Intro to React

What is React?

React is a JavaScript library for building user interfaces

— Declarative
— Component-Based
— Ubiquitous.

Declarative

…expresses the logic of a computation without describing its control flow.

```
[1, 2, 3]
  .map(x => x * 2)
  .map(x => `${x} €`)
  .map(x => console.log(x))
```

Component Based

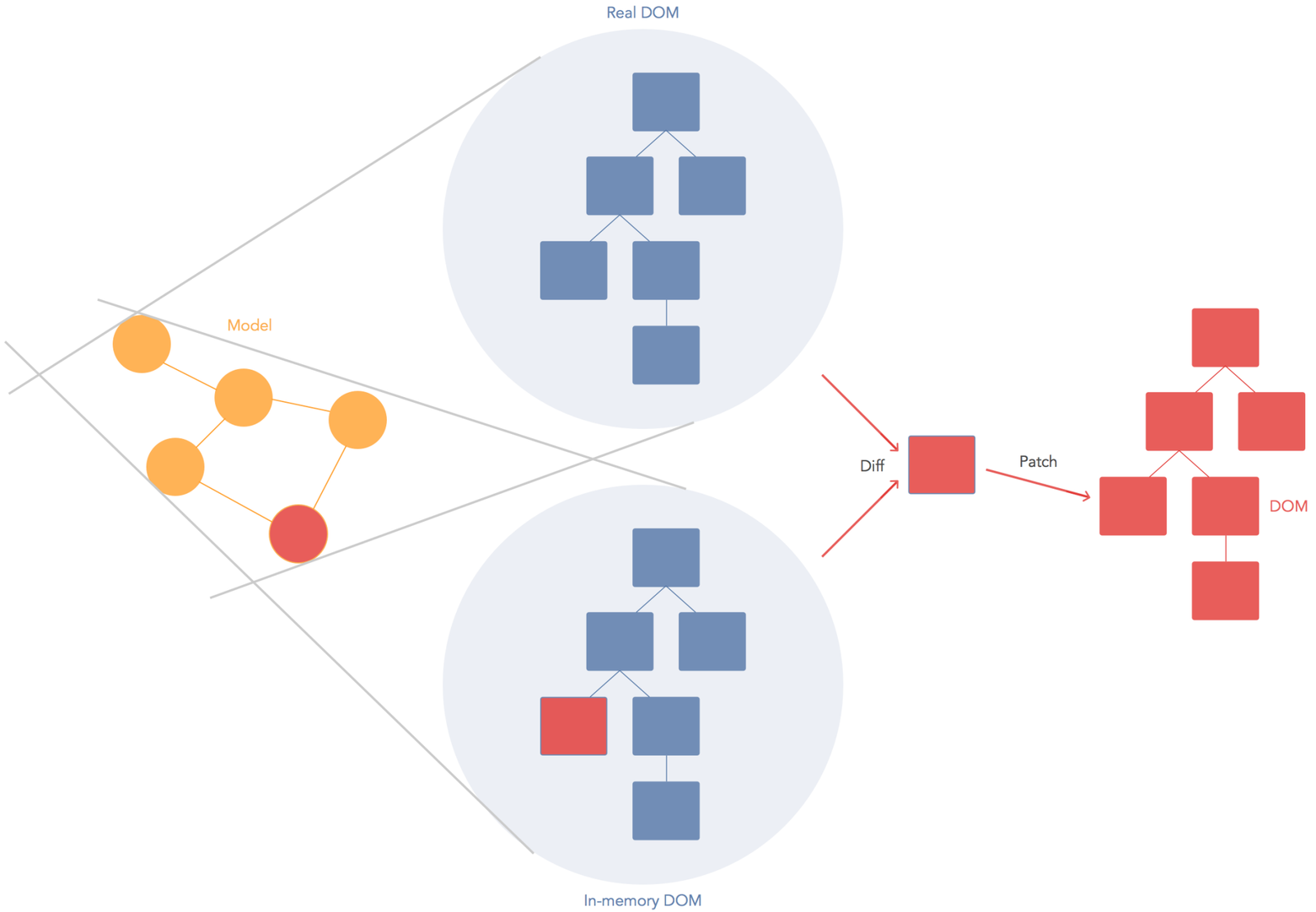
Build encapsulated components that manage their own state, then compose them to make complex UIs.

