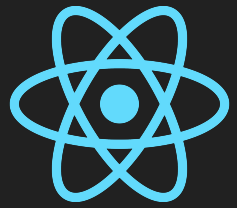


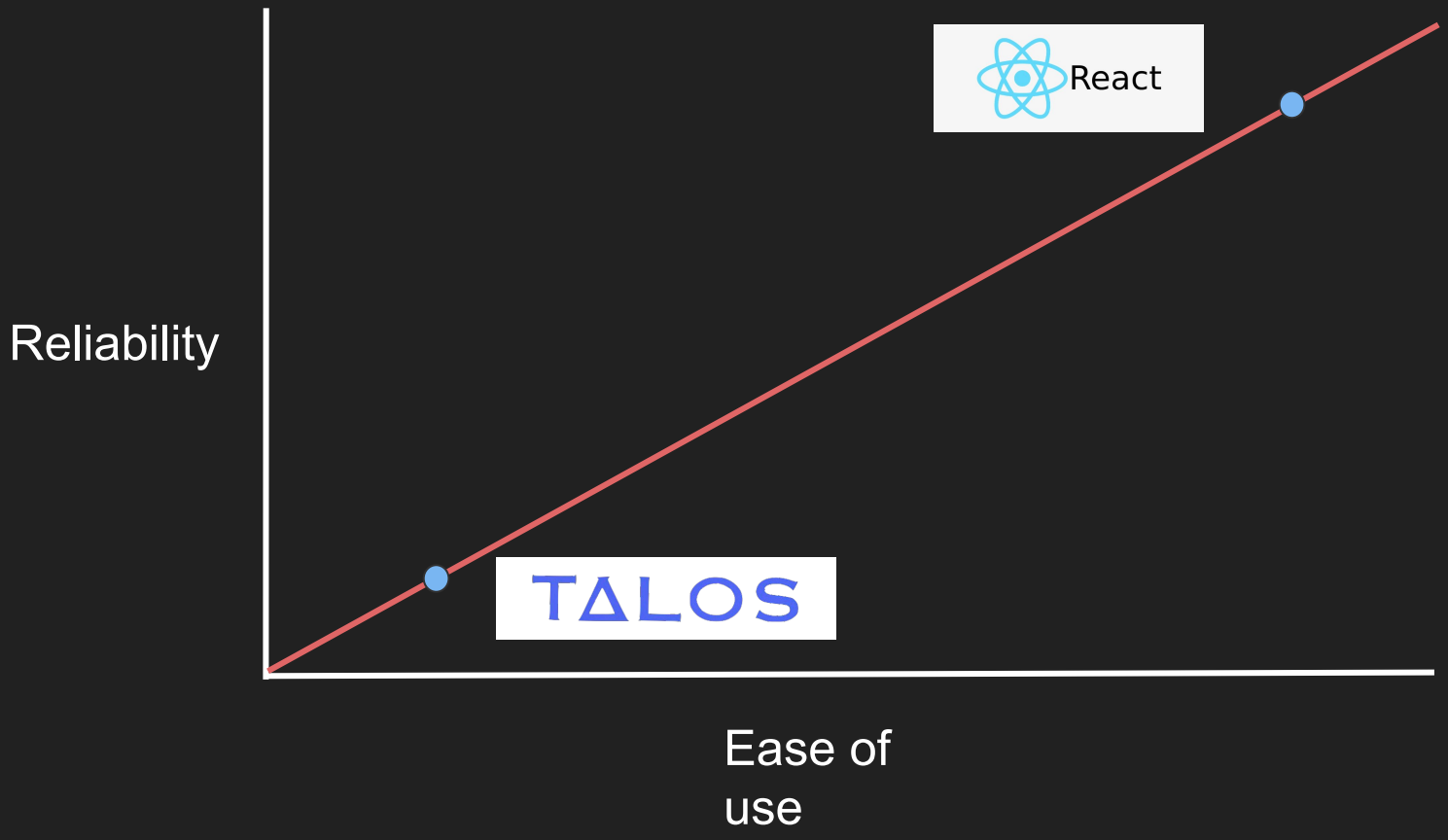
by

Meta

Alif and Jesse present:

A look into React

(and why it’s kinda cool)


Reliability
React
TALOS
Ease of use

The Utility of React

- React is commonly used because it is so versatile:
  - React framework
    - Used for single page applications to quickly rewrite and update code and content on a page without needing to refresh
  - React native
    - Used for developing mobile applications, and performs great
  - Open-source
    - There are many available tools that can be found on the github for react
  - JSX
    - Next slide!

