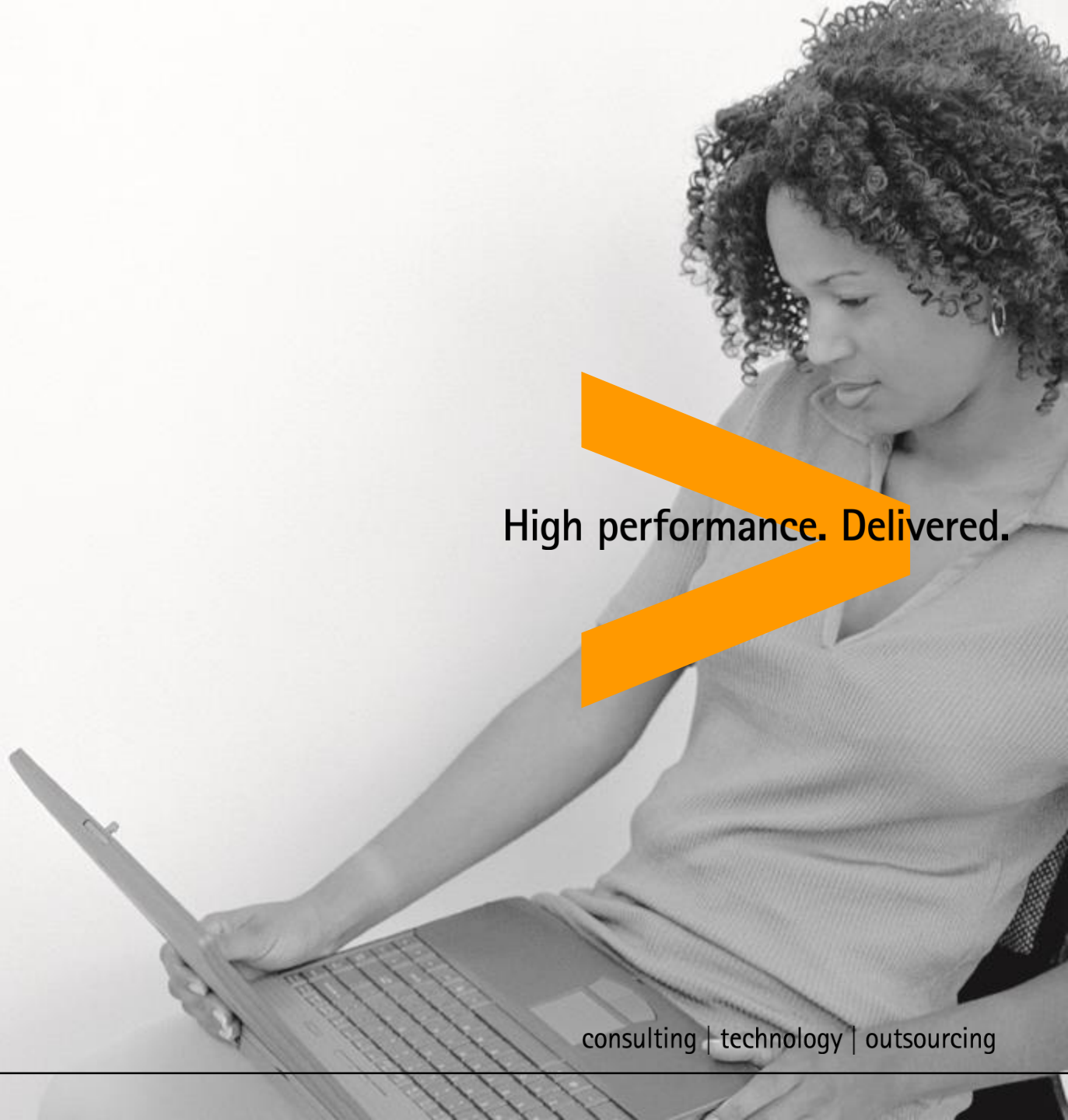
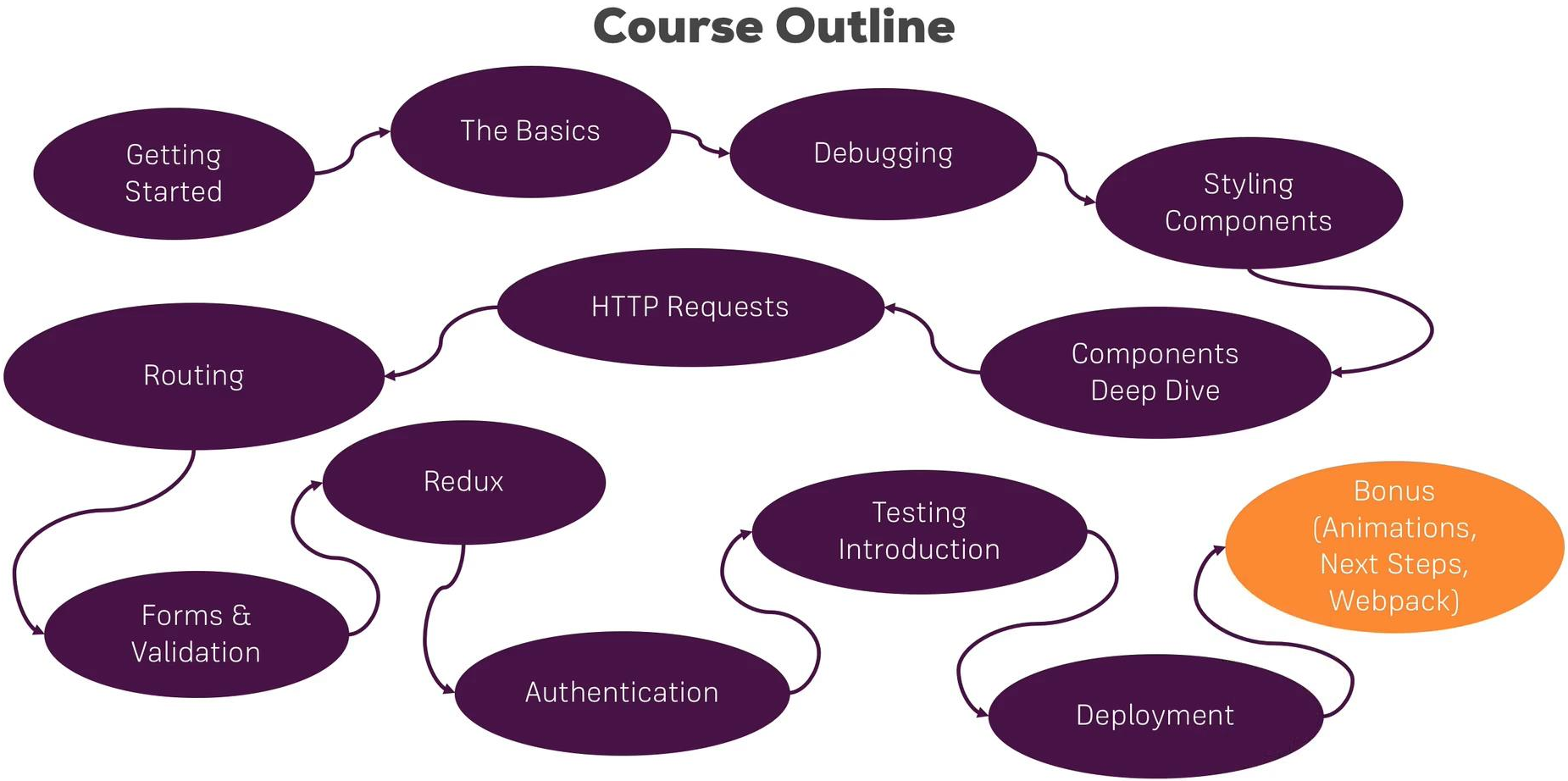
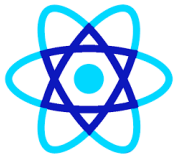
# React js

Parameswari E



# React


Course Outline
Getting Started
The Basics
Debugging
Styling Components
Components Deep Dive
HTTP Requests
Routing
Forms & Validation
Redux
Authentication
Testing Introduction
Deployment
Bonus (Animations, Next Steps, Webpack)

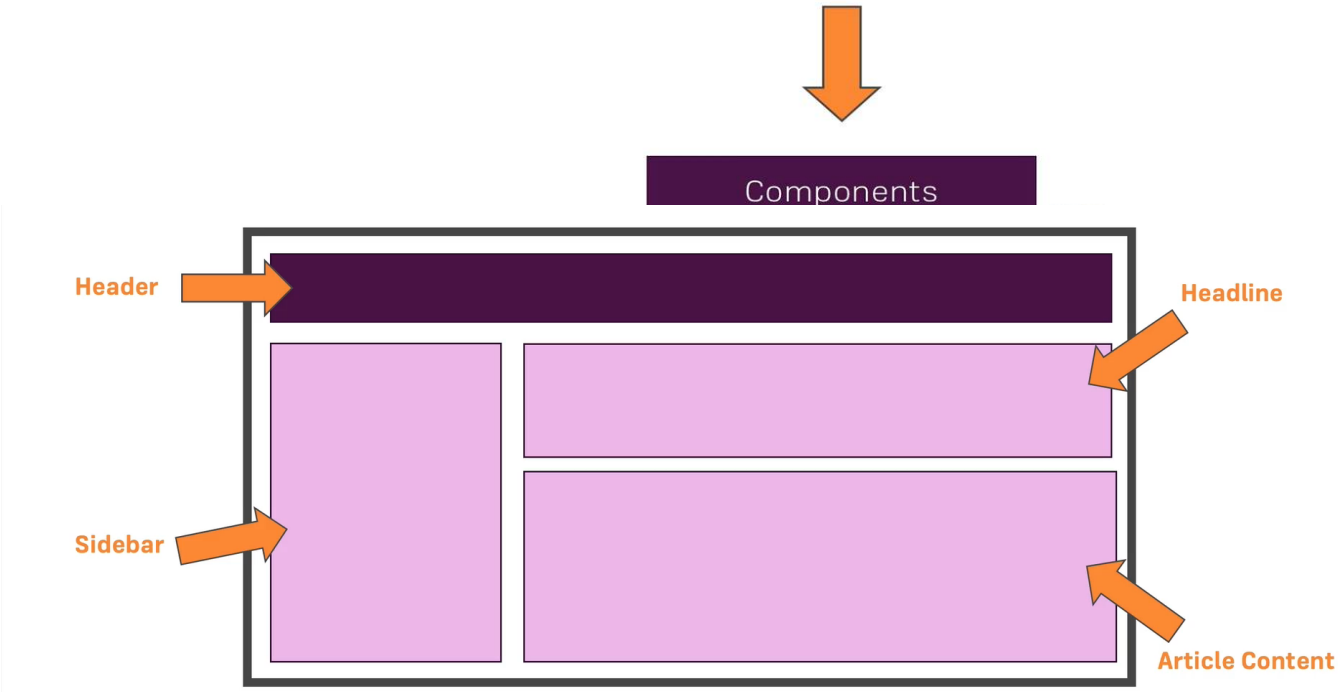


# Getting Started

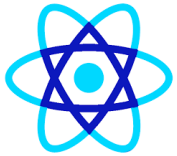
What? Why? How?

# What is React?

"A JavaScript Library for building User Interfaces"

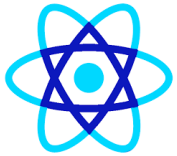


# What Is React?



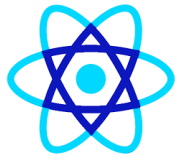
- React is a declarative, efficient, and flexible JavaScript library for building user interfaces.
- It lets you compose complex UIs from small and isolated pieces of code called “components”.
- React is a component-based library which is used to develop interactive UI’s (*User Interfaces*).
- It is currently one of the most popular JavaScript front-end libraries which has a strong foundation and a large community supporting it.
- *NOTE: ReactJS is only a frontend library and not the whole framework, which deals with the **View** component of **MVC** (Model – **View** – Controller).*

# What Is React?



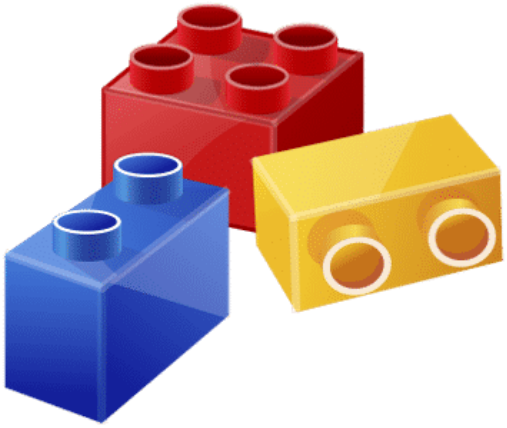
- A ReactJS application is made up of multiple components, each component responsible for outputting a small, reusable piece of HTML code.
- The components are the heart of all React applications.
- These Components can be nested with other components to allow complex applications to be built of simple building blocks.
- ReactJS uses virtual DOM based mechanism to fill data in HTML DOM.
- The virtual DOM works fast as it only changes individual DOM elements instead of reloading complete DOM every time.

# What Is React?

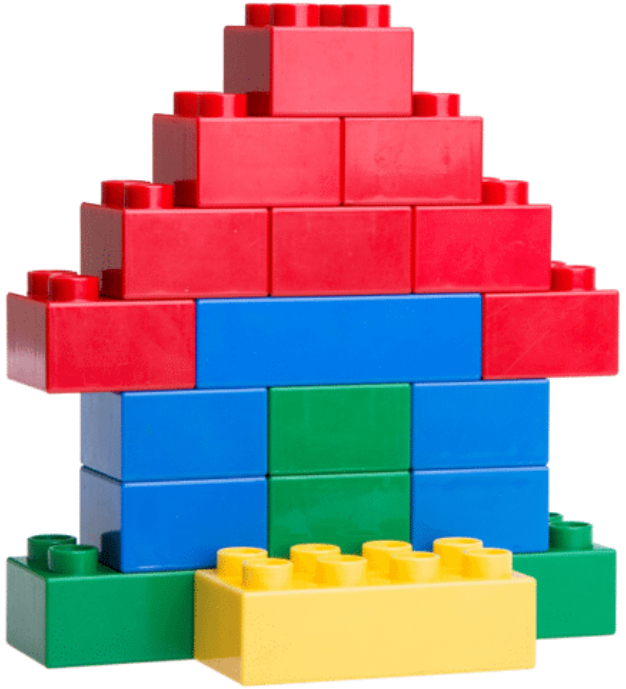


- To create React app, we write React components that correspond to various elements.
- We organize these components inside higher level components which define the application structure.
- For example, we take a form that consists of many elements like input fields, labels, or buttons.
- We can write each element of the form as React components, and then we combine it into a higher-level component, i.e., the form component itself.
- The form components would specify the structure of the form along with elements inside of it.

# What Is React?

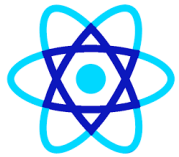


Building Blocks



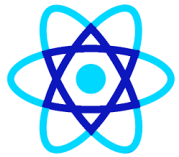
Final Product

# What is React



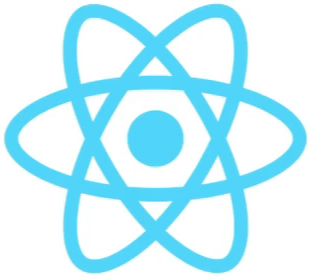
- ReactJS was developed by *Jordan Walke*, a software engineer working at Facebook.
- Facebook implemented ReactJS in 2011 in its *newsfeed* section, but it was released to the public in May 2013.
- After the implementation of ReactJS, Facebook's UI underwent drastic improvement.
- This resulted in satisfied users and a sudden boost in its popularity.

# Java script Frameworks



# React Alternatives

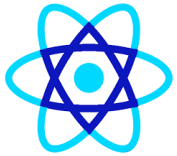
Angular



React

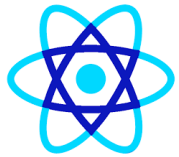
Vue

# Advantages of JavaScript Frameworks

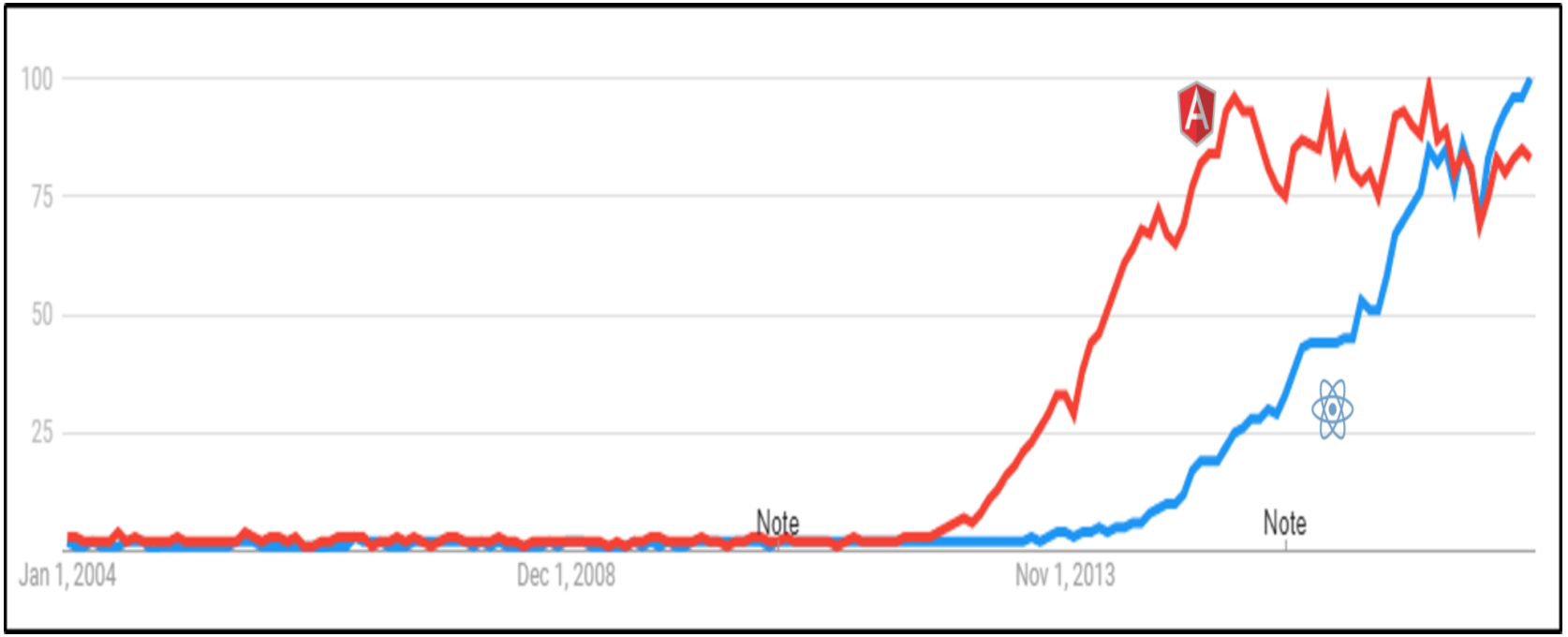


- **Efficiency:** With the use of pre-built patterns and functions, the development of the applications became easy. The projects which used to take months to develop can now be developed in very less time. This increased the efficiency as well as reduced the time and effort involved.
- **Security:** As JavaScript is an open source community, its top frameworks have strong security arrangements. Frameworks are supported by these large communities in which, the members and the users can also act as testers. This increases the chances of detecting any kind of backdoor or bug present in the framework. Thus providing better security at less cost.
- **Cost Reduction:** JavaScript Frameworks are free for public use as they are open source. So, when we develop a web application using these frameworks, the overall cost of the application is much lesser.

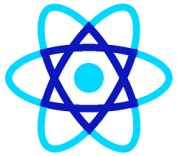
# JavaScript Frameworks



- ***Even though React is young, it is giving a neck to neck competition to Angular***

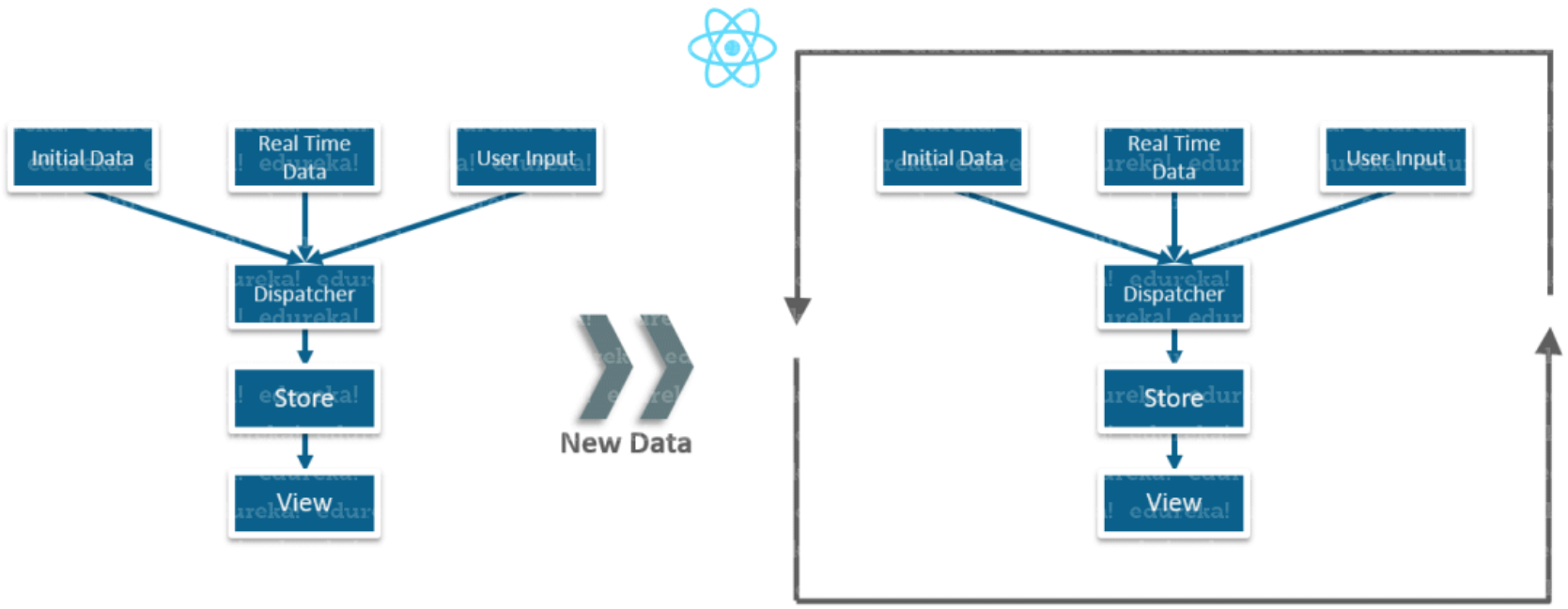


# React vs Angular

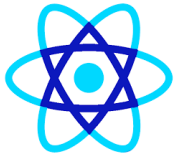


| TOPIC | REACT | ANGULAR |
|---|---|---|
| *1. ARCHITECTURE* | Only the View of MVC | Complete MVC |
| *2. RENDERING* | Server-side rendering | Client-side rendering |
| *3. DOM* | Uses virtual DOM | Uses real DOM |
| *4. DATA BINDING* | One-way data binding | Two-way data binding |
| *5. DEBUGGING* | Compile time debugging | Runtime debugging |
| *6. AUTHOR* | Facebook | Google |

# Why React?

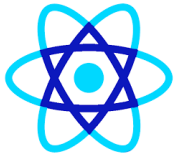

edureka!
Initial Data
Real Time Data
User Input
Dispatcher
Store
View
New Data
Initial Data
Real Time Data
User Input
Dispatcher
Store
View

# Why React



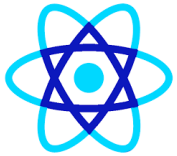
- The previous frameworks follow the traditional data flow structure, which uses the DOM (Document Object Model).
- DOM is an object which is created by the browser each time a web page is loaded.
- It dynamically adds or removes the data at the back end and when any modifications were done, then each time a new DOM is created for the same page.
- This repeated creation of DOM makes unnecessary memory wastage and reduces the performance of the application.
- A new technology ReactJS framework invented which remove this drawback.
- ReactJS allows you to divide your entire application into various components.

