React JS

U of A Level Up Hackathon
Winter 2019

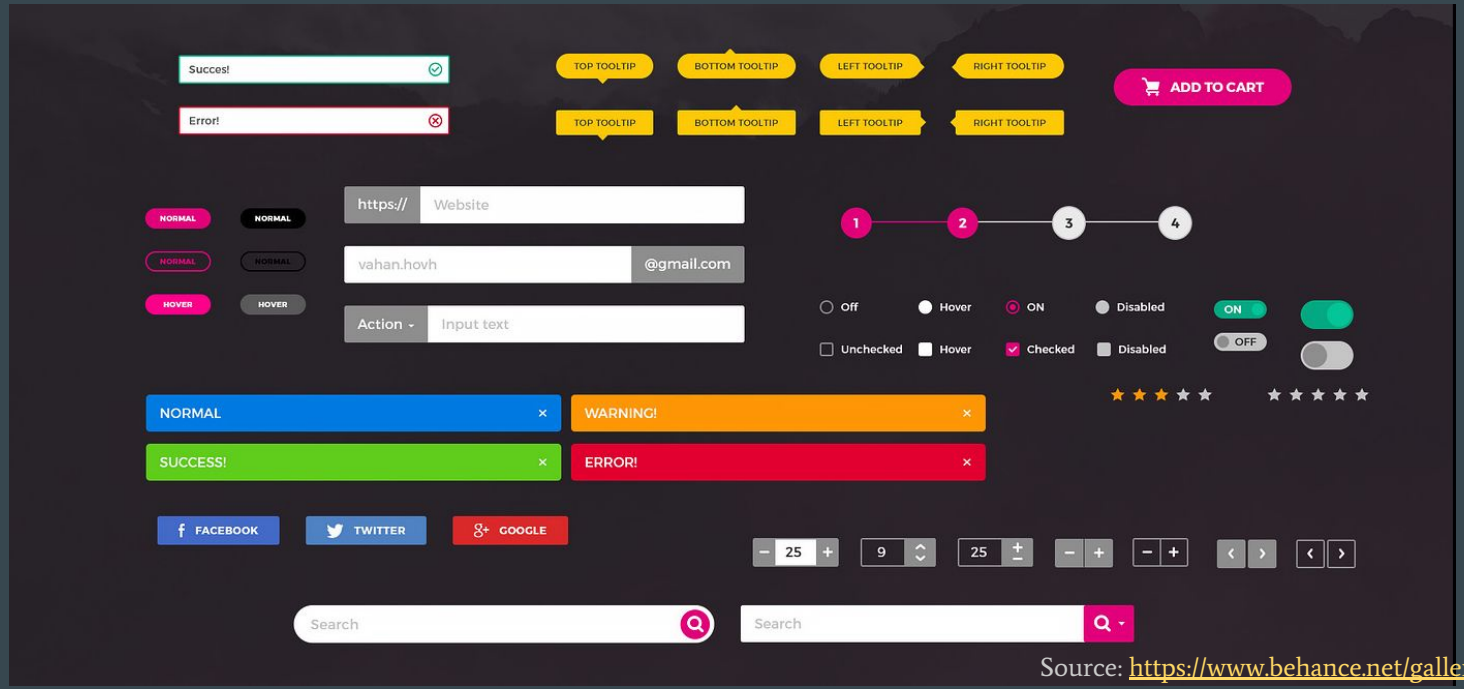
What is React? In one short sentence:

React is a JavaScript library used for building UI components for web applications.

Why Components?

Users love consistent UI across application, throughout an organization. It also give developers great benefits to separate rendering concerns, allowing us to compose their web apps with ease.

https://reactjs.org/docs/composition-vs-inheritance.html



Source: https://www.behance.net/gallery/31643855/August-UI-kit

Credit: Vahan Hovhannisyan

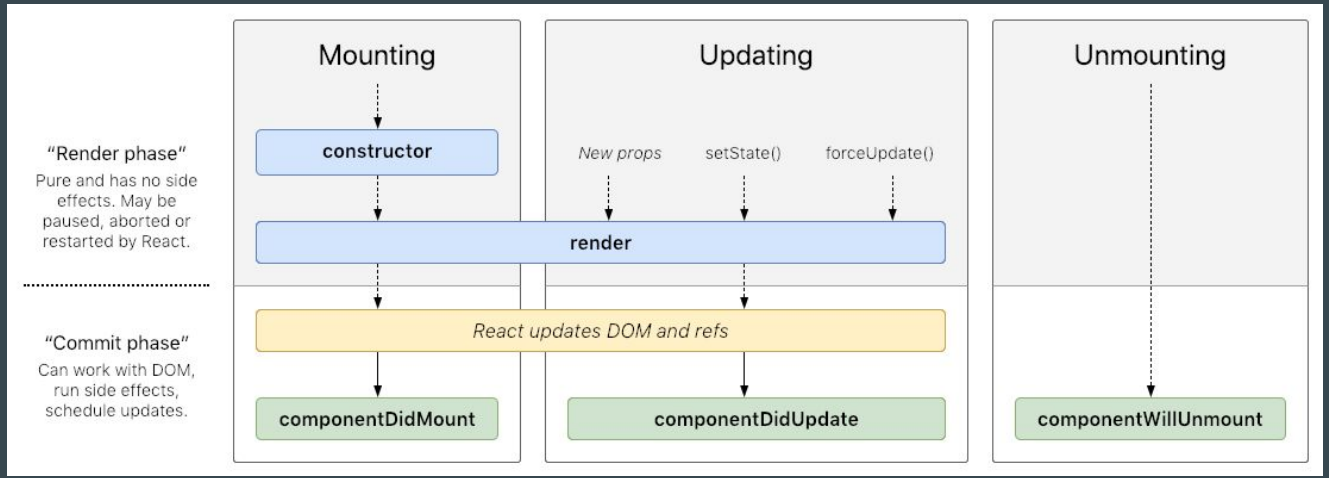
Internals of a React component:

```
class MyComponent extends React.Component {
  state: {
    someInformation: 123,
  };

  componentDidMount() {
    fetch("example.com/some_data.json")
      .then(responseData => {
        this.setState({ someInformation: responseData.newInformation });
      });
  }

  render() {
    return (
      <div>
        {this.state.someInformation}
      </div>
    );
  }
}
```

