React Everywhere

DEVIEW
2015

Leonardo YongUk Kim
Feeling Sunday

contents

1. 리액트
2. 웹팩, 핫 모듈 대체, 유니버설 렌더링
3. 리액트 네이티브, 일렉트론
4. 다음은?

1.
리액트

1.1 왜 React인가요?

Why React?

Edit on GitHub

React is a JavaScript library for creating user interfaces by Facebook and Instagram. Many people choose to think of React as the **V** in **MVC**.

We built React to solve one problem: **building large applications with data that changes over time**.

Simple

Simply express how your app should look at any given point in time, and React will automatically manage all UI updates when your underlying data changes.

Declarative

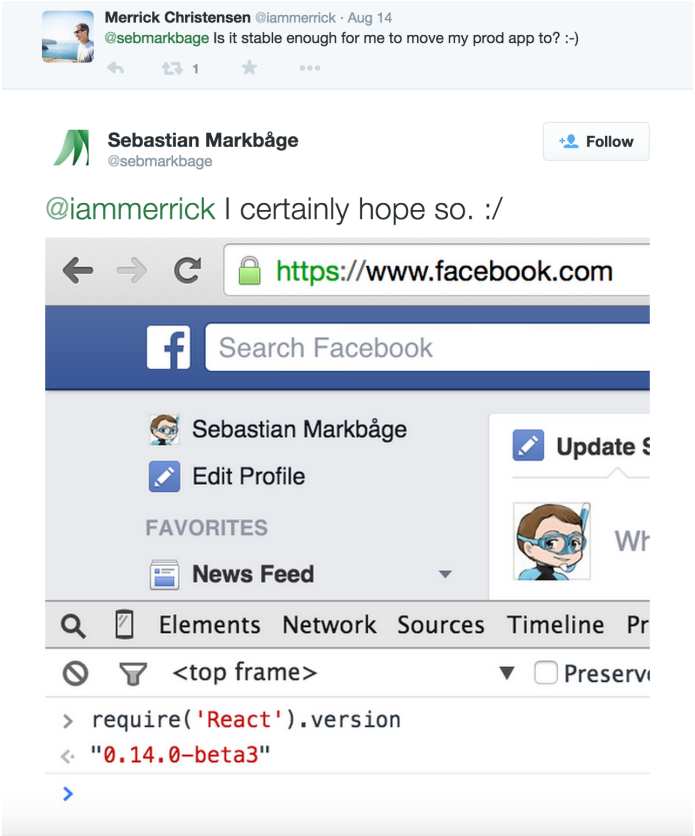
When the data changes, React conceptually hits the "refresh" button, and knows to only update the changed parts.

Build Composable Components

React is all about building reusable components. In fact, with React the *only* thing you do is build components. Since they're so encapsulated, components make code reuse, testing, and separation of concerns easy.

1.1 왜 React인가요?

DEVIEW
2015



페이스북이 만든 View 라이브러리

페이스북과 인스타그램에 직접 사용.

심지어 베타 버전으로 개밥 복용.

https://twitter.com/sebmarkbage/status/632257978003951616

1.2 Hello, world!

패키지를 먼저 설치합시다.

```
npm install babel-core --save
npm install react --save
npm install react-dom --save (0.14가 정식 버전이 되면)
```

1.2 Hello, world!

나는 최신 버전(현재로서는 0.14.0-RC 버전)을 사용하고 싶어요.

```
npm install babel-core --save
npm install react@0.14.0-rc1 --save
npm install react-dom@0.14.0-rc1 —save
```

0.14.0 이상 버전도 react-dom 모듈도 script 태그로 포함시키는 것을 잊지 마세요.

```
1  <!DOCTYPE html>
2  <html>
3    <head>
4      <meta charset="UTF-8" />
5      <title>Hello React!</title>
6      <script src="node_modules/react/dist/react.js"></script>
7      <script src="node_modules/babel-core/browser.js"></script>
8    </head>
9    <body>
10     <div id="example"></div>
11     <script type="text/babel">
12       React.render(
13         <h1>Hello, world!</h1>,
14         document.getElementById('example')
15       );
16     </script>
17   </body>
18 </html>
```

