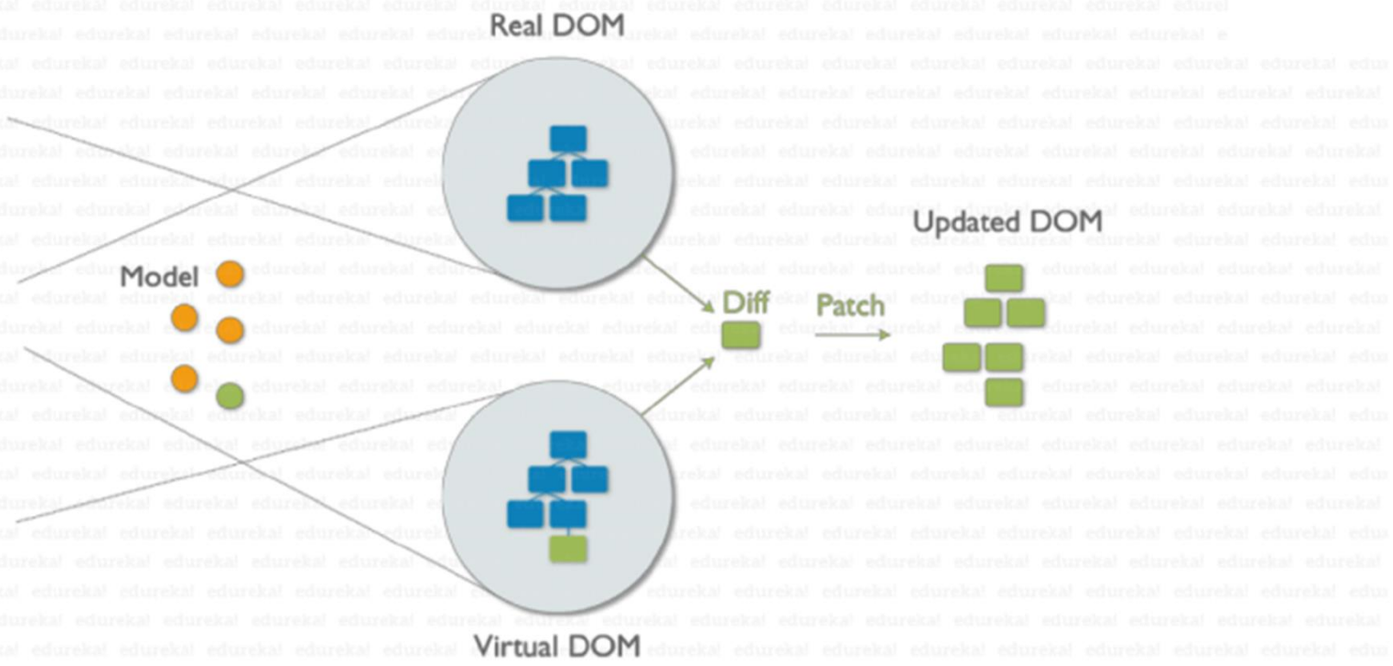
VDOM


DIFF
COMPONENT
Render
OLD VIRTUAL DOM
NEW VIRTUAL DOM
Update
LIVE DOM
UI

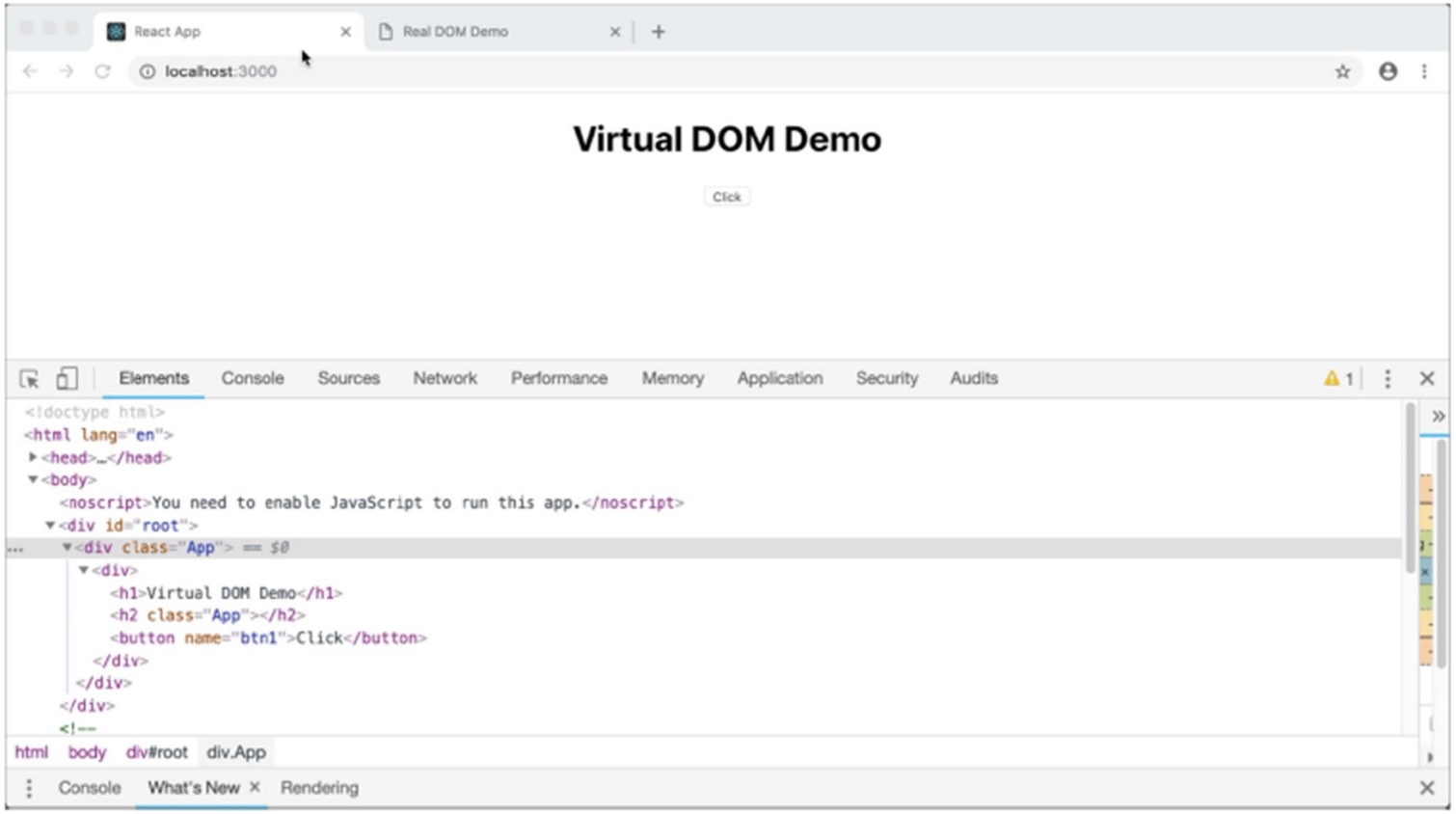
Virtual DOM…

- Manipulating the DOM is slow whereas manipulating the virtual DOM is fast as nothing gets drawn on the screen.
- Once the virtual DOM is updated React compares it with its previous state.
- "**Diffing**" is the process by which React figures the virtual DOM objects that have changed.
- React updates only the changed objects in the real DOM.
- Changes on the real DOM cause the screen to change.

The Virtual DOM

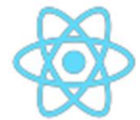


Virtual DOM / Real DOM



Setting up the Environment - Workflow

- Why do we need to setup a workflow?
  - Code optimization
    - This increases the performance of the application.
  - Use ES6 features
    - This is the de-facto standard for react that makes the code easier to read, leaner and faster.
    - Though code is written in ES6 the shipped code needs to run in multiple browsers.
    - The workflow needs to compile ES6 features into browser comprehend able ES.
  - Increase Productivity
    - CSS auto-prefixing to achieve broad browser support.
- The build workflow needs to allow developers to write modern code that facilitates all of the above.



Setting up the Environment - Workflow

- A dependency management tool - ***npm***
  - npm (Node Package Manager) is a command line utility that ships with NodeJS (downloadable from http://www.nodejs.org)
  - Third party libraries and packages are dependencies
  - Compiler to convert ES6 to ES5 is a dependency
  - Build tools are dependencies
- A bundler - ***webpack***
  - Code is authored in a modular manner and split over multiple files.
  - These files need to be bundled when shipped as browsers do not support segmented files.
  - Webpack allows setup of build steps before bundling
    - Babel + presets can be configured in webpack to be a part of the bundling process
- A development Web server

Setting up a React project

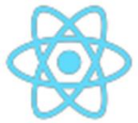
- A tool called “Create React App” that helps create a project.

