ReactJS Training Course (Session 1)

TUAN MAI CHUNG / NashTech

Sep/2020

Agenda

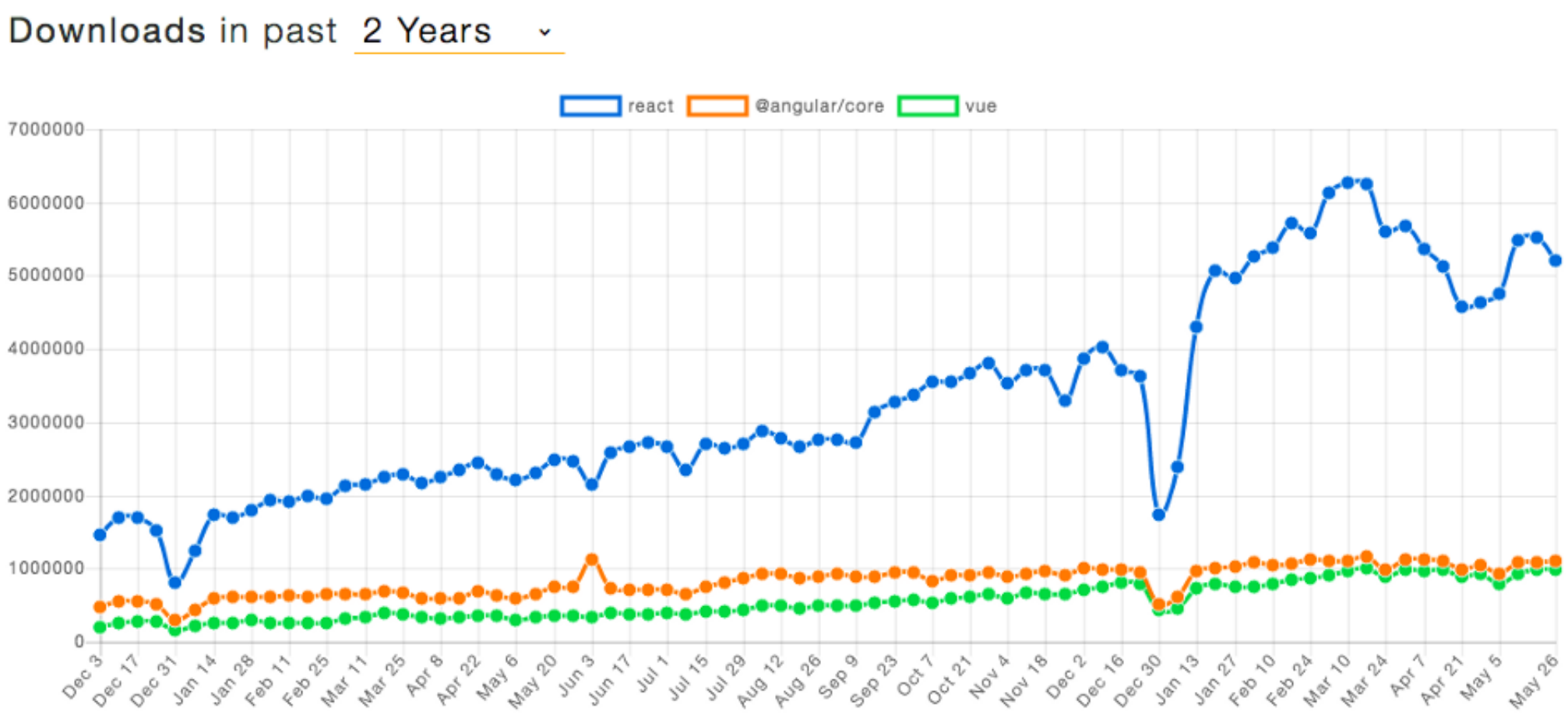
1. REACT INTRODUCTION
2. CRA & APPLICATION STRUCTURE
3. COMPONENT & JSX

React Introduction

- 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 not a framework. It is just a library developed by Facebook to solve some problems that we were facing earlier.

Frontend trends



Who is using React?

React-based Frameworks

- **NextJS:**
  https://nextjs.org/

NEXT.JS

- **GatsbyJS**
  https://www.gatsbyjs.com/

Gatsby

ReactJS

Why ReactJS?

