GETTING STARTED WITH

React

What is React.js?

A JavaScript library for building user interfaces. Reactjs is created by the Facebook for the V of MVC by reusable and interactive UI components.

Who is using Reactjs?

- Facebook – Partial
- Instagram
- Khan Academy - Partial

Why React.js?

- **V(view) of MVC -** Solution of View in MVC
- **Virtual DOM** - Reactjs use the concept of virtual DOM which helps in the performance
- **Unidirectional Data Flow -** Compare to the 2 way data binding. Reactjs use the concept of Unidirectional data flow which improve the over all performance.
- **UI Components - Reusable and interactive components**
- **SEO Friendly -** Components are client side as well as server side render hence they are SEO friendly and no 3rd party plugin required
- **Coding is simpler because of JSX**
- Reactjs **own debugger**
- React Native is going to be **next big thing**
- **Big minds** are backing Reactjs

Core Concept of Reactjs

JSX

Components

Unidirectional Data Flow

Virtual DOM

JSX

JSX - javascript XML syntax transform.

It helps in making our writing code easier and faster. JSX lets us writeHTML (not 100%) with XML based object representation.

```
8   <script type="text/jsx">
9   var MyApp = React.createClass({
10      render : function(){
11          return (
12                  <h1>Hello React!!</h1>
13              )
14      }
15
16  })
17
18  React.render(<MyApp />, document.getElementById('content'));
19  </script>
20  </body>
21  </html>
```

Line 8-19 : Telling browser the code between the script block is JSX and not normal JavaScript.

Line no 12 : This is how we write the HTML in JSX

```
// without JSX
React.render(
        React.DOM.h1(null, "Hello React!"),
        document.getElementById('content');
    )
```

If you don’t want to use JSX, you can still work in react however the code would be like the above as compare to the last slide without JSX

Components

In Reactjs the whole application is break into the components. Components are interactive, reusable and stageful too.

```
8   <script type="text/jsx">
9   var MyApp = React.createClass({
10      render : function(){
11          return (
12              <h1>Hello React!!</h1>
13          )
14      }
15
16  })
17
18  React.render(<MyApp />, document.getElementById('content'));
19  </script>
20  </body>
21  </html>
```

Line 9 : We are creating our component with name MyApp
Line 18 : We are calling/rendering our component in the content div on the HTML page

Unidirectional Data Flow

- As compare to other MVC frameworks/Library Reactjs use the concept of unidirectional data flow.

- In Reactjs application the data flow from the parent to the children component by the state and the props.

- Only one parent is responsible to update the states and passing the value to the children components via *props*.

- *setState* is used to update/refresh the UI when the state change and the value can be pass to the children component by the *this.props*

Virtual DOM

- Reactjs uses the concept of the virtual DOM.
- It selectively render the subtree of DOM elements into the rendering of the DOM on state change
- Use different algorithm with the browser DOM tree to identify the changes
- Instead of creating new object, Reactjs just identify what change is took place and once identify update that state.
- This way it is creating a virtual DOM and improving the performance too
- Can be render on server and sync on Local

Lets Start

- Download react
- Download JSX
- Install React debugger tool (Chrome)

Start your HTML Page

```
<!DOCTYPE html>
<html>
<head>
    <title>hello React</title>
<script src="https://cdnjs.cloudflare.com/ajax/libs/react/0.13.3/JSXTransformer.js"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/react/0.13.3/react.js"></script>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js"></script>
</head>

<body>
<div id="content"></div>
<script type="text/jsx">

var MyComponent = React.createClass({
    render: function(){
        return (
            <div>
            <h1>Hello, {this.props.name}!</h1>
            </div>
        );
    }
});

React.render(<MyComponent name="Neha" />, document.getElementById('content'));
</script>
</body>
</html>
```

Components

- Everything in reactjs is components. The core building blocks of React application is components only. Components interact with each other and maintain the state too. In Reactjs whole application is need to be break into the component only.

```
 8  <script type="text/jsx">
 9  var MyApp = React.createClass({
10      render : function(){
11          return (
12                  <h1>Hello React!!</h1>
13              )
14      }
15
16  })
17
18  React.render(<MyApp />, document.getElementById('content'));
19  </script>
20  </body>
21  </html>
```

props

- In Reactjs props are like the HTML Properties. They are used to pass the data between the components or via the states. In Reactjs the props can be accessed by *this.props.propsname*

- Props can be define by name=“value”. To access this we have to call this.props.name

```
<body>
<div id="content"></div>
<script type="text/jsx">

var MyComponent = React.createClass({
    render: function(){
        return (
            <div>
            <h1>Hello, {this.props.name}!</h1>
            </div>
        );
    }
});

React.render(<MyComponent name="Neha" />, document.getElementById('content'));
</script>
</body>
</html>
```

React.render(<MyComponent name="Neha" />) : *name* is the one of the props of the component *MyComponent* . To access the value of the props in the react we use {this.props.*name*}.

states

Every component has a State object. Can be set by using setState. setState triggers UI updates and to get the initial state before the setState : getInitialState.getDefaultProps

```
<script type="text/jsx">

var MyComponent = React.createClass({
    getInitialState: function(){
        return {
            count : 5
        }

    },
    render: function(){
        return (
            <div>
            <h1>Hello, {this.props.name} {this.state.count}!</h1>
            </div>
        );
    }
});

React.render(<MyComponent name="Neha" />, document.getElementById('content'));
</script>
```

Reactjs maintained the state – getIntialState, setState, getDefaultProps.

In the above example we are getting the count variable to 5 and accessing it by {this.state.count}

Component Lifecycle

- componentWillMount – Client and server side componenet Will Occur only once (before)
- componentDidMount – Only once (after)
- shouldComponentUpdate – Return value determines weather component should update
- componentWillUnmount – Before unmounting component

Events

- Reactjs has the events that are attached with the components as the props of the components and can trigger methods.

```
<div id="content"></div>
<script type="text/jsx">

var MyComponent = React.createClass({
    getInitialState: function(){
        return {
            count : 5
        }

    },
    clickCount : function(){
        this.setState({
            count : this.state.count + 1
        });
    },
    render: function(){
        return (
            <div>
            <h1>Hello, {this.props.name}  !</h1>
            <button type="button" onClick={this.clickCount}>{this.state.count} Clicked </button>
            </div>
        );
    }
});

React.render(<MyComponent name="Neha" />, document.getElementById('content'));
```

We have created a custom method clickCount and onClick of the button we are calling it onClick = {this.clickCount}

Unidriectional Data Flow

- In reactjs, application data flows unidirectional via the state and props not like angular js where we have 2-way data binding. Which means in multiple component hierarchy , a common parent component should manage the state and pass it down the chain by props.
- setState - state should be updated by setState to ensure UI will refresh/update
- this.props – to pass the value to the child components

	Angular	Backbone	React
Type	*MV**	*MVC*	*V*
Technology	*HTML/CSS/JS/Angular*	*HTML/CSS/JS/backbone*	*JSX*
Core	*MVC*	*MVC*	*Components*
View	*HTML*	*HTML*	*Virtual DOM*
Data Flowing	*2 way binding*	*-*	*Unidirectional*
Creator	*Google*	*-*	*FB* & Instagram*
Architchure	*-*	*-*	*React Native & Flux*
SEO Support	*Phantom js*	*-*	*SEO friendly*

Sources

google.com & Scotch.io

Thank you!