Ubiquitous

— React Web.
— React Native.
— Client Side rendering.
— Server Side rendering.

Hello World

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


Real DOM
Model
Diff
Patch
DOM
In-memory DOM

Components

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

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

<Welcome name="Lance"/>
```

Will output <h1>Hello, Lance</h1>

Components as Classes

- Import React.
- Create an ES6 class that extends React.Component.
- Add `render()` method.
- Return your JSX in the render method.
- USE `this.props` for props.

Example

```
class Clock extends React.Component {
  render() {
    return (
      <div>
        <h1>Hello, {this.props.name}}!</h1>
        <h2>It is {this.state.date.toLocaleTimeString()}.</h2>
      </div>
    );
  }
}
```

Use as

```
<Clock name="McFly"/>
```

Adding a State

```
class Clock extends React.Component {
  constructor(props) {
    super(props);
    this.state = {date: new Date()};
  }

  render() {
    return (
      <div>
        <h1>Hello, world!</h1>
        <h2>It is {this.state.date.toLocaleTimeString()}.</h2>
      </div>
    );
  }
}
```

Updating the State and LifeCycle

```
componentDidMount() {
  this.timerID = setInterval(
    () => this.setState({
      date: new Date()
    }),
    1000
  );
}

componentWillUnmount() {
  clearInterval(this.timerID);
}
```

Events

Handling events with React elements is very similar to handling events on DOM elements.

Some differences:

— React events are named using camelCase, rather than lowercase.
— With JSX you pass a function as the event handler, rather than a string.

Example

```
const Greeter = (props): JSX.Element => (
  <span
    onClick={() => console.log(`hello ${props.name}!`)}
  >
    Greet!
  </span>
)

<Greeter name="Lance"/>
```

Nested Components

In JSX expressions that contain both an opening tag and a closing tag, the content between those tags is passed as a special prop:

`props.children`

Example of a Nested Component

```
const Page = (props): JSX.Element => (
  <>
    <h1>My Website</h1>
    <div>
      {this.props.children}
    </div>
  </>
)
```

Can be consumed as:

```
<Page>
  <div>My Content</div>
</Page>
```

Thinking in React

Let's Build Something

Hooks

Hooks are inline states.

```
import React, { useState } from 'react';

function Example() {
  // Declare a new state variable, which we'll call "count"
  const [count, setCount] = useState(0);

  return (
    <div>
      <p>You clicked {count} times</p>
      <button onClick={() => setCount(count + 1)}>
        Click me
      </button>
    </div>
  );
}
```

Hooks API

```
const [state, setState] = useState(initialState);
```

— Returns a stateful value, and a function to update it.

— During the initial render, the returned state (state) is the same as the value passed as the first argument (initialState).

— The setState function is used to update the state. It accepts a new state value and enqueues a re-render of the component.

Error Boundaries

A JavaScript error in a part of the UI shouldn’t break the whole app. To solve this problem for React users, React 16 introduced the concept of an “error boundary”.

Example of an Error Boundary Component

```
class ErrorBoundary extends React.Component {
  constructor(props) {
    super(props);
    this.state = { hasError: false };
  }

  static getDerivedStateFromError(error) {
    // Update state so the next render will show the fallback UI.
    return { hasError: true };
  }

  componentDidCatch(error, info) {
    // You can also log the error to an error reporting service
    logErrorToMyService(error, info);
  }

  render() {
    if (this.state.hasError) {
      // You can render any custom fallback UI
      return <h1>Something went wrong.</h1>;
    }

    return this.props.children;
  }
}
```

Usage of an Error Boundary Component

```
<ErrorBoundary>
  <MyWidget />
</ErrorBoundary>
```

FIN