# Why React

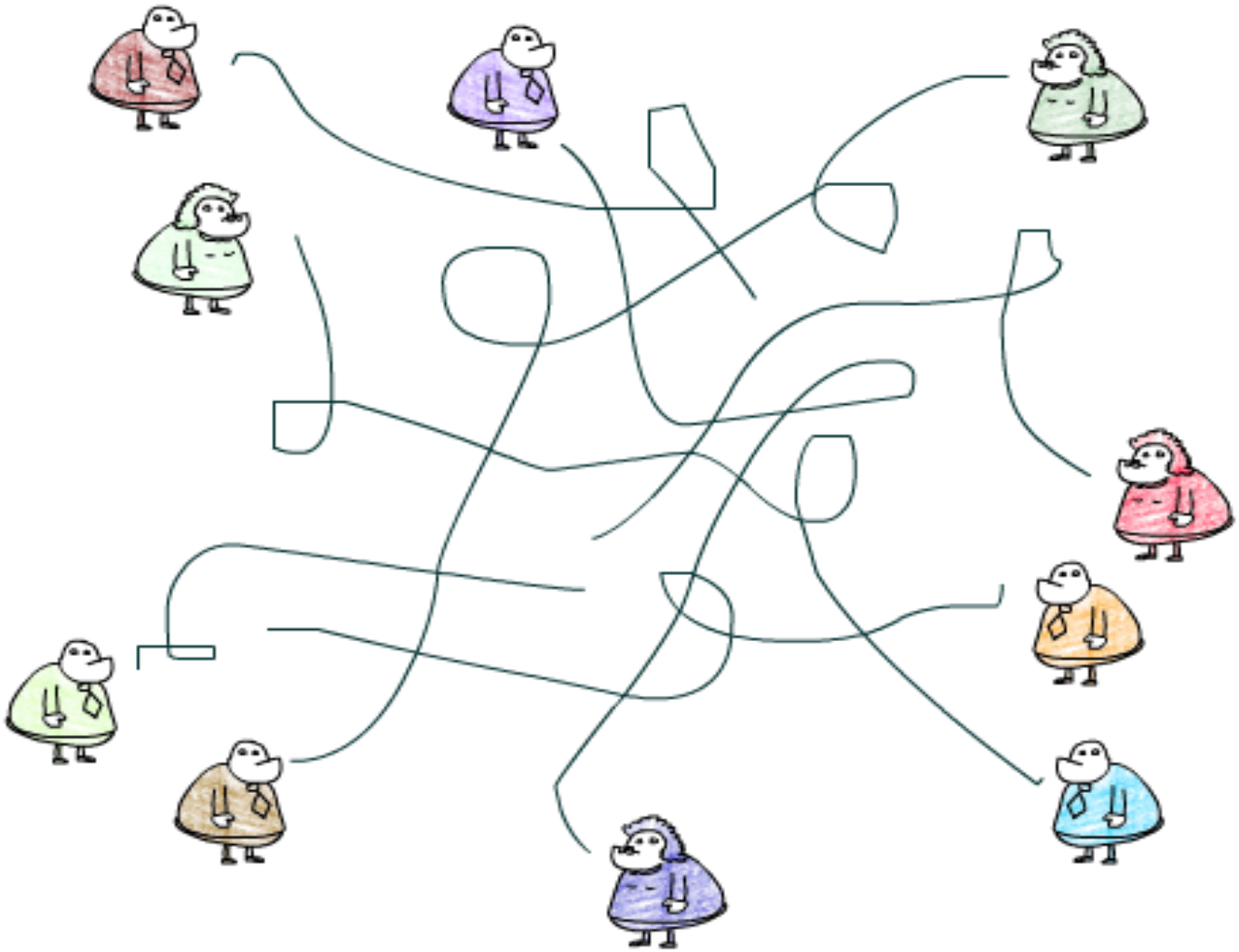


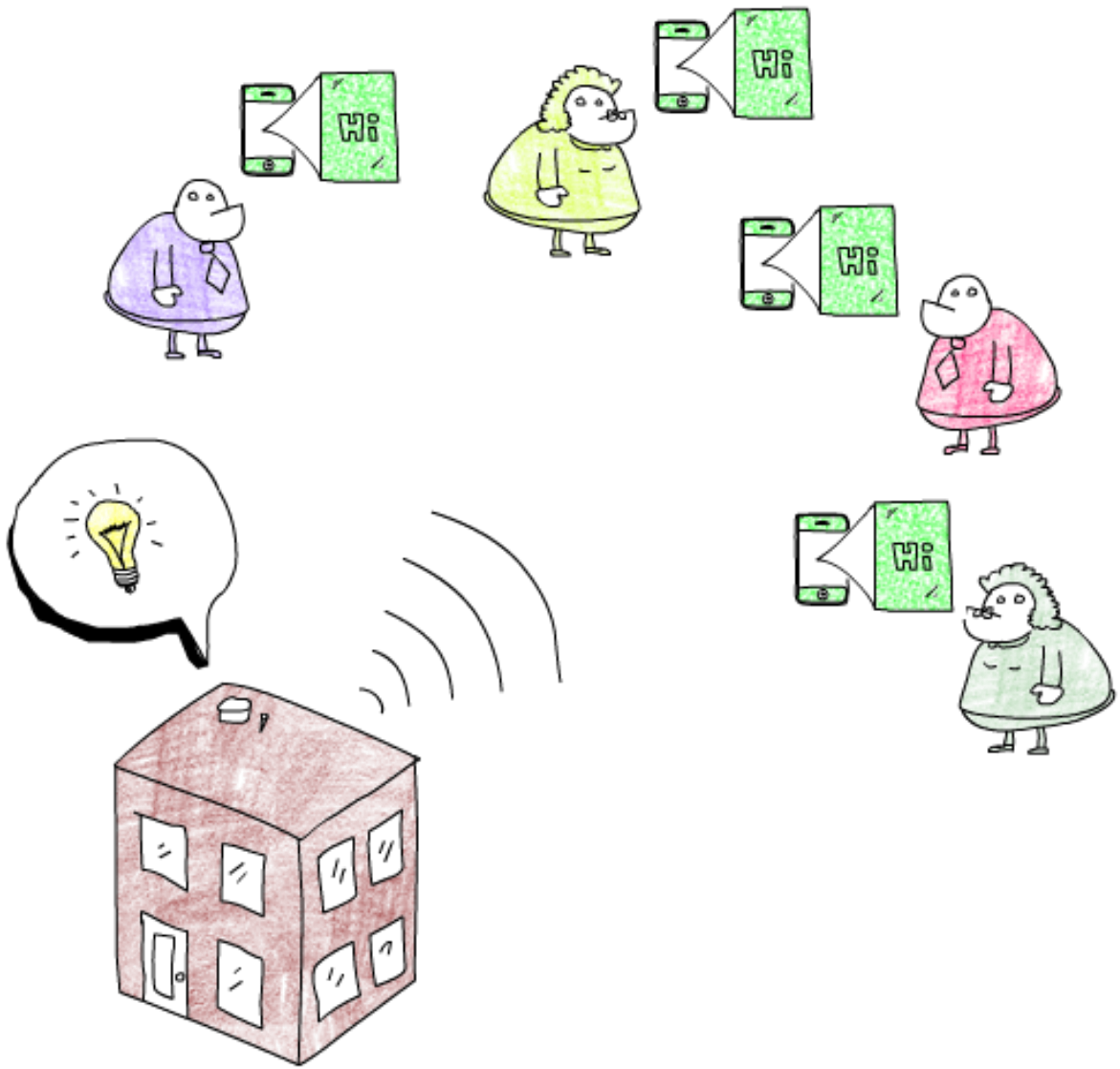
- ReactJS still used the same traditional data flow, but it is not directly operating on the browser's Document Object Model (DOM) immediately; instead, it operates on a virtual DOM.
- It means rather than manipulating the document in a browser after changes to our data, it resolves changes on a DOM built and run entirely in memory.
- After the virtual DOM has been updated, React determines what changes made to the actual browser's DOM.
- The React Virtual DOM exists entirely in memory and is a representation of the web browser's DOM.
- Due to this, when we write a React component, we did not write directly to the DOM; instead, we are writing virtual components that react will turn into the DOM.

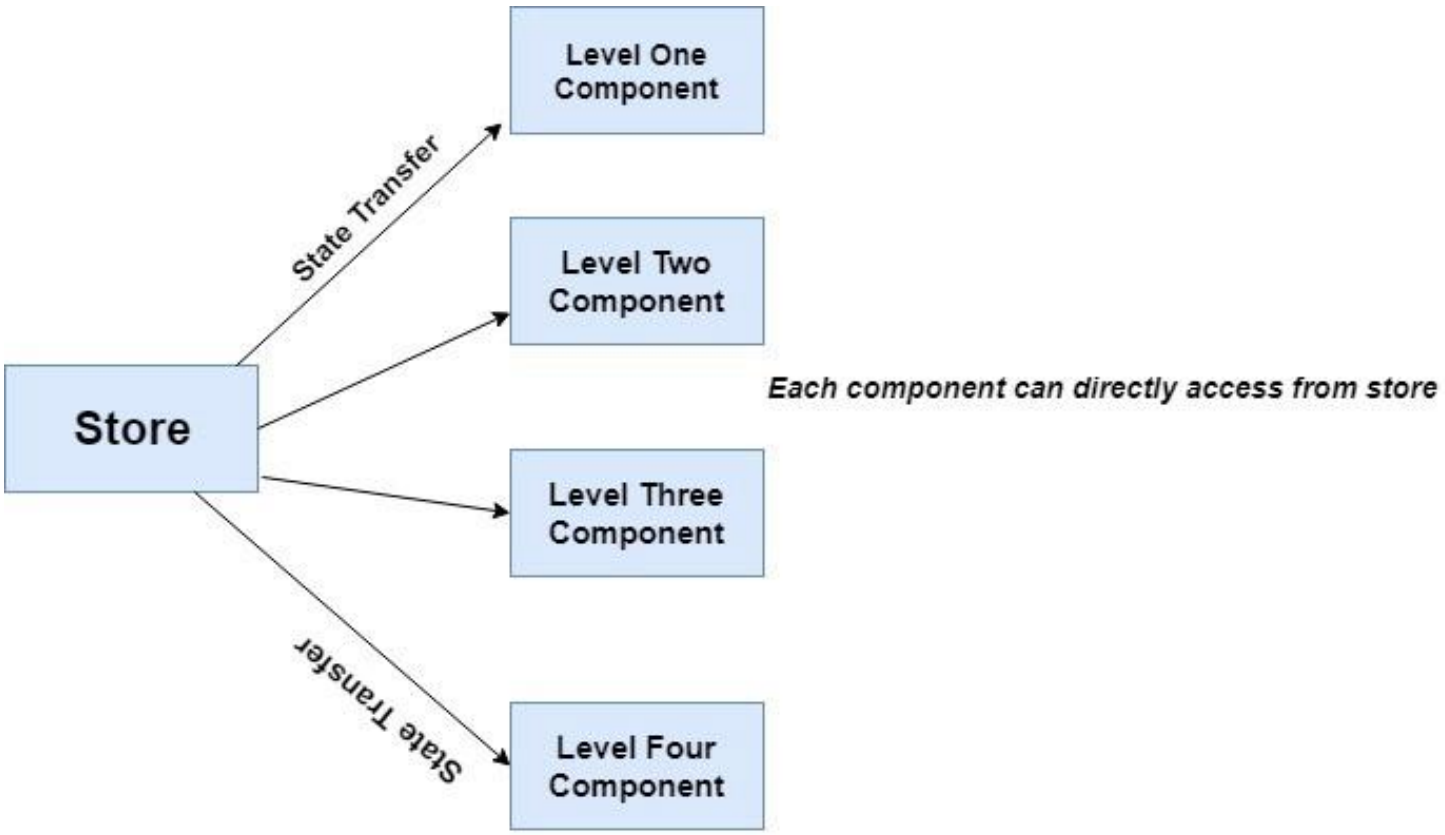
# Why React



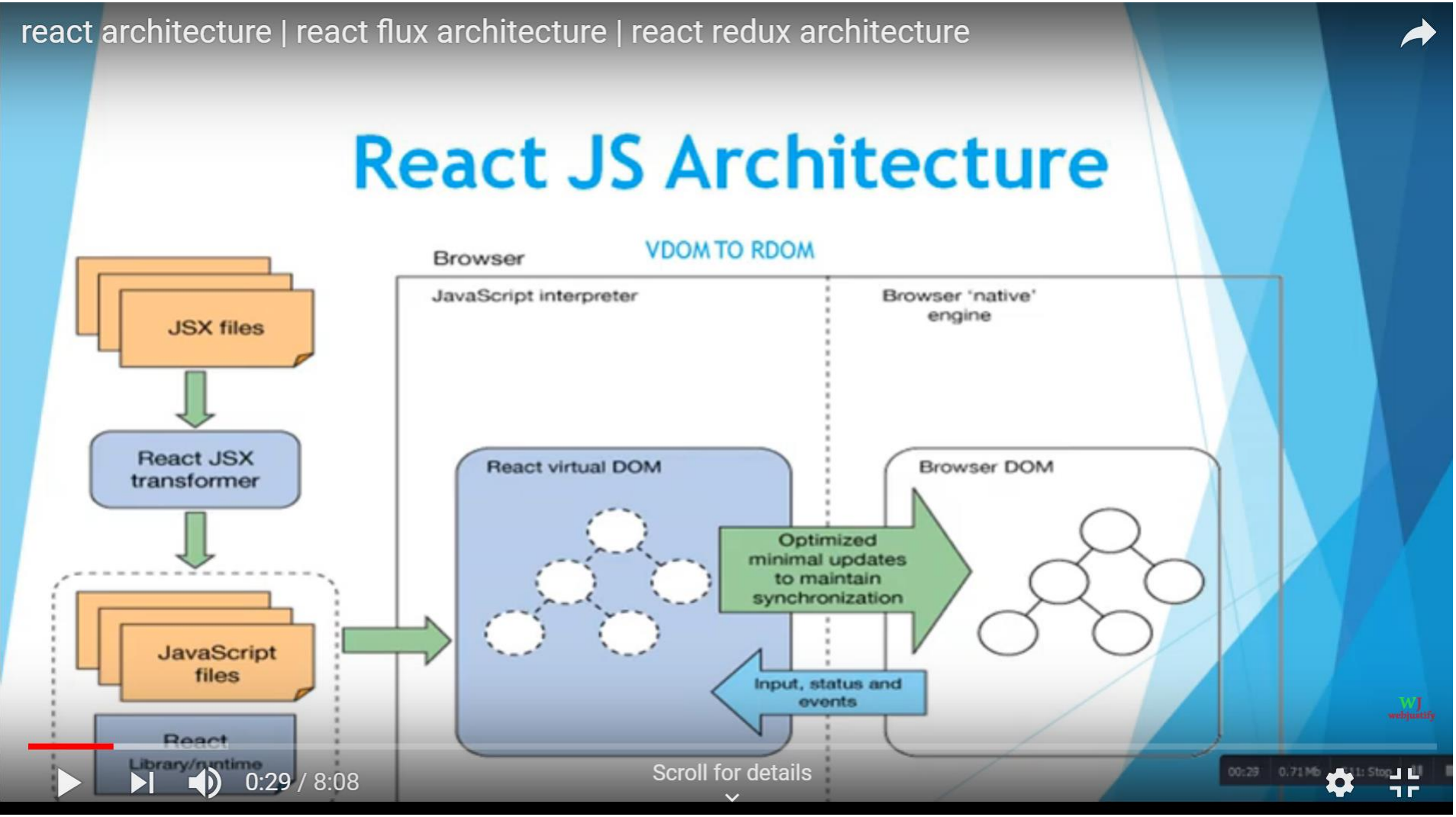
- The biggest advantage of using components is that you can change any component at any point in time without affecting the rest of the applications.
- This feature is most effective when implemented with larger and real-time applications where data changes frequently.
- Each time any data is added or updated, ReactJS automatically updates the specific component whose state has *actually* changed.
- This saves the browser from the task of reloading the whole application to reflect the changes.

