Please install Yarn Globally...

```
npm install -g yarn
```

Props, Lists, and Stateful Components

Web Development

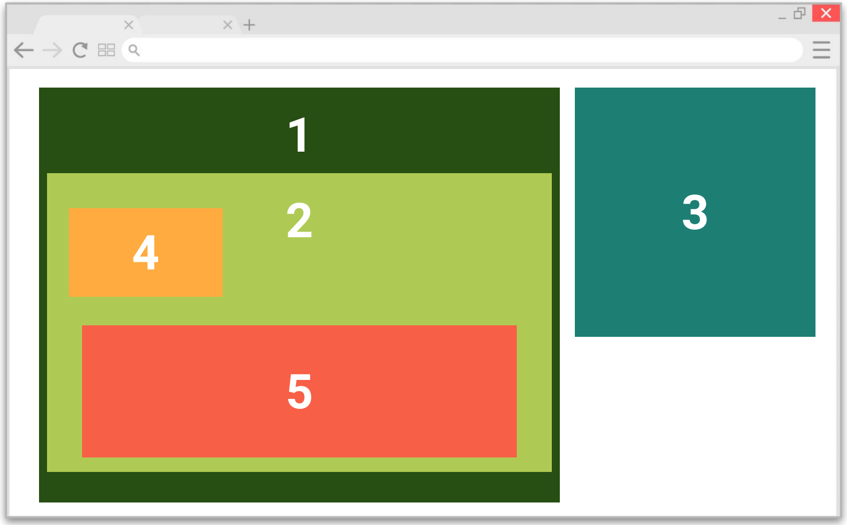
Lesson 19.3

Components

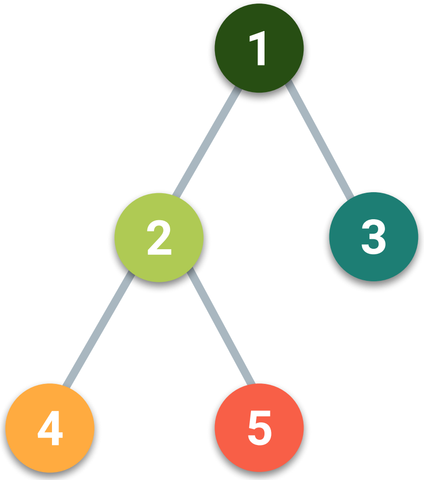
Components

Components are JavaScript functions that return a part of an application's UI. Think of them as the building blocks of a React application.

