**Class 41: REACT (JSX & Props)**

**REACT Introduction**

React is a **JavaSript** library for building the **user interfaces**. It is maintained by Facebook, Instagram and a community of individual developers and corporations. It aims primarily to provide speed, simplicity, and scalability. React was created by **Jordan Walke**, a software engineer at **Facebook**. It was first deployed on Facebook’s newsfeed in 2011 and later on Instagram in 2012. It was open sourced at JSConf US in May 2013.

**Three Important Features**

**Component-based**

Build encapsulated components that manage their own state, then compose them to make complex UIs. Since component logic is written in JavaScript instead of templates, you can easily pass rich data through your app and keep state out of DOM.

**Declarative**

REACT makes it painless to create interactive UIs. Design simple views for each state in your application. REACT will efficiently update and render the right components when your data changes. Declarative views make your code more predictable and easier to debug.

**Learn Once, Write anywhere**

We don’t make assumptions about the rest of your technology stack, so you can develop new features in REACT without re-writing existing code. REACT can also render on the server using Node and power mobile apps using REACT Native.

**JSX www.babeljs.io**

JSX is a syntax extension to JavaScript. We recommend using it with REACT to describe what the UI should look like. JSX may remind you of a template language but it comes with the full power of JavaScript.

JSX produces REACT “elements”. To render this react element:

ReactDOM.render(element, document.getElementById(‘root’));

JSX Represents Objects

Two identical examples:

Constant element = <h1 className=”greeting”>Hello, world!</h1>;

Const element = React.createElement(

‘h1’,

{className: ‘greeting’},

‘Hello, world!’

);

React

**Embedded Expression in JSX**

**JSX is an expression too**

**Inline Style**

**Component & Props**

**Component** let you split the UI into independent, reusable pieces, and think about each piece in isolation. Conceptually, components are like JavaScript **functions**. They accept arbitrary **inputs**(called “**props**”) and return react elements.

**Component props are read only**

LOOP Generation:

Conditional Rendering:

Rendering the

REACT only updates what’s necessary

**Visual DOM**:

DOM stands for document object model, which is an abstraction of a structured text. For web developers, this text is HTML. Elements of HTML becomes nodes in the DOM.

The HTML DOM provides an interface (API) to traverse and modify the nodes. It contains methods like **getElementsById, .innerHTML and createElement**.

Visual DOM: when a change occurs, a new virtual DOM is created from scratch. That new virtual DOM will reflect the new state of the data model. REACT has two virtual DOM data management: the old one and the new one. It then runs a diffing algorithm on two virtual DOMs, to get the set of changes between them. Those changes are applied to the real DOM. Every node in React’s virtual DOM tree is a ReactElement.

**Home Work**: Extracting Components

**Class 44: REACT (State & Event)**

**Convert a function** **to a class**: 5 steps (44-2, 3:40)

**Adding local state to a class:**

Add a class constructor that assign

Note how we pass props to the

**Add lifecycle methods** to a class: in applications with many components, it is very important to free up resources taken by the components when they are destroyed.

**Using State Correctly**

**DO not modify state directly**: never mutate this.state directly, as calling setState() afterwards may replace the mutation you made.

The only place you can assign this.state is the **constructor.**

**State Updates may be asynchronous**

React may batch multiple setState() calls into a single update for performance. Because this.props and this.state may be updated asynchronously, you should not rely on their values for calculating the next state.

**Class 45: REACT (State & Event)**