SIX WEEKS OF REACT JS

WEEK 0

INTRODUCTION, JSX AND THE VIRTUAL DOM

INTRODUCTION

WHAT ARE WE DOING HERE?

FRONTEND
FROM A NOVICE PERSPECTIVE

SKILLS
TO BUILD YOUR PROJECT

INSPIRATION
TO GO LEARN MORE

CODE
THAT RUNS AND RUNS WELL

HOW DOES THIS WHOLE THING WORK?

SIX WEEKS
ONE HOUR A WEEK

SLIDES
FOR BRIEF BACKGROUND

TUTORIALS
FOR HANDS ON LEARNING

EXERCISES
PROGRESSIVELY BUILD A THING

MODERN WEB APPLICATIONS ARE HARD

DATA
CHANGES ON THE FLY

TEMPLATES
HUNDREDS OF THEM

VIEWS
OVERLY COMPLEX

EVENTS
HARD TO MANAGE

MEMORY
LEAKS EVERYWHERE

RENDERING
TAKES A TON OF TIME

WHAT IS REACT JS

LIGHTWEIGHT OPEN-SOURCE JS LIBRARY

SIMPLE
MANAGES ALL UI UPDATES

DECLARATIVE
ONLY CHANGES UPDATED UI

COMPOSABLE
BUILD REUSABLE COMPONENTS

WHY USE IT?

COMPONENTS ARE THE FUTURE

CUSTOMIZABLE
EASILY SUBCLASSED AND FLEXIBLE

REUSABLE
BUILD A UI LIBRARY

NESTABLE
EASY MANAGEMENT OF DESCENDANTS

EFFICIENCY GAINS

VIRTUAL DOM
FEWER REDRAWS

MEMORY LIGHT
AUTOBINDING & SINGLE EVENT HANDLER

LESS TYPING
TEMPLATES AND VIEWS IN ONE PLACE

A HUGE COMMUNITY

zendesk

WhatsApp

asana:

KHANACADEMY

BBC

UBER

Instagram

SQUARESPACE

airbnb

facebook

The New York Times

Dropbox

NETFLIX

codecademy

reddit

venmo

imgur

Atlassian

WIRED

GitHub

JSX

STATICALLY-TYPED
OBJECT-ORIENTED
XML-LIKE SYNTAX
ECMASCRIPT EXTENSION

```
var HelloMessage = React.createClass({
  render: function() {
    return <div>Hello {this.props.name}</div>;
  }
});

React.render(<HelloMessage name="John" />,
mountNode);
```

WHAT IS THE POINT OF JSX?

FASTER
OPTIMIZED ON COMPILATION

SAFER
STATICALLY TYPED AND TYPE-SAFE

EASIER
SOLID CLASS SYSTEM AND FAMILIARITY

JSX TRANSFORMATION

```
var HelloMessage = React.createClass({
  render: function() {
    return <div>Hello {this.props.name}</div>;
  }
});

React.render(<HelloMessage name="John" />, mountNode);
```

TO

```
var HelloMessage = React.createClass({displayName: "HelloMessage",
  render: function() {
    return React.createElement("div", null, "Hello ", this.props.name);
  }
});

React.render(React.createElement(HelloMessage, {name: "John"}), mountNode);
```

IN-BROWSER TRANSFORMATION

```html
<!DOCTYPE html>
<html>
  <head>
    <script src="build/react.js"></script>
    <script src="build/JSXTransformer.js"></script>
  </head>
  <body>
    <div id="example"></div>
    <script type="text/jsx">
      React.render(
        <h1>Hello, world!</h1>,
        document.getElementById('example')
      );
    </script>
  </body>
</html>
```

PRE-COMPILED TRANSFORMATION

```
npm install -g react-tools
jsx --watch src/ build/
```

build/helloworld.js

```
React.render(
  React.createElement('h1', null, 'Hello, world!'),
  document.getElementById('example')
);
```

HTML FILE

```
<html>
  <head>
    <script src="build/react.js"></script>
  </head>
  <body>
    <div id="example"></div>
    <script src="build/helloworld.js"></script>
  </body>
</html>
```

UPDATING THE DOM

VIRTUAL DOM (PRE)

```
<h1>Header about things</h1>
  <ul>
    <li>Thing 1</li>
    <li>Thing 2</li>
  </ul>
```