Browser



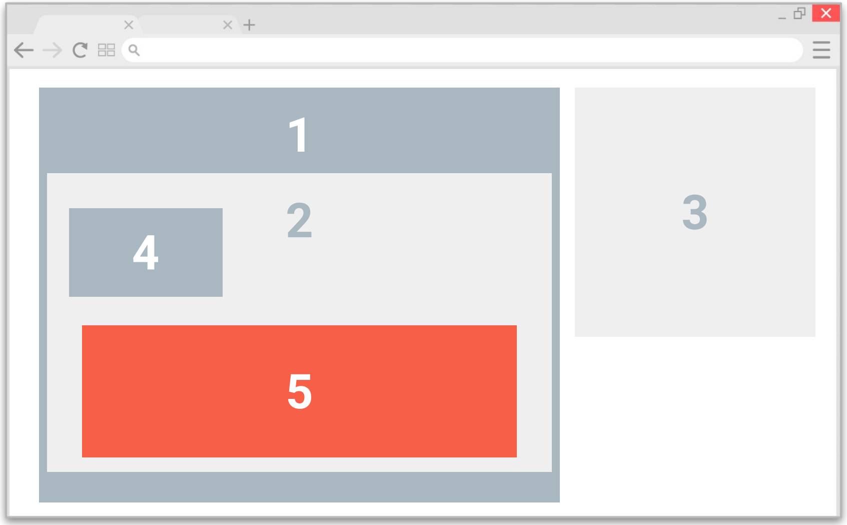
UI Tree



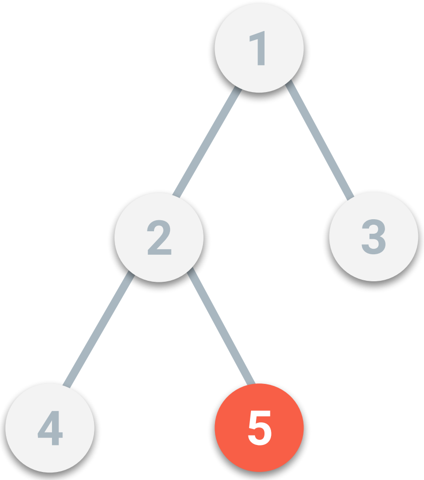
Components

Components let you split the UI into independent, reusable pieces and to consider each piece in isolation.

Browser

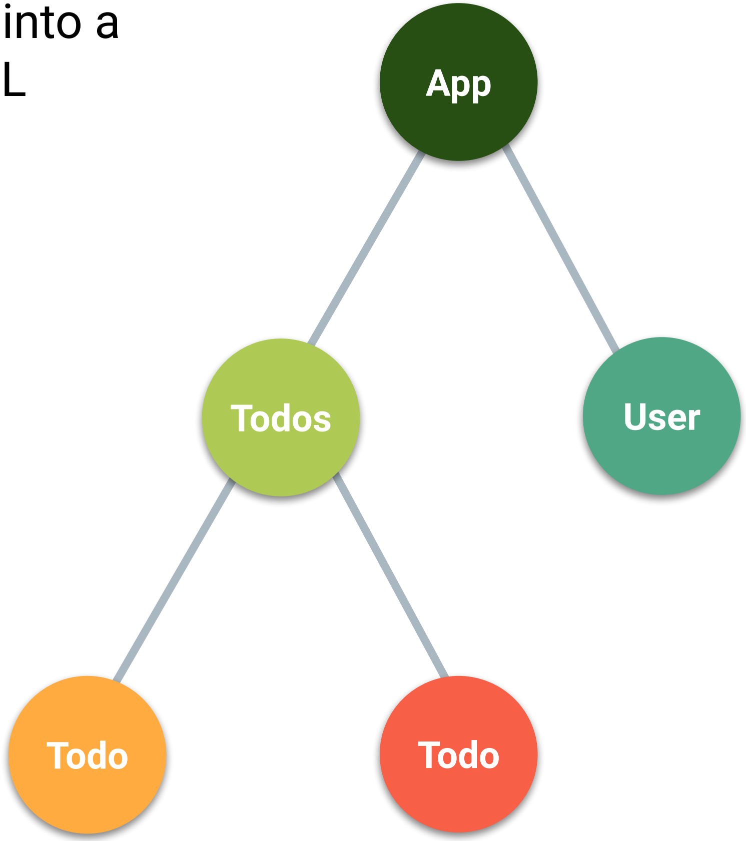


UI Tree



Components

Components are organized into a tree structure, just like HTML elements in the DOM.



Components

By separating elements into components:

01

Layout and logic are bundled together in a self-contained package.

02

Components can be reused throughout the application without needing to be re-coded.

03

Testing is easier (e.g., having one reusable component means only one UI element needs to be tested).