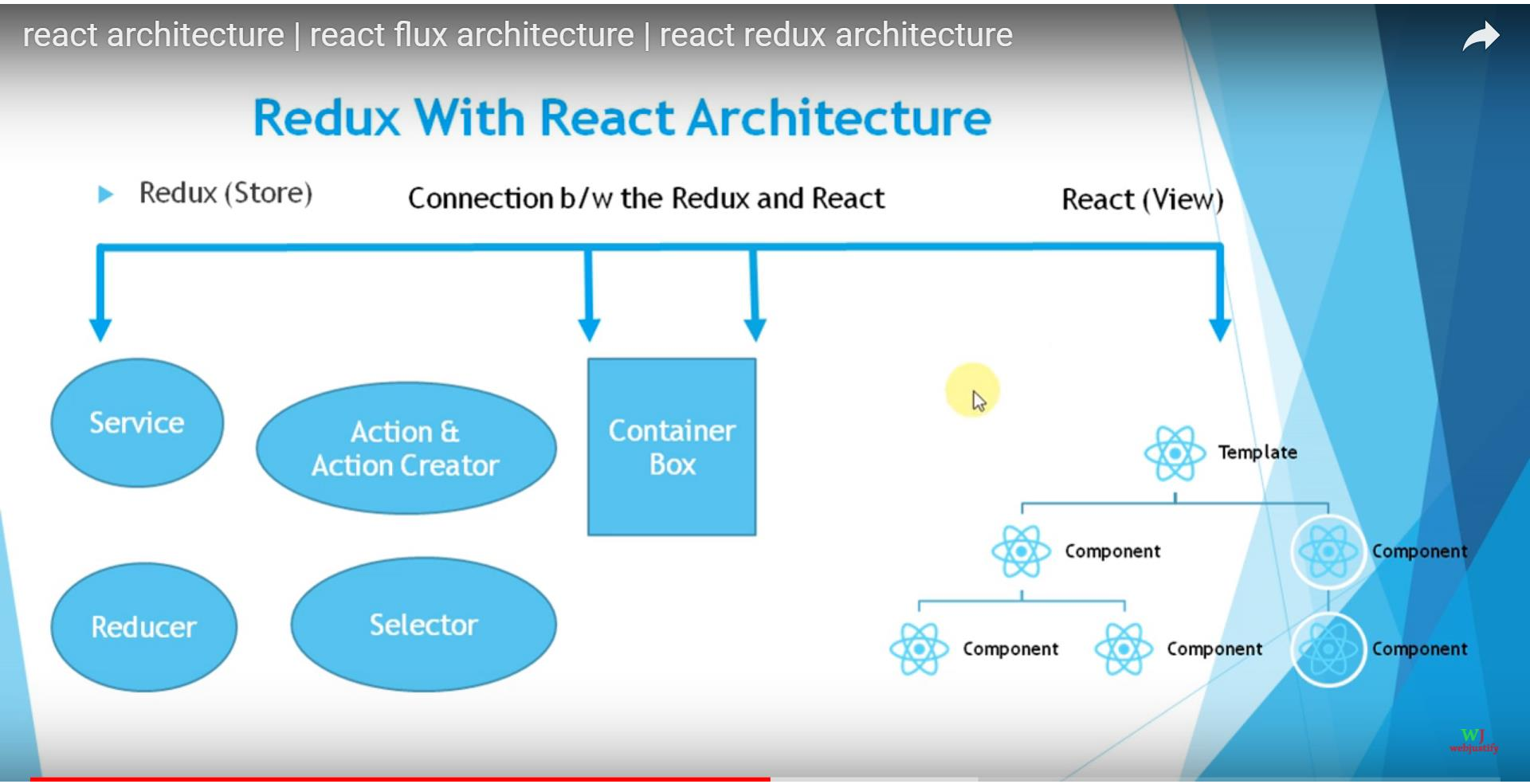


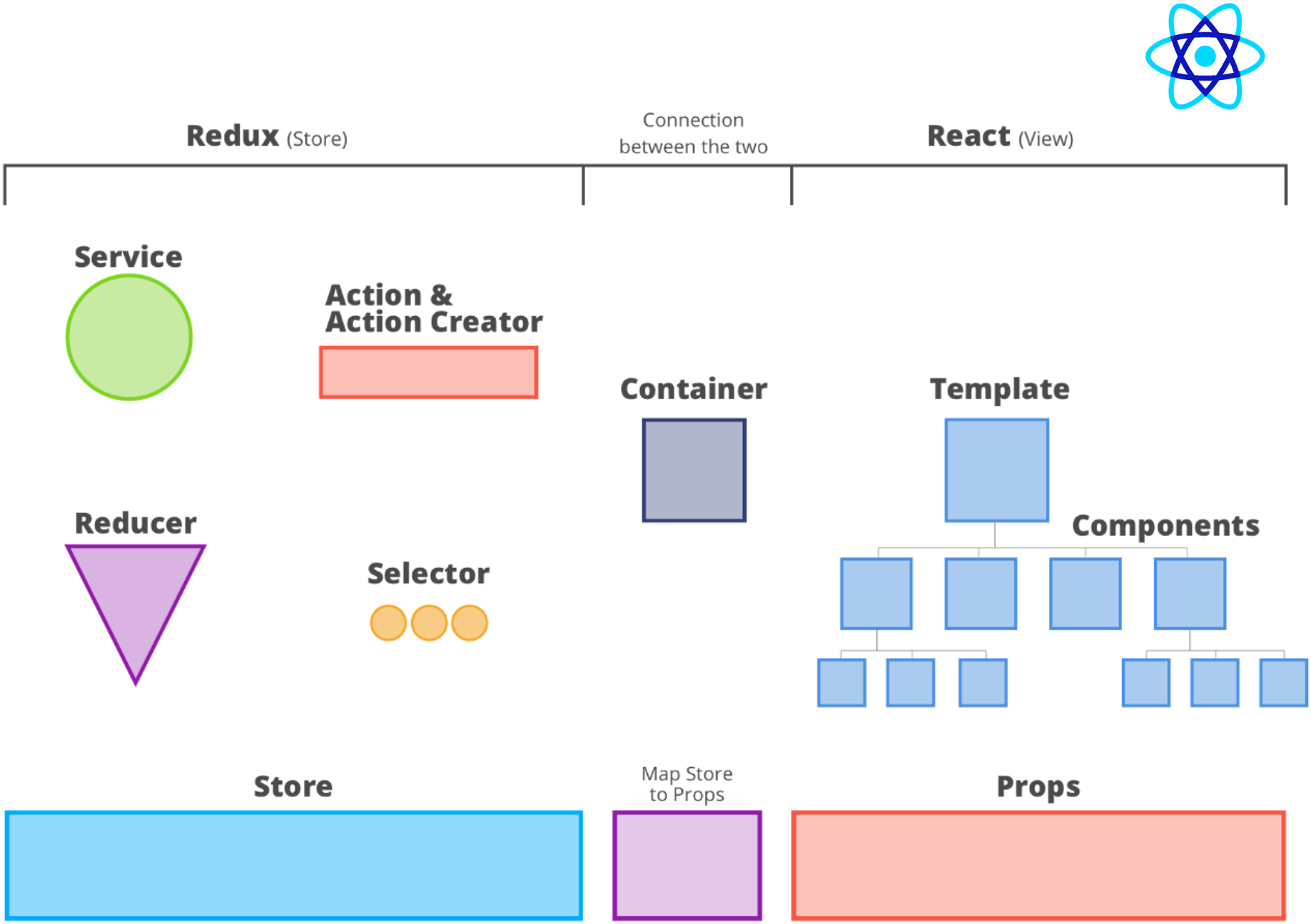

Hi
Hi
Hi
Hi

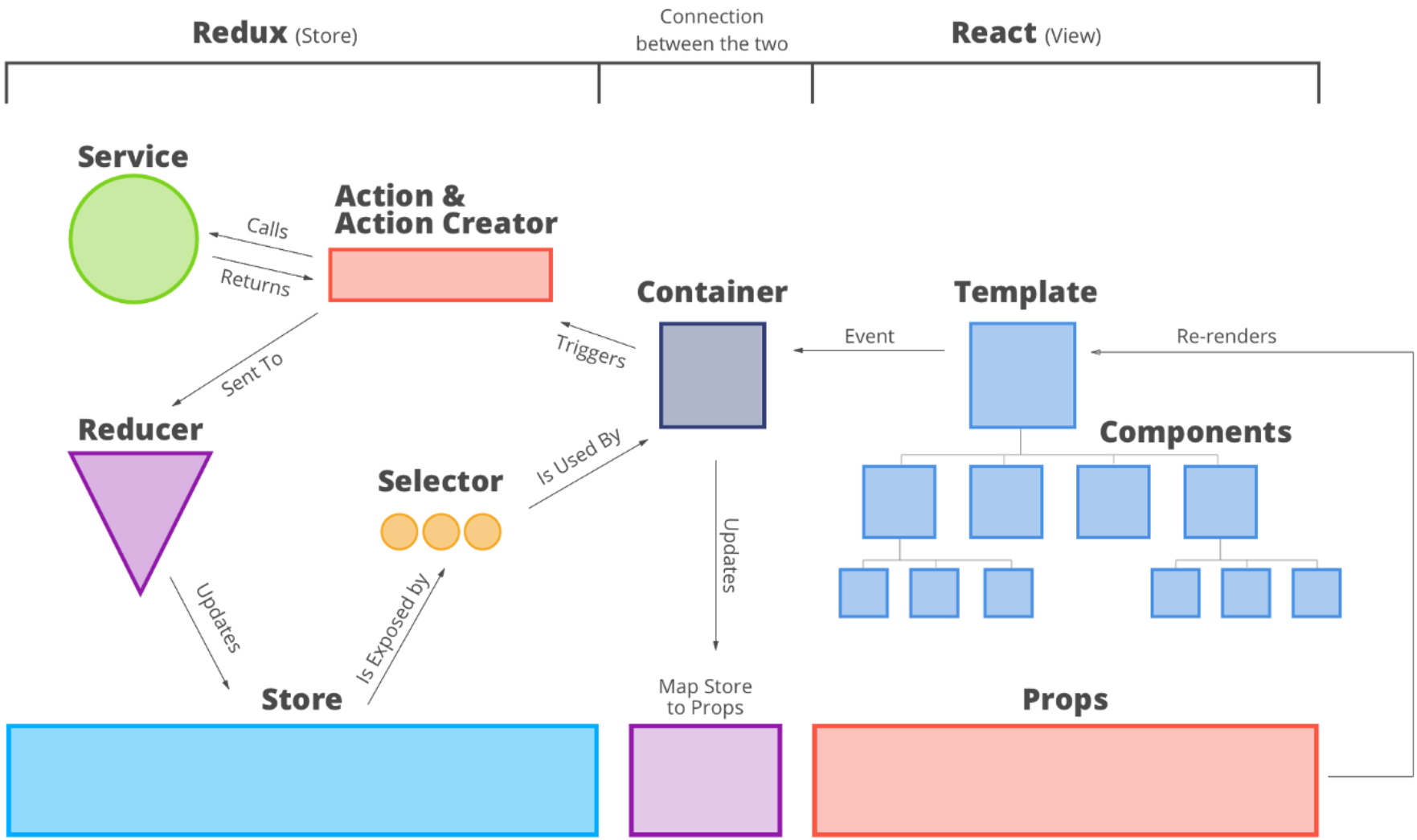


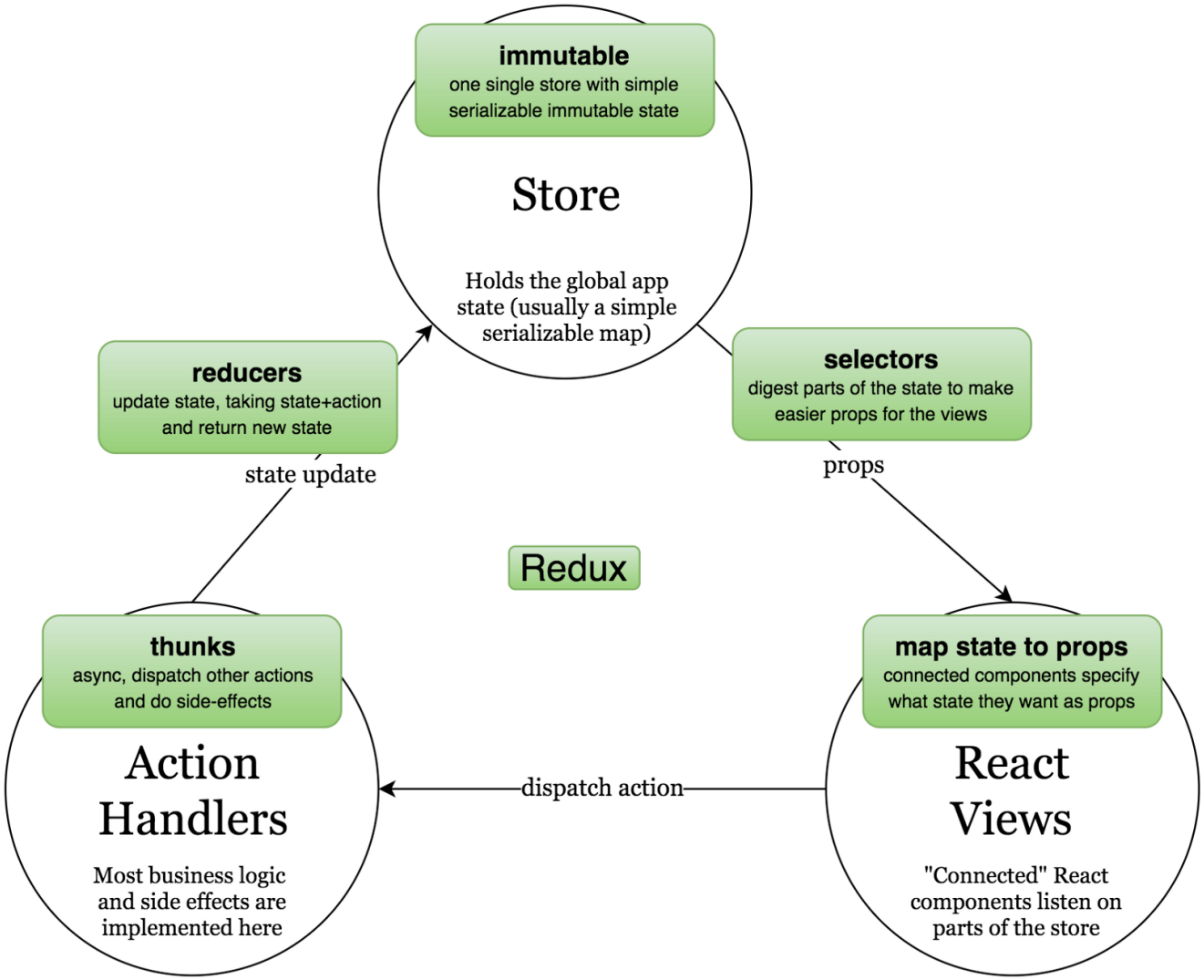

Redux State Transfer

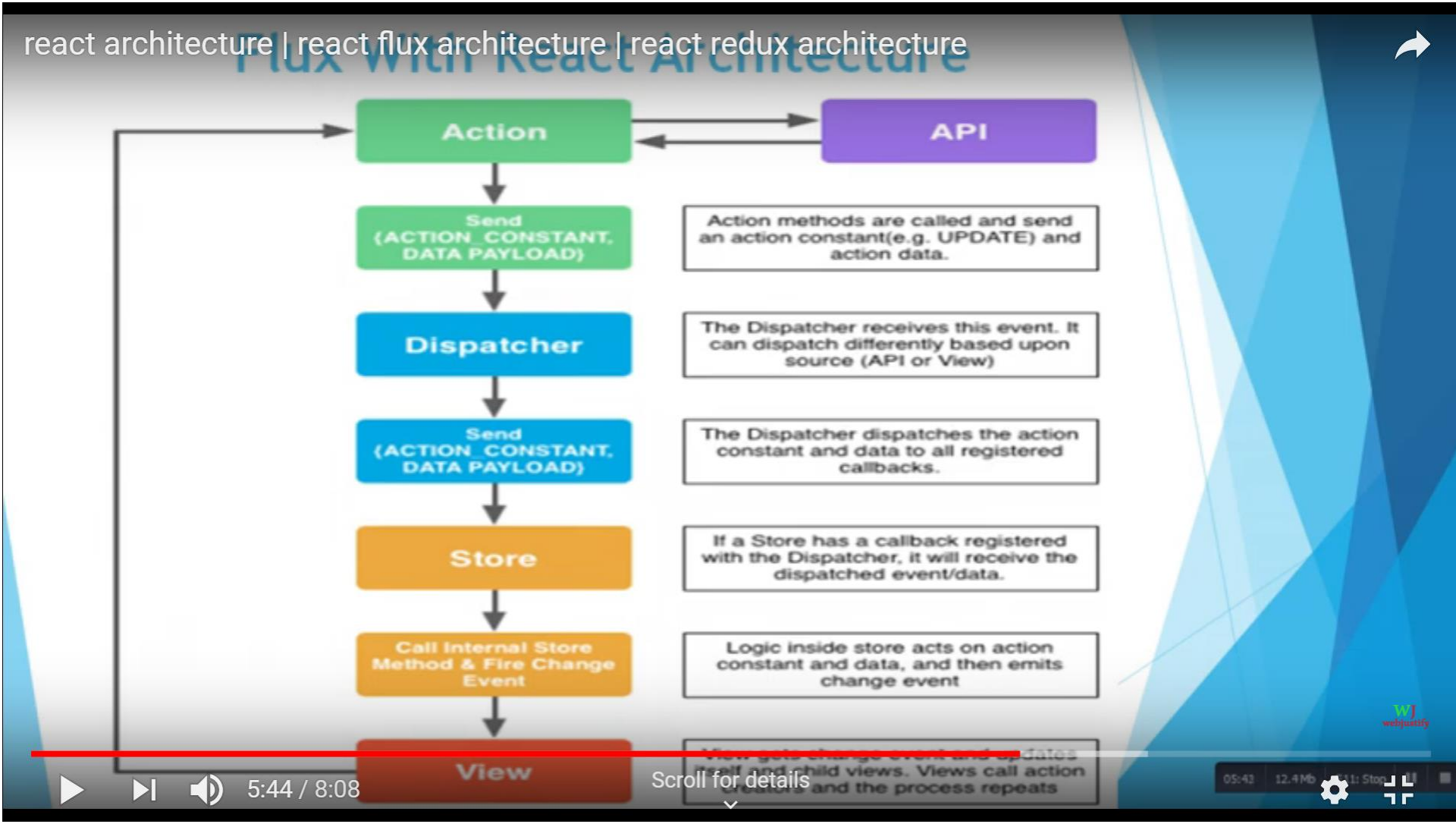

react architecture | react flux architecture | react redux architecture
React JS Architecture
VDOM TO RDOM
Browser
JavaScript interpreter
Browser 'native' engine
JSX files
React JSX transformer
JavaScript files
React
Library/runtime
React virtual DOM
Browser DOM
Optimized minimal updates to maintain synchronization
Input, status and events
0:29 / 8:08
Scroll for details

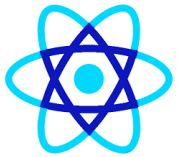

react architecture | react flux architecture | react redux architecture
Redux With React Architecture
Redux (Store)
Connection b/w the Redux and React
React (View)
Service
Action & Action Creator
Container Box
Reducer
Selector
Template
Component
Component
Component
Component
Component


Redux (Store)
Connection between the two
React (View)
Service
Action & Action Creator
Container
Template
Components
Reducer
Selector
Store
Map Store to Props
Props


Redux (Store)
Connection between the two
React (View)
Service
Action & Action Creator
Calls
Returns
Sent To
Reducer
Container
Triggers
Event
Template
Re-renders
Components
Selector
Is Used By
Updates
Updates
Is Exposed by
Store
Map Store to Props
Props


immutable
one single store with simple serializable immutable state
Store
Holds the global app state (usually a simple serializable map)
reducers
update state, taking state+action and return new state
selectors
digest parts of the state to make easier props for the views
state update
props
Redux
thunks
async, dispatch other actions and do side-effects
map state to props
connected components specify what state they want as props
Action Handlers
Most business logic and side effects are implemented here
dispatch action
React Views
"Connected" React components listen on parts of the store


react architecture | react flux architecture | react redux architecture
Flux With React Architecture
Action
API
Send (ACTION_CONSTANT, DATA PAYLOAD)
Action methods are called and send an action constant(e.g. UPDATE) and action data.
Dispatcher
The Dispatcher receives this event. It can dispatch differently based upon source (API or View)
Send (ACTION_CONSTANT, DATA PAYLOAD)
The Dispatcher dispatches the action constant and data to all registered callbacks.
Store
If a Store has a callback registered with the Dispatcher, it will receive the dispatched event/data.
Call Internal Store Method & Fire Change Event
Logic inside store acts on action constant and data, and then emits change event
View
5:44 / 8:08
Scroll for details



# Why React?

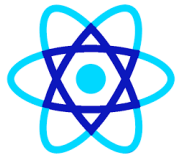
UI State becomes difficult to handle with Vanilla JavaScript

Focus on Business Logic, not on preventing your App from exploding

Plus

Framework Creators probably write better Code

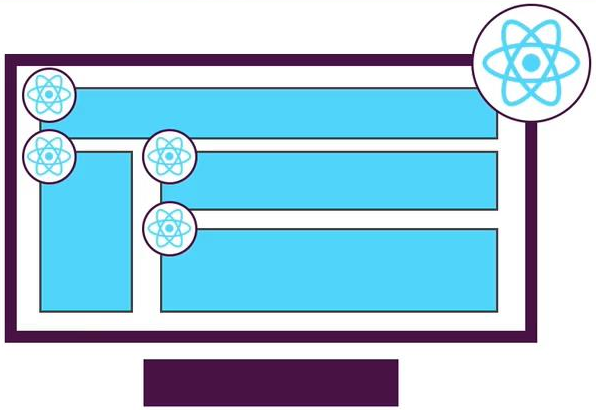
Huge Ecosystem, Active Community, High Performance



# Two Kinds of Applications

**Single Page Applications**

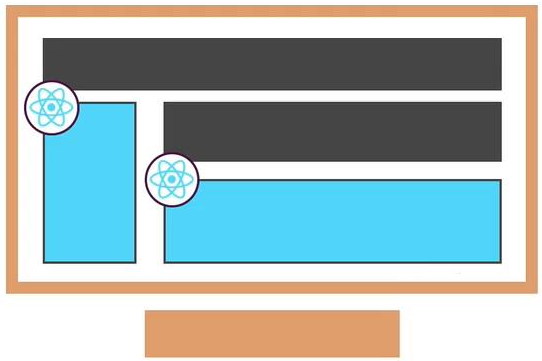
Only ONE HTML Page, Content is (re)rendered on Client



Typically only ONE ReactDOM.render() call

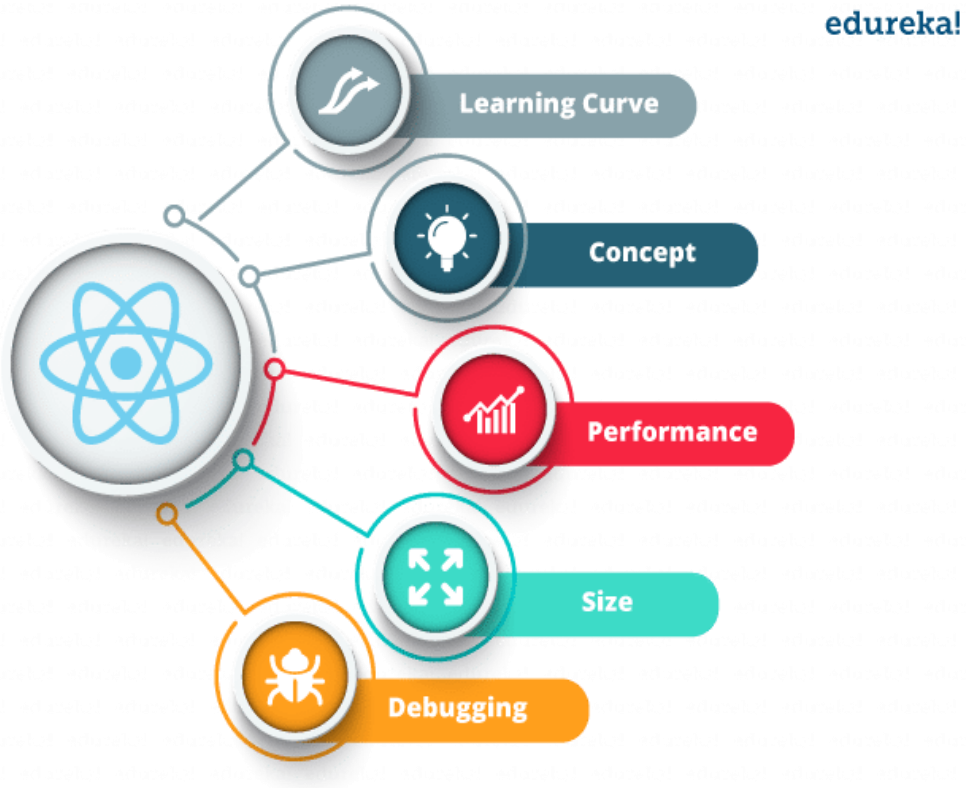
**Multi Page Applications**

Multiple HTML Pages, Content is rendered on Server

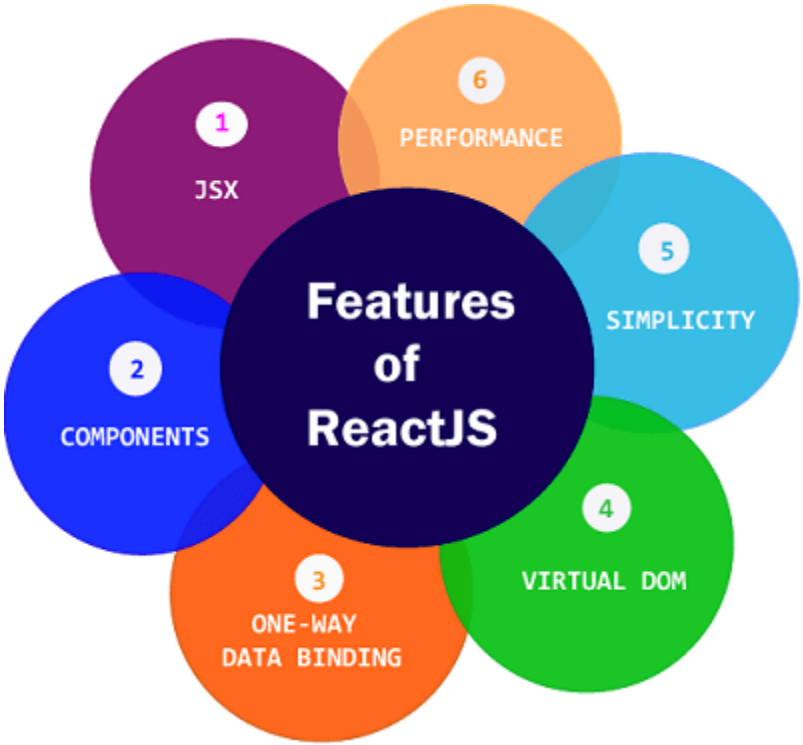


One ReactDOM.render() call per “widget”

# Features of React

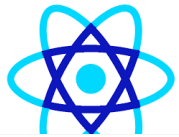


# Features of React



# Features Of React

# React Versions



# Using a Build Workflow

WHY?

HOW?

Recommended for SPAs and MPAs

**Optimize** Code

Use **Next-Gen** JavaScript Features

ES6 vs ES5

Be **More Productive**

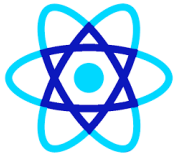
Use **Dependency Management** Tool
**npm** or **yarn**

Use **Bundler**
Recommended: **Webpack**

Use **Compiler** (Next-Gen JavaScript)
**Babel** + Presets

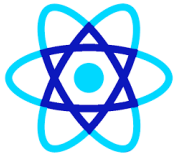
Use a **Development Server**

# React Environment Setup



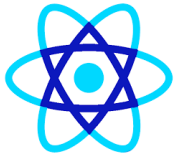
- Pre-requisite for ReactJS
  - NodeJS and NPM
  - React and React DOM
  - Webpack
  - Babel
- Ways to install ReactJS
  - There are two ways to set up an environment for successful ReactJS application. They are given below.
  - Using the npm command
  - Using the create-react-app command

# Create react app



- Starting a new React project is very complicated, with so many build tools.
- It uses many dependencies, configuration files, and other requirements such as Babel, Webpack, ESLint before writing a single line of React code.
- Create React App CLI tool removes all that complexities and makes React app simple.
- For this, you need to install the package using NPM, and then run a few simple commands to get a new React project.
- The create-react-app is an excellent tool for beginners, which allows you to create and run React project very quickly.
- It does not take any configuration manually. This tool is wrapping all of the required dependencies like Webpack, Babel for React project itself and then you need to focus on writing React code only.
- This tool sets up the development environment, provides an excellent developer experience, and optimizes the app for production.

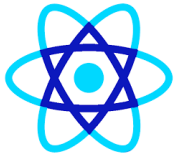
# Requirements



- The Create React App is maintained by **Facebook** and can works on any **platform**, for example, macOS, Windows, Linux, etc.
- To create a React Project using create-react-app, you need to have installed the following things in your system.
- Node version >= 8.10
- NPM version >= 5.6
- Let us check the current version of Node and NPM in the system.
- Run the following command to check the Node version in the command prompt.
- $ node -v
- npm -v

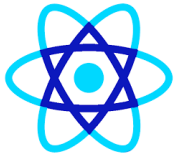

node
Interactive
NORTH AMERICA
Vancouver, Canada
October 4-6, 2017
Join us for Node.js Interactive
happening in Vancouver, Canada
October 4 - 6, 2017
Node.js® is a JavaScript runtime built on Chrome's V8 JavaScript engine. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient. Node.js' package ecosystem, npm, is the largest ecosystem of open source libraries in the world.
Important security releases, please update now!
Download for macOS (x64)
v6.11.3 LTS
Recommended For Most Users
v8.5.0 Current
Latest Features
Download v8.5.0 Current

# .npmrc settings



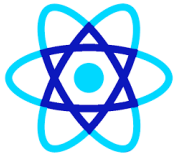
- npm config set strict-ssl=false
- npm config set proxy http://username:password@proxyname:port
- npm config set https-proxy http://username:password@proxyname:port
- npm config set registry http://registry.npmjs.org/
- Or
- Npm config edit
- proxy = http://username:password@proxyname:port
- https-proxy = http://username:password@proxyname:port
- Registry= http://registry.npmjs.org/

# Yarn vs NPM



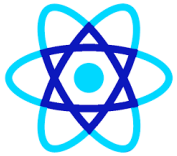
- There were two major shortcomings to npm.
- First of all, npm didn't use a lockfile.
- A lockfile contains all the information about the exact version of each dependency.
- Considering packages add new versions all the time, there's a big risk your code can break if it's not compatible with the latest versions of certain dependencies.
- That's why it's important to lock dependencies to a single version. This couldn't be done with npm 5 or below.
- Yarn solved this problem by generating a yarn.lock file that stores exactly which version of which dependency was installed.

# Yarn vs NPM



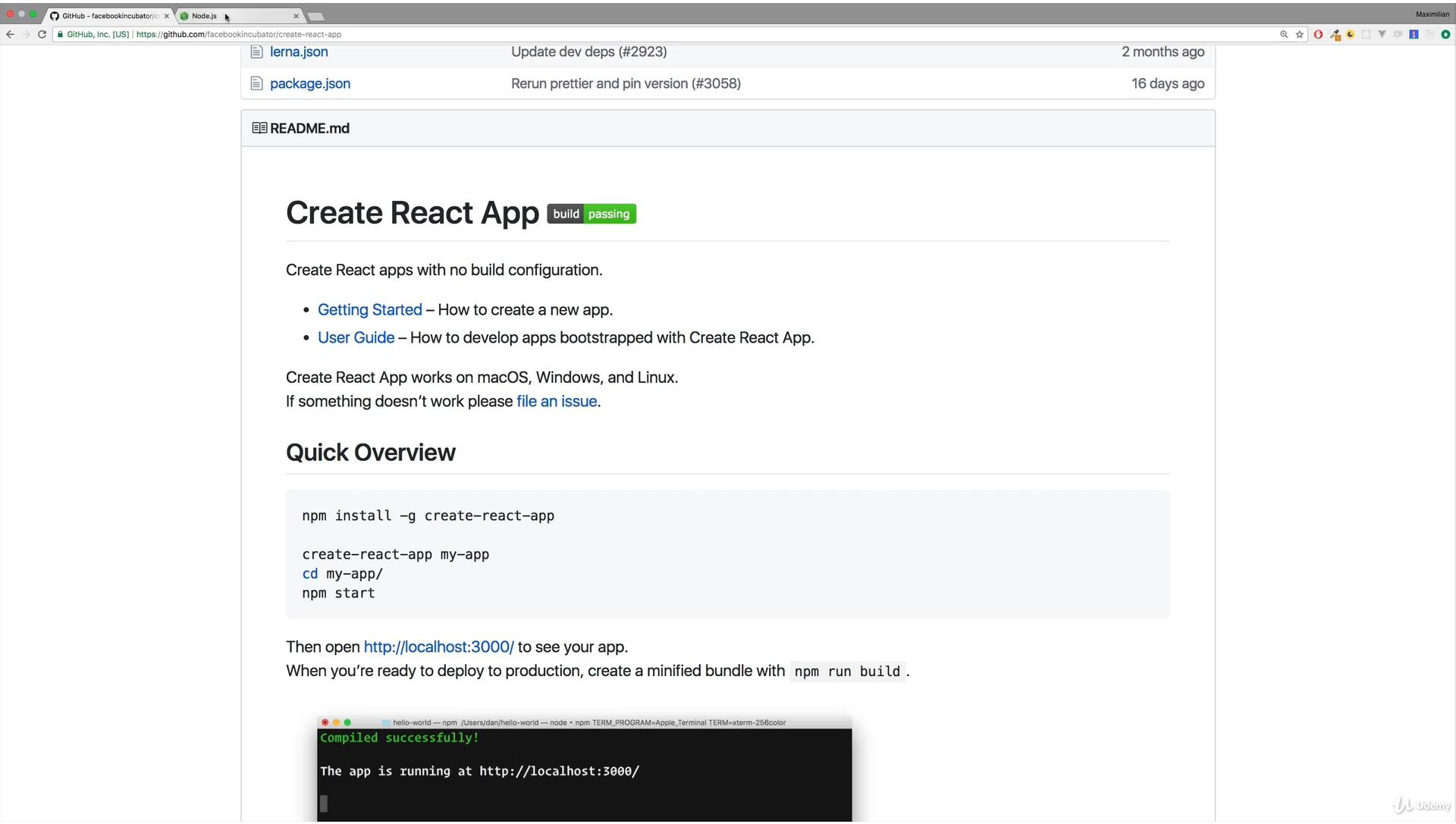
- The second major shortcoming of npm was that it was non-deterministic.
- Your node_modules folder is likely to differ from the node_modules folder of your colleague, or even of the different testing and production servers.
- Yarn is a deterministic package manager, which means that all computers with a given package.json file will have the exact same dependencies installed in their node_modules folder.
- This helps avoid scenarios where code would work on your computer, but not on a different computer.