- Simplicity
- Easy to learn
- Native Approach
- Performance
- Testability

Create-react-app (CRA)

Create-react-app (CRA)

- Requirement:
  - NodeJS (https://nodejs.org/en/download/)
  - IDE (Visual Studio Code, Sublime,…)

Create-react-app (CRA)

```
npx create-react-app my-app
cd my-app
npm install reactstrap
npm start
```

Folder Structure in React

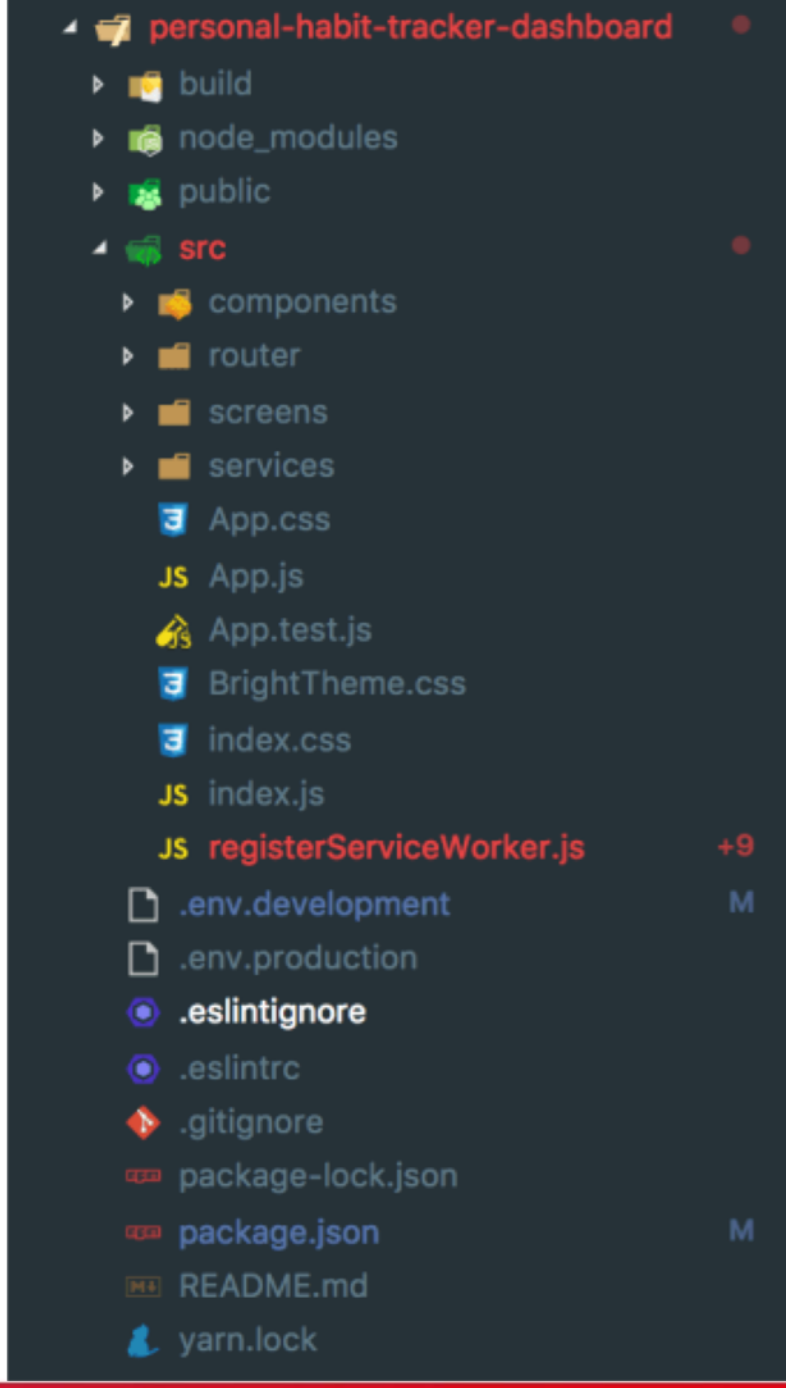
2 structures ReactJS docs recommended:

- Group by features or routers:

```
common/
  Avatar.js
  Avatar.css
  APIUtils.js
  APIUtils.test.js
feed/
  index.js
  Feed.js
  Feed.css
  FeedStory.js
  FeedStory.test.js
  FeedAPI.js
profile/
  index.js
  Profile.js
  ProfileHeader.js
  ProfileHeader.css
  ProfileAPI.js
```

- Group by file type:

```
api/
  APIUtils.js
  APIUtils.test.js
  ProfileAPI.js
  UserAPI.js
components/
  Avatar.js
  Avatar.css
  Feed.js
  Feed.css
  FeedStory.js
  FeedStory.test.js
  Profile.js
  ProfileHeader.js
  ProfileHeader.css
```



Boilerplate vs CRA tool vs Build from scratch

- Boilerplate

  Boilerplate code means a piece of code which can be used over and over again.

- CRA tool

  Create React App is a tool built by developers at Facebook to help you build React applications. It saves you from time-consuming setup and configuration.

- Build from scratch

  If you do something from scratch, you do it without making use of anything that has been done before.

Component & JSX

Hello world

- Index.js

```
ReactDOM.render(
    <h1>Hello,world!</h1>,
    document.getElementById('root')
);
```

Introducing JSX

- A syntax extension that bind HTML element to a JS variable

```
const element = <h1>Hello, world!</h1>;
```

JSX in expression

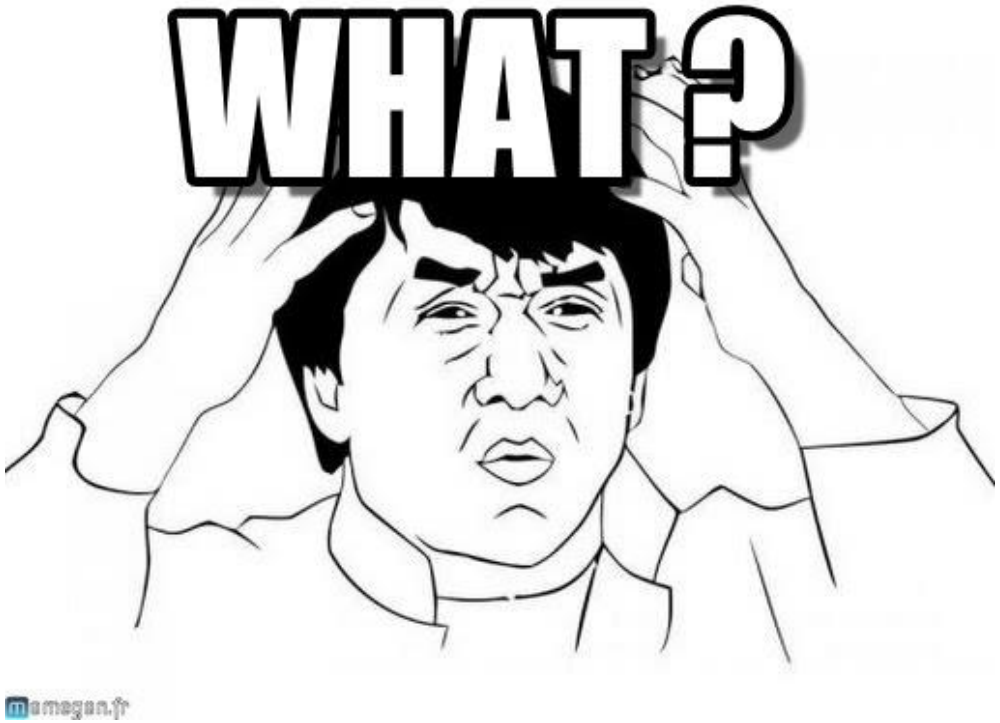
```
const name = 'Josh Perez';
const element = <h1>Hello, {name}</h1>;

ReactDOM.render(
  element,
  document.getElementById('root')
);
```