JSX

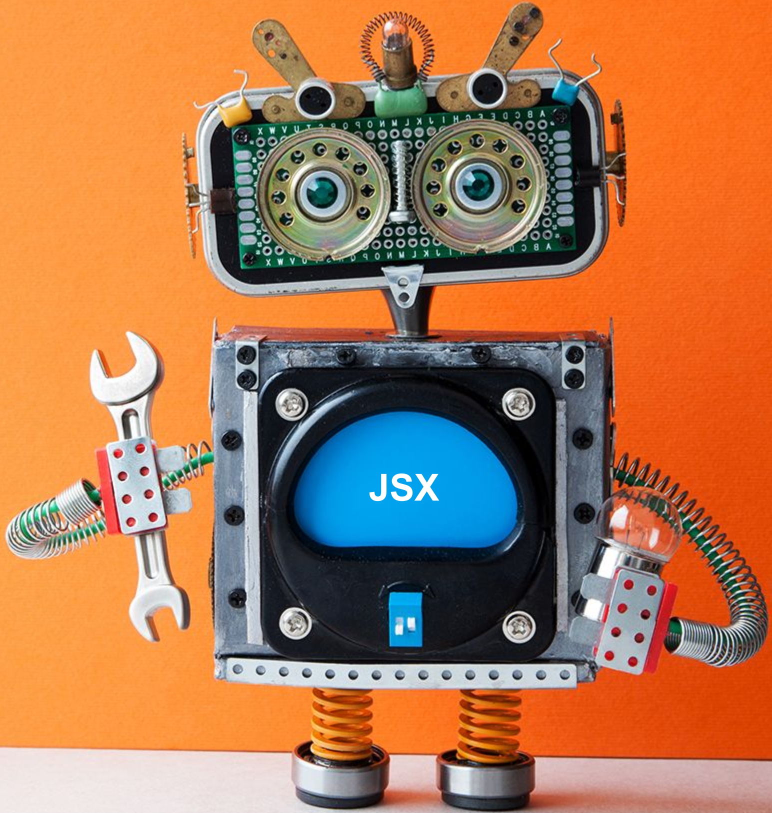
JSX is a preprocessor step that adds XML syntax to JavaScript.

JSX makes React more elegant.

JSX

JSX looks like HTML, but there are some differences you need to know about.

React’s documentation covers the gotchas to look out for.



JSX Curly Braces

Use curly braces `{ }` in JSX code to embed JavaScript expressions.

Evaluating a JavaScript variable:

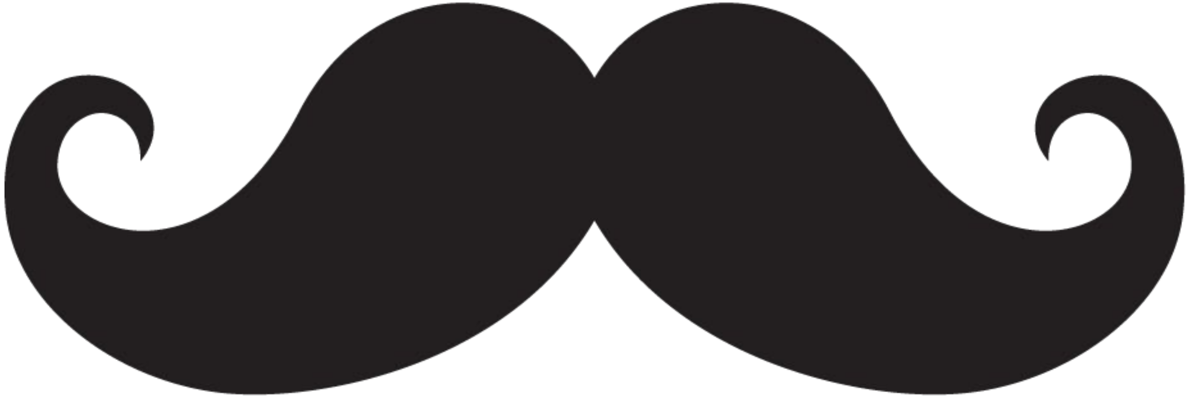
```
const yellowStyle={color: 'yellow'}
<Star style={yellowStyle} />
```

Which is same as:

```
<Star style={{color: 'yellow'}} />
```

JSX Curly Braces

JSX curly braces { } can be compared to the double curly braces {{ }} in Handlebars.



Props

Props are like function arguments that you can pass to components for them to use.

Props

01

Can be passed to a component by attaching attributes to the JSX that render the component, similar to how you attach attributes to HTML elements.

02

Considered immutable (something you can’t change).

03

Can be used to change the default behavior of a component.