VIRTUAL DOM (POST)

```
<h1>Header about things</h1>
  <ul>
    <li>Thing 1</li>
    <li>Thing 2</li>
    <li>Thing 3</li>
  </ul>
```
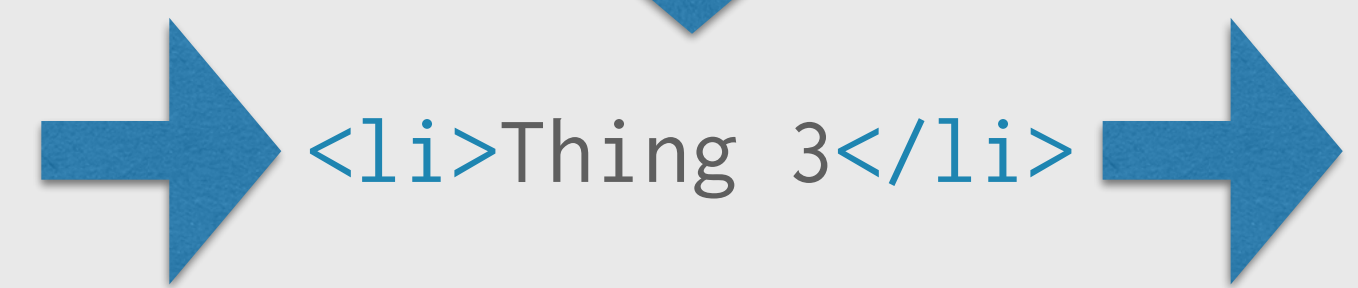
DIFF

PATCH

```
<li>Thing 3</li>
```

REAL DOM (PRE)

```
<h1>Header about things</h1>
  <ul>
    <li>Thing 1</li>
    <li>Thing 2</li>
  </ul>
```

APPLY

```
<li>Thing 3</li>
```

STREAM OF DOM OPERATIONS

REAL DOM (POST)

```
<h1>Header about things</h1>
  <ul>
    <li>Thing 1</li>
    <li>Thing 2</li>
    <li>Thing 3</li>
  </ul>
```

CHANGE DETECTION IN THE VIRTUAL DOM

VIRTUAL DOM (PRE)

```
<h1>Header about things</h1>
  <ul>
    <li>Thing 1</li>
    <li>Thing 2</li>
  </ul>
```

DIFF

VIRTUAL DOM (POST)

```
<h1>Header about things</h1>
  <ul>
    <li>Thing 1</li>
    <li>Thing 2</li>
    <li>Thing 3</li>
  </ul>
```

ASSUMPTIONS

COMPONENTS

SAME CLASS > SIMILAR TREE

DIFFERENT CLASS > DIFFERENT TREE

UNIQUE KEYS

OPTIONAL ATTRIBUTE

STABLE ACROSS RENDERS

PAIR-WISE DIFFING

DIFFERENT NODE TYPES

```
renderA: <div />
renderB: <span />
=> [removeNode <div />], [insertNode <span />]
```

EVEN ON CUSTOM COMPONENTS

```
renderA: <Header />
renderB: <Content />
=> [removeNode <Header />], [insertNode <Content />]
```

IMMEDIATE REPLACEMENT OF THE NODE

PAIR-WISE DIFFING

CHANGED ATTRIBUTES

```
renderA: <div id="before" />
renderB: <div id="after" />
=> [replaceAttribute id "after"]
```

KEY VALUE PAIRS FOR STYLES

```
renderA: <div style={{color: 'before'}}/>
renderB: <div style={{fontWeight: 'bold'}}/>
=> [removeStyle color], [addStyle font-weight 'bold']
```

UPDATE ATTRIBUTES AND RECURSE

LIST-WISE DIFFING

INSERT AT END

```
renderA: <div><span>first</span></div>
renderB: <div><span>first</span><span>second</span></div>
=> [insertNode <span>second</span>]
```

INSERT AT BEGINNING

```
renderA: <div><span>first</span></div>
renderB: <div><span>second</span></div><div><span>first</span></div>
=> [replaceAttribute textContent 'second'], [insertNode <span>first</span>]
```

MUTATIONS DIFFS ARE COMPLEX AND SLOW

LIST-WISE DIFFING

USE KEYS AS OPTIONAL ATTRIBUTES

```
renderA: <div><span key="first">first</span></div>
renderB: <div><span key="second">second</span></div><div><span key="first">first</span></div>
=> [insertNode <span>second</span>]
```

FINDING A STABLE, UNIQUE KEY IS EASY

EXERCISES

SET UP AN ENVIRONMENT

BUILD A HELLO WORLD APP

BUILD A SIMPLE TIMER