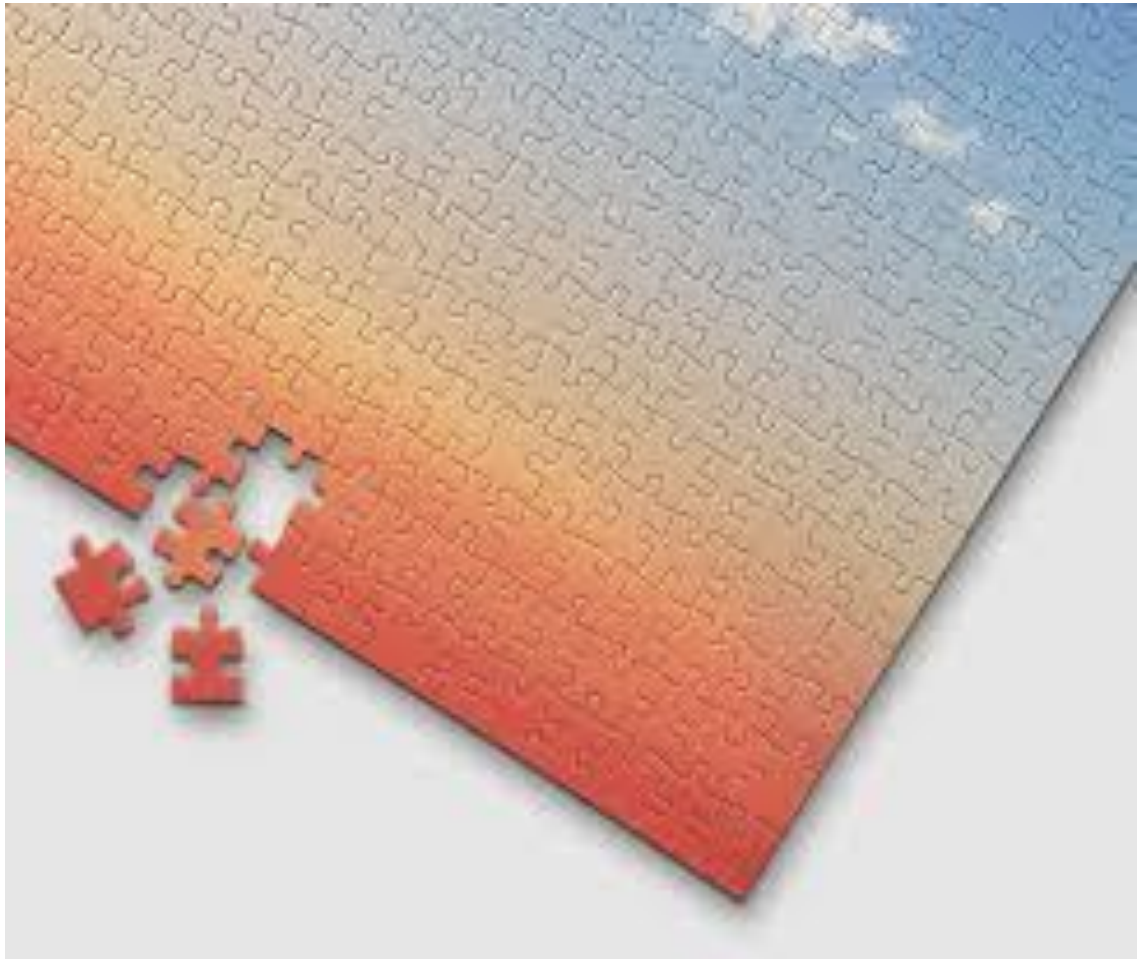
JSX in expression

```
function formatName(user) {
  return user.firstName + ' ' + user.lastName;
}

const user = {
  firstName: 'Harper',
  lastName: 'Perez'
};

const element = (
  <h1>
    Hello, {formatName(user)}!
  </h1>
);
```

