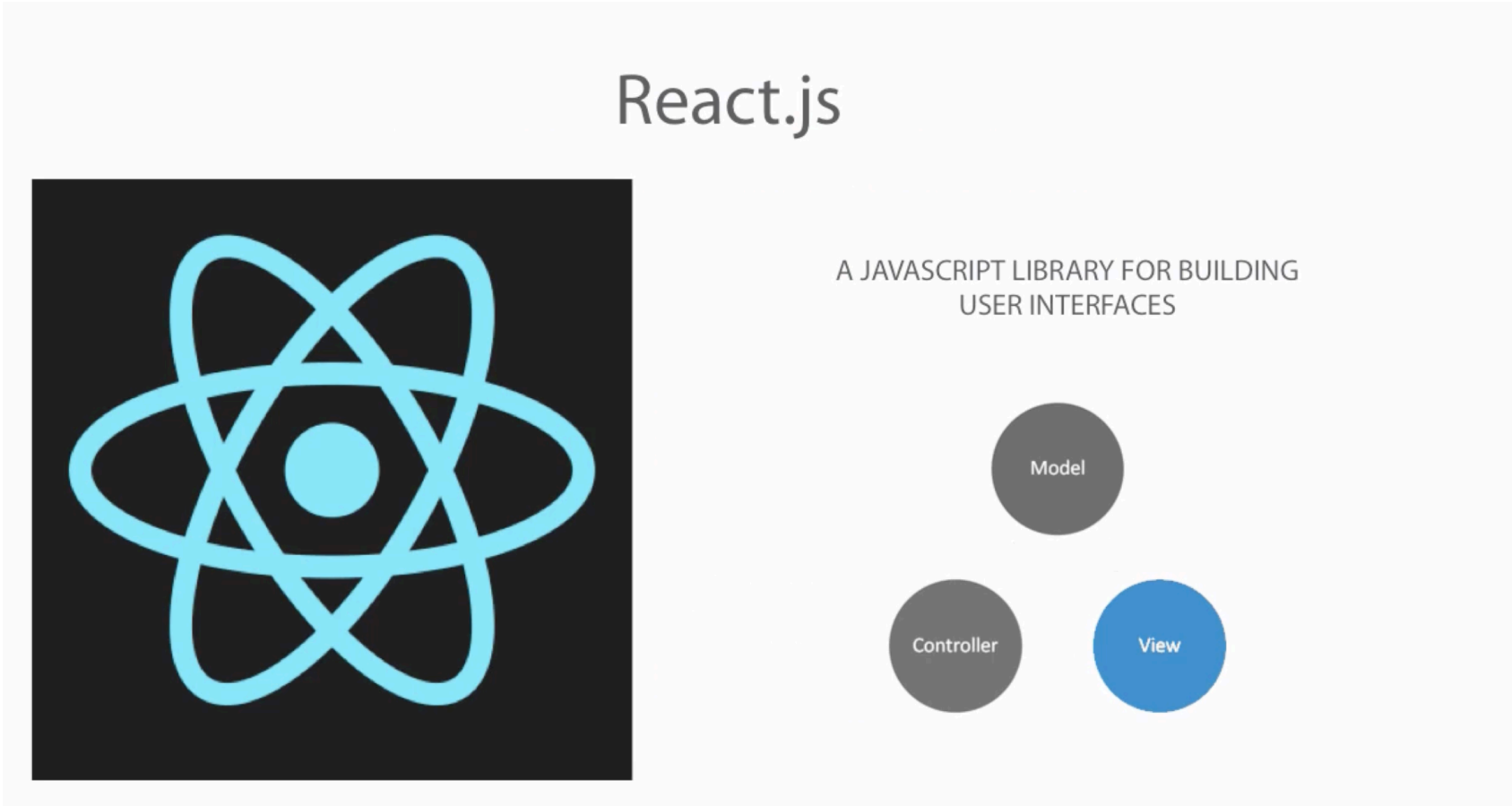
React

Introduction & Core Concepts

What is React

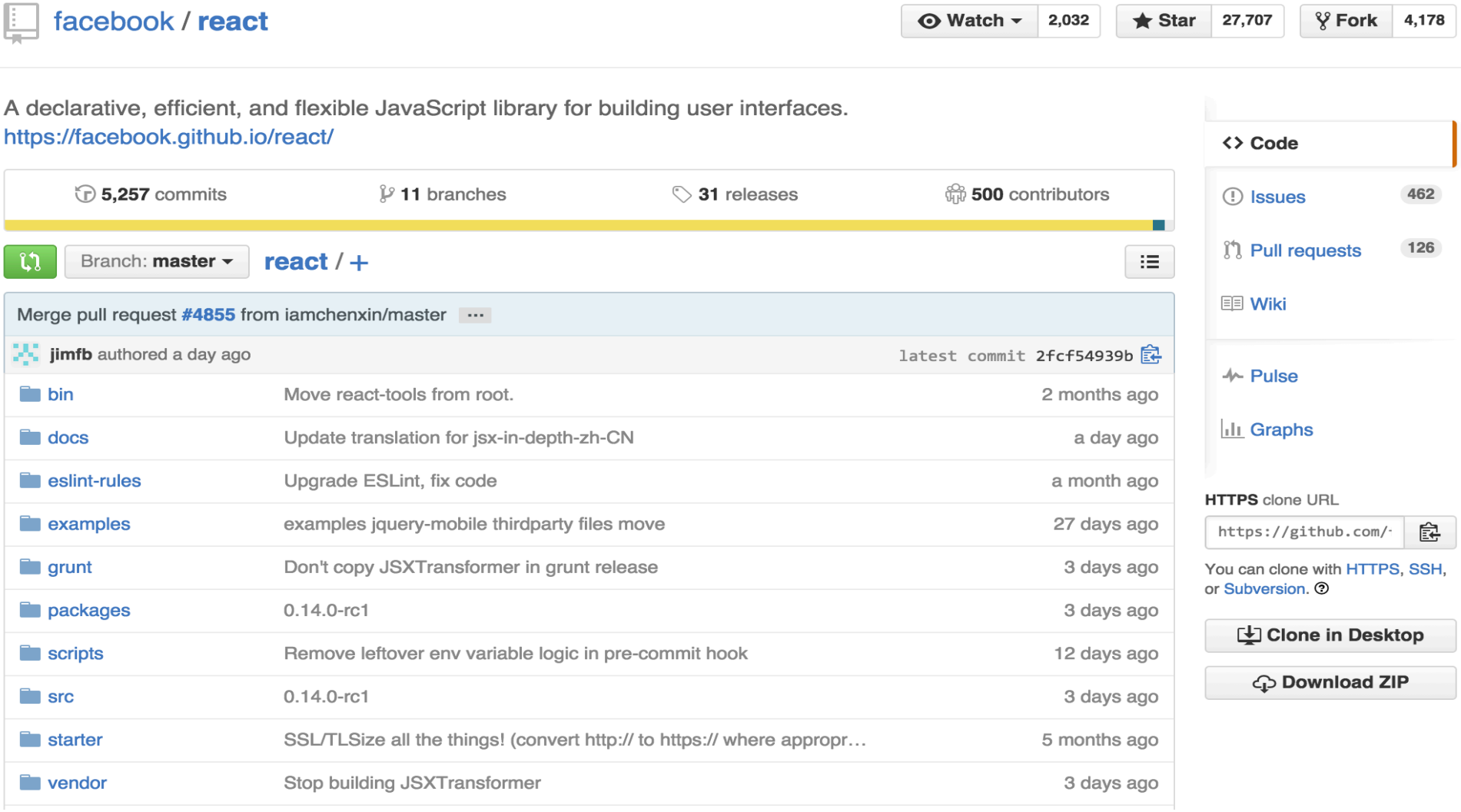
Frameworks

ReactJS


React.js
A JAVASCRIPT LIBRARY FOR BUILDING
USER INTERFACES
Model
Controller
View

Creating ReactJS Apps

Open Source


facebook / react
Watch 2,032
Star 27,707
Fork 4,178
A declarative, efficient, and flexible JavaScript library for building user interfaces.
https://facebook.github.io/react/
5,257 commits
11 branches
31 releases
500 contributors
Branch: master
react / +
Merge pull request #4855 from iamchenxin/master
jimfb authored a day ago
latest commit 2fcf54939b
bin Move react-tools from root. 2 months ago
docs Update translation for jsx-in-depth-zh-CN a day ago
eslint-rules Upgrade ESLint, fix code a month ago
examples examples jquery-mobile thirdparty files move 27 days ago
grunt Don't copy JSXTransformer in grunt release 3 days ago
packages 0.14.0-rc1 3 days ago
scripts Remove leftover env variable logic in pre-commit hook 12 days ago
src 0.14.0-rc1 3 days ago
starter SSL/TLSize all the things! (convert http:// to https:// where appropr… 5 months ago
vendor Stop building JSXTransformer 3 days ago
Code
Issues 462
Pull requests 126
Wiki
Pulse
Graphs
HTTPS clone URL
https://github.com/
You can clone with HTTPS, SSH, or Subversion.
Clone in Desktop
Download ZIP