JSX: A Look

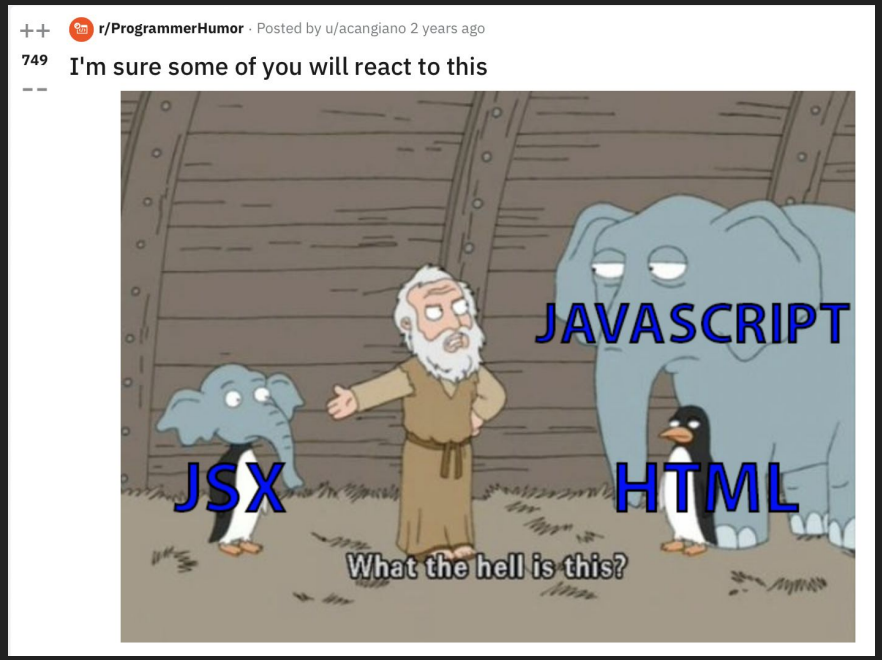
The simplest way to think of JSX is as HTML tags that contain JS Code.

```
const hellow = <h1> Hellow, World! </h1> ;
```

You can incorporate JS within your JSX expressions using curly braces.

```
let olleh = "bruh";

let amog = <h1> Hellow, World, {olleh}! </h1> ;
```



React Components

React components aren't very different from JS functions, since they accept inputs and return an output or element. We will refer to inputs as properties, or props.

Function Component

```
function Welcome(props) {

  return <h1>Hello, {props.name}</h1>;

}
```

Function components are the same as their JS counterparts, based on their structure.

Class Component

```
class Welcome extends React.Component {

  render() {

    return <h1>Hello, {this.props.name}</h1>;

  }

}
```

Class components facilitate the creation of objects, and let us declare variables and functions for the class. These two chunks of code function the same.

Properties and Rendering Components

To actually use the component function we built, we must make a tag with our defined component-name. The render() method is built in, and called a lifecycle method.

Component Calls

```
let greeting = <Welcome name="World!" />
```

OR

```
const world = {name: "World!"}

let greeting = <Welcome name={world.name} />
```

Take a look at our class definition for component

```
class Welcome extends React.Component {

  render() {

    return <h1>Hello, {this.props.name}</h1>;

  }

}
```

Notice that the render function in the class definition of a component intakes a variable named props - this is similar to the “attribute” keywords you would see in a normal DOM tag. Within a **component call**, supply the attribute you defined with appropriate input - be it string, number, object, etc. Surround your input in brackets.

Rendering Components 2: Electric Boogaloo

To display the return value of our components on our page, we simply use the .render method that is built into html elements like as root.

```
let greeting = <Welcome name="World!" />
let root = ReactDOM.createRoot(document.getElementById('root'));
root.render(greeting);
```

If we want to render multiple components, we would have to pass our elements to be rendered through an array.

More about Class Components

If we want to store information within these components, we can use constructors and states to do so with class components. We would use the constructor() lifecycle method to pass props to the base constructor and assign our information to this.state.

```
class Fruit extends React.Component {
  constructor(props) {
    super(props);
    this.state = {fruit: props.name};
  }
  render() {
    return <h2> {this.state.fruit} good </h2>;
  }
}
```

Updating the Local State

To update our local state, we call the setState method and input an object containing the value that we want to update.

Following our previous example, a method that would update the value of ‘fruit’ in our state would look something like this:

```
updateFruit() {
  this.setState({fruit: "apple"}) ;
}
```

Interactive Components

```
class Alert extends React.Component {
  constructor(props) {
    super(props);
    this.methodName = this.methodName.bind(this) ;
  }

  methodName() {
    console.log('click') ;
  }

  render() {
    return(<div><button onClick={this.methodName}>Amog</button></div>) ;
  }

}
```

```
const ele = <Alert/>;
const root = ReactDOM.createRoot(document.getElementById("root")) ;

root.render(ele);
```

```
this.methodName = this.methodName.bind(this) ;
```

Learn It

The Root Of It All:

https://reactjs.org/docs/hello-world.html

Try It Out:

https://jsfiddle.net/reactjs/69z2wepo/