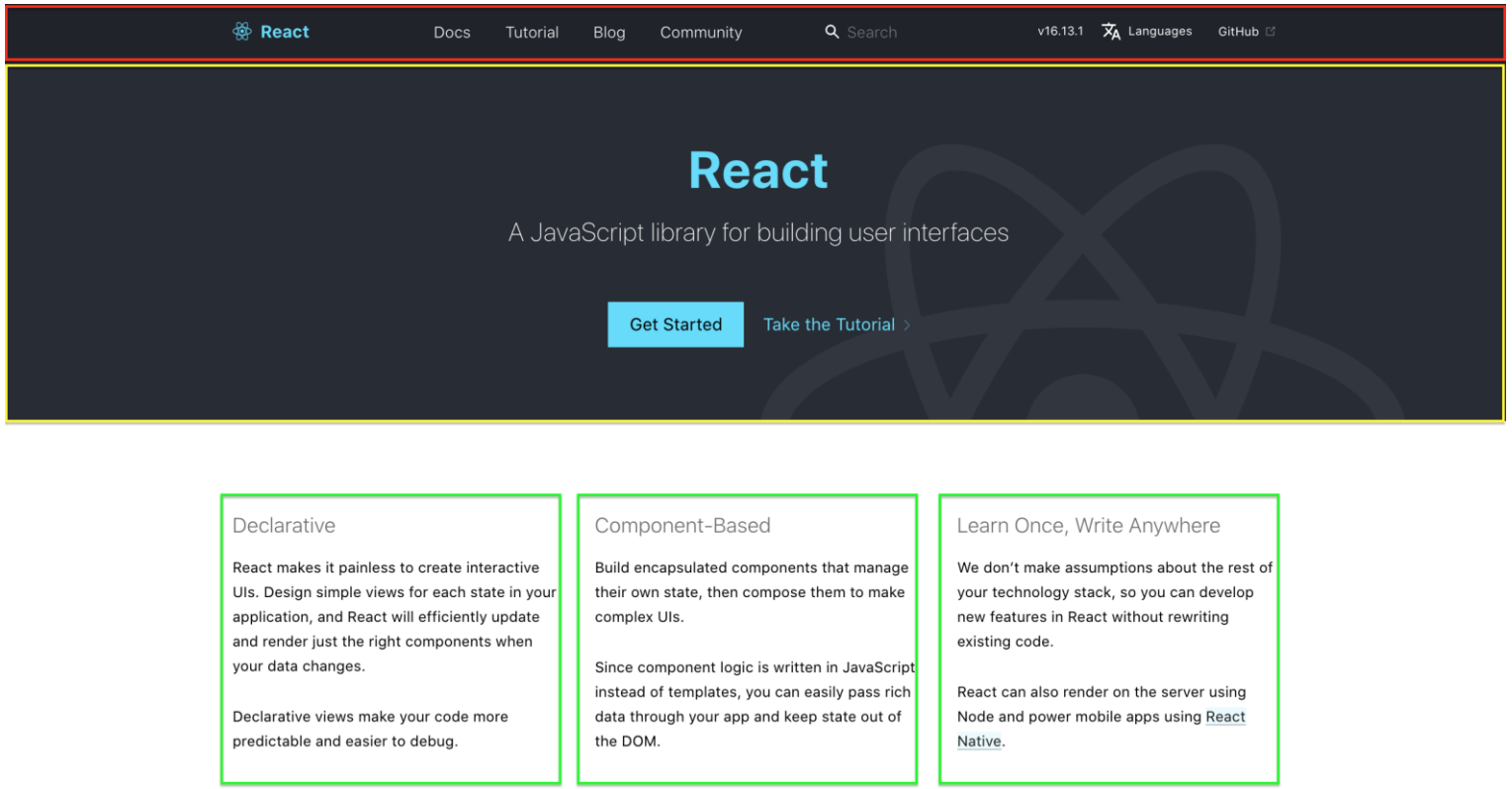
What is web component?

- Basic unit of a website
- A set of HTML elements, which:
  - Work individually
  - Reusable


WHAT ?
memegen.fr



Example of web component



Function vs Class component

```jsx
function Welcome(props) {
  return <h1>Hello, {props.name}</h1>;
}
```