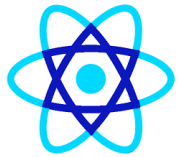
# Yarn



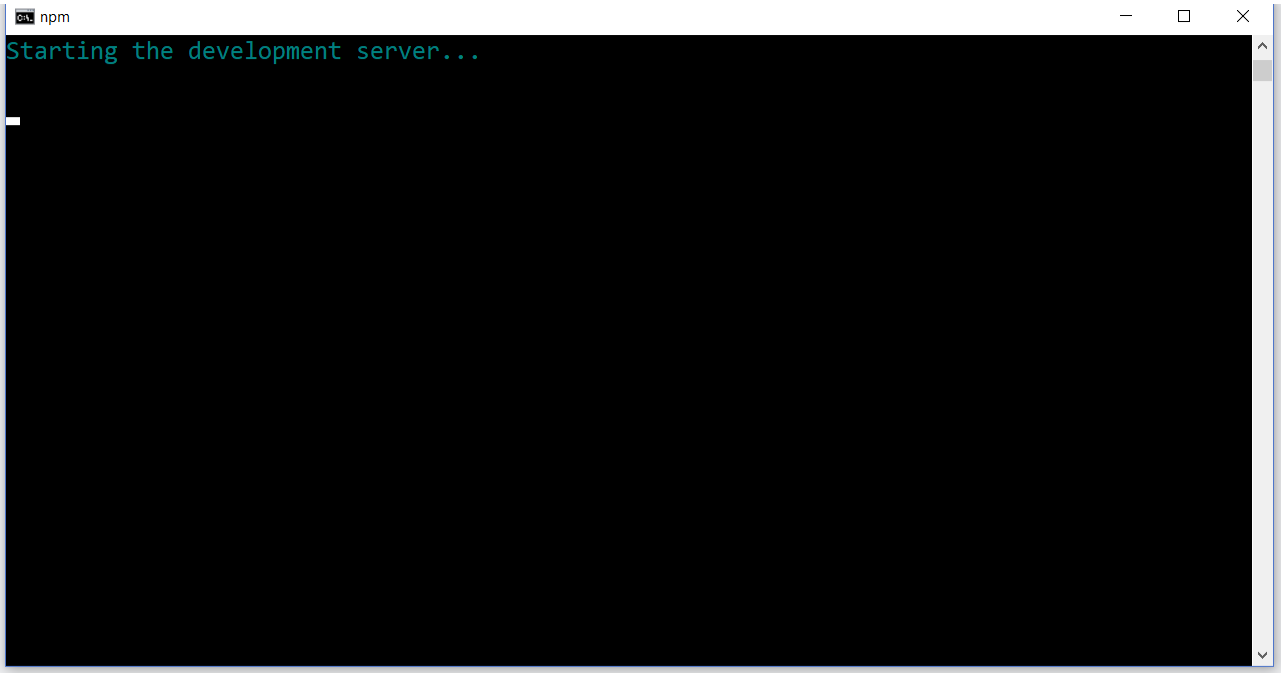
- Installing Yarn
- There are several ways of installing Yarn. If you have npm installed, you can just install Yarn with npm:
- npm install yarn --global
- Yarn create react-app activiti-app
- Yarn start

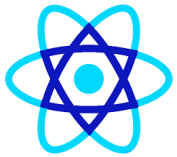
# Create React Project



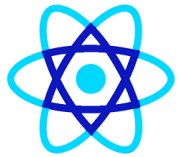


```
Maximilians-MBP:udemy maximilianschwarzmuller$
Maximilians-MBP:udemy maximilianschwarzmuller$
Maximilians-MBP:udemy maximilianschwarzmuller$ sudo npm install create-react-app -g
Password:
/usr/local/bin/create-react-app -> /usr/local/lib/node_modules/create-react-app/index.js
+ create-react-app@1.4.0
added 2 packages, removed 4 packages and updated 4 packages in 4.344s
Maximilians-MBP:udemy maximilianschwarzmuller$ create-react-app react-complete-guide

Creating a new React app in /Users/maximilianschwarzmuller/development/reactjs/tutorials/udemy/react-compl
ete-guide.

Installing packages. This might take a couple of minutes.
Installing react, react-dom, and react-scripts...

yarn add v0.27.5
info No lockfile found.
[1/4] Resolving packages...
[2/4] Fetching packages...
[3/4] Linking dependencies...
                                                                                15726/20175
```


npm
Starting the development server...

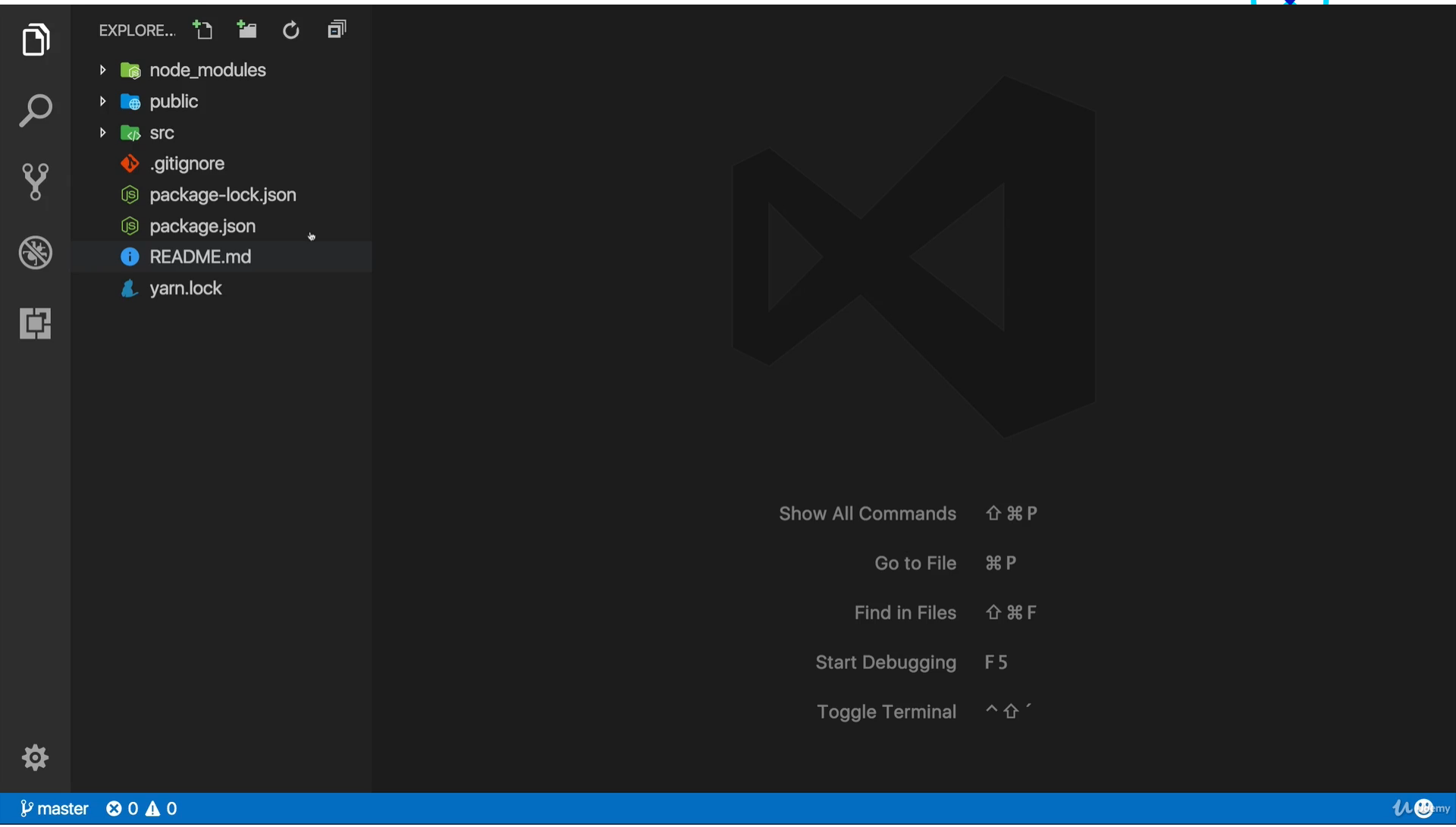


```
Compiled successfully!

You can now view react-complete-guide in the browser.

  Local:            http://localhost:3000/
  On Your Network:  http://192.168.178.23:3000/

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

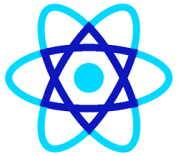


Edit `src/App.js` and save to reload.

Learn React

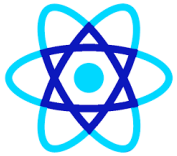

EXPLORE...
node_modules
public
src
.gitignore
package-lock.json
package.json
README.md
yarn.lock
Show All Commands ⇧ ⌘ P
Go to File ⌘ P
Find in Files ⇧ ⌘ F
Start Debugging F 5
Toggle Terminal ^ ⇧ ´
master 0 0

# JSX



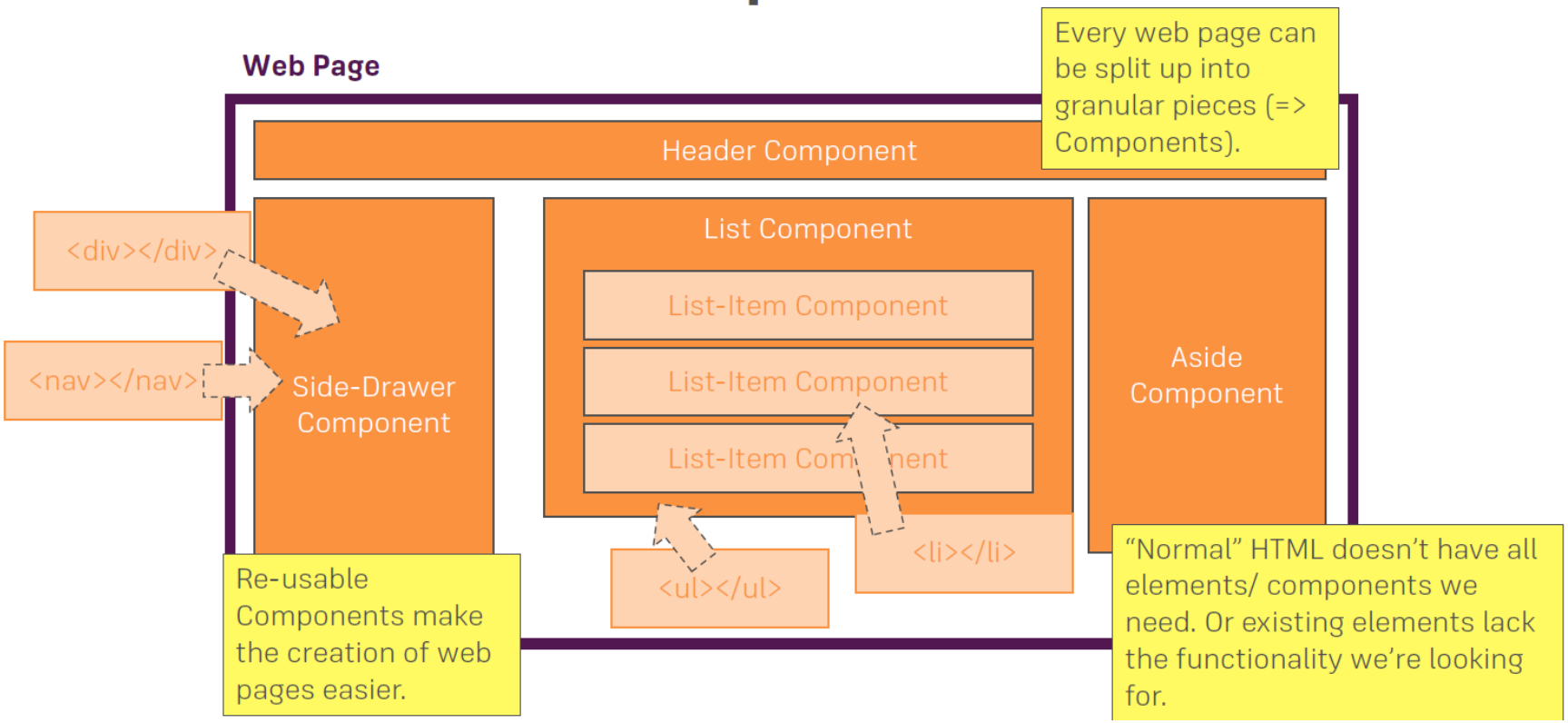
- **JSX:** JSX stands for JavaScript XML.
- Its an XML/ HTML like syntax used by React.
- It extends the ECMAScript so that XML/ HTML like text can co-exist along with JavaScript react code.
- This syntax is used by the pre-processors like *Babel* to transform HTML like text found in JavaScript files into standard JavaScript objects.
- With JSX, we can go a step further by again embedding the HTML code inside the JavaScript.
- This makes HTML codes easy to understand and boosts JavaScript's performance while making our application robust.

# JSX



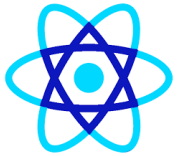
- It is faster than regular JavaScript because it performs optimization while translating the code to JavaScript.
- Instead of separating technologies by putting markup and logic in separate files, React uses components that contain both.
- It is type-safe, and most of the errors can be found at compilation time.
- It makes easier to create templates.

# Components


Web Page
Every web page can be split up into granular pieces (=> Components).
Header Component

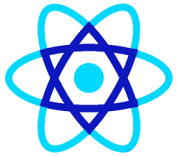
<nav></nav>
Side-Drawer Component
List Component
List-Item Component
List-Item Component
List-Item Component
Aside Component
<ul></ul>
<li></li>
Re-usable Components make the creation of web pages easier.
"Normal" HTML doesn't have all elements/ components we need. Or existing elements lack the functionality we're looking for.

# JSX



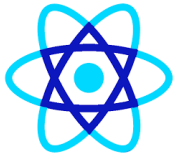
```
class App extends Component{
   render(){
      return(
         <div>
            <h1>React Training</h1>
          <h2>Training Institutes</h2>
            <p>Rocks on Hands on</p>
         </div>
      );
   }
}
export default App;
```

# JSX Internals



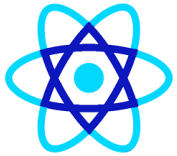
- //jsx internals
- //type,props,children
- //return React.createElement('div',null,'h1','Novice Developer!!!!')
-
- //css class name as props
- //return React.createElement('div',{className:"App"},React.createElement('h1',null,'Novice Developer!!!!'))
-

# JSX Restrictions



- 1.For Style class referred in Element as className=“”
- 2. Internally className becomes class.
- 3.JSX can return at the max only one root element. All other elements are sub elements.
-

# Core Functional Components



- import React from 'react'
- const person=()=>{
- return <h1>I am new bie!!!</h1>
- }

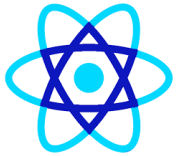
- export default person

- import Person from './Person/Person.js'

- import './App.css';

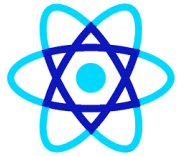
- class App extends Component

# Working with reusable components