  `https://github.com/facebook/create-react-app`

- This tool is maintained by Facebook and has a community around it.
- It is the officially recommended tool for creating *React* projects.
- Considering that we have node installed from **`http://nodejs.org`**

  `npx create-react-app <projectname>`

  - **`create-react-app`** is a node package.
  - **`npx`** is a **`npm`** package runner



Setting up a React project…

- A folder with the name specified will be created in the current path and the project files downloaded into it.

```
cd <projectname>
npm start
```

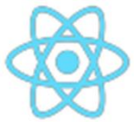
  - This starts the development server, loads the application that can be accessed at **`http://localhost:3000`**
  - The process started with **`npm start`** needs to be kept running since the server can detect changes made to the code and reload the page in the browser.

Folder structure

- Open the folder in VS Code (IDE).
- The general dependencies of the project are defined in **`package.json`**
  - **`react`** is the library to help with logic and components
  - **`react-dom`** helps render the components into the DOM
  - **`react-scripts`** is a package that helps set up the build workflow, development server and required ES6 support.
- The **`node_modules`** folder contains all the dependencies.
- The **`public`** folder is the root folder that gets served by the web server.
  - The **`index.html`** is the single html page residing in this folder.
  - There will be no more html pages in this folder.
- Script files are placed in the **`src`** folder.

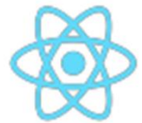
File structure

- The script files will get injected into **`index.htm`** by the build workflow.
  - The **`<div id="root"></div>`** is the mount point for the react application.
- **`manifest.json`** is added by the *create-react-app* to provide a progressive web application out of the box.
- **`index.js`** gets access to the **`root`** element in the DOM and renders the react application with the **`render()`**
- **`App.js`** contains the react component created by *create-react-app*
- **`registerServiceWorker.js`** created by *create-react-app* is used to help pre-cache files and is related to progressive web application.