React component lifecycle (simplified):



http://projects.wojtekmaj.pl/react-lifecycle-methods-diagram/

React component lifecycle (continued):

`constructor`

Like OOP languages, the constructor method allow us to setup the component foundation prior to mounting. Constructor is the only place where you should assign `this.state` directly. We can also set up event bindings here.

`componentDidMount`

Most commonly used event hook. Typically used to fetching external data after the component is fully mounted in the DOM and ready for a re-render.

`componentDidUpdate`

Can be used to compare component states, to determine certain condition are met for render or data fetching.

`componentWillUnmount`

Invoked before the component is unmounted and fully removed. This method allow us to perform cleanup tasks.

https://reactjs.org/docs/react-component.html

Commonly used React terminologies:

State:

A description of data and its condition (the state) of a component. It is managed and used by the component itself.

Props:

Information being passed from a parent component. This is a read-only property and cannot be manipulated by the consumer.

Containers:

An unofficial term often used by the community, to distinguish a component that holds many smaller components but have very little work to do on its own.

Other Essential Information:

Create-react-app

https://facebook.github.io/create-react-app/docs/getting-started

JSX

https://reactjs.org/docs/introducing-jsx.html

React debug tool

https://chrome.google.com/webstore/detail/react-developer-tools/fmkadmapgofadopljbjfkapdkoienihi?hl=en

Thinking in React

In a nutshell:

A technique to componentize a React application by dividing rendering responsibility, planning data flows, and maintaining code reusability.

Let's take a look at the class project:

http://kingston-test-weather-app.s3-website-us-east-1.amazonaws.com

Let’s code!

1. Install Node environment (includes npm): https://nodejs.org/en/download/
2. Check node and npm versions:

   ```
   node -v
   npm -v
   ```
3. Download project starter file using git:

   ```
   git clone git@github.com:kingstonfung/starter-react-weather-app.git
   ```

   Or, direct download (.zip file): `https://bit.ly/2019-uofa-react-app`
4. Install 'create-react-app' node package globally (global install allow easier effort to spin up additional projects later):

   ```
   npm install -g create-react-app
   ```
5. Create the "weather" React app using Create React App, and then cd into the "weather" folder:

   ```
   create-react-app weather
   cd weather
   ```
6. Install 'node-sass' node package inside the "weather" project folder:

   ```
   npm install node-sass
   ```
7. Open up your favorite code editor.
   a. Alternatively, we can use VSCode to get going: https://code.visualstudio.com/download
8. Run the preinstalled project:

   ```
   npm run start
   ```

More Advanced Topics:

Higher Order Component:
https://reactjs.org/docs/higher-order-components.html

Routing:
https://reacttraining.com/react-router/web/guides/quick-start

Redux:
https://redux.js.org/basics/usage-with-react

Unit Testing with Jest:
https://jestjs.io/docs/en/getting-started.html

React Native:
https://facebook.github.io/react-native/docs/getting-started.html

So... is React the best framework?

It depends!

It can be the best tool for some application projects, in some organization, at a certain time.

Always keep an open mind to learn more tools, then you can evaluate what is the best tool for the job.

So what is the best front end framework?

One that can keep the user interface in sync the application state while delivering business value and great user experience to its users.

They can be: React, Vue, Angular, Ember, or plain vanilla JavaScript!

In closing, here are few pro tips when it comes to front end development:

1. Ship small changes, frequently!
2. Test, Test, Test, and more Tests!
3. Often ask yourself: “What happens if...?”
4. Think of your users: When, How, and With What they will use your product?
5. Don’t be afraid to make mistakes! Fail fast so you can learn even faster.
6. Keep an open mind! JavaScript and community is a very dynamic environment. Always be willing to learn.
7. Nothing is perfect. Shipping out “good enough” code is often better than “perfect” code.
8. Write (front end) code for humans to understand. The machine can optimize & handle the rest.

Thank You! And happy hacking!

Questions? Feel free to email me at: kingstonfung@gmail.com
I will also send out classroom training information to anyone interested.

Learn more about Trust Science: www.trustscience.com

Other inquiries: info@trustscience.com