React

A JAVASCRIPT LIBRARY FOR BUILDING USER INTERFACES

Get Started | Download React v0.14.0

JUST THE UI

Lots of people use React as the V in MVC. Since React makes no assumptions about the rest of your technology stack, it's easy to try it out on a small feature in an existing project.

VIRTUAL DOM

React abstracts away the DOM from you, giving a simpler programming model and better performance. React can also render on the server using Node, and it can power native apps using React Native.

DATA FLOW

React implements one-way reactive data flow which reduces boilerplate and is easier to reason about than traditional data binding.

A Simple Component

Getting Started

Edit on GitHub

JSFiddle

The easiest way to start hacking on React is using the following JSFiddle Hello World examples:

- **React JSFiddle**
- React JSFiddle without JSX

Using React from npm

We recommend using React with a CommonJS module system like browserify or webpack. Use the `react` and `react-dom` npm packages.

Code

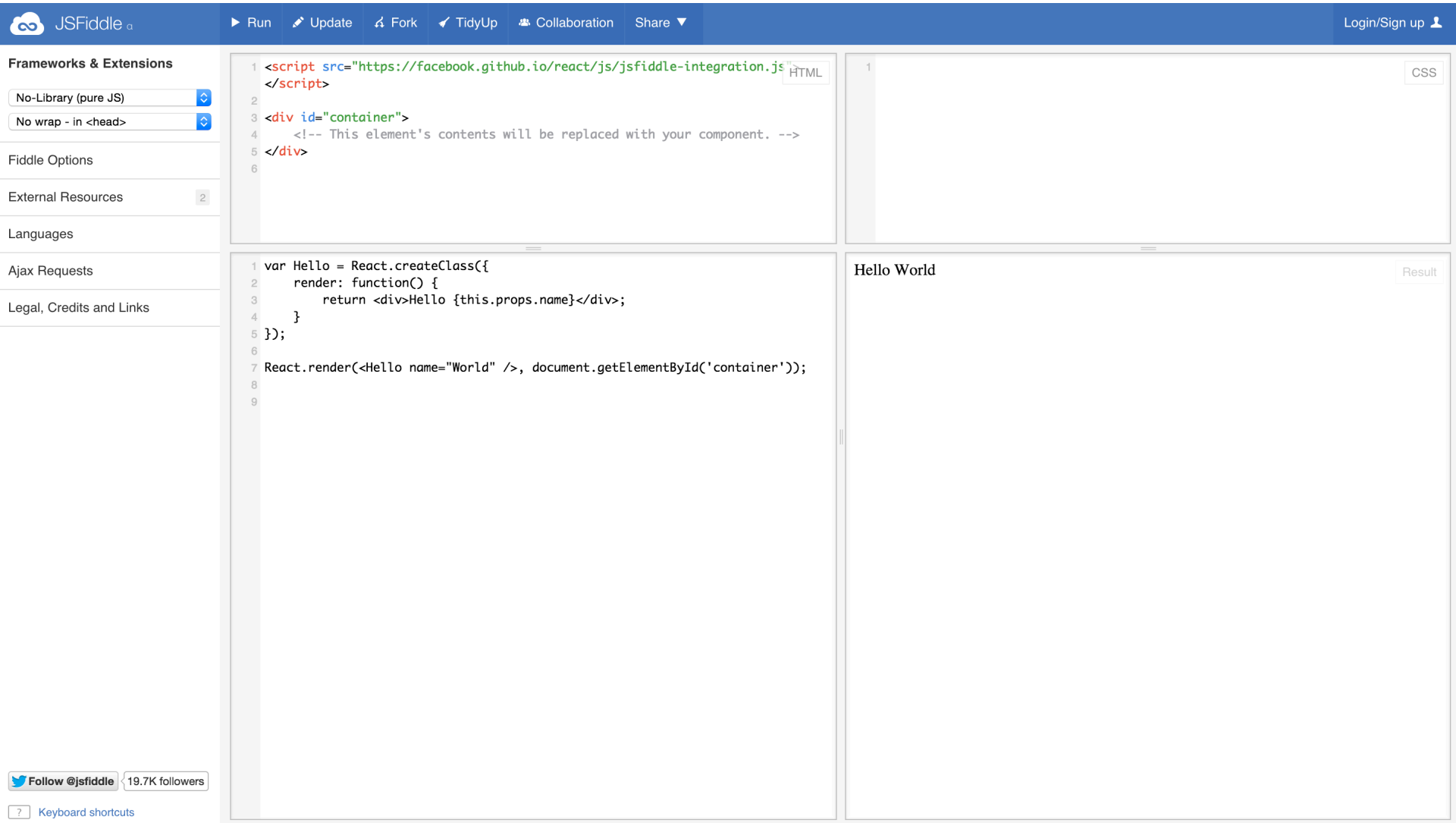
```
// main.js
var React = require('react');
var ReactDOM = require('react-dom');

ReactDOM.render(
  <h1>Hello, world!</h1>,
  document.getElementById('example')
);
```

