* React is a JavaScript framework

**History of React:**

* React was originally created by engineers at Facebook
* It was released in 2013,
* React Native is a new open source library for creating native user interfaces utilizing the same principles as React's JavaScript library.
* React can be used to solve specific problems encountered when building complex user interfaces.
* Rebuilt the interface each time the data changed or the user interacted with the application.
* Virtual DOM that they use each time they encounter an update to calculate the minimum set of changes needed to update the application's actual DOM.
* JSX is the transform layer that transforms XML syntax for writing React components into the syntax that React uses to render elements in JavaScript.
* Properties are the set of options that a component holds. this.props is a plain JavaScript object in React.
* The state should not be accessed from outside of the component, unless a parent component is adding or setting the initial state of the component.
* Flux is Facebook's application architecture for how to get data to interact with React components in an organized and meaningful way

**The core of React:**

React component can be created in three ways

### Using React.createClass

var MyComponent = React.createClass({

render: function() {

return (

<div>

{this.props.name}

</div>

);

}

});

React.render(<MyComponent name="frodo" />, document.getElementById('container'));

1. ES6 classes by inheriting from React.Component.

class MyComponent extends React.Component {

render() {

return (

<div>

{this.props.name}

</div>

);

}

};

React.render(<MyComponent name="frodo" />, document.getElementById('container'));

**Helper functions:**

### React.Children.map

### Perform a function on each of the immediate children contained and will return an object

var MyComponent = React.createClass({

render: function() {

React.Children.map(this.props.children, function(child){

console.log(child)

});

return (

<div>

{this.props.name}

</div>

);

}

});

React.render(<MyComponent name="frodo" >

<p key="firsty">a child</p>

<p key="2">another</p>

</MyComponent>, document.getElementById('container'));

### React.Children.forEach

React.Children.map function except that it does not return an object

### React.Children.count

### return the number of components that are contained in this.props.children

### React.Children.only

The only method will return the only child that is in this.props.children

### React.createElement

### The createElement method will generate a new ReactElement

### React.cloneElement

### will clone a ReactElement based on a target base element provided as a parameter.

### React.DOM

### This object provides utility functions that help to create DOM elements if you are not utilizing JSX

### React.createFactory

### React.createFactory is a function that will call createElement on a given ReactElement type

### React.render

### React.render will take a ReactElement and render it to the DOM.

### React.renderToString

### React.renderToString is a function that will allow you to render a ReactElement to its initial HTML markup.

### React.findDOMNode

### will return the DOM element of the supplied React component or element that is passed into the function

## **Discovering React Components**

### React components are created when you extend from the base React.Component class using ES6

### Functions in React Component

### To use setState, you can either pass a function or a plain object to the setState method.

setState( function( currState, currProps ) {

return { X: currState.X + "state changed" };

});

setState using an object directly setting the state.

setState( { X: "state changed" } );

When setState is called, you are really queuing the new object into the React update queue, which is the mechanism React uses to control when things are changed.

state object as immutable and only allow React and the setState process of queuing and merging state to control changes to state.

1. forceUpdate( callback )

Bypassing one portion of a component's lifecycle, ComponentShouldUpdate

## **Understanding Component Properties and Methods**

var ReactClassComponent = function() {};

assign(

ReactClassComponent.prototype,

ReactComponent.prototype,

**ReactClassMixin**

);

The assign method is based on Object.assign( target, …sources ) and it will take all the enumerable own properties of the sources and assign them to the target

 In the end the ReactClassMixin is added to the component and has several methods.

1. replaceState( nextState, callback );
2. bool isMounted();
3. setProps( nextProps, callback );

## **Component Lifecycle and Rendering**

### Render - render function will accept a ReactElement and provide a container location where the component will be added or mounted to the DOM

### GetInitialState - unction is invoked one time, just before the component renders.  you will actually be setting the state, via this.state, within the constructor function of the class

### GetDefaultProps - getDefaultProps is invoked one time and then is cached. This function returns an object that will represent the default state of this.props on the component

### Mixins - A mixin can share the lifecycle events of your component and you can be assured that the functionality will execute during the proper time during the component's lifecycle

### propTypes - checks for types for each of the props passed to your component.

React.PropTypes.array

React.PropTypes.bool

React.PropTypes.func

React.PropTypes.number

React.PropTypes.object

React.PropTypes.string

React.PropTypes.any

### 6. statics - Your static functions live in the component and can be invoked without creating instances of the function

### 7. displayName - debugging messages from your React app

### 8. componentWillMount

### 9. componentDidMount

### 10. componentWillReceiveProps

### 11. shouldComponentUpdate

### 12. componentWillUpdate

### 13. componentDidUpdate

### 14. componentWillUnmount

### https://cdn2.percipio.com/1624440246.cb4dc5683901a1426aadef3df6fc389d7c29fc9a/eod/books/97511/images/ch02_fig01a_0.jpg

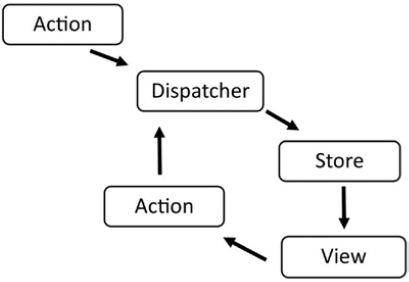
### https://cdn2.percipio.com/1624440246.cb4dc5683901a1426aadef3df6fc389d7c29fc9a/eod/books/97511/images/ch02_fig03_0.jpg

## React Elements

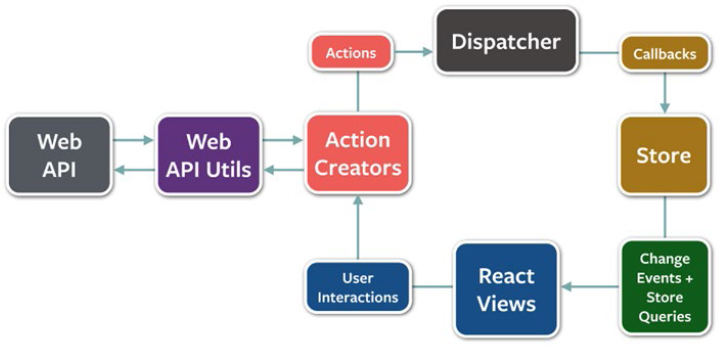
You can create a React element using JSX,

### 

**Flux Architecture**



**Full Architecture of Flux**



Flux is composed of four components

### Dispatcher

1. the ability for the dispatcher to actually dispatch the actions to the stores

### Stores

Stores contain the logic and state of a Flux application.

### Actions

Actions are actually any form of data that has been dispatched to the stores.

### Views

The view layer is where React fits into this architecture. React, with its ability to render the virtual DOM and minimize complex DOM updates, is particularly useful when creating a Flux application.