ReactDOM.render

ReactDOM.render

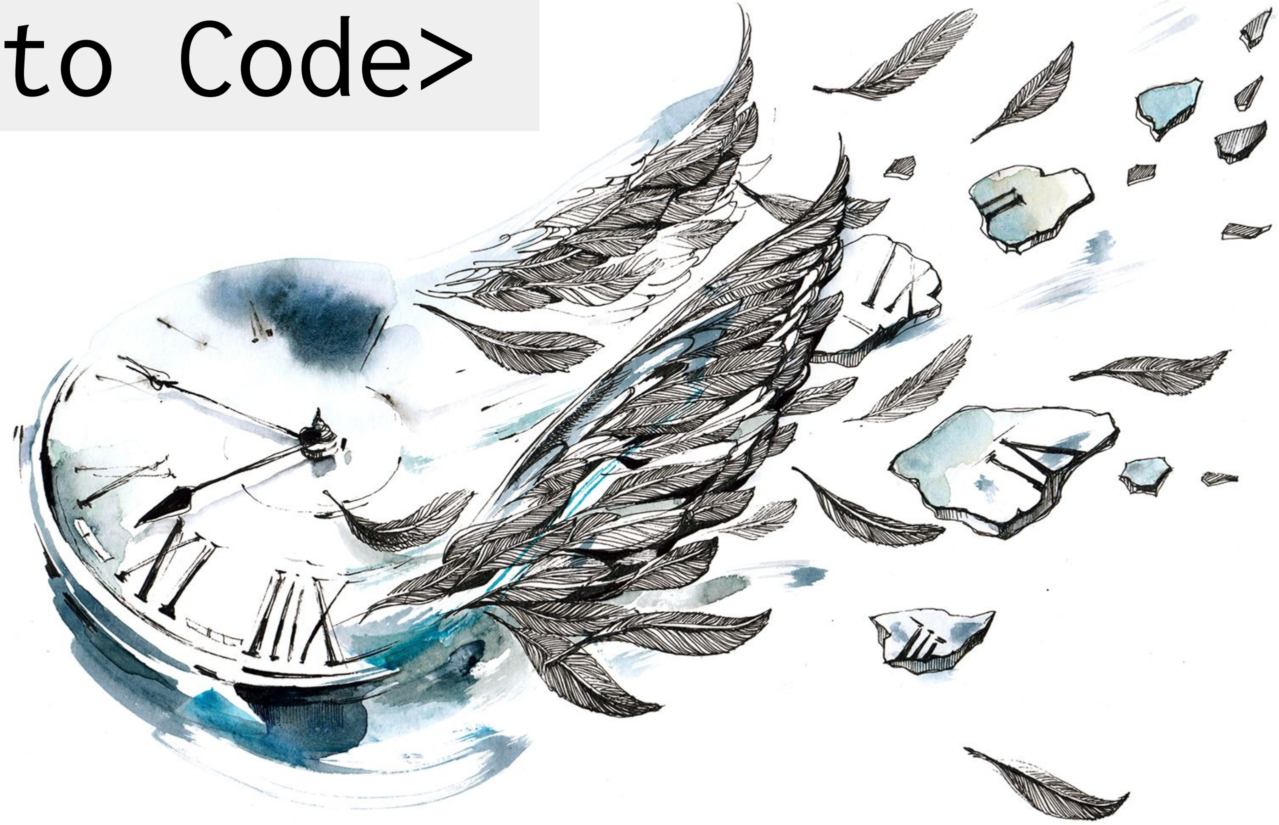
01 ReactDOM

ReactDOM is a library separate from React with methods for working with the DOM.

02 ReactDOM.render()

`ReactDOM.render()` renders a single root component to the Virtual DOM, and then the real HTML DOM. We typically call this method only once per React application.