To install React DOM and build your bundle after installing browserify:

Code

```
$ npm install --save react react-dom
$ browserify -t babelify main.js -o bundle.js
```


JSFiddle
Run
Update
Fork
TidyUp
Collaboration
Share
Login/Sign up
Frameworks & Extensions
No-Library (pure JS)
No wrap - in <head>
Fiddle Options
External Resources
2
Languages
Ajax Requests
Legal, Credits and Links
<script src="https://facebook.github.io/react/js/jsfiddle-integration.js">
</script>

<!-- This element's contents will be replaced with your component. -->

HTML
CSS
var Hello = React.createClass({
render: function() {
return Hello {this.props.name} ;
}
});
React.render(<Hello name="World" />, document.getElementById('container'));
Hello World
Result
Follow @jsfiddle
19.7K followers
Keyboard shortcuts

React Advantages

- Speed
- Declarative
- Composable
- Not going Dark!

React is Not

1. Handle Data Retrieval
2. Handle Data Routing

Composable

- What does Composable mean?

Performant

- Introducing the Virtual DOM

React is Not

- What React does not to..

1. Data Retrieval

2. Data Routing

React Component

```
var React = require('React');

React.render(
  React.createElement(
    'h1', null, 'Hello, world!'),
    document.getElementById('page'))
)
```

createClass

```
var Greeting = React.createClass({
  render: function() {
    return React.createElement(
      'h1', null, 'Hello, world!');
  }
});

React.render(
  React.createElement(Greeting), document.body);
```

Nesting Components

```
var Heading = React.createClass({
    render: function () {
        return React.createElement('h1', null, this.props.children)
    }
});

var Greeting = React.createClass({
    render: function () {
        return React.createElement(Heading, null, 'Hello, world.')
    }
});

React.render(React.createElement(Greeting), document.body);
```

Sample React App

```
var Heading = React.createClass({
  render: function () {
    return (
      React.createElement("header", {
          style: {
            backgroundColor: 'red',
            padding: 20,
            textAlign: "center"}},
        React.createElement(HelloWorld),
        React.createElement('hr'),
        React.createElement('p', null, "Lorem...."))
    )
  }
});

var HelloWorld = React.createClass({
  render: function () {
    return React.createElement("h1", null, "Hello World")
  }
});

module.exports = Heading;
```

this.props

```
var Heading = React.createClass({
  render: function () {
    return React.createElement(
      'h1', null, this.props.children);
  }
});

var Greeting = React.createClass({
  render: function () {
    var name = this.props.name;
    return React.createElement(
      Heading, null, 'Hello, ' + name);
  }
});

React.render(
  React.createElement(
    Greeting, {name: 'world'}), document.body);
```

this.state

```
var ClickMe = React.createClass({
  getInitialState: function() {
    return {clicked: false};
  },

  handleClick: function() {
    this.setState({toggled: !this.state.toggled})
  },
  render: function() {
    var classString = this.state.toggled ? "box toggled": "box";
    return React.createElement(
      'div', {className: classString, onClick: this.handleClick},
        '<span>Click Me</span>')
    );
  }
});

React.render(
  React.createElement(ClickMe, null, document.body);
```

onChange

```
var HelloWho = React.createClass({
  getInitialState: function() {
    return {name: 'world'};
  },
  handleChange: function(event) {
    this.setState( {name: event.target.value });
  },
  render: function() {
    var name = this.state && this.state.name;
    return (
      React.createElement('h1', {className: 'helloWho'},
        'Hello, ' + name,
        React.createElement(
          'input',
          {value: name, onChange: this.handleChange, id: 'helloInput'})
      )
    )
  }
})

React.render(
  React.createElement(HelloWho, null), document.getElementById("page"));
```

props vs state

Props
-Set by parent of component
-Should be immutable within the component
-Props are like components initialization options
-Best to use props in a component