```
class App extends Component {
  render() {
   return (
      <div className="App">
       <h1>Novice Developer!!!!</h1>
       {/* Nested element*/}
        <p>Creates Nested Element under Root!!</p>
        <Person/>
          <Person/>
          <Person/>
          <Person/>
      </div>
    );
```

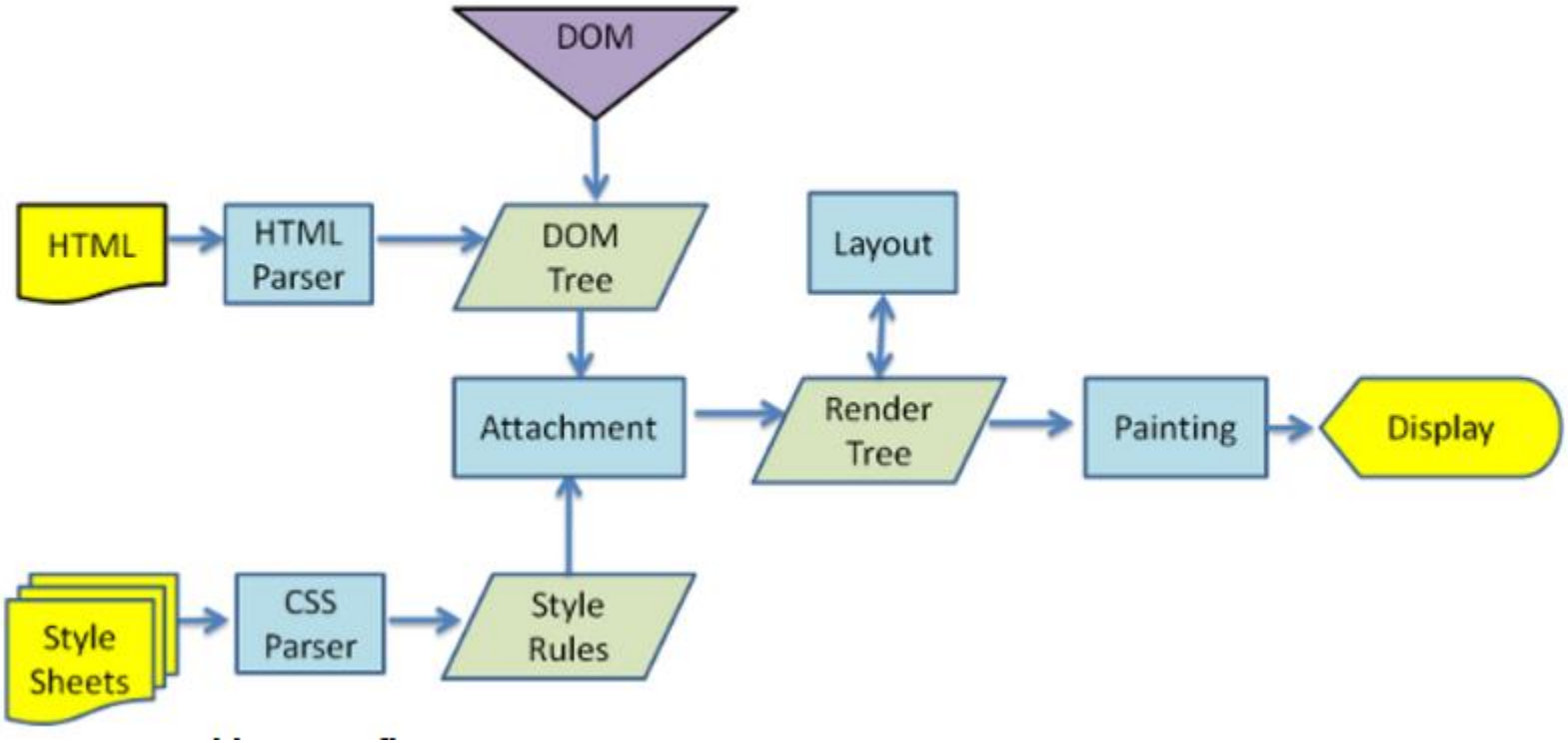
# Output Dynamic Content



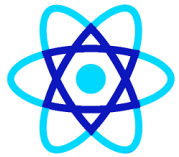
- import React from 'react'
- const person=()=>{
- return <h1>I am new bie!!!, having {Math.floor(Math.random()*30)} years experience</h1>
- }

- export default person

# Why updating Real DOM is slow:

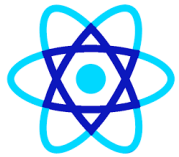


# Why updating Real DOM is slow:



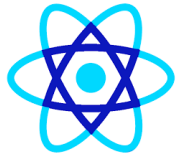
- **document.getElementById('elementId').innerHTML = "New Value“**
- Browser have to parses the HTML
- It removes the child element of elementId
- Updates the DOM with the “New Value”
- Re-calculate the CSS for the parent and child
- Recalculating the CSS and changed layouts uses **complex algorithm and they effect the performance.**
- Update the layout i.e. each elements exact co-ordinates on the screen
- Traverse the render tree and paint it on the browser display

# Virtual DOM



- The virtual DOM (VDOM) is a programming concept where an ideal, or “virtual”, representation of a UI is kept in memory and synced with the “real” DOM by a library such as ReactDOM. This process is called reconciliation.
- This approach enables the declarative API of React: You tell React what state you want the UI to be in, and it makes sure the DOM matches that state.
- This abstracts out the attribute manipulation, event handling, and manual DOM updating that you would otherwise have to use to build your app.

# Virtual DOM

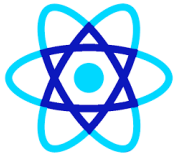


- Virtual DOM is in-memory representation of Real DOM.
- It is lightweight JavaScript object which is copy of Real DOM.
- Updating virtual DOM in ReactJS is faster because ReactJS uses

**Virtual DOM inner working**

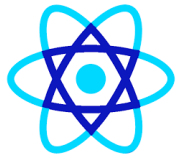
- Efficient diff algorithm
- Batched update operations
- Efficient update of sub tree only
- Uses observable instead of dirty checking to detect change

# Virtual DOM



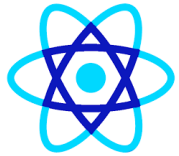
- AngularJS uses dirty checking to find the models which has changed.
- This dirty checking process runs in cycle after a specified time.
- As the application grows, checking the whole model reduces the performance and thus makes the application slow.

# Virtual DOM



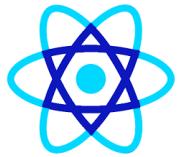
- ReactJS uses observable's to find the modified components.
- Whenever setState() method is called on any component, ReactJS makes that component dirty and re-renders it.
- Whenever setState() method is called, ReactJS creates the whole Virtual DOM from scratch.
- Creating a whole tree is very fast so it does not affect the performance.
- At any given time, ReactJS maintains two virtual DOM, one with the updated state Virtual DOM and other with the previous state Virtual DOM.

# Virtual DOM



- ReactJS using diff algorithm compares both the Virtual DOM to find the minimum number of steps to update the Real DOM.
- Finding minimum number of modifications between two trees have complexity in the order of O(n^3).
- But react uses heuristic approach with some assumptions which makes the problems to have complexity in the order of O(n).

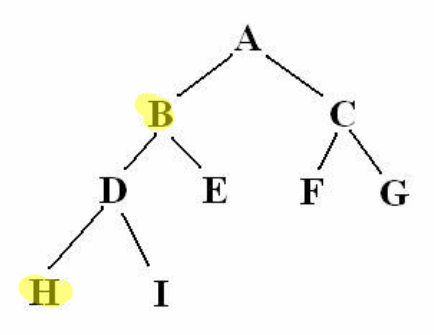
# Virtual DOM Steps for DIFF Algorithm



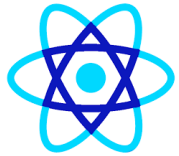
- Re-render all the children if parent state has changed.
- If the state of a component has changed, then ReactJS re-renders all the child components even if child components are not modified.
- To prevent the unwanted re-render of the child components we can use shouldComponentUpdate() component life cycle method.
- This will further help in boosting the performance.

# Virtual DOM Steps for DIFF Algorithm

- Breadth First Search. ReactJS traverse the tree using BST.
- Consider the below tree. States of element B and H have changed.
- So when using BST ReactJS reached element B it will by default re-render the element H.
- This is the reason to use BST for tree traversal

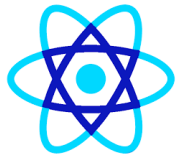


# Virtual DOM Steps for DIFF Algorithm

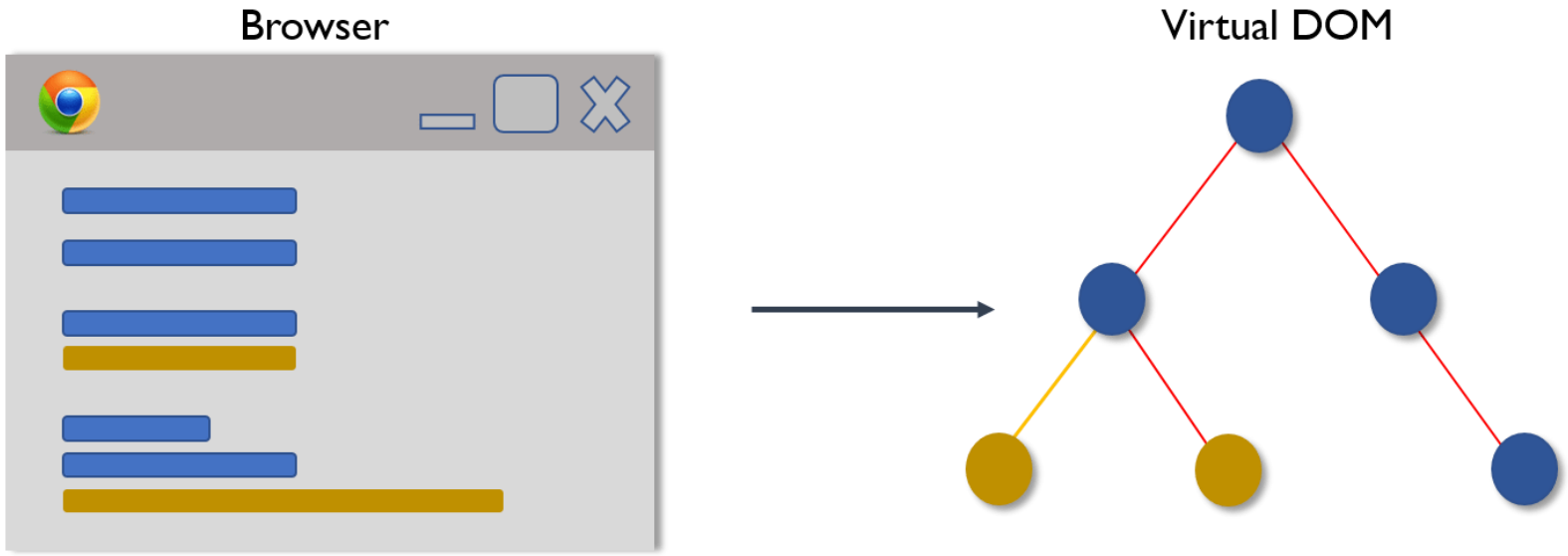


- Reconciliation. It is the process to determine which parts of the Real DOM need to be updated. It follow below steps:
- Two elements of different types will produce different trees.
- The developer can hint at which child elements may be stable across different renders with a key prop.

# Virtual DOM

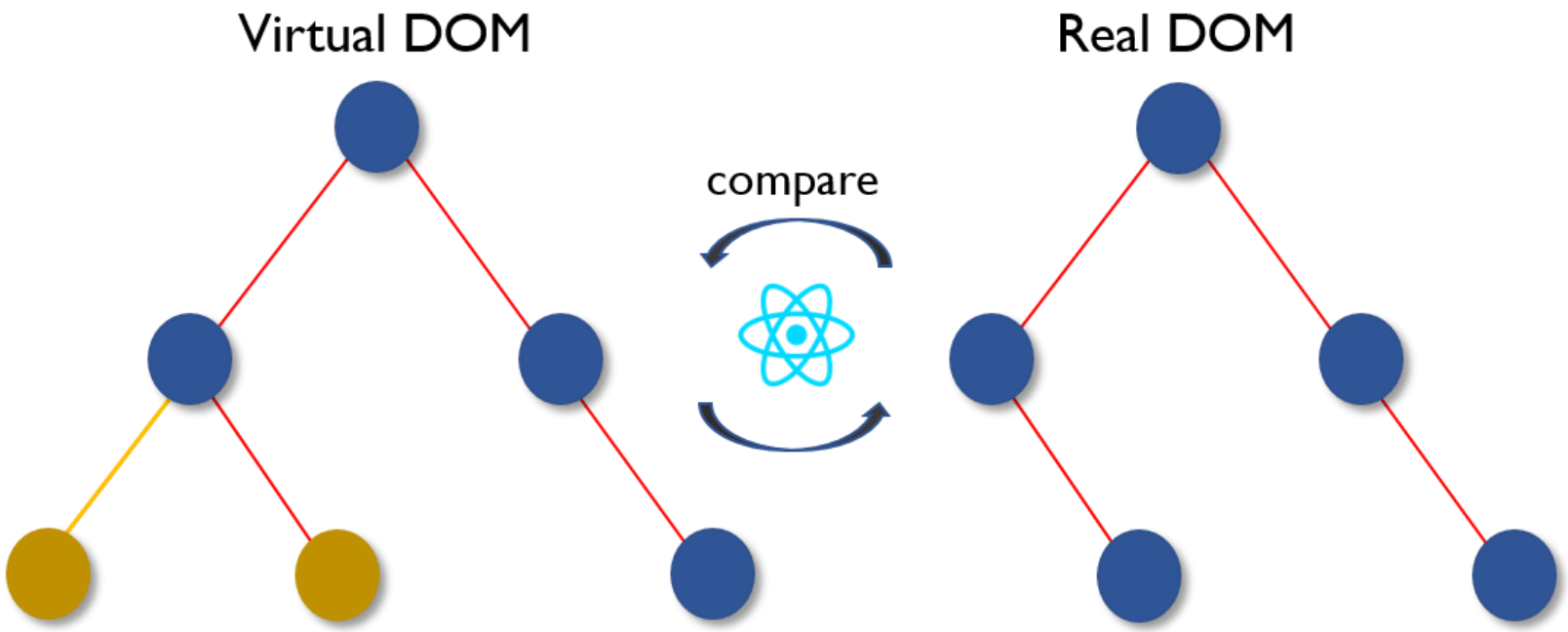


- Whenever any underlying data changes, the entire UI is re-rendered in Virtual DOM representation.

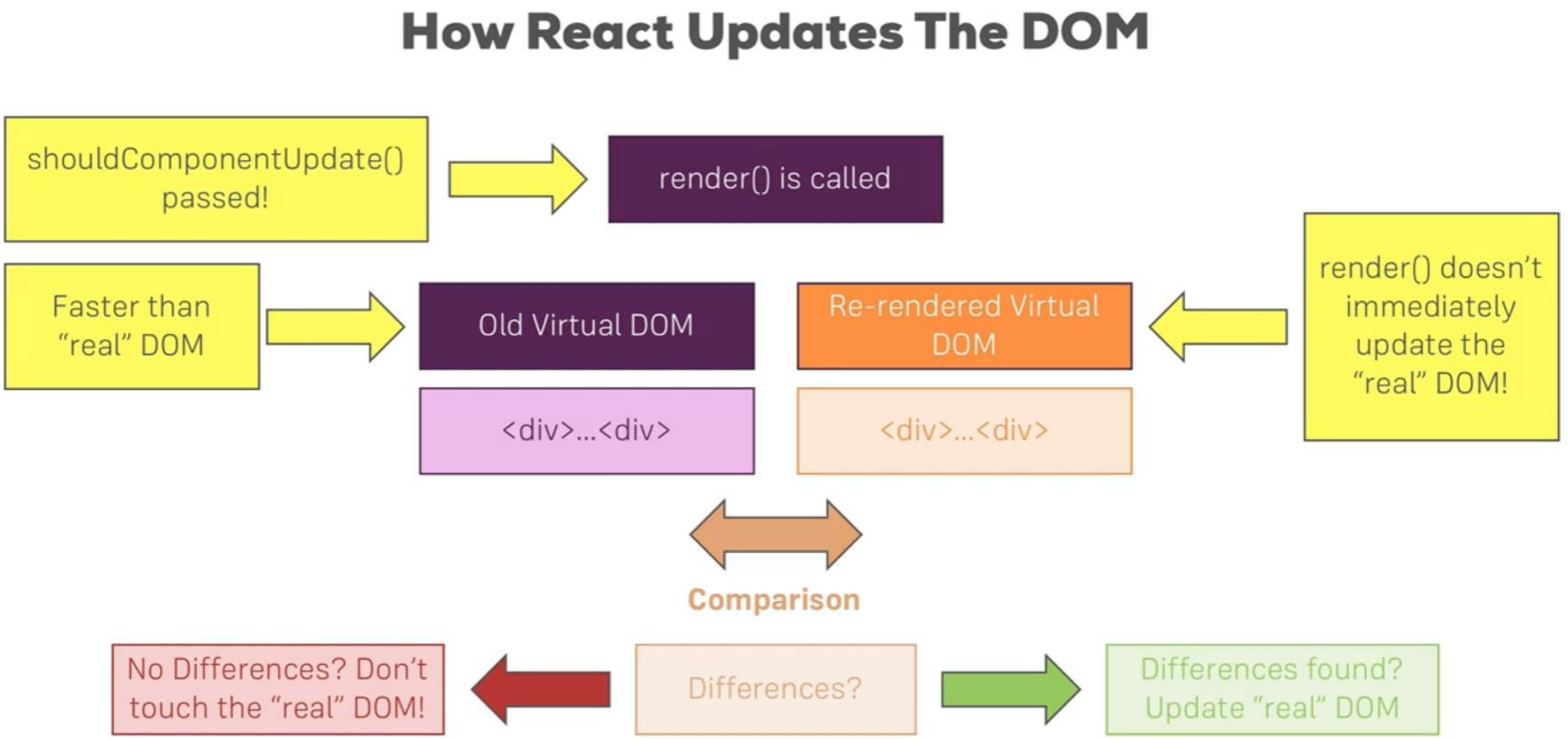


# Virtual DOM

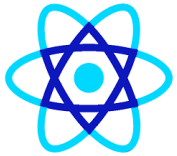
- Then the difference between the previous DOM representation and the new one is calculated..



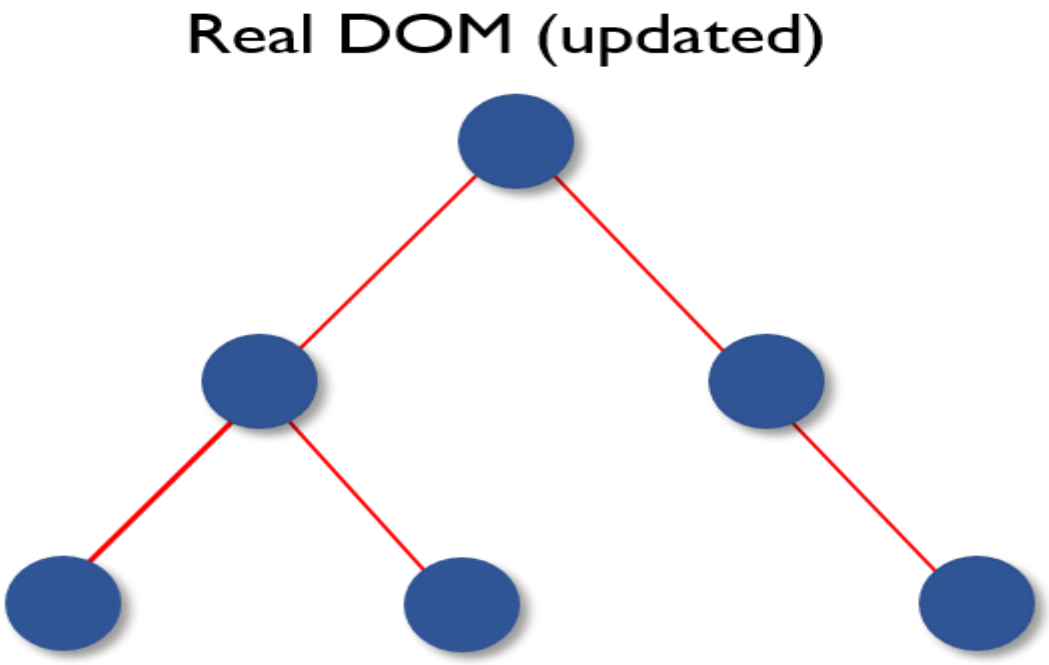
# Virtual DOM


How React Updates The DOM
shouldComponentUpdate() passed!
render() is called
Faster than "real" DOM
Old Virtual DOM
...
Re-rendered Virtual DOM
...
render() doesn't immediately update the "real" DOM!
Comparison
No Differences? Don't touch the "real" DOM!
Differences?
Differences found? Update "real" DOM

# Virtual DOM

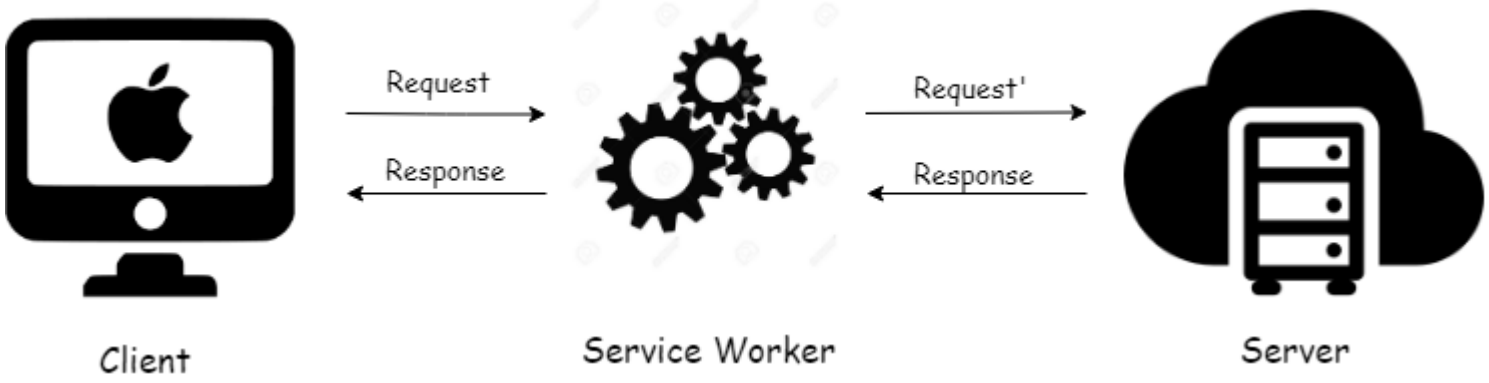


- Once the calculations are done, the real DOM will be updated with only the things that have actually changed. You can think of it as a patch. As patches are applied only to the affected area, similarly, the virtual DOM acts as patches and are applied to the elements which are updated or changed, in the real DOM.

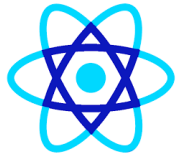


# Service Worker

- A service worker is basically a JavaScript file.
- One thing that differentiate a service worker file from a normal JavaScript file, is that service worker runs in the background.
- Service worker plays a very vital role when it's comes to Progressive Web Apps (PWA), as it is responsible for offline caching, push notifications, background sync etc.



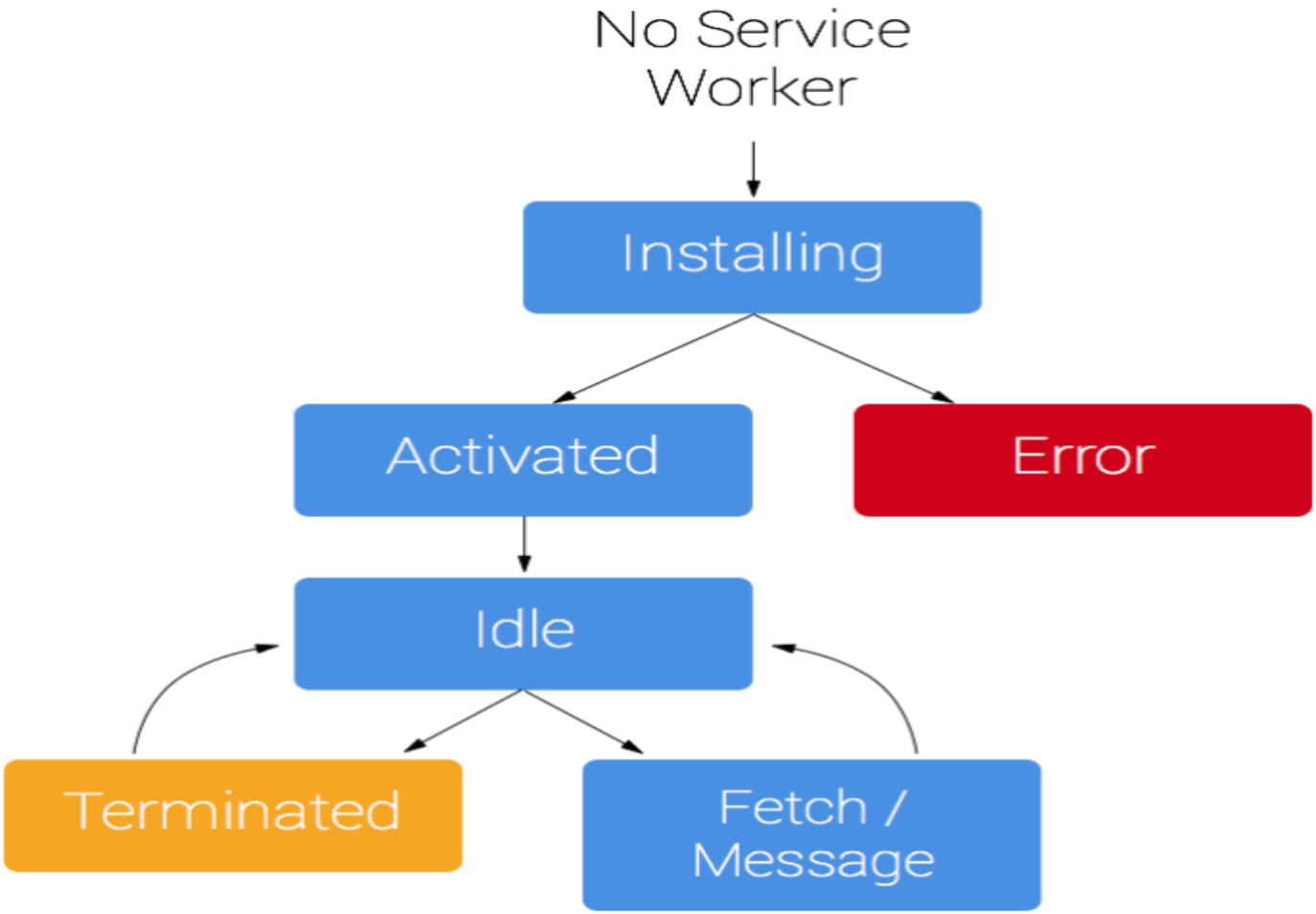
# Service Worker



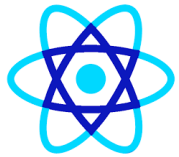
- For effective use of service worker, an understanding of the service lifecycle is essential. The service worker lifecycle consists of mainly 3 phases, which are:

- Registration
- Installation
- Activation

# Service Worker



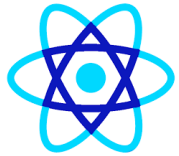
# Service Worker



- This is the first phase of the phase of the lifecycle. Since service worker is not currently supported in all browsers yet.
- When registering a service worker, we must first check to make sure the browser supports service worker.

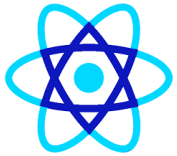
```
if ('serviceWorker' in navigator) {
    navigator.serviceWorker.register('/sw.js')
    .then(function (registration) {
        console.log('Service worker registered!');
    })
    .catch(function (err) {
        console.log('Registration failed!');
    })
}
```

# Service Worker



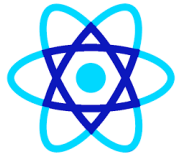
- **Installation**
- **Upon successful registration of the service worker, the script is downloaded and then the browser will attempt to install the service worker. The service worker will only be installed in either of these cases:**
- **The service worker hasn't been registered before**
- **The service worker script changes (even if it's by one byte).**
- **Once a service worker has been installed, an install event is fired.**

# Service Worker



- **Activation**
- **If the installation was successful, the service worker enters an installed state (though not yet active), during which it waits to take control of the page from the current service worker.**
- **It then moves on to the next phase in the lifecycle, which is the activation phase. A service worker is not immediately activated upon installation.**

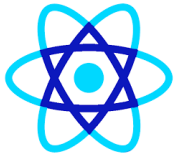
# Service Worker



- **Activation**
- **A service worker will only be active (that is, be activated) in any of these cases:**

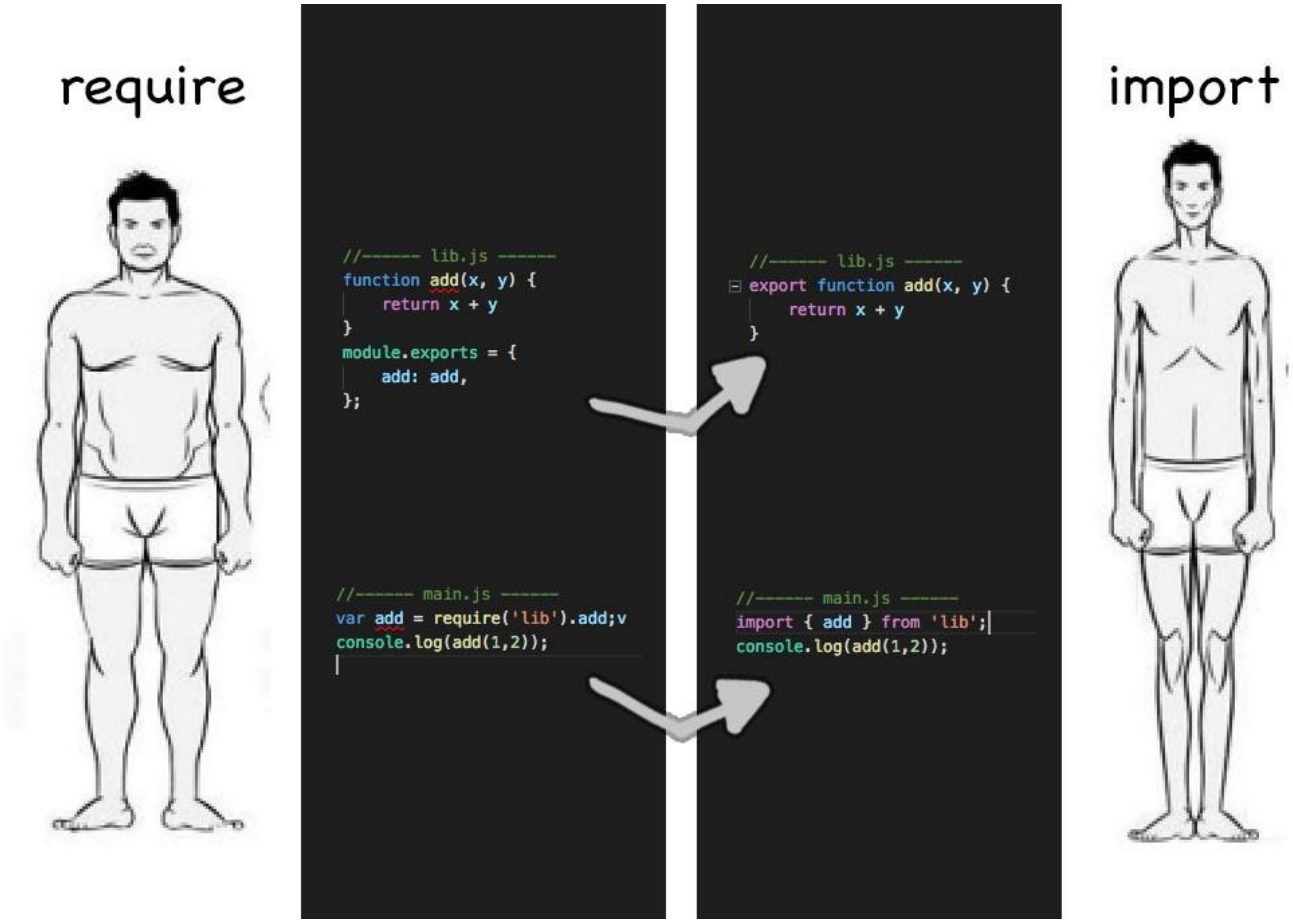
- **If there is no service worker currently active**
- **If the self.skipWaiting() is called in the install event handler of the service worker script**
- **If the user refreshes the page**

# Service Workers

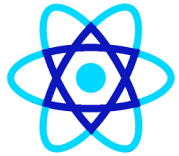


- Service Worker, in simple terms, is a script file that runs in the background without interfering with the user interactions.
- Service Worker acts as a proxy server that intercepts the network requests sent by your web application to the server.
- In the sense, requests to fetch javascript or css files, images goes through service worker to the server.
- Service Worker has the ability to modify this request or send a custom response back to the client.
- It uses caching to its full advantage to cache some of the resources, so that subsequent requests for these resources can be met out speedily with a cache hit instead of going all the way back to the server to fetch it (More on caching later).
- Service Worker operates as an event-driven system.

# Require vs Import

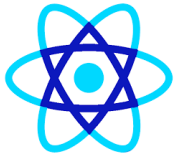


# Different types of export



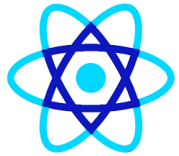
- 1— Named exports (several per module)
- 2— Default exports (one per module)
- 3—Mixed named & default exports
- 4— Cyclical Dependencies

# Named Export



```
//------ lib.js ------
export const sqrt = Math.sqrt;
export function square(x) {
return x * x;
}
export function diag(x, y) {
return sqrt(square(x) + square(y));
}

//------ main.js ------
import { square, diag } from 'lib';
console.log(square(11)); // 121
console.log(diag(4, 3)); // 5
```

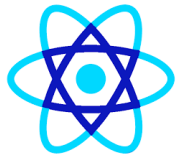
# Named Export



or

```
//------ main.js ------
import * as lib from 'lib';
console.log(lib.square(11)); // 121
console.log(lib.diag(4, 3)); // 5
```

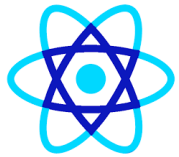
# Default exports (one per module)



- //------ myFunc.js ------
- export default function () { ... };

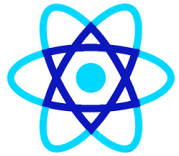
- //------ main1.js ------
- import myFunc from 'myFunc';
- myFunc();

# Mixed Default and Named Exports



