# Advanced Concepts

19 Nov, 2017

---

React Components, Elements, and Instances

Traditional UI model:  you need to take care of creating and destroying child component instances.

Elements Describe the Tree
Element

- a plain object describing a component instance or DOM node and its desired properties
- a way to tell React what you want to see on the screen
- DOM Elements, Component Elements

Component Elements

- Function Components
- Class Components
- Fundamentally, they take the props as their input, and return the elements as their output

Top-Down Reconciliation

- What is reconciliation process?
    - At a single point in time, you can think of the render() function as creating a tree of React elements.
    - On the next state or props update, that render() function will return a different tree of React elements.
    - React then needs to figure out how to efficiently update the UI to match the most recent tree.
    - Example: Ref
    - React implements a heuristic O(n) algorithm based on two assumptions:
        - Two elements of different types will produce different trees.
        - The developer can hint at which child elements may be stable across different renders with a key prop.
- By the end of the reconciliation, React knows the result DOM tree, and a renderer like **react-dom** or **react-native** applies the minimal set of changes necessary to update the DOM nodes (or the platform-specific views in case of React Native).

Difference between traditional UI and React

- Need to take care of child component instances
- Functional components don’t have instances at all
- Only components declared as classes have instances, and React create them for you

- While mechanisms for a parent component instance to access a child component instance exist, they are only used for imperative actions (such as setting focus on a field), and should generally be avoided

---

Project: Build a Github Card Component

"deploy": "yarn run build && surge -p build -d your-domain.surge.sh"

---

JSX In Depth
Fundamentally, JSX just provides syntactic sugar for the React.createElement(component, props, ...children) function.

React Without JSX
- JSX is not a requirement for using React.
- Babel

Specifying The React Element Type
- React Must Be in Scope
- Using Dot Notation for JSX Type
- User-Defined Components Must Be Capitalized
- Choosing the Type at Runtime

Props in JSX
- JavaScript Expressions as Props
- String Literals
- Props Default to “True”
- Spread Attributes

Children in JSX
- String Literals
- JSX Children
- JavaScript Expressions as Children
- Functions as Children
- Booleans, Null, and Undefined Are Ignored(false, null, undefined, and true)

---

Typechecking With PropTypes(Run typechecking on the props for a component)
- PropTypes
- Default Prop Values

Static Type Checking
- Typescript
- **Flow**

---

Refs and the DOM
An escape hatch for parent components to modify their children(React component or DOM element)
When to Use Refs

- Managing focus, text selection, or media playback
- Triggering imperative animations.
- Integrating with third-party DOM libraries.

Adding a Ref to a DOM Element

- React will call the ref callback with the DOM element when the component mounts, and call it with null when it unmounts.

Adding a Ref to a Class Component
Refs and Functional Components

- You may not use the ref attribute on functional components because they don't have instances

Don't Overuse Refs

Exposing DOM Refs to Parent Components?

---

Assignment Project: Todo List

---

Toolchain:
https://teletype.atom.io/
https://github.com/axios/axios
http://www.mockapi.io/
https://surge.sh/

Javascript:
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Template_literals
http://2ality.com/2014/09/es6-modules-final.html
https://twitter.com/sebmarkbage/status/926290535974932480

Reference:
https://reactjs.org/blog/2015/12/18/react-components-elements-and-instances.html
https://reactjs.org/docs/reconciliation.html
https://reactjs.org/docs/typechecking-with-proptypes.html