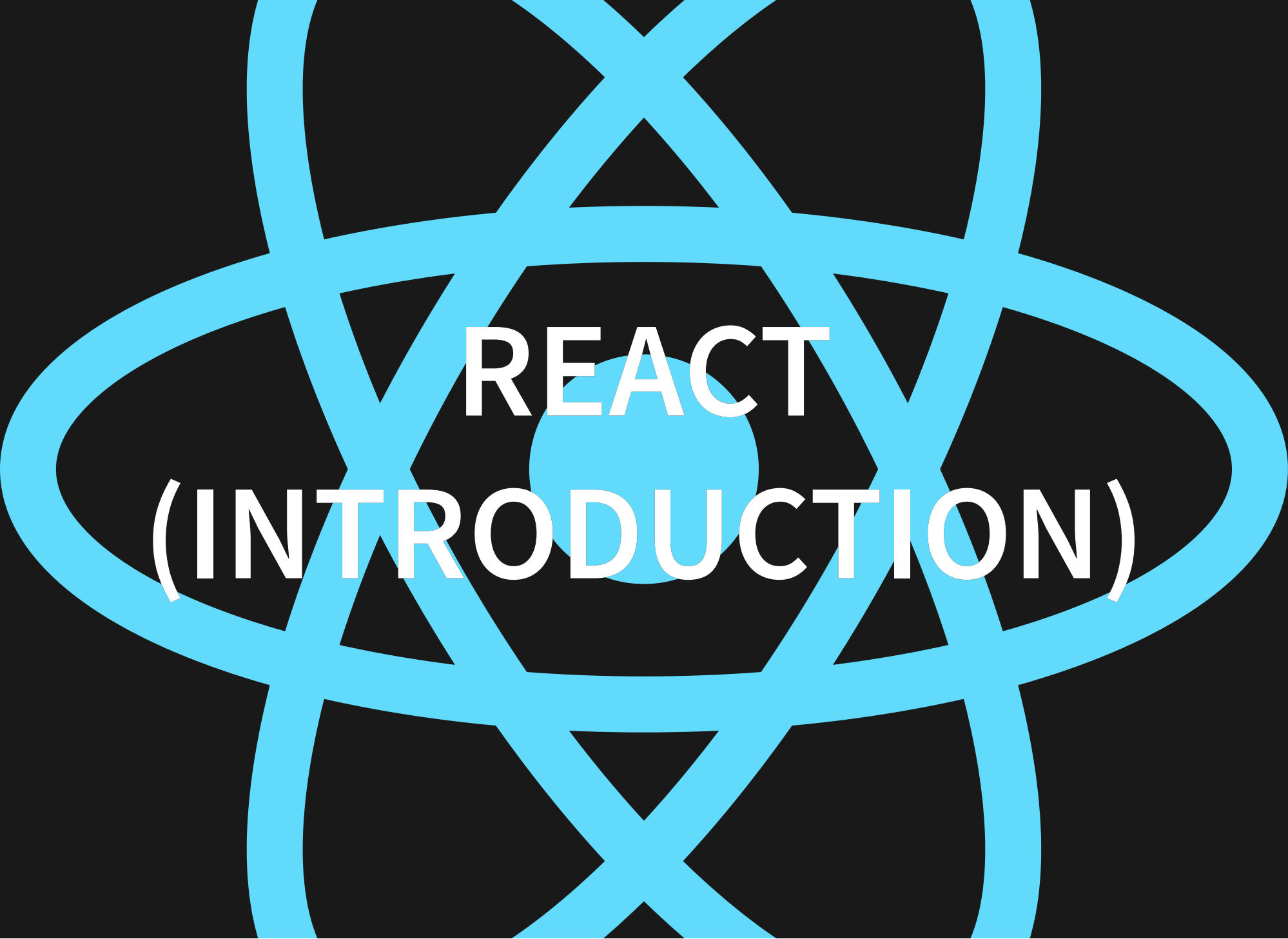
REACT
(INTRODUCTION)

A few popular sites built using React:

- Airbnb
- BBC
- Facebook
- Imgur
- Instagram
- Netflix
- Paypal
- Reddit
- Uber

" React is a library for building composable user interfaces. It encourages the creation of reusable UI components which present data that changes over time. "

Source Why did we build React? (2013)

We will cover:

1. Components
2. JSX
3. State

What are

Components?

" A React component is basically any part of a UI that can contain React nodes "

Source What Is a React Component?

The simplest way to define a component is to write a JavaScript function:

The simplest way to define a component is to write a JavaScript function:

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

The simplest way to define a component is to write a JavaScript function:

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

This is a valid component because it:

1. accepts data (props)
2. returns a React element

The simplest way to define a component is to write a JavaScript function:

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

This is a valid component because it:

1. accepts data (props)
2. returns a React element

This is known as a *functional* component.

Let's look at the other type of component, the

Class Component

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

Notice anything strange about this code?

Let's talk about **JSX**

" by using JSX you can write concise HTML/XML-like structures (e.g., DOM like tree structures) in the same file as you write JavaScript code "

Source What Is JSX?

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

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

- It is not (Vanilla) JavaScript ❌
- It is not a string ❌
- It is not HTML ❌

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

- It is not (Vanilla) JavaScript ❌
- It is not a string ❌
- It is not HTML ❌
- It is JSX ✔️

JSX allows us to express HTML DOM objects in a much easier, human readable way

Let's look at the previous code in Vanilla JavaScript (without React)

Let's look at the previous code in Vanilla JavaScript (without React)

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

```
1 const element = document.createElement('h1');
2 element.textContent = 'Hello, world!;
```

Can you read the relationships between these objects?

Can you read the relationships between these objects?

```
const root = document.createElement('div');
const header = document.createElement('h1');
const subtitle = document.createElement('span');

root.append(header);
header.append(subtitle);
```

```
const element = (<div>
    <h1> <span></span></h1>
</div>);
```

You can include JavaScript expressions inside JSX:

You can include JavaScript expressions inside JSX:

```
1 const array = ['bah!', 'humbug'];
2 const element = <ul>
3     {array.map((word) => <li>{word}</li>)}
4 </ul>
```

You can include JavaScript expressions inside JSX:

```
1 const array = ['bah!', 'humbug'];
2 const element = <ul>
3     {array.map((word) => <li>{word}</li>)}
4 </ul>
```

You can put any valid JavaScript expression inside the curly braces

You can include JavaScript expressions inside JSX:

```
1 const array = ['bah!', 'humbug'];
2 const element = <ul>
3     {array.map((word) => <li>{word}</li>)}
4 </ul>
```

You can put any valid JavaScript expression inside the curly braces

JSX is an expression!

You don't have to use JSX (it is optional)

You don't have to use JSX (it is optional)

Pros:

- Easier to write elements than using JavaScript DOM
- Easier to see relationship between DOM objects
- Code is more optimised

You don't have to use JSX (it is optional)

Pros:

- Easier to write elements than using JavaScript DOM
- Easier to see relationship between DOM objects
- Code is more optimised

Cons:

(Can) mix HTML and JavaScript

What is

state?

" the state of a computer program shows its current values or contents "

Source What does State mean?

Application State

Data which (can) affect the whole application

Application State

Data which (can) affect the whole application

Example: (Todo list) The list of things to do

Application State

Data which (can) affect the whole application

Example: (Todo list) The list of things to do

Component State

Local state, which affects only the component

Application State

Data which (can) affect the whole application

Example: (Todo list) The list of things to do

Component State

Local state, which affects only the component

Example: Checkbox, is it on or off?

So let's create a React app...

```
npx create-react-app my-app
cd my-app
npm run start
```