Overview

- Front-end tool to render interactive web user interfaces.
- Declarative immutable views.
- Component-based: embed HTML-like markup within JavaScript using JSX XML-like syntax (violates separation of concerns).
- From Facebook.

Reactjs Hello World

Hello World reactjs app.

- `<h1>Hello, world!</h1>` is an example of JSX embedded within JavaScript.
- `text/babel` mime-type used to specify use of Babel, which allows supporting newer JavaScript features in browsers using syntax transformers. Used for translating embedded JSX.

Building JavaScript for the Browser

- Example uses a self-contained HTML file which translates at runtime. **Not recommended for production**.
- Modern JavaScript development for the browser typically uses a **build** step on the server to build the artifacts deployed in the browser.

  Package Manager: Examples are npm or yarn (yarn is from Facebook; better reproducibility and performance than npm).

  Bundler: Allows writing modular code with module inclusion directives like `require`. Bundle everything together to minimize HTTP requests. Examples: webpack, browserify and parceljs.

  Compiler: Allows writing code in more modern (or alternate) dialects of JavaScript and have it compiled to dialect supported by browser. Examples: babel, typescript (from MS, used in angular), dart (from Google, in ng2).

JSX

```
const element = <h1>hello world</h1>;
```

- `<h1>...</h1>` represents JSX, an extension to JavaScript syntax. It is not a JavaScript string; it is not HTML.
- JSX is syntactically a JavaScript **expression**.
- A single JSX expression can be written over multiple lines; recommend wrapping in parentheses to avoid automatic semicolon insertion pitfalls.
- Can embed JavaScript within braces inside JSX: `const msg = <h2>hello {user.firstName}<h1>`.
- JSX elements can have attributes:

```
const msg = <h2>hello {user.firstName}
                <img src={user.avatarUrl}/>
              </h2>
```

A Clock Example

Clock 1 application from *ReactJs Tutorial*

- `setInterval()` calls function `tick()` every 1 second.
- `tick()` creates a new JSX element and renders it within the `root` element.

Components

Welcome

- Can define JSX components using a JavaScript function which takes a single argument `props` representing the attributes the component is called with.
- Properties are **immutable** during the lifetime of a component.
- User-defined component names must start with upper-case character.
- We are rendering a list of JSX elements.
- Each JSX element in a list must have a `key` attribute which makes it easy for react to identify it.

Making Clock into a Component

Clock 2 application from *React Tutorial*.

- Setting up timer should be part of clock component and not an external requirement to use it.
- Component needs to maintain state; move from implementing components using functions to implementing components using ES6 classes.

Making Clock into a Component Continued

Clock 3 application from *React Tutorial*.

- Component can be a class with a `render` method.
- Constructor for component class is called with `props` argument specifying attributes for component.
- Our clock value does not change after load; need to set up tick handler after clock component has been loaded.

Making Clock into a Component Continued

Clock 4 application from *React Tutorial*.

- Using component lifecycle hooks:

  `componentDidMount()` Runs after component has been rendered to the DOM. Used for setting up timer in example.

  `componentWillUnmount()` Runs before component removed from DOM. Used for removing the timer in example.

- `tick()` uses `this.setState()` to schedule update to component state.
- Never modify `state` directly; **always** modify only using `setState()` so that react is notified and can set up state modification appropriately (possibly batching with other state modifications).

React Events

Toggle from *React Tutorial*

- Since react is basically JavaScript, event names must be camel-cased.
- Need to make sure `this` in `handleClick()` event-handler is bound to class instance (normally `this` within an event-handler is bound to the DOM element which detected the event).
- Event handler can take argument.

Handling Forms

Simple form from *React Tutorial*

- Single source of truth: mirror state of form controls in component.
- HTML `<textarea>` contents defined by children; react uses `value` attribute on `<textarea>` component instead.
- HTML `<select>` defines selected option using `selected` attribute on `<option>` tag; react uses `value` attribute on `<select>` component instead.

A more complex example: *reservation form* from *React Tutorial*

React Lifecycle Methods

Commonly used *lifecycle methods*:

`constructor()` Initialize state, bind handlers.

`componentDidMount()`, `componentWillUnmount()` Invoked immediately after / before component is being inserted / removed from DOM. Use for initialization which requires DOM nodes, remote services access.

`componentDidUpdate()` Called after state or props update. If `setState()` called, then wrap in condition else infinite loop.