<Time to Code>



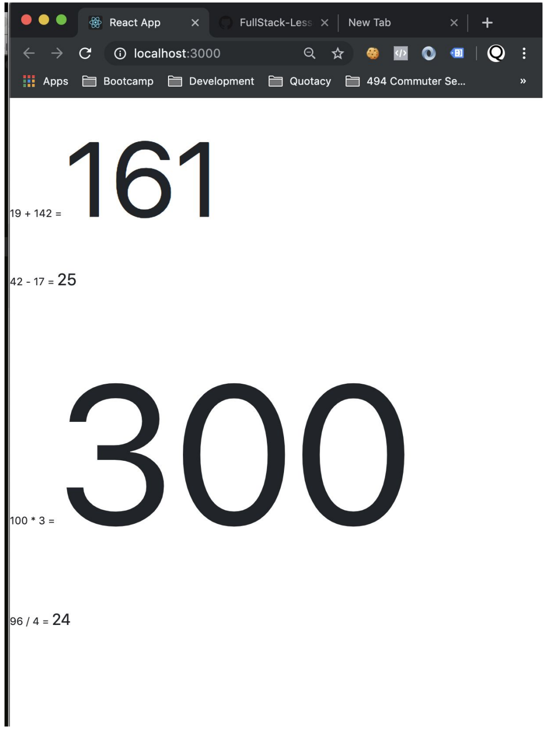
Class Objectives

- To deepen understanding of passing props between React components.
- To gain a firm understanding of the concept of child-parent relationships in React
- To be able to programmatically render components from an array of data.
- To introduce the concept of class components and component state.
- To complete the Mongo Checkpoint.

Calculator Props (10 mins)

write a component that accepts props, performs arithmetic and renders the result.

- Replace your React application's `src` folder with Unsolved/src. Stop the dev server if it is already running. Start the app in dev mode by running `npm start`.
- Open your web browser to localhost:3000. Take a moment to observe the rendered app.
- Open `src/components/Calculator.js`in your editor and take a moment to study the application's code.
- Write a component named "Math" and render one Math component in the place of each "?" mark.
- The Math component should perform some arithmetic using 3 props:
  - `num1` - a number
  - `operator` - a string representing an arithmetic operator, e.g. "+", "-", "*", "/"
  - `num2` - a number
- The Math component should render the result of problem in a `span` tag. e.g. 19 + 341 = 360

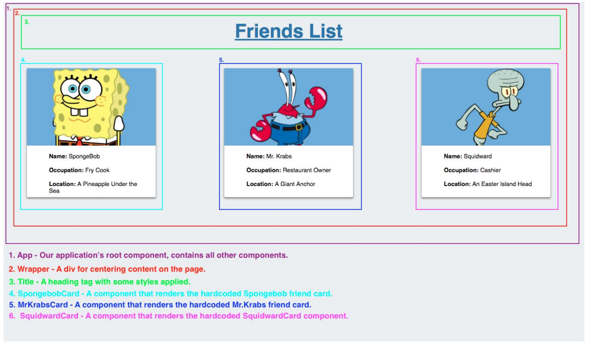


```
19 + 142 = 161

42 - 17 = 25

100 * 3 = 300

96 / 4 = 24
```

Props Review

Make an existing React application more DRY through the use of reusable components and props

- Modify the `FriendCard` component so that it accepts and renders all of the passed props in place of the currently hard coded values. Once complete, check your browser to make sure the first `FriendCard` is still being properly rendered.
- Inside of `src/App.js`, render another `FriendCard` component for the second and third piece of friend data. Pass down the appropriate JSON data for each as props. If successful, you should see each friend being rendered to the browser, utilizing the same same `FriendCard` component three times.



Friends List

Name: SpongeBob
Occupation: Fry Cook
Location: A Pineapple Under the Sea

Name: Mr. Krabs
Occupation: Restaurant Owner
Location: A Giant Anchor

Name: Squidward
Occupation: Cashier
Location: An Easter Island Head

1. App - Our application's root component, contains all other components.
2. Wrapper - A div for centering content on the page.
3. Title - A heading tag with some styles applied.
4. SpongebobCard - A component that renders the hardcoded Spongebob friend card.
5. MrKrabsCard - A component that renders the hardcoded Mr.Krabs friend card.
6. SquidwardCard - A component that renders the hardcoded SquidwardCard component.

List Map

App.js and List.js Component

```
const fastFood = [
 {
   id: 5,
   name: "McDonalds"
 },
 {
   id: 6,
   name: "Wendy’s"
 }
];
function App() {
 return <List fastFood={fastFood} />;
```

```
function List(props) {
 return (
   <ul className="list-group">
     {props.groceries.map(item => (
       <li className="list-group-item" key={item.id}>
         {item.name}
       </li>
     ))}
   </ul>
 );
}

export default List;
```

Component Map

Utilize the map method in order to render JSX from an array of objects.