State
-Private to the component
-Mutable (via this.setState)
-Mutation triggers re-render of component
-Used for tracking of component state:

- --toggle state
- --changes to input value

JSX

```
var HelloWho = React.createClass({
  getInitialState: function() {
    return {name: 'world'};
  },
  handleChange: function(event) {
    this.setState( {name: event.target.value });
  },
  render: function() {
    var name = this.state && this.state.name;
    return (
        <div className='helloWho'>
            <h1>Hello, {name}!</h1>
            <input value={name}
                onChange={this.handleChange}
                id='helloInput' />
        </div>
    )
  }
})

React.render(
    <HelloWho />, document.getElementById("page"));
```

Attributes on JSX element

```
var HelloWho = React.createClass({
  getInitialState: function() {
    return {name: this.props.whoName};
  },
  handleChange: function(event) {
    this.setState( {name: event.target.value });
  },
  render: function() {
    var name = this.state && this.state.name;
    return (
        <div className='helloWho'>
          <h1>Hello, {name}!</h1>
          <input
            value={name}
            onChange={this.handleChange}
            id='helloInput' />
        </div>
    )
  }
})

React.render(<HelloWho whoName='world'/>,
    document.getElementById("page"));
```

Parent-Child

```
var Parent = React.createClass({
  getInitialState: function() {
    return {value: 0};
  },
  update: function(value) {
    this.setState({value: value});
  },
  render: function () {
    return (
      <div className="clickCounter">
        <h1># Clicks: {this.state.value}</h1>
        <Child value={this.state.value}
            onChange={this.update}/>
      </div>
    );
  }
});

var Child = React.createClass({
  render: function () {
    var props = this.props;
    var onChange = function () {
      props.onChange(props.value + 1);
    }
    return (
      <button onClick={onChange}>
        Click me!
      </button>
    );
  }
});
```

Parent-Child: Example

```
var Parent = React.createClass({
  getInitialState: function() {
    return {value: 0};
  },
  update: function(value) {
    this.setState({value: value});
  },
  render: function () {
    return (
      <div className="clickCounter">
        <h1># Clicks: {this.state.value}</h1>
        <Child value={this.state.value}
          onChange={this.update}/>
      </div>
    );
  }
});
var Child = React.createClass({
  render: function () {
    var value = this.props.value;
    var onChange = this.props.onChange;
    return (
      <button
        onClick={onChange.bind(null, value + 1)}>
        Click me!
      </button>
    );
  }
});
```

Lifecycle of Components

1. componentWillMount
2. componentDidMount
3. shouldComponentUpdate
4. componentWillReceiveProps
5. componentWillUpdate
6. componentDidUpdate
7. componentWillUnmount

componentDidMount()

1. Called once, right after component's first render
2. This is used a hook to trigger asynchronous data retrieval or register event listeners
3. Useful when using react-router to trigger behaviors when user visits a route

shouldComponentUpdate()

1. Called anytime, a component is about to receive new props or state, but not before initial render

2. Boolean: return true or false. If false, component will not update

3. Useful when you need to compare new state or props values to their old counterparts in order to decide to update.

componentWillReceiveProps()

1. Called once, right after component's first render

componentDidUpdate()

1. Called once, right after component's first render

2. This is used a hook to trigger asynchronous data retrieval or register event listeners

3. Useful when using react-router to trigger behaviors when user visits a route

componentWillUnmount()

1. Called once, right after component is loaded
2. This is used to remove any components in the DOM
3. Useful when using having many components loaded

Mixins

1. componentWillMount
2. componentDidMount
3. shouldComponentUpdate
4. componentWillReceiveProps
5. componentWillUpdate
6. componentDidUpdate
7. componentWillUnmount

React Router

1. Lazy code loading

2. Dynamic Route matching

3. Location transition handling