```
//------ underscore.js ------
export default function (obj) {
    ...
};
export function each(obj, iterator, context) {
    ...
}
export { each as forEach };

//------ main.js ------
import _, { each } from 'underscore';
...
```

# Cyclical Dependency



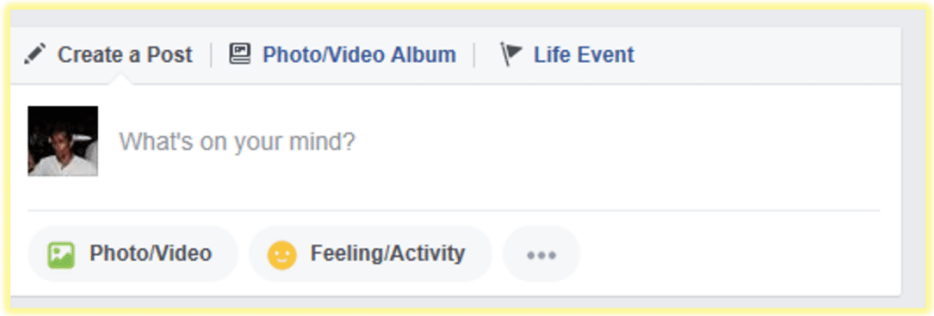
- // lib.js
- import Main from 'main';
- var lib = {message: "This Is A Lib"};
- export { lib as Lib };

- // main.js
- import { Lib } from 'lib';
- export default class Main {
- // ....
- }

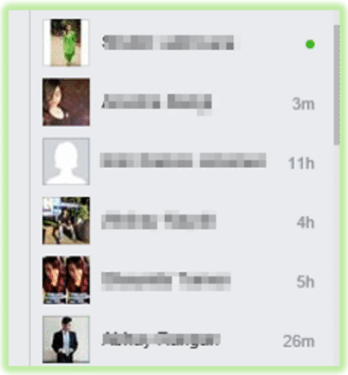
# Facebook Components


Search Bar

Notifications Bar

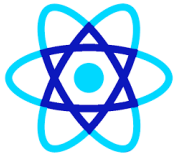




Add Post



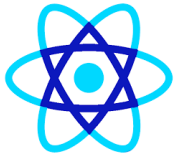

Chat Window

# Building Blocks



- Components
- Props
- State
- State Lifecycle
- Event handling
- Keys

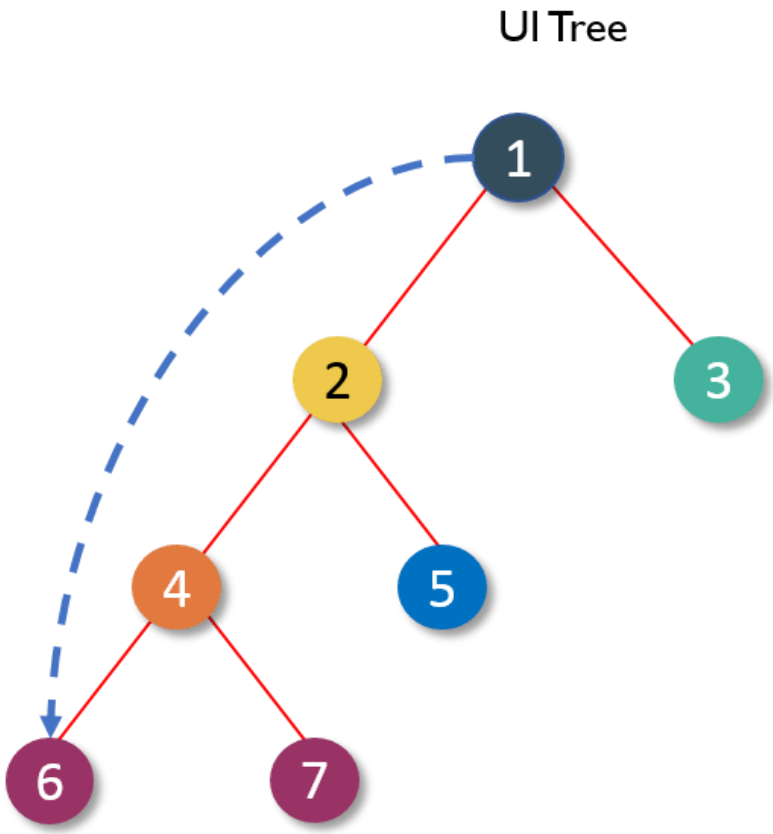
# Components

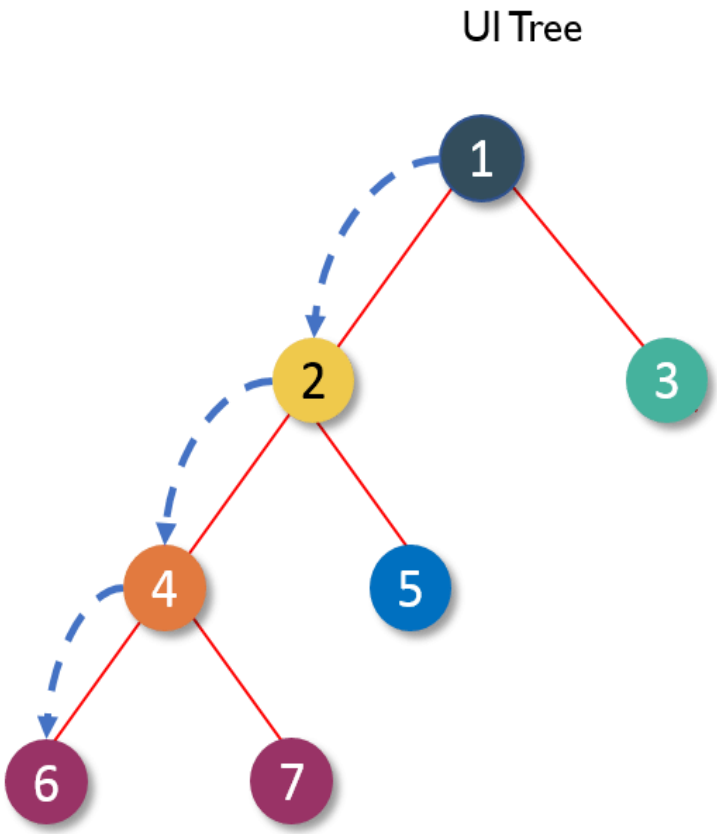


- The entire application can be modeled as a set of independent components.
- Different components are used to serve different purposes.
- This enables us to keep logic and views separate.
- React renders multiple components simultaneously. Components can be either stateful or stateless.

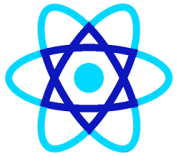
# Props

Prop is a way of passing data from parent to child component.



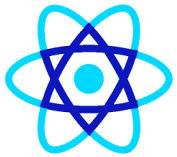



# Props



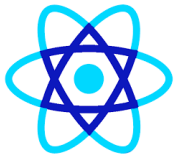
- const person=(props)=>{
- return <h1>I am {props.name}!!!, having {Math.floor(Math.random()*props.value)} years experience</h1>
- }
- <Person name="Parameswari" value="48"/>
- <Person name="Bala" value="53"/>
- <Person name="Vignesh" value="24" />
- <Person/>

# React Props Validation



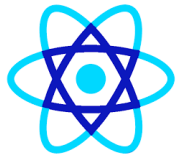
| SN | PropsType | Description |
|---|---|---|
| **1.** | PropTypes.any | The props can be of any data type. |
| **2.** | PropTypes.array | The props should be an array. |
| **3.** | PropTypes.bool | The props should be a boolean. |
| **4.** | PropTypes.func | The props should be a function. |
| **5.** | PropTypes.number | The props should be a number. |
| **6.** | PropTypes.object | The props should be an object. |
| **7.** | PropTypes.string | The props should be a string. |

# React Props Validation



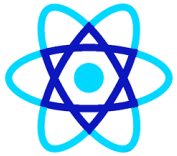
| | | |
|---|---|---|
| **8.** | PropTypes.symbol | The props should be a symbol. |
| **9.** | PropTypes.instanceOf | The props should be an instance of a particular JavaScript class. |
| **10.** | PropTypes.isRequired | The props must be provided. |
| **11.** | PropTypes.element | The props must be an element. |
| **12.** | PropTypes.node | The props can render anything: numbers, strings, elements or an array (or fragment) containing these types. |
| **13.** | PropTypes.oneOf() | The props should be one of several types of specific values. |
| **14.** | PropTypes.oneOfType([PropTypes.string,PropTypes.number]) | The props should be an object that could be one of many types. |

# Understanding the Children Property



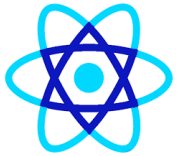
- return ( //important
- 
- <h1>I am {props.name}!!!, having {Math.floor(Math.random()*props.value)} years experience</h1>
- {props.children}
- 
- )
- <Person name="Bala" value="53">Hobby:Listening Music</Person>

# Understanding and using state



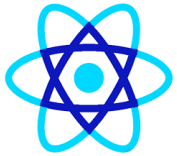
```
state={
    persons:[
        {
          name:"Parameswari",value:48
        },
        {
            name:"Bala",value:53
        }
    ],
    customers:{
        Id:102,
        name:"Arjun"
    }
  }
```

# Understanding and using state



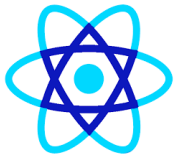
```
handler=()=>{
   // console.log( "clicked");
      this.setState({
          persons:[
              {
                  name:"ParameswariBala",value:49
              },
              {
                  name:"BalaManickam",value:54
              }
          ]
      })
  }
```

# Understanding and using state



```
render() {
    return (
      <div className="App">

        <h1>Novice Developer!!!!</h1>
        {/* Nested element*/}
          <button onClick={this.handler}>Show State</button>
        <p>Creates Nested Element under Root!!</p>
        <Person name={this.state.persons[0].name} value={this.state.persons[0].value}/>
          <Person name={this.state.persons[1].name} value={this.state.persons[1].value}>Hobby:Listening Music</Person>
          <Person name="Vignesh" value="24" />
          <Person/>