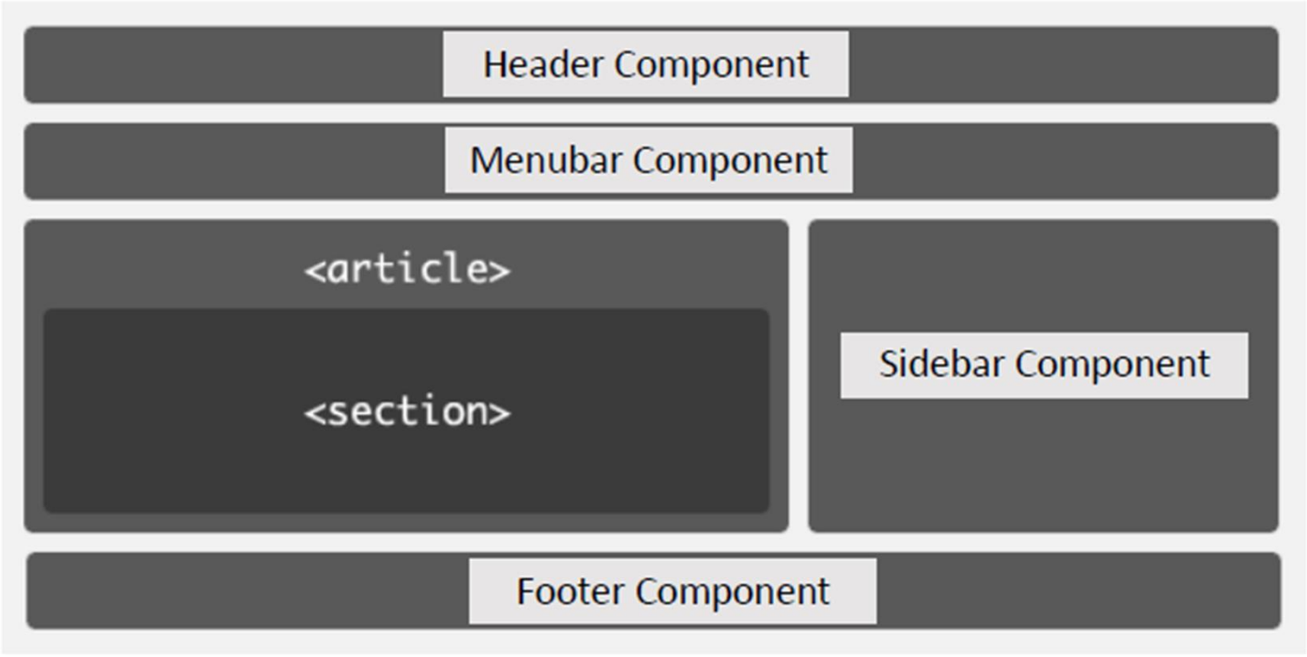
Components

- A Component can be considered as a custom HTML element.
- React therefore addresses the problem of having to build sophisticated user interfaces with HTML and JavaScript.
- *A component is a simple, manageable, maintainable and reusable piece of code*.
- The component can be plugged into web pages desiring a specific functionality.
  - Enhancements to the component lead to enhancement of the entire page.

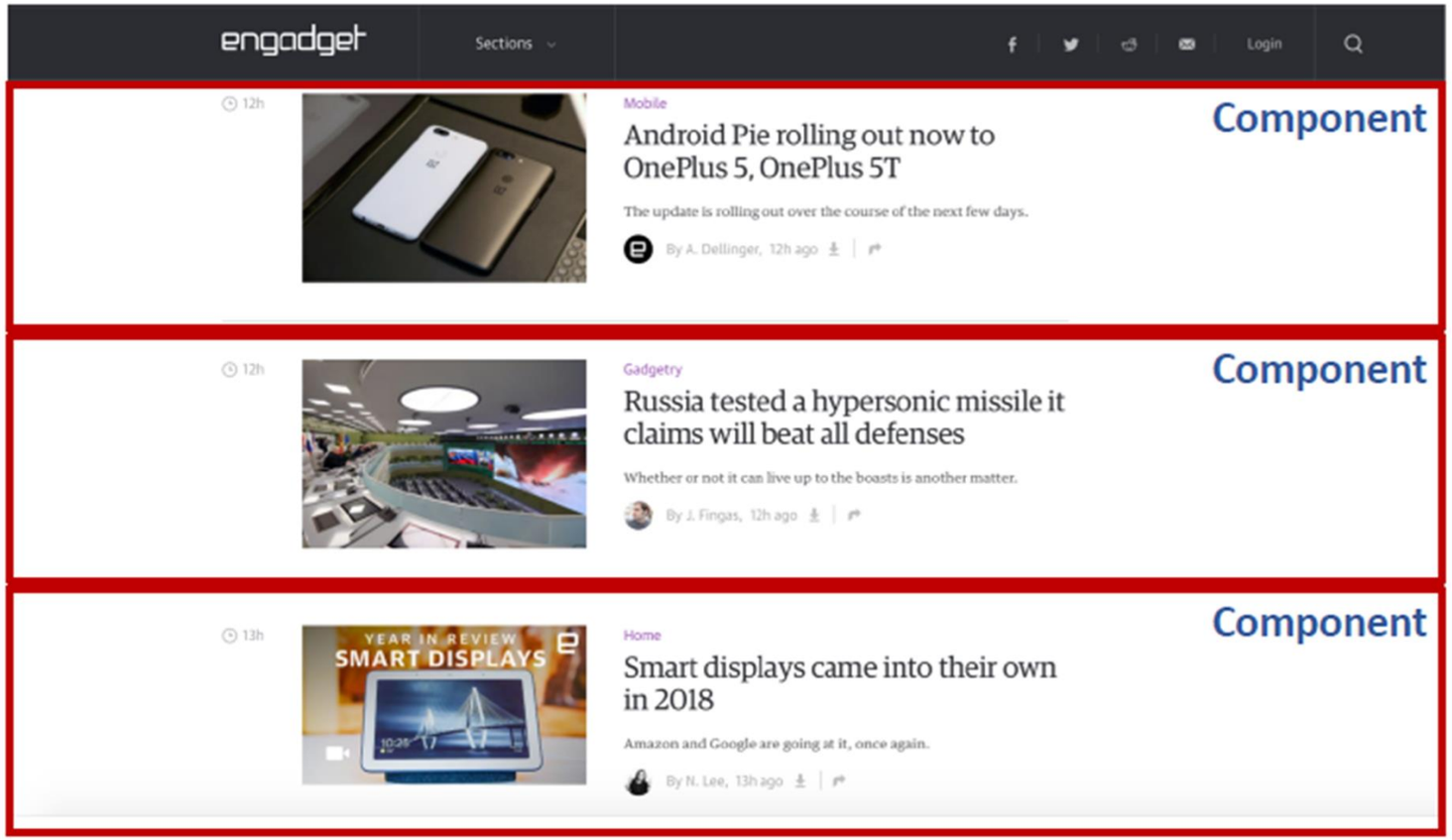


Components?