- This activity uses Bootstrap. Be sure to add the Bootstrap CDN to your React app's `index.html` file:
```
<link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/twitter-bootstrap/4.0.0/css/bootst
rap.min.css"/>
```
- Open the application in your web browser and study the rendered application. Then take a minute to study the included components:
    - App: Our Application's root component.
    - List: Responsible for rendering an unordered list from `props.data`.
- Modify the `List` component so that inside of its `ul` tags, it renders one `li` tag for each item in array of grocery objects being passed via props. Each `li` tag should display the `text` property of each grocery object. The array map method should be used for this.
- For styling purposes, give each `li` tag a class of `list-group-item`.

Decrement Counter

Add a "Decrement" button and event handler to the previous Click Counter example.

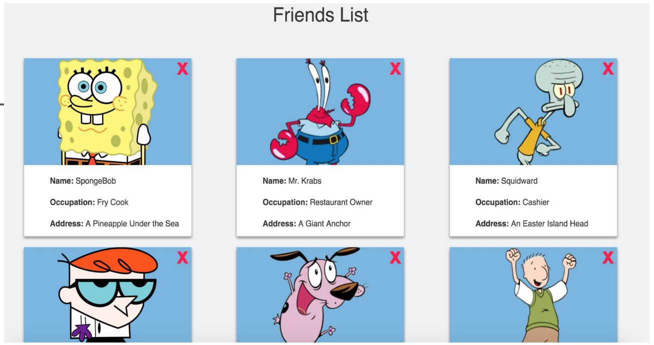
- This example uses Bootstrap. Be sure to add the Bootstrap CDN to your `index.html` file.

```
  <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/twitter-bootstrap/4.0.0/css/bootst
rap.min.css"/>
```

- Open localhost:3000 in your web browser. This application's starter code is identical to the last example.
- Add code to the `Counter` component to add a `Decrement` button which *decreases* the value of `this.state.count` by one each time it is clicked.

Take a Break!

Friend Refactor


Friends List

X

Name: SpongeBob

Occupation: Fry Cook

Address: A Pineapple Under the Sea

X

Name: Mr. Krabs

Occupation: Restaurant Owner

Address: A Giant Anchor

X

Name: Squidward

Occupation: Cashier

Address: An Easter Island Head

X

X

X

Further refactor the Friends List application from earlier to use class components, events, and programmatically render the FriendCard components.

Attempt to complete each of the following:

i. Refactor the `App` component so that it's a class component. Set the component's Initial state to the `friends`JSON array. Inside of the `render` method, map over `this.state.friends`to render each `FriendCard`component.

ii. Refactor the `App` component so that rather than rendering each `FriendCard` component manually, use a map to render one `FriendCard` component for each object in the `friends` JSON, passing in the appropriate props.

iii. Add functionality to the application so that when the red X icon on a `FriendCard` is clicked, that `FriendCard`is removed from the page. To accomplish this, you should define a method inside of `App` which accepts an `id` parameter, and then utilize filter to create a new array of friends without the passed `id`. Then set `this.state.friends`to this new filtered array. You'll want to pass this method into each `FriendCard`component and attach an `onClick` listener to the "remove" span.

Fun With Forms

Add some new functionality to the previous form example.

i. A new input field that updates `this.state.password`
   a. Set the initial value of `this.state.password`to an empty string.
ii. Whenever a user clicks the "Submit" button, add code to accomplish the following:
   a. If the user hasn't provided a first and last name, throw an alert saying: "Fill out your first and last name please!".
   b. If the user has provided a first and last name, but their password is less than 6 characters, throw an alert saying, "Choose a more secure password," followed by the full name. E.g. "Choose a more secure password, John Smith!"
   c. Else, throw an alert to greet the user. E.g. "Hello, John Smith!".
iii. Do not allow the user to type in a password that is longer than 15 characters. i.e. the length of the password state should never go beyond 15 characters.