      </div>
    );
```

# Props vs State



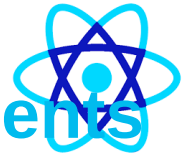
| SN | Props | State |
|---|---|---|
| **1.** | Props are read-only. | State changes can be asynchronous. |
| **2.** | Props are immutable. | State is mutable. |
| **3.** | Props allow you to pass data from one component to other components as an argument. | State holds information about the components. |
| **4.** | Props can be accessed by the child component. | State cannot be accessed by child components. |
| **5.** | Props are used to communicate between components. | States can be used for rendering dynamic changes with the component. |
| **6.** | Stateless component can have Props. | Stateless components cannot have State. |
| **7.** | Props make components reusable. | State cannot make components reusable. |
| **8.** | Props are external and controlled by whatever renders the component. | The State is internal and controlled by the React Component itself. |

# Passing method references between components

- App.js
- {/* click here*/}
- <Person name="Vignesh" value="24" click={this.handler} />

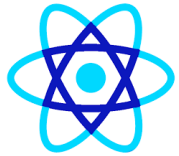
- Person.js
- <h1 onClick={props.click}>I am {props.name}!!!, having {Math.floor(Math.random()*props.value)} years experience</h1>
-

# Passing method references between components



- App.js
- <button onClick={()=>this.handler("Parameswari","Bala")}>Show State</button>

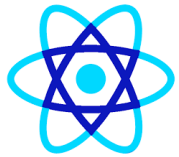
# Adding Two way binding



- Person.js
- <input type="text" onChange={props.changed}  />

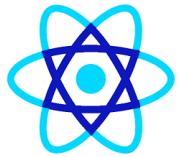
- App.js
- <Person name="Vignesh" value="24"
- click={this.handler.bind(this,"Bala","Parameswari")}
- changed={this.changeHandler}
- />
- <Person/>

# Adding Two way binding



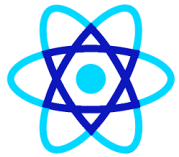
```
changeHandler=(event)=>{
    this.setState({
        persons:[
            {
                name:"Parameswari",value:49
            },
            {
                name:event.target.value,value:54
            }
        ]
    })
}
```

# Adding Styling with Stylesheets



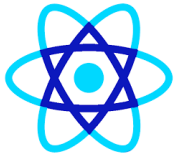
- import './Person.css'
- const person=(props)=>{
- return ( //important
- 
- <h1 onClick={props.click}>I am {props.name}!!!, having {Math.floor(Math.random()*props.value)} years experience</h1>
- <input type="text" onChange={props.changed}  />
- {props.children}
- 
- )
- }

# Adding Inline Style



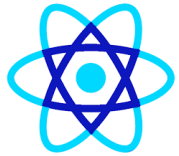
```
render() {
    const styleProp={
        backgroundColor:'blue',
        border:'2px solid red',
        color:'white'
    }
  return (
    <div className="App" >
     <h1>Novice Developer!!!!</h1>
     {/* Nested element*/}
        {/* second syntax */}
        <button  style={styleProp}
```

# Rendering Lists and Conditional Styles



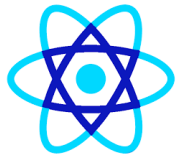
- let persons = null;
- if ( this.state.showPersons )
- {
- persons=(
- 
- <Person name={this.state.persons[0].name} value={this.state.persons[0].value}/>
- < Person name={this.state.persons[1].name} value={this.state.persons[1].value}>Hobby:Listening Music</Person>
- {/* click here*/}
- {/*first syntax*/}

# Rendering Lists and Conditional Styles



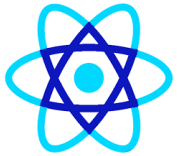
```
<Person name="Vignesh" value="24"
                    click={this.handler.bind(this, "Bala", "Parameswari")}
                    changed={this.changeHandler}
                />
                <Person/>
            </div>
        )
    }
```

# Outputting Lists

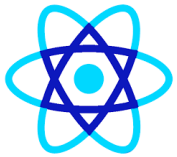


- persons=(
- 
- {
- this.state.persons.map(person=>{
- return<Person name={person.name} value={person.value} click={this.handler.bind(this, "Bala", "Parameswari")}
- changed={this.changeHandler}/>
- })
- }

# Index.js

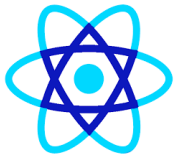


- npm start
- Below adding individual links for above commands for node @8.9.4
- @node_modules/react-scripts/bin/react-scripts.js >> line number 35
- @node_modules/react-scripts/scripts/start.js >> line number 46
- @node_modules/react-scripts/config/webpack.config.dev.js >> line number 21, 102
- @node_modules/react-scripts/config/paths.js >> line number 56



# Stateless vs Stateful Components

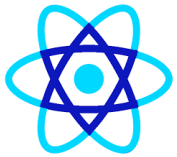
| Stateful (Containers) | Stateless |
| --- | --- |
| class XY extends Component | const XY = (props) => { ... } |
| ✓ Access to State | ✗ Access to State |
| ✓ Lifecycle Hooks | ✗ Lifecycle Hooks |
| Access State and Props via "**this**" | Access Props via "**props**" |
| this.state.XY & this.props.XY | props.XY |
| Use only if you need to manage State or access to Lifecycle Hooks! | Use in all other Cases |



# Component Lifecycle

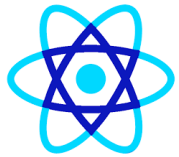
Only available in Stateful Components!

- constructor()
- componentWillReceiveProps()
- componentWillUpdate()
- componentDidCatch()
- componentWillUnmount()
- componentWillMount()
- shouldComponentUpdate()
- componentDidUpdate()
- componentDidMount()
- render()



# Component Lifecycle – Creation

| | |
|---|---|
| constructor() | componentWillMount() |
| componentWillReceiveProps() | shouldComponentUpdate() |
| componentWillUpdate() | componentDidUpdate() |
| componentDidCatch() | componentDidMount() |
| componentWillUnmount() | render() |



# Component Lifecycle – Creation

Default ES6 class Feature

Exists for Historic Reasons

constructor(props)

componentWillMount()

render()

Render Child Components

componentDidMount()

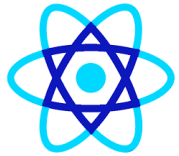
Call super(props)
**DO:** Set up State
**DON'T:** Cause Side-Effects

**DO:** Update State, last-minute Optimization
**DON'T:** Cause Side-Effects

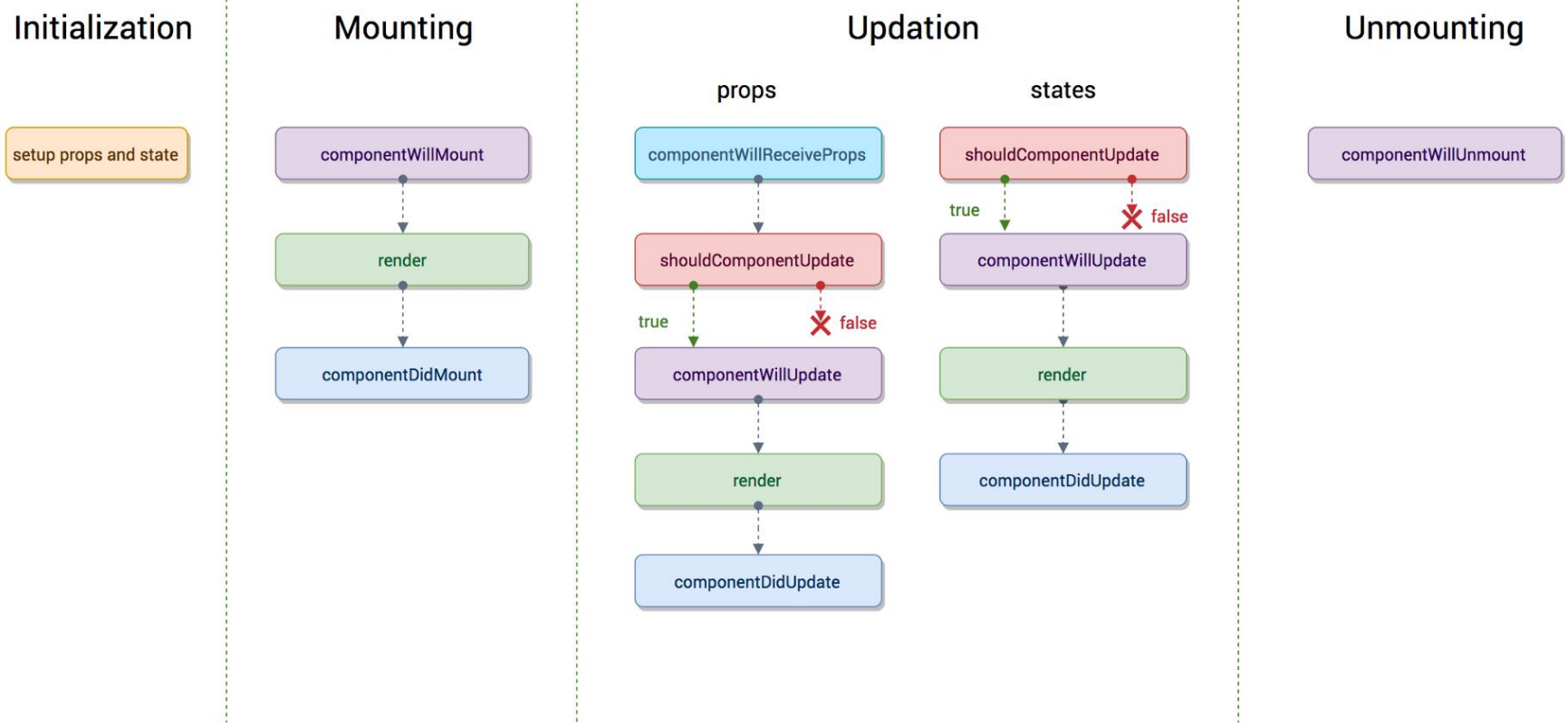
Prepare & Structure your JSX Code

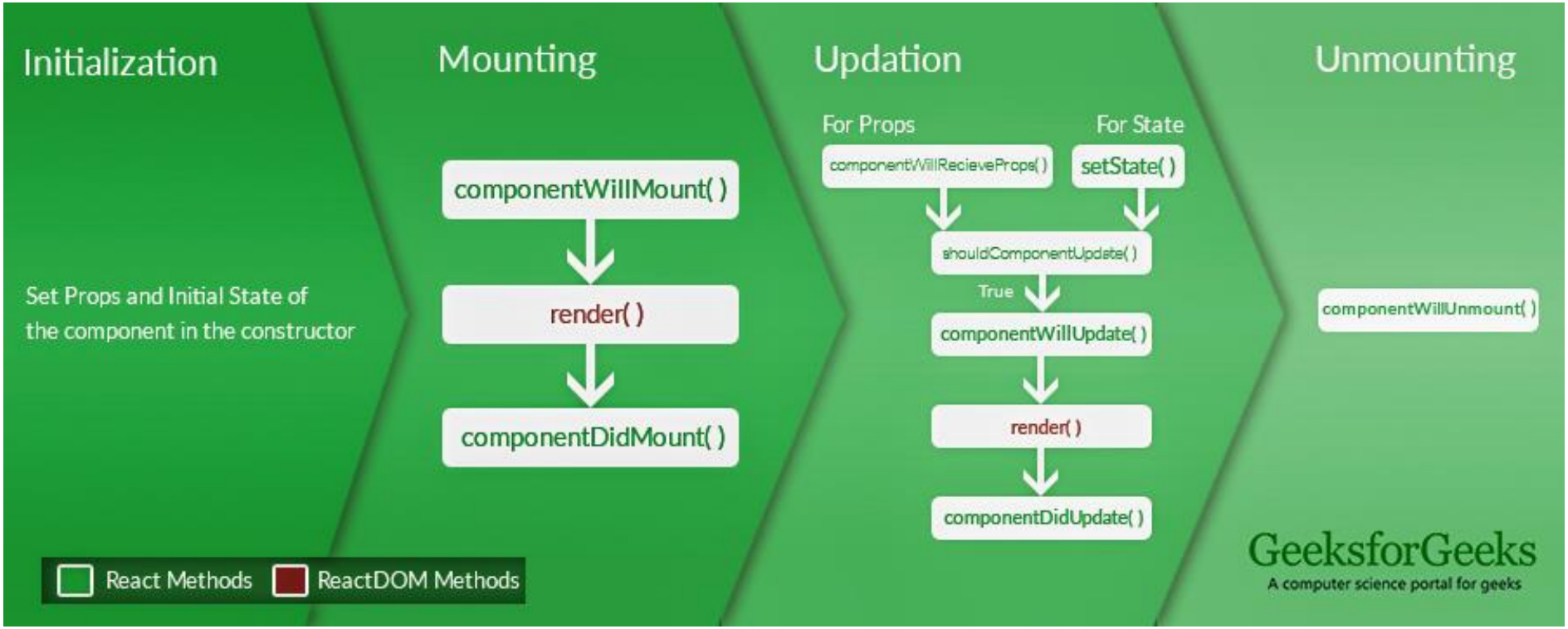
**DO:** Cause Side-Effects
**DON'T:** Update State (triggers re-render)

# React Component Lifecycle

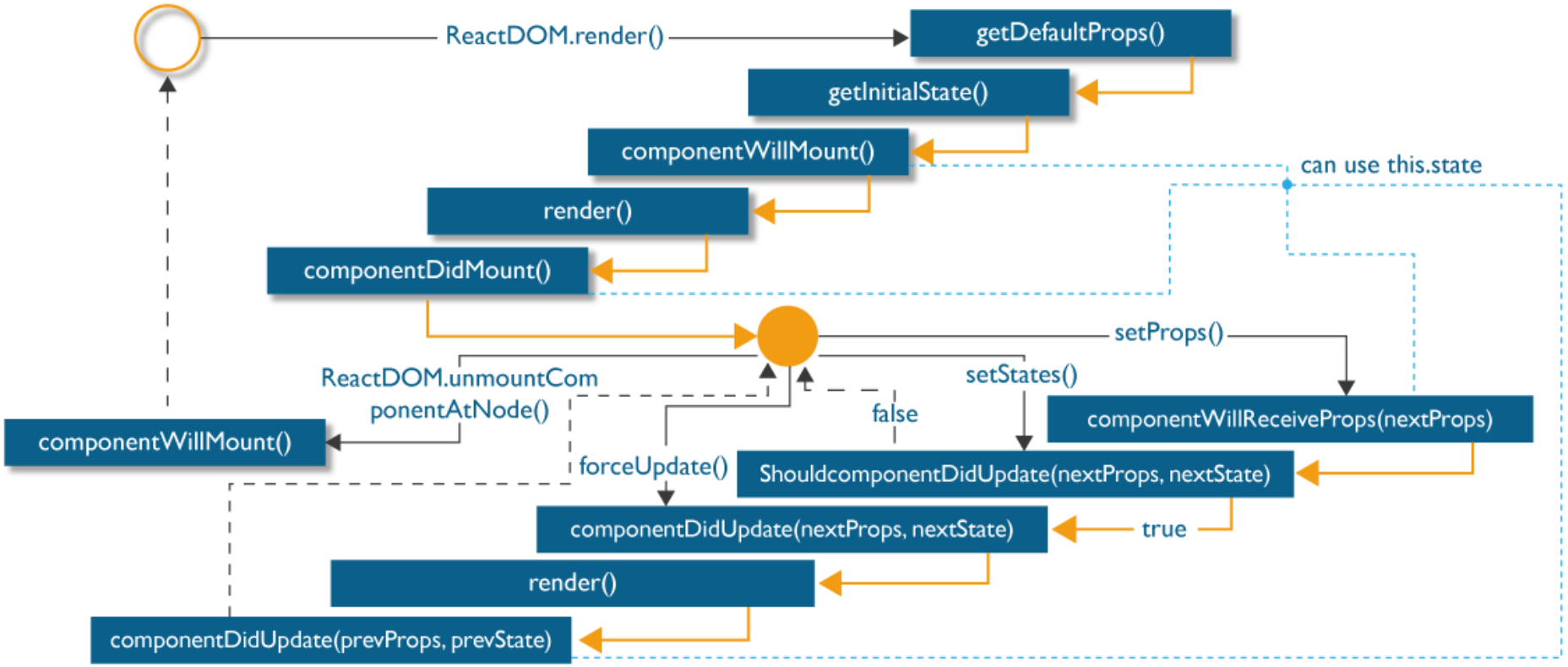


- Initial Phase
- Updating Phase
- Props change Phase
- Unmounting Phase

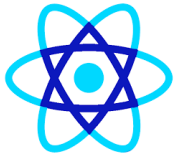

Initialization
setup props and state
Mounting
componentWillMount
render
componentDidMount
Updation
props
componentWillReceiveProps
shouldComponentUpdate
true
false
componentWillUpdate
render
componentDidUpdate
states
shouldComponentUpdate
true
false
componentWillUpdate
render
componentDidUpdate
Unmounting
componentWillUnmount


Initialization
Set Props and Initial State of
the component in the constructor
Mounting
componentWillMount( )
render( )
componentDidMount( )
Updation
For Props
For State
componentWillRecieveProps( )
setState( )
shouldComponentUpdate( )
True
componentWillUpdate( )
render( )
componentDidUpdate( )
Unmounting
componentWillUnmount( )
React Methods
ReactDOM Methods
GeeksforGeeks
A computer science portal for geeks

# React Component Lifecycle

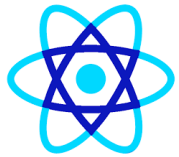


# Initial Phase



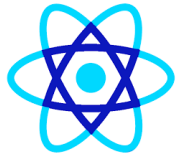
- ***getDefaultProps():*** This method is used to specify the default value of **this.props**. It gets called before your component is even created or any props from the parent are passed into it.
- **getInitialState():** This method is used to specify the default value of **this.state** before your component is created.
- ***componentWillMount():*** This is the last method that you can call before your component gets rendered into the DOM. But if you call **setState()** inside this method your component will not re-render.
- ***render():*** This method is responsible for returning a single root HTML node and must be defined in each and every component. You can return **null** or **false** in case you don't want to render anything.
- ***componentDidMount():*** Once the component is rendered and placed on the DOM, this method is called. Here you can perform any DOM querying operations.

# Updating Phase



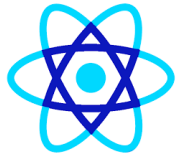
- ***shouldComponentUpdate():*** Using this method you can control your component's behavior of updating itself. If you return a true from this method, the component will update. Else if this method returns a false, the component will skip the updating.
- ***componentWillUpdate():*** This method is called just before your component is about to update. In this method, you can't change your component state by calling **this.setState**.
- ***render():*** If you are returning false via **shouldComponentUpdate()**, the code inside **render()** will be invoked again to ensure that your component displays itself properly.
- ***componentDidUpdate():*** Once the component is updated and rendered, then this method is invoked. You can put any code inside this method, which you want to execute once the component is updated.

# Props Change Phase



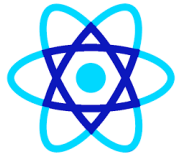
- ***componentWillReceiveProps():*** This method returns one argument which contains the new prop value that is about to be assigned to the component.
  *Rest of the lifecycle methods behave identically to the methods which we saw in the previous phase.*
- ***shouldComponentUpdate()***
- ***componentWillUpdate()***
- ***render()***
- ***componentDidUpdate()***

# The Unmounting Phase

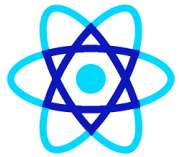


- ***componentWillUnmount():*** Once this method is invoked, your component is removed from the DOM permanently. In this method, you can perform any clean-up related tasks like removing event listeners, stopping timers, etc.

# Create React App

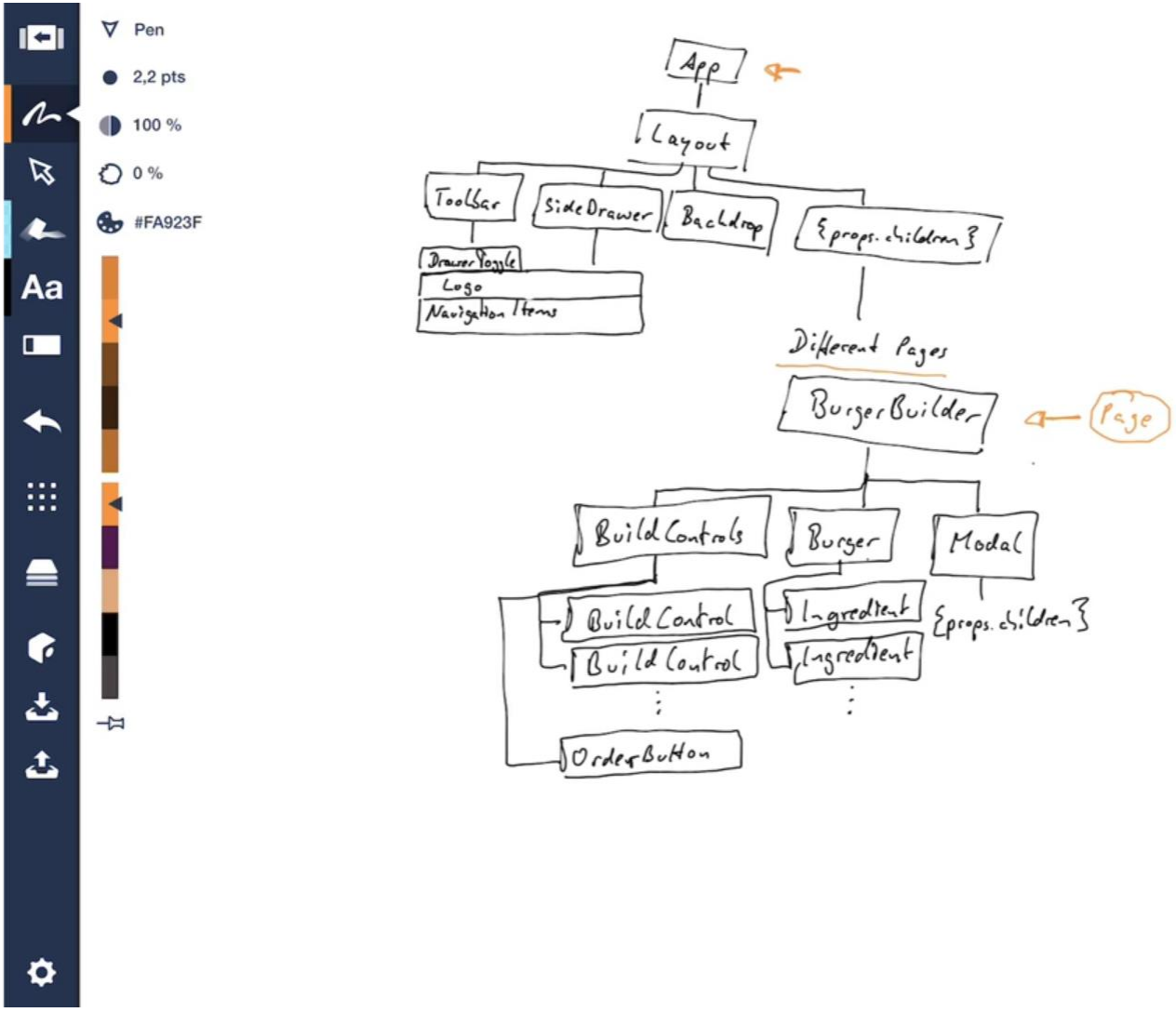


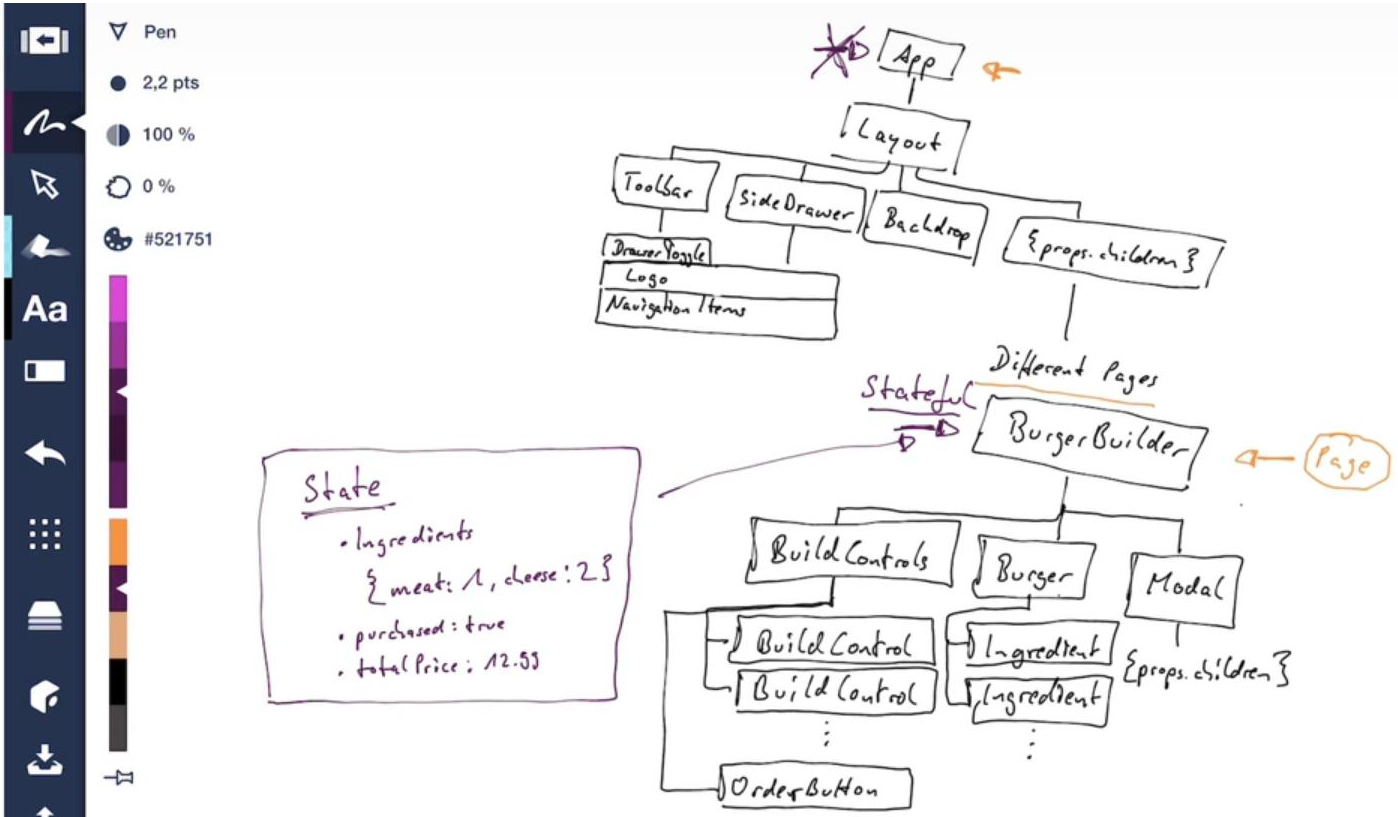
- First, install create-react-app globally with node package manager (npm).
- npm **install** -g create-react-app
- Then run the generator in your chosen directory.
- create-react-app my-app
- Navigate to the newly created directory and run the start script.
- **cd** my-app***/***
- npm **start**

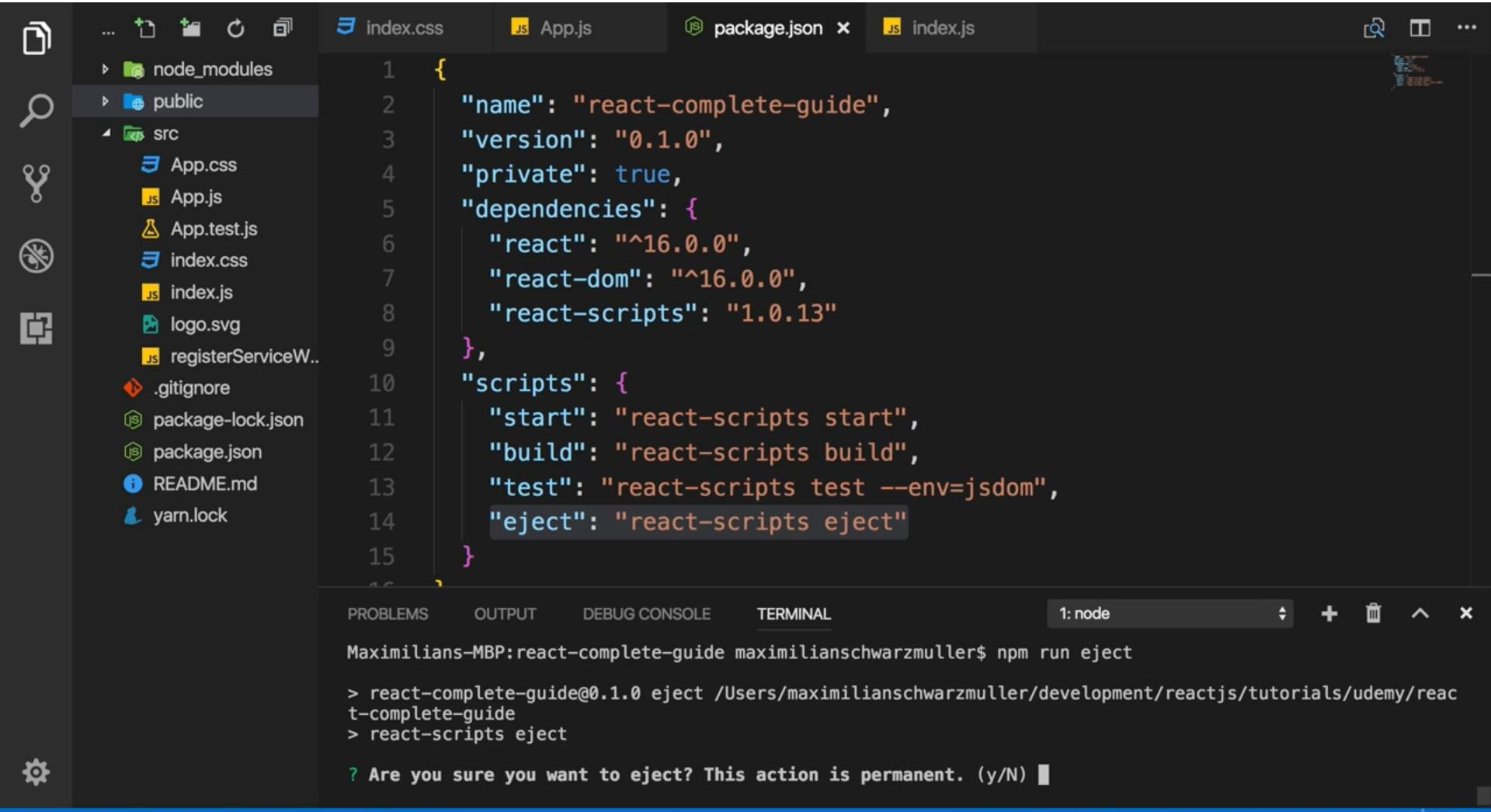


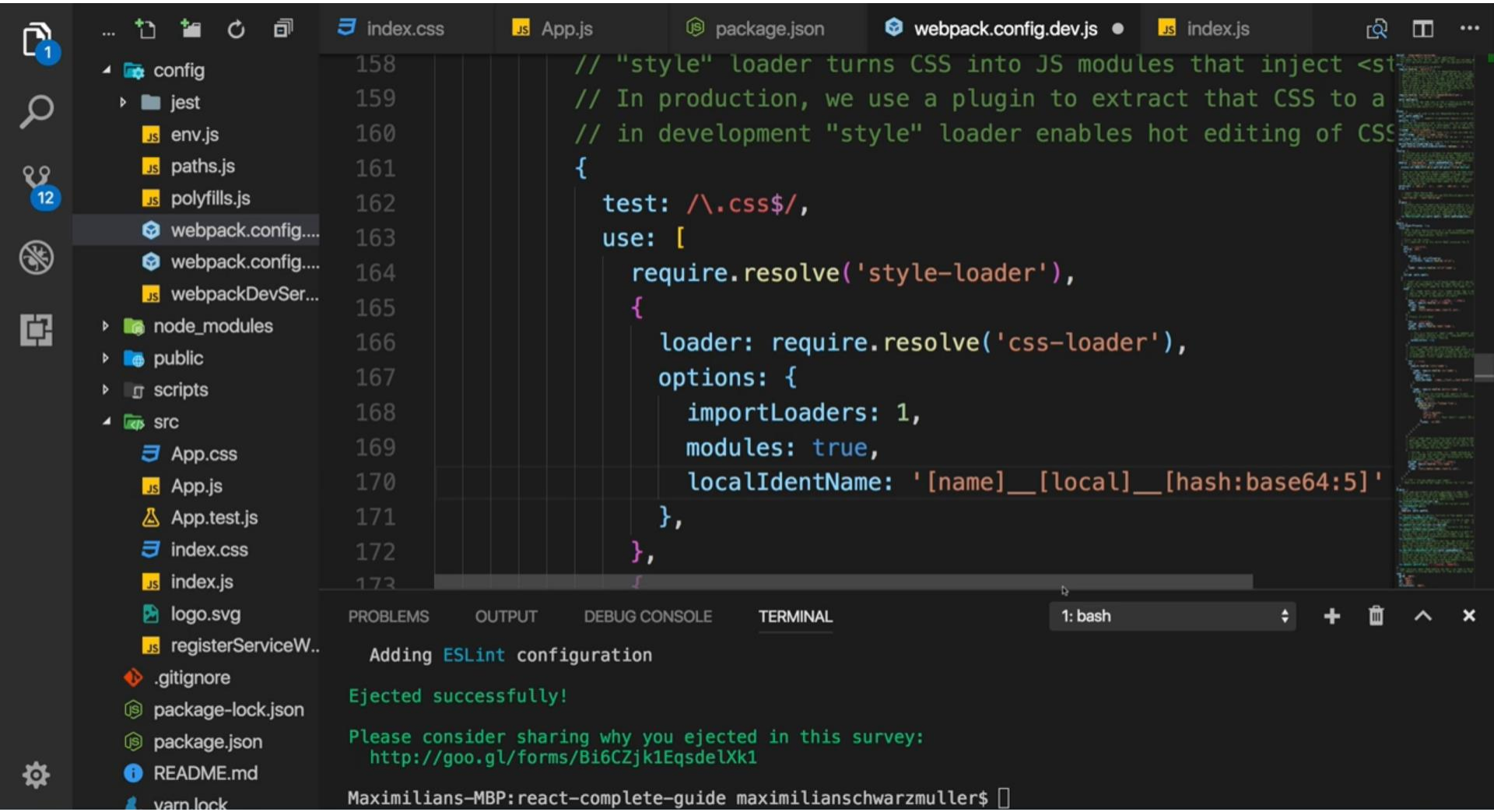
# Planning a React App

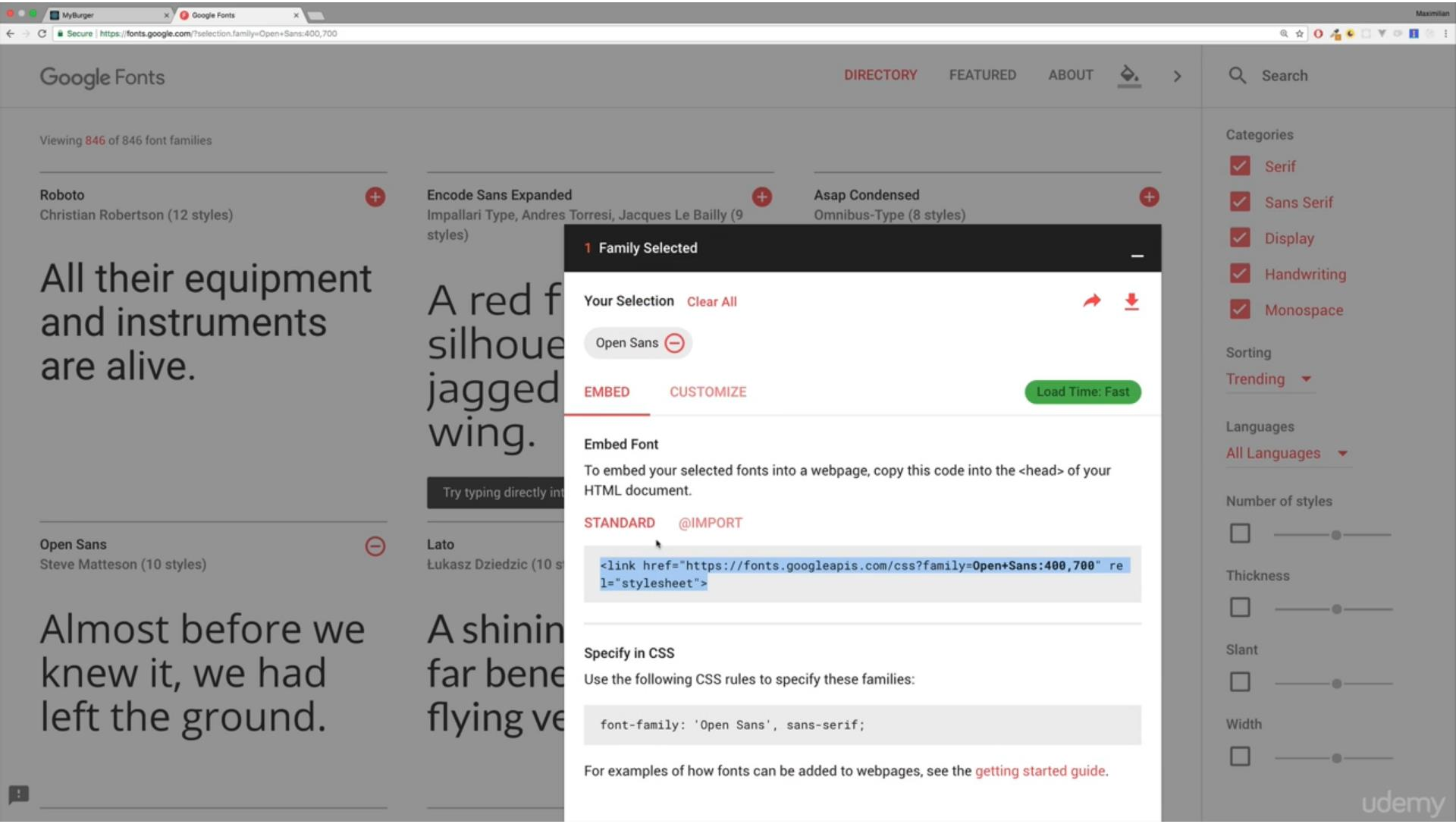
1. Component Tree / Component Structure
2. Application State (Data)
3. Components vs Containers


Pen
2,2 pts
100 %
0 %
#FA923F
App
Layout
ToolBar
SideDrawer
BackDrop
{props.children}
DrawerToggle
Logo
NavigationItems
Different Pages
BurgerBuilder
Page
BuildControls
Burger
Modal
BuildControl
BuildControl
Ingredient
Ingredient
{props.children}
OrderButton

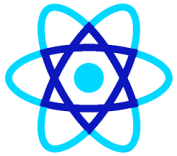

Pen
2,2 pts
100 %
0 %
#521751
App
Layout
ToolBar
SideDrawer
Backdrop
{props.children}
DrawerToggle
Logo
NavigationItems
Different Pages
Stateful
BurgerBuilder
Page
State
• Ingredients
{meat: 1, cheese: 2}
• purchased: true
• totalPrice: 12.99
BuildControls
Burger
Modal
BuildControl
BuildControl
Ingredient
Ingredient
{props.children}
OrderButton


index.css
App.js
package.json
index.js
node_modules
public
src
App.css
App.js
App.test.js
index.css
index.js
logo.svg
registerServiceW..
.gitignore
package-lock.json
package.json
README.md
yarn.lock
{
"name": "react-complete-guide",
"version": "0.1.0",
"private": true,
"dependencies": {
"react": "^16.0.0",
"react-dom": "^16.0.0",
"react-scripts": "1.0.13"
},
"scripts": {
"start": "react-scripts start",
"build": "react-scripts build",
"test": "react-scripts test --env=jsdom",
"eject": "react-scripts eject"
}
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
1: node
Maximilians-MBP:react-complete-guide maximilianschwarzmuller$ npm run eject
> react-complete-guide@0.1.0 eject /Users/maximilianschwarzmuller/development/reactjs/tutorials/udemy/react-complete-guide
> react-scripts eject
? Are you sure you want to eject? This action is permanent. (y/N)
07-my-burger-app-basic
{..} : Scanning..
Ln 14, Col 5 (30 selected)
Spaces: 2
UTF-8
LF
JSON


index.css
App.js
package.json
webpack.config.dev.js
index.js
config
jest
env.js
paths.js
polyfills.js
webpack.config....
webpack.config....
webpackDevSer...
node_modules
public
scripts
src
App.css
App.js
App.test.js
index.css
index.js
logo.svg
registerServiceW...
.gitignore
package-lock.json
package.json
README.md
yarn.lock
// "style" loader turns CSS into JS modules that inject <st
// In production, we use a plugin to extract that CSS to a
// in development "style" loader enables hot editing of CSS
{
test: /\.css$/,
use: [
require.resolve('style-loader'),
{
loader: require.resolve('css-loader'),
options: {
importLoaders: 1,
modules: true,
localIdentName: '[name]__[local]__[hash:base64:5]'
},
},
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
1: bash
Adding ESLint configuration
Ejected successfully!
Please consider sharing why you ejected in this survey:
http://goo.gl/forms/Bi6CZjk1EqsdelXk1
Maximilians-MBP:react-complete-guide maximilianschwarzmuller$


MyBurger
Google Fonts
Secure | https://fonts.google.com/?selection.family=Open+Sans:400,700
Google Fonts
DIRECTORY
FEATURED
ABOUT
Search
Viewing 846 of 846 font families
Roboto
Christian Robertson (12 styles)
All their equipment and instruments are alive.
Encode Sans Expanded
Impallari Type, Andres Torresi, Jacques Le Bailly (9 styles)
A red f
silhoue
jagged
wing.
Try typing directly int
Asap Condensed
Omnibus-Type (8 styles)
Open Sans
Steve Matteson (10 styles)
Almost before we knew it, we had left the ground.
Lato
Łukasz Dziedzic (10 s
A shinin
far bene
flying ve
1 Family Selected
Your Selection
Clear All
Open Sans
EMBED
CUSTOMIZE
Load Time: Fast
Embed Font
To embed your selected fonts into a webpage, copy this code into the <head> of your HTML document.
STANDARD
@IMPORT
<link href="https://fonts.googleapis.com/css?family=Open+Sans:400,700" rel="stylesheet">
Specify in CSS
Use the following CSS rules to specify these families:
font-family: 'Open Sans', sans-serif;
For examples of how fonts can be added to webpages, see the getting started guide.
Categories
Serif
Sans Serif
Display
Handwriting
Monospace
Sorting
Trending
Languages
All Languages
Number of styles
Thickness
Slant
Width
udemy

# React Router

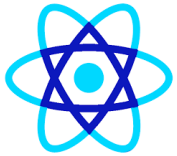


Npm install react-router-dom

Index.js

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

# React Router

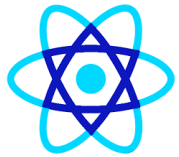


Npm install react-router-dom

Index.js

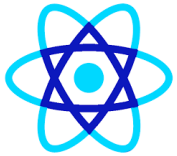
```
const app=(
    <BrowserRouter>
        <App/>
    </BrowserRouter>
)
ReactDOM.render(app,
document.getElementById('root'));
```

# Data Fetching



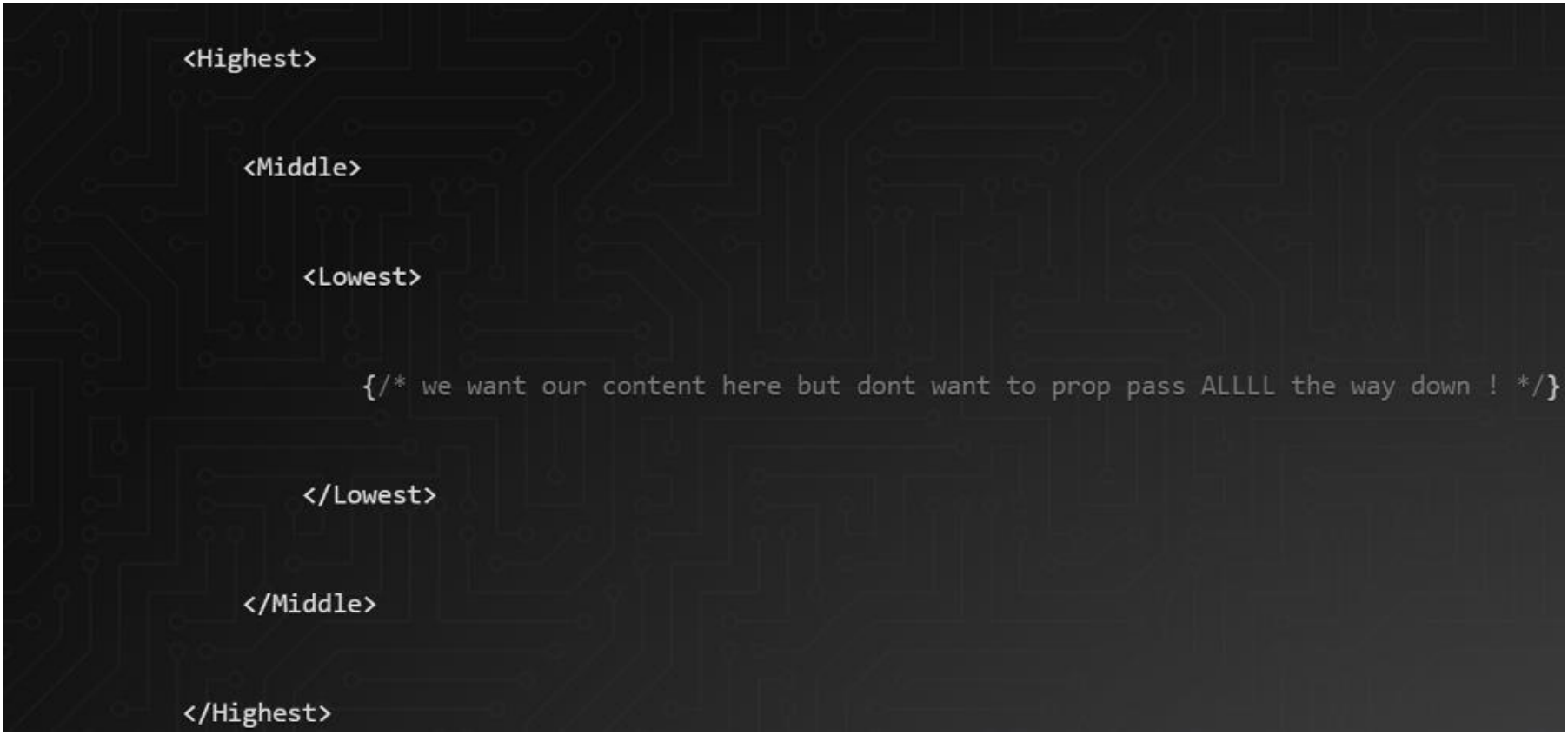
- Apollo: Easy to set up and use GraphQL client.
- Axios: Promise based HTTP client for the browser and node.js.
- Relay Modern - A JavaScript framework for building data-driven React applications.
- Request: Simplified HTTP request client.
- Superagent: A lightweight “isomorphic” library for AJAX requests.

# Data Fetching



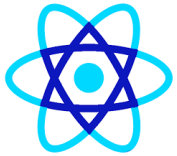
- npm install axios --save

# Context API