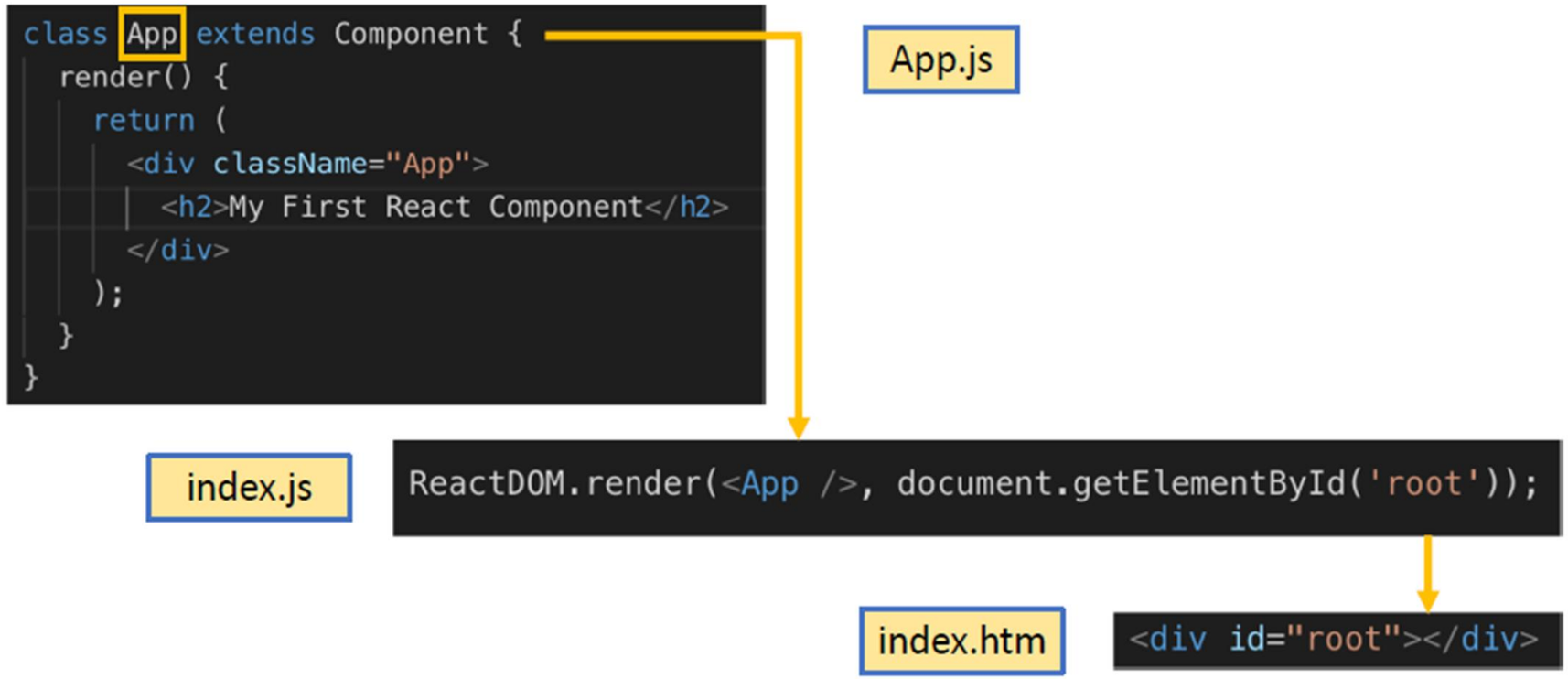
- Every web page can be split into components.
- Components can be reused and hence help simplify the creation of web pages