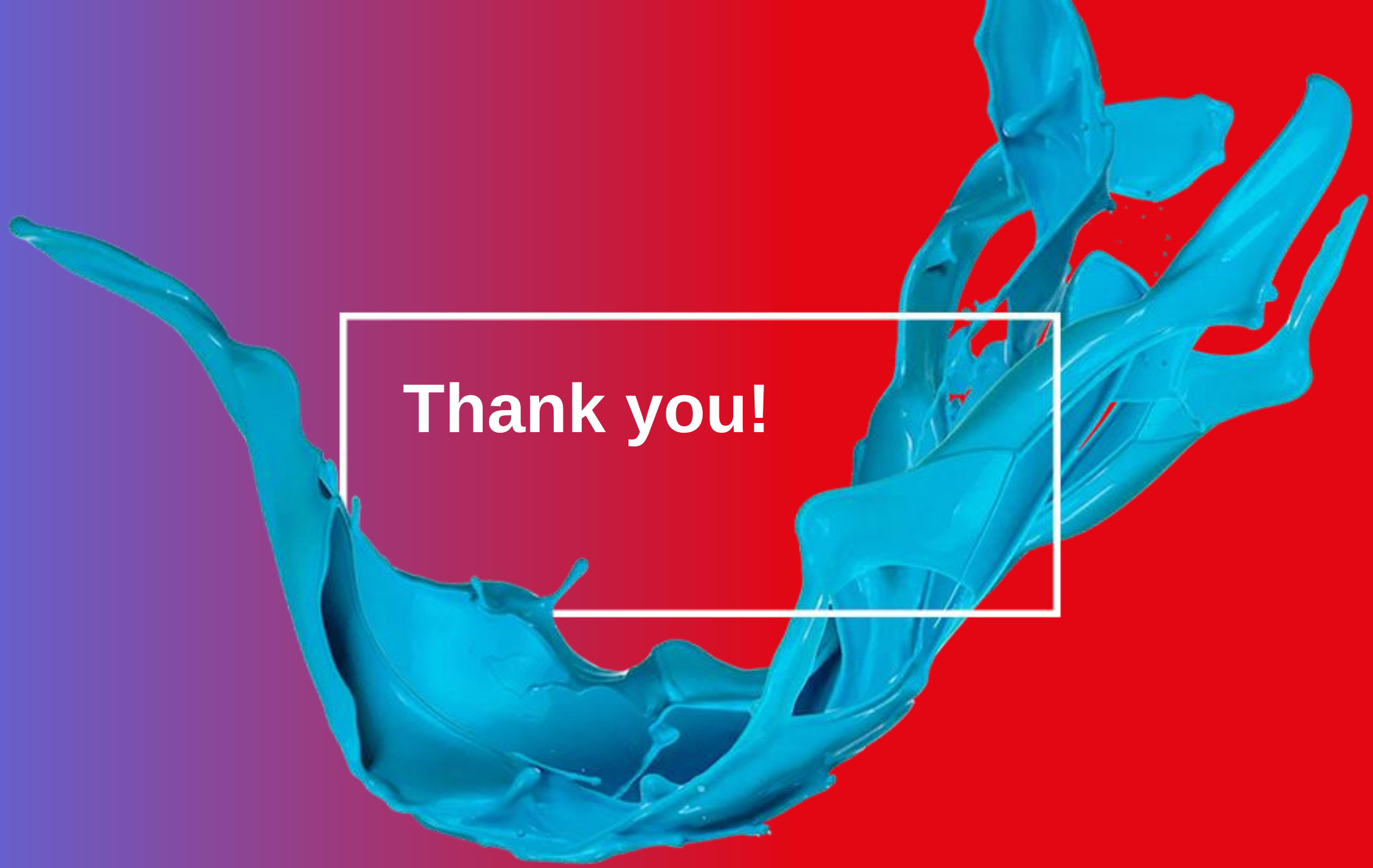
```jsx
class Welcome extends React.Component {
  render() {
    return <h1>Hello, {this.props.name}</h1>;
  }
}
```

React Component

```jsx
import React from 'react';
import logo from './logo.svg';
import './App.css';

function App() {
  return (
    <div className="App">
      <header className="App-header">
        <img src={logo} className="App-logo" alt="logo" />
        <p>
          Edit <code>src/App.js</code> and save to reload.
        </p>
        <a
          className="App-link"
          href="https://reactjs.org"
          target="_blank"
          rel="noopener noreferrer"
        >
          Learn React
        </a>
      </header>
    </div>
  );
}

export default App;
```

Exercise

1. Use `**create-react-app**` to create application **react-fundamental**
2. Install `**bootstrap**` and `**reactstrap**` from **npm**
3. Add `*productItems.json*` and `*images*` to **public** folder
4. Create `*components*` *folder and* add 2 components `**Header**` and `**TopBanner**`
5. Use logo and slide images from *images* folder in 2 previous components



Thank you!

Presentational vs Container Component

Presentational

- Are concerned with *how things look*.
- Have no dependencies on the rest of the app
- Don’t specify how the data is loaded or mutated
- Are written as *functional components*

Container

- Are concerned with *how things work*
- Provide the data and behavior to presentational or other container components
- Are often stateful
- Are written as *class components*

Presentational vs Container Component