```
<Highest>

    <Middle>

        <Lowest>

            {/* we want our content here but dont want to prop pass ALLLL the way down ! */}

        </Lowest>

    </Middle>

</Highest>
```

# Redux

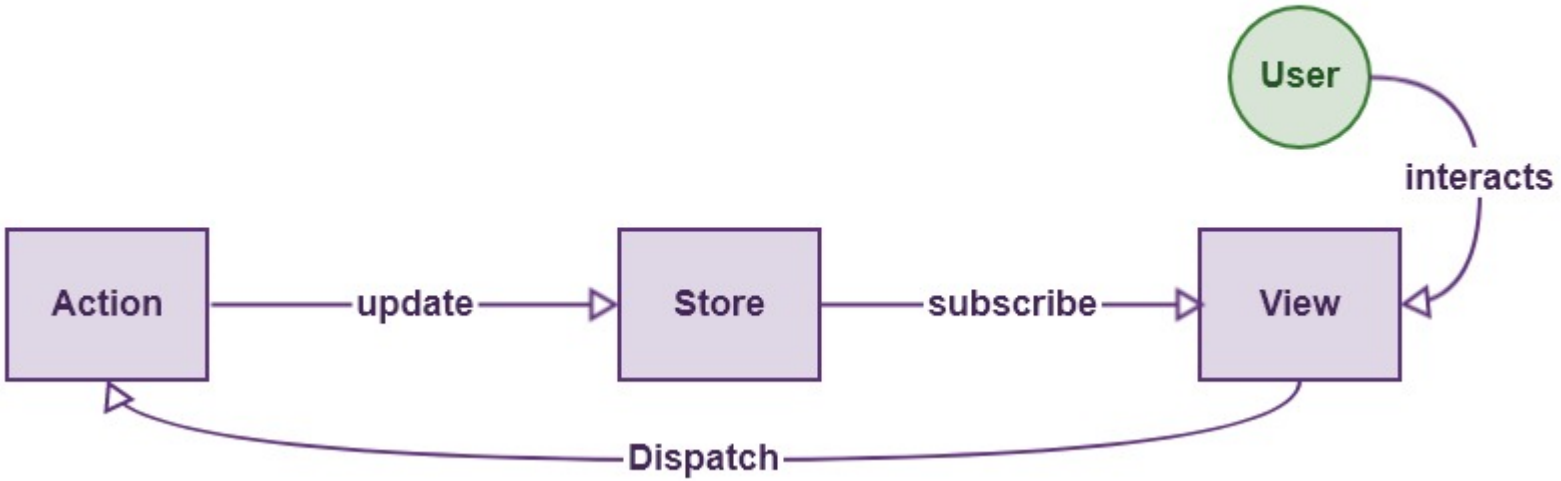


- Redux is the predictable state container for JavaScript applications. Redux is also following the Unidirectional flow, but it is entirely different from Flux. Flux has multiple stores.

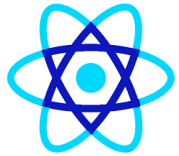
- Redux has a Single Store.

- Redux can not have multiple stores. The store is divided into various state objects. So all we need is to maintain the single store, or we can say the only source of truth.

# REDUX

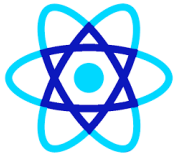

User
interacts
Action
update
Store
subscribe
View
Dispatch

# THREE PRINCIPLES OF REDUX



- Single source of truth.
- The state is read-only.
- Changes are made with pure functions.

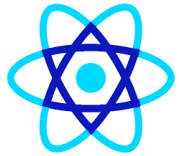
# Store



- A store is an object that brings them together. A store has the following responsibilities:

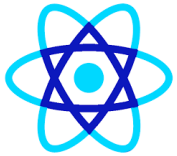
- Holds application state;
- Allows access to state via getState();
- Allows state to be updated via dispatch(action);
- Registers listeners via subscribe(listener);
- It handles unregistering of listeners via the function returned by subscribe(listener)

# Redux Project



- create-react-app postreactredux
- npm install redux react-redux –*save*
- npm install bootstrap –save
- *npm install classnames (to treat boostrap as css classes)*
- *Copy bootstrap.min.css to assets folder*
- npm install uuid *--save*
- Go to index.js
- import '../node_modules/bootstrap/dist/css/bootstrap.min.css';

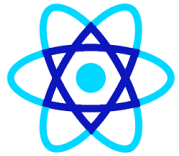
# Redux



- Dumb Components
- Smart Component(Containers)
- Dumb components render the data, and they did not care about any logic. They have nothing to do with a redux store.

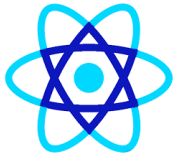
- Smart components are concern about logic and directly connected to the store.

# Redux with mongodb



- Go to the project folder and install dependencies using this command: **npm install**
- Start the mongodb server using this command: **mongod**
- Start the Node.js server using this command: **nodemon server/server**
- Start the Reacyt.js dev server using this command: **yarn start**

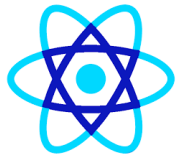
# Redux Async



- npm install redux-thunk –*save*

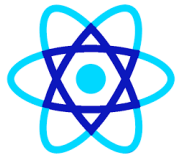
- *For middleware*
- npm install express body-parser mongoose –*save*
- npm install nodemon *--save-dev*

# React Native



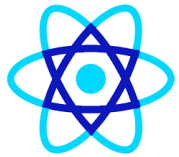
- React Native is like React, but it uses native components instead of web components as building blocks.

# React Native Features



- **React** − This is a Framework for building web and mobile apps using JavaScript.
- **Native** − You can use native components controlled by JavaScript.
- **Platforms** − React Native supports IOS and Android platform.

# Advantages

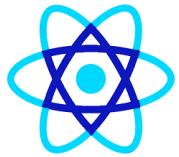


- JavaScript − You can use the existing JavaScript knowledge to build native mobile apps.

- Code sharing − You can share most of your code on different platforms.

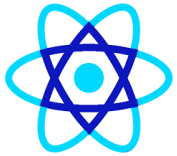
- Community − The community around React and React Native is large, and you will be able to find any answer you need.

# Limitations



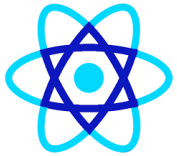
- Native Components − If you want to create native functionality which is not created yet, you will need to write some platform specific code.

# Installation



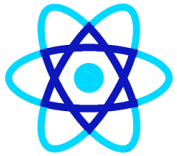
- npm install -g create-react-native-app
- create-react-native-app MyReactNative
- npm install -g react-native-cli
- react-native init rncreate

# Emulators



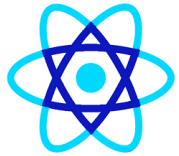
- Go to project folder
- Open in ios
- react-native run-ios
- react-native run-android
- react-native start (by mistake if you close window)

# Bundling



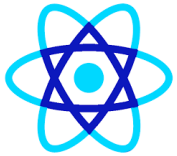
- (in project directory) mkdir android/app/src/main/assets
- react-native bundle --platform android --dev false --entry-file index.js --bundle-output android/app/src/main/assets/index.android.bundle --assets-dest android/app/src/main/res
- react-native run-android

# Enzymes



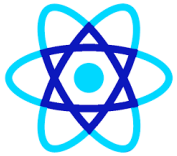
- Enzyme is a JavaScript Testing utility for React that makes it easier to assert, manipulate, and traverse your React Components' output.

# Installation



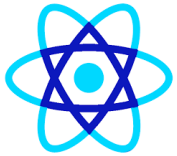
- npm i --save-dev enzyme enzyme-adapter-react-16
- npm install --save enzyme enzyme-adapter-react-16 react-test-renderer
- Npm install –save-dev chai
- npm install --save jest-enzyme
- Npm install –save sinon
- Npm test

# Jest  Unit Testing



- Jest is a JavaScript unit testing framework, used by Facebook to test services and React applications.
- CRA comes bundled with Jest; it does not need to be installed separately.
- Jest acts as a test runner, assertion library, and mocking library.
- Jest also provides Snapshot testing, the ability to create a rendered ‘snapshot’ of a component and compare it to a previously saved ‘snapshot’.
- The test will fail if the two do not match. Snapshots will be saved for you beside the test file that created them in an auto-generate __snapshots__ folder.

# Parcel Bundler

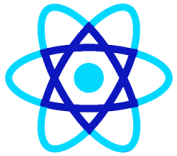


## Benchmarks

Based on a reasonably sized app, containing 1726 modules, 6.5M uncompressed. Built on a 2016 MacBook Pro with 4 physical CPUs.

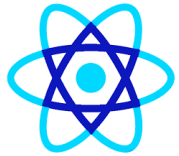
| Bundler | Time |
| --- | --- |
| browserify | 22.98s |
| webpack | 20.71s |
| **parcel** | **9.98s** |
| **parcel - with cache** | **2.64s** |

# Parcel Bundler



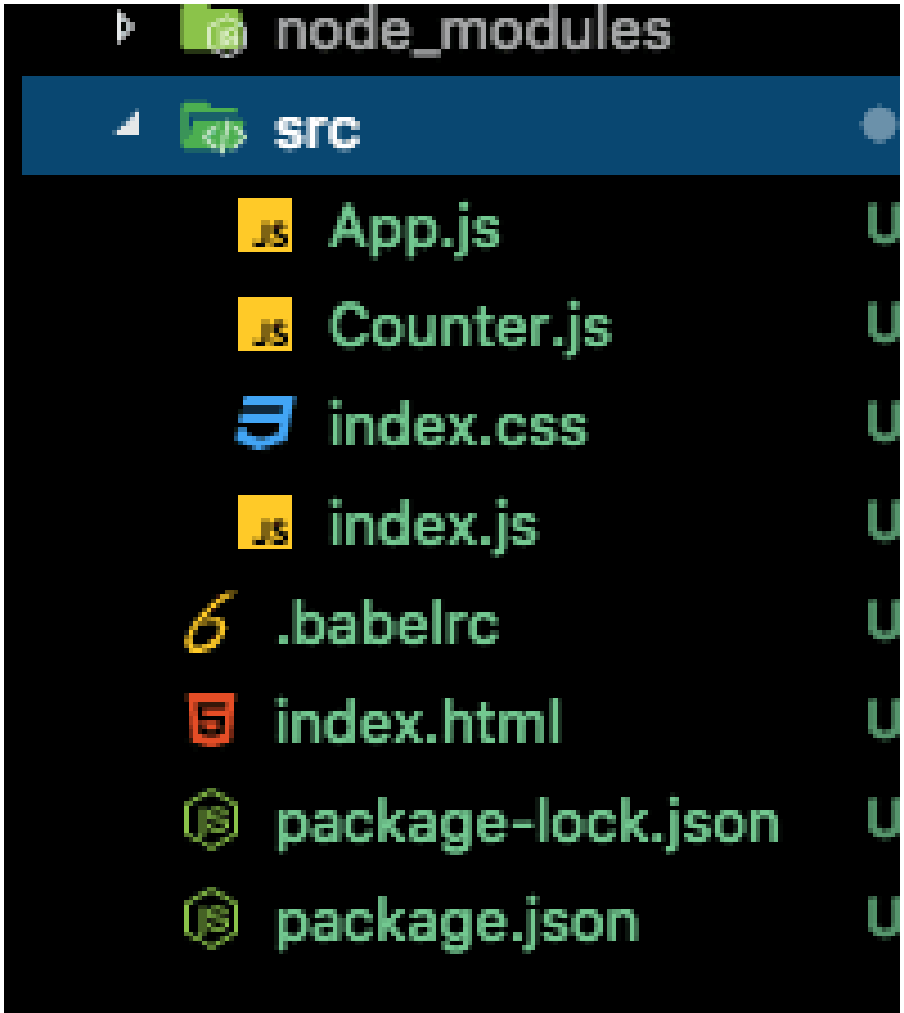
- Assets Bundling - Parcel has out of the box support for JS, CSS, HTML, file assets.
- Automatic transforms - All your code are automatically transformed using Babel, PostCSS, and PostHTML.
- Code Splitting - Parcel allows you to split your output bundle by using the dynamic import() syntax.
- Hot module replacement (HMR) - Parcel automatically updates modules in the browser as you make changes during development, no configuration needed.
- Error Logging - Parcel prints syntax highlighted code frames when it encounters errors to help you pinpoint the problem.

# Parcel Bundler

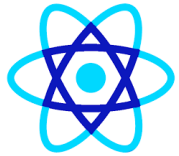


- mkdir react-parcel
- cd react-parcel
- npm init -y
- npm i --save-dev parcel-bundler
- npm i --save react react-dom

# Parcel Bundler

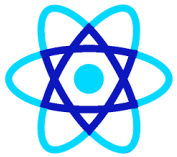


# E2e testing



- npm install nightwatch
- npm install chromedriver --save-dev
- npm install geckodriver --save-dev
- Keep web driver in current folder

# React Production Deployment



fiservpune2019 [F:\fiservpune2019] - WebStorm

Administrator: Node.js command prompt - serve -s build

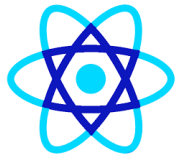
```
F:\fiservpune2019\bankingapp>npm install netlify-cli -g
npm WARN deprecated flatten@1.0.2: I wrote this module a very long time ago; you should u
se something else.
C:\Users\Balasubramaniam\AppData\Roaming\npm\ntl -> C:\Users\Balasubramaniam\AppData\Roam
ing\npm\node_modules\netlify-cli\bin\run
C:\Users\Balasubramaniam\AppData\Roaming\npm\netlify -> C:\Users\Balasubramaniam\AppData\
Roaming\npm\node_modules\netlify-cli\bin\run

> core-js@2.6.10 postinstall C:\Users\Balasubramaniam\AppData\Roaming\npm\node_modules\ne
tlify-cli\node_modules\core-js
> node postinstall || echo "ignore"

Thank you for using core-js ( https://github.com/zloirock/core-js ) for polyfilling JavaS
cript standard library!

The project needs your help! Please consider supporting of core-js on Open Collective or
Patreon:
> https://opencollective.com/core-js
> https://www.patreon.com/zloirock
```

Microsoft Edge

Type here to search

ENG 22:03 16/10/2019

# React Production Deployment



Reactjs_v1.pptx - PowerPoint

Administrator: Node.js command prompt - serve -s build

```
es/netlify-cli/node_modules/node-fetch/lib/index.js:1455:11)

F:\fiservpune2019\bankingapp>netlify deploy
Please provide a publish directory (e.g. "public" or "dist" or "."):
F:\fiservpune2019\bankingapp
? Publish directory F:\fiservpune2019\bankingapp\build
Deploy path: F:\fiservpune2019\bankingapp\build
Deploying to draft URL...
√ Finished hashing 17 files
√ CDN requesting 0 files
√ Finished uploading 0 assets
√ Draft deploy is live!

Logs:              https://app.netlify.com/sites/banking-app-veb/deploys/5da743c6f4ee609bdb2
b7deb
Live Draft URL: https://5da743c6f4ee609bdb2b7deb--banking-app-veb.netlify.com

If everything looks good on your draft URL, take it live with the --prod flag.
netlify deploy --prod
```

Type here to search

ENG 22:05 16/10/2019

# React Production Deployment

## Yarn global add serve

Reactjs_v1.pptx - PowerPoint

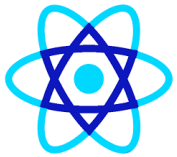
Administrator: Node.js command prompt - serve -s build

```
F:\fiservpune2019\react-voting-app-master>serve -s build

   Serving!

   - Local:            http://localhost:5000
   - On Your Network:  http://172.29.53.177:5000

   Copied local address to clipboard!
```

151 152 153 154 155

Type here to search

ENG 22:05 16/10/2019



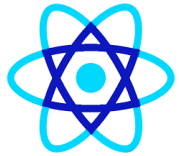
lite-server

```
 UI External: http://localhost:3003
 ---------------------------------------
[Browsersync] Serving files from: ./
[Browsersync] Watching files...
19.11.12 15:05:22 404 GET /index.html
^CTerminate batch job (Y/N)? y

F:\daimlerreactoct2019\blog2019>lite-server --baseDir=build/
Did not detect a `bs-config.json` or `bs-config.js` override file. Using lite-server defaults...
** browser-sync config **
{
  injectChanges: false,
  files: [ './**/*.{html,htm,css,js}' ],
  watchOptions: { ignored: 'node_modules' },
  server: { baseDir: 'build/', middleware: [ [Function], [Function] ] }
}
[Browsersync] Access URLs:
 ---------------------------------------
       Local: http://localhost:3002
    External: http://172.29.53.177:3002
 ---------------------------------------
          UI: http://localhost:3003
 UI External: http://localhost:3003
 ---------------------------------------
[Browsersync] Serving files from: build/
[Browsersync] Watching files...
19.11.12 15:05:53 200 GET /index.html
```

Type here to search

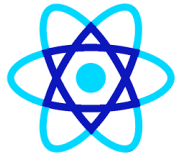
ENG 15:07 12/11/2019

# React Optimization



- **Using Immutable Data Structures**
- **Function/Stateless Components and React.PureComponent**
- **Multiple Chunk Files**
- **Using Production Mode Flag in Webpack**
- **Dependency optimization**
- Use React.Fragments to Avoid Additional HTML Element Wrappers

# React Optimization



- **Avoid Inline Function Definition in the Render Function.**
- Throttling and Debouncing Event Action in JavaScript
- **Spreading props on DOM elements**
- **CSS Animations Instead of JS Animations**

# Questions

# Module Summary

- Reactjs
- JSX, Props and State
- Events
- Component Interaction