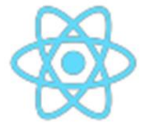


Components…


engadget
Sections
Login
12h
Mobile
Android Pie rolling out now to OnePlus 5, OnePlus 5T
The update is rolling out over the course of the next few days.
By A. Dellinger, 12h ago
Component
12h
Gadgetry
Russia tested a hypersonic missile it claims will beat all defenses
Whether or not it can live up to the boasts is another matter.
By J. Fingas, 12h ago
Component
13h
YEAR IN REVIEW
SMART DISPLAYS
Home
Smart displays came into their own in 2018
Amazon and Google are going at it, once again.
By N. Lee, 13h ago
Component

React Components





React Components

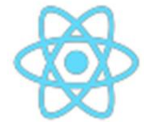
- In a react application generally, one root component is rendered.
- All other components needed by the application are nested in there.
- A react component is extended from the **`Component`** class that is imported from the **`react`** library.
- The component has a method called **`render()`**
  - The **`render()`** method returns "HTML like" content.
  - This method is called by ***react*** to emit content into the DOM and thereby render content to the view.

JSX?

- JavaScript eXtension (JSX) is used in React to describe the UI.
- JSX is a preprocessor step that adds XML syntax to JavaScript.
  - Like in XML, JSX tags have a tag name, attributes and children.
  - If a value is enclosed in quotes, it is regarded as a string.

```
const appElement = <h2 className='App'>React Component</h2>;
```

- JSX makes React a lot more elegant though React can be used without JSX.
- JSX allows HTML to be put into JavaScript.



JSX

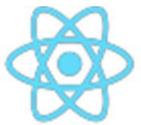
- React embraces "*separation of concerns*" with loosely coupled units called components that contain both markup and logic.
- React uses JSX for templating instead of regular JavaScript.
- JSX syntax is intended to be used by preprocessors (like Babel) to transform HTML-like text into standard JavaScript objects that can be parsed by a JavaScript engine.
- Familiarity of HTML helps create templates quicker.
- JSX is faster since optimization is performed when compiling code into JavaScript.
- JSX produces React "Nodes"

React Nodes

- A React Node is defined as a light, stateless, immutable and virtual representation node.
- React nodes are not real DOM nodes themselves, but a representation of a potential DOM node.
- The representation is considered the virtual DOM.
- React is used to define a virtual DOM using React nodes.
- React nodes can be created using JSX or JavaScript.

Creating React Nodes

- Creating React nodes using JavaScript is accomplished by **`React.createElement()`**
- This method is used to create a virtual DOM representation of an element node.
- To create the virtual DOM the React element node should be rendered to a real DOM.
- This is achieved using the **`ReactDOM.render()`** method.



JSX...

- JSX is simply converting XML-like markup into JavaScript.

```
<h2>React Component</h2>
```

- gets compiled to

```
React.createElement('h2',null,'React Component')
```

  - The first argument is the element that is to be rendered
  - The second argument is a JavaScript object and is optional
  - The third argument represents the children (the content nested in the element mentioned as the first argument)

JSX…

```
<div className="App"><p>React Component</p></div>
```

- transforms to

```
React.createElement('div',{className:'App'},
                    React.createElement('p', null, 'React
                                                    Component'))
```

- React uses **`className`** instead of the traditional DOM **`class`**.
  - **`class`** is a keyword in ES6

Expressions

- The **{ }** brackets in JSX indicate that the content is JavaScript.
- It is eventually parsed by the JavaScript engine.
- **{ }** can be used anywhere among the JSX expressions as long as the result is a valid JavaScript.

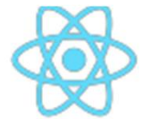
```
const contentNode = 'React Component';

<h2>{ contentNode }</h2>
```

Conditional statements

- `if-else` statements do not work in JSX.
- `if` statements can be used outside JSX to determine the content.
- Ternary operator can be employed in JSX

```
<select name='selCity'>
  <option>Delhi</option>
  <option>Mumbai</option>
  { currCity==='Bangalore' ? null :
                    <option>Bangalore</option>
  }
</select>
```



IIFE

- Immediately invoked function expression can be used in JSX for logic.

```
{(() => {
      if(localStorage.getItem('user'))
        this.userName = localStorage.getItem('user')
      else
        this.userName = 'Guest';
     return <p>{this.userName}</p>
    }
)()}
```

Creating Components

- A React application can be depicted as a component tree - having one root component ("App") and a number of nested child components.
- Components can be created in two possible ways:
  - **Functional components**
    - Also known as *presentational* or *stateless* components
  - **Class-based components**
    - Also known as *containers* or *stateful* components

