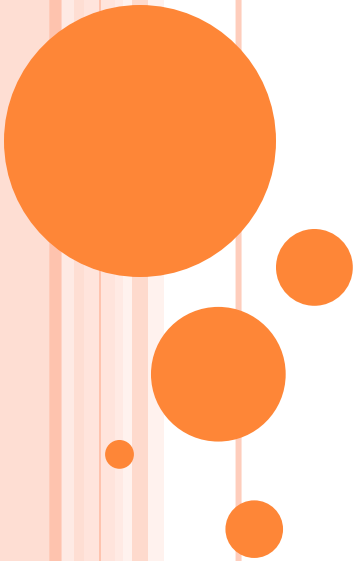

Master ReactJS

A JavaScript UI Framework

What is React

- React is a javascript library or framework for creating UI (user interfaces) by Facebook & Instagram.
- We can think of React as *View* in Model View Controller
- React gives you a template language and some function hooks to essentially render HTML.

Why React

- UI code is readable and maintainable
- Creating UI components allows for modularity and code reusability
- React can be rendered on both client & server side
- It uses a concept called Virtual DOM that selectively renders subtrees of nodes upon state changes.

Component State & Life Cycle

- render
- getInitialState
- getDefaultProps
- propTypes
- mixins
- statics
- displayName

Component State & Life Cycle…

- States & Methods
- Mount
- **componentWillMount**
- **componentDidMount**
- Update
- **componentWillReceiveProps**
- **shouldComponentUpdate**
- **componentWillUpdate**
- **componentDidUpdate**
- Unmount
- **componentWillUnmount**

Code Example

- render
- getInitialState
- getDefaultProps
- propTypes

Events in ReactJS

- **Clipboard Events**
- onCopy onCut onPaste
- Properties
- DOMDataTransfer clipboardData
- **Keyboard Events**
- onKeyDown onKeyPress onKeyUp
- Properties
- boolean altKey
- Number charCode
- boolean ctrlKey
- function getModifierState(key)
- String key
- Number keyCode
- String locale
- Number location
- boolean metaKey
- boolean repeat
- boolean shiftKey
- Number which
- **Focus Events**
- onFocus onBlur

Events in ReactJS

- **Focus Events**
- onFocus onBlur
- Properties
- DOMEventTarget relatedTarget
- **Form Events**
- onChange onInput onSubmit
- **Mouse Events**
- onClick onContextMenu onDoubleClick onDrag onDragEnd onDragEnter onDragExit onDragLeave onDragOver onDragStart onDrop onMouseDown onMouseEnter onMouseLeave onMouseMove onMouseOut onMouseOver onMouseUp
- Properties
- boolean altKey Number button Number buttons Number clientX Number clientY boolean ctrlKey function getModifierState(key) boolean metaKey Number pageX Number pageY DOMEventTarget relatedTarget Number screenX Number screenY boolean shiftKey

Events in ReactJS

- **Touch events**
- To enable touch events, call React.initializeTouchEvents(true)
- onTouchCancel onTouchEnd onTouchMove onTouchStart
- Properties
- boolean altKey DOMTouchList changedTouches boolean ctrlKey function getModifierState(key) boolean metaKey boolean shiftKey DOMTouchList targetTouches DOMTouchList touches

Events in ReactJS

- **UI Events**
- onScroll
- Number detail
- DOMAbstractView view
- **Wheel Events**
- onWheel
- Number deltaMode
- Number deltaX
- Number deltaY
- Number deltaZ

Form Components

- **Interactive Props**
- value, supported by <input> and <textarea> components.
- checked, supported by <input> components of type checkbox or radio.
- selected, supported by <option> components.
- **Controlled Components**
- **Uncontrolled Components**

Accessing DOM

- getDOMNode
- refs

- DOM Event Listeners in a Component

- Special Non-DOM Attributes
- key
- ref
- dangerouslySetInnerHTML

Communicate Between Components

- To communicate between Parent-Child we can make use of props
- To communicate between Child-Parent components we can make use of some pure JavaScript concept like subscribing to events

Flux - Introduction

- Flux is application architecture which can be used to build client-side web application.

- This architecture works by utilizing unidirectional data flow.
- Its pretty easy to use without a lot of new code

Flux - Architecture

- Flux applications have three major parts
- Dispatcher
- Stores
- Views [React Components]

- When a user interacts with a React view, the view propagates an action through a central dispatcher, to the various stores that hold the application's data and business logic, which updates all of the views that are affected. This works especially well with React's declarative programming style, which allows the store to send updates without specifying how to transition views between states

- The stores accept updates and reconcile them as appropriate, rather than depending on something external to update its data in a consistent way. Nothing outside the store has any insight into how it manages the data for its domain, helping to keep a clear separation of concerns. Stores have only a single way of getting new data into their self-contained world — the callback they register with the dispatcher.

Structure and Data Flow

- Data in a Flux application flows in a single direction:

- Action ->Dispatcher->Store->View
- A unidirectional data flow is central to the Flux pattern, and the above diagram should be **the primary mental model for the Flux programmer**. The dispatcher, stores and views are independent nodes with distinct inputs and outputs. The actions are simple objects containing the new data and an identifying *type* property.

Creating Course Cart

- Display a course
- Add courses to the cart
- Remove courses from the cart
- Display the number of courses in the cart
- Display the total price of the course in the cart
- Display the cart after adding a course / clicking the “View Cart” button