Creating a Functional Component

- A React component is basically any part of a UI that can contain React nodes.

```
import React from 'react';

const user = () => {
    const userName = '   Rahul    '
    return <p>{userName}</p>
}

export default user
```

Creating a Class based Component

- A class based component is a JavaScript class.
  - Class in ES6 can have constructor, methods etc.

```
class userContainer extends React.Component
```

- It extends the **`React.Component`** and requires a method **`render()`**
- React expects the class to return a react element from the **`render()`**

```
render(){
        return(
                <div>
                        <p>{this.userName}</p>
                </div>
                );
        }
```

Component Props

- Component ***Props*** (short for properties) function like HTML attributes.
- Props provide configuration values for the component.
  - Props is **readonly**

```
<User userName=' Rahul ' />
<User userName='Sam' />
```

App.js

```
const user = (props) => {
    return <p>{props.userName}</p>
}
```

User.js

Props – Children property

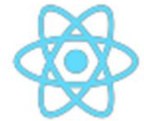
- React provides access to a special prop called children.
- Children is a reserved term referring to the all the content between the component tag.

```
<User userName = 'Lakshman'>Lakshman M N</User>
<User userName = 'Sam' />
```

App.js

User.js

```
<p>User Name : {props.userName}</p>
<p>Name : {props.children}</p>
```



Props Validation

- With the developmental growth of an application a number of bugs can be screened using type checking.
- The `prop-types` library can be used to run type checking on the props for a component.

```
npm install --save prop-types
```

- `prop-types` can be used to document the intended types of properties passed to components.
- React will check props passed to components against those definitions and warn if they do not match.

Props Validation

```
import PropTypes from 'prop-types';
```

- `PropTypes` exports a range of validators that can be used to ensure valid data.

```
course.propTypes = {
    name : PropTypes.string,
    duration : PropTypes.number
    }
```

- If default props are set for the React component, the values are first resolved before type-checking against `propTypes`.
  - Default values are also subject to prop type definitions.
- `propTypes` type-checking only happens in development mode.

Props Validation

- Some of the prominent validators available include:
  - `PropTypes.bool`
  - `PropTypes.number`
  - `PropTypes.string`
  - `PropTypes.func`
  - `PropTypes.array`
  - `PropTypes.object`
  - `PropTypes.string.isRequired` – `isRequired` can be chained to any prop validator

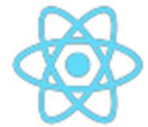
Props Validation

- Multiple Types
  - `PropTypes.oneOf` – the prop is limited to a specific set of values

```
prodCondition : PropType.oneOf(['New', 'Used', 'NA'])
```

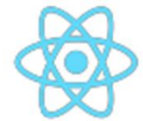
  - `PropTypes.oneOfType` – the prop should be one of a specified set of types

```
courseDuration : PropType.oneOfType([
                    PropType.string,
                    PropType.number
                    ])
```



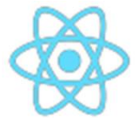
Props Validation

- Custom validators
  - `prop-types` allow custom validation functions for type-checking
  - The validation function accepts three arguments
    - **props** – an object containing all the props passed to the component
    - **propName** – the name of the prop to be validated
    - **componentName** – the name of the component
  - It should return an `Error` object if the validation fails.



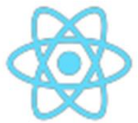
Component **State**

- Most components should take in *props* and render.
- Components also offer ***state*** and this is used to store information about the component that can change over time.
- A ***state*** change implicitly re-renders the component.
- The ***state*** object should only contain minimal amount of data needed for the UI.
- ***state*** is available only in components that extend the **`Component`** class.



Props and State

- **Props** and **State** are core concepts in React.
- Both are plain JavaScript objects.
- Both can have default values.
- Both should be accessed using **`this`** (`this.props` or `this.state`)
  - Both are **readonly** when using `this`
- Changes to **Props** and/or **State** trigger React to re-render components and potentially update the DOM in the browser.



Props and State

- **Props**
  - Props are passed into the component from above (generally parent component)
  - Props are intended as configuration values passed into the component (like arguments passed to a function)
  - Props are immutable to the component receiving them.
- **State**
  - State is a serializable representation of data (JS object) generally associated with the UI.
  - Only class based components can define and use state.
  - State should always start with a default value.
  - State can only be mutated by the component that contains it.
  - *State should be avoided if possible*