React 라이브러리

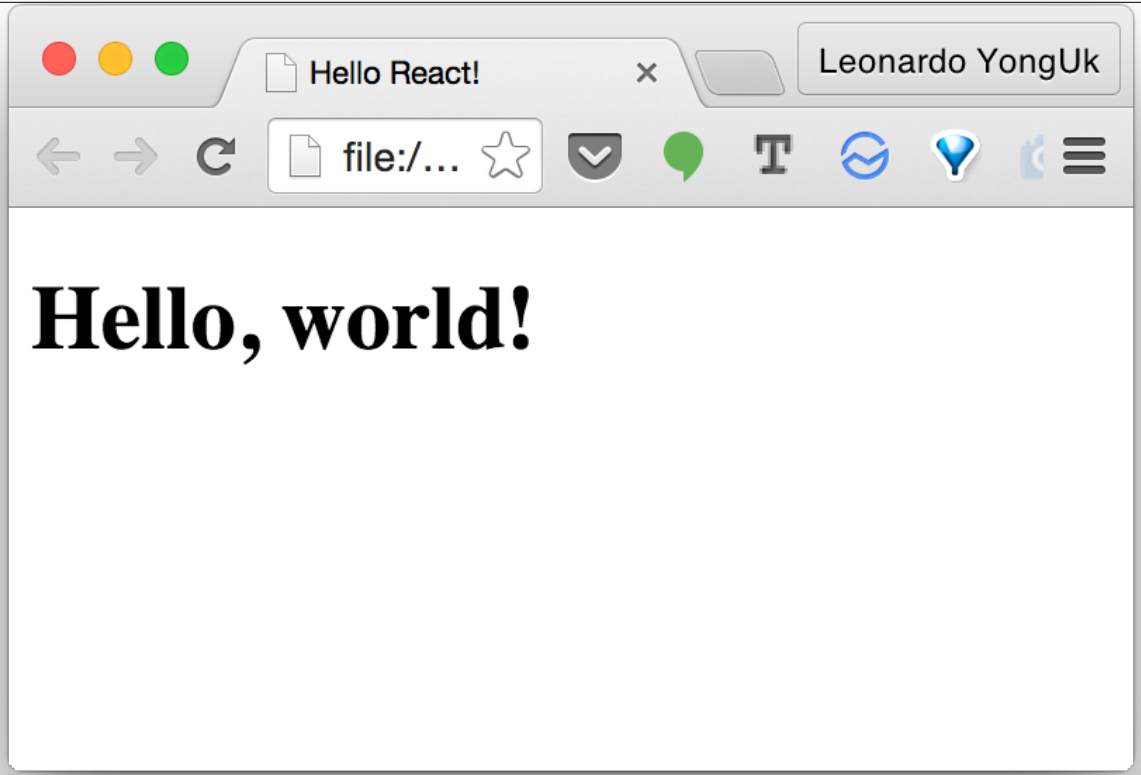
Babel Browser (브라우저 번역기)

번역될 영역에 text/babel 표시

렌더링 메서드

그려질 내용

그려질 위치


Hello React!
Leonardo YongUk
file:/...
Hello, world!

1.3 JSX


```
<script type="text/babel">
  React.render(
    <h1>Hello, world!</h1>,
    document.getElementById('example')
  );
</script>
```

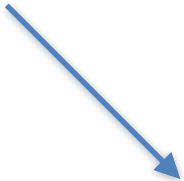
1. 자바스크립트 안에 HTML이에요? HTML과 유사한 문법을 가진 JSX입니다.
2. 어떻게 실행되는 거에요? text/babel 영역을 바벨이 번역해서 자바스크립트로 만들어요.
3. 실행시간에 번역하면 느리지 않나요? 프로덕션에선 번역된 코드를 씁니다.
4. 튜토리얼에서는 바벨을 안쓰던데 text/jsx쓰던데? 그 방법 0.14에서 폐기됩니다. (곧)

1.3 JSX


```
<script type="text/babel">
  React.render(
    <h1>Hello, world!</h1>,
    document.getElementById('example')
  );
</script>
```

Babel에 의한 번역

(0.14는 자바스크립트를 쓰지 않는 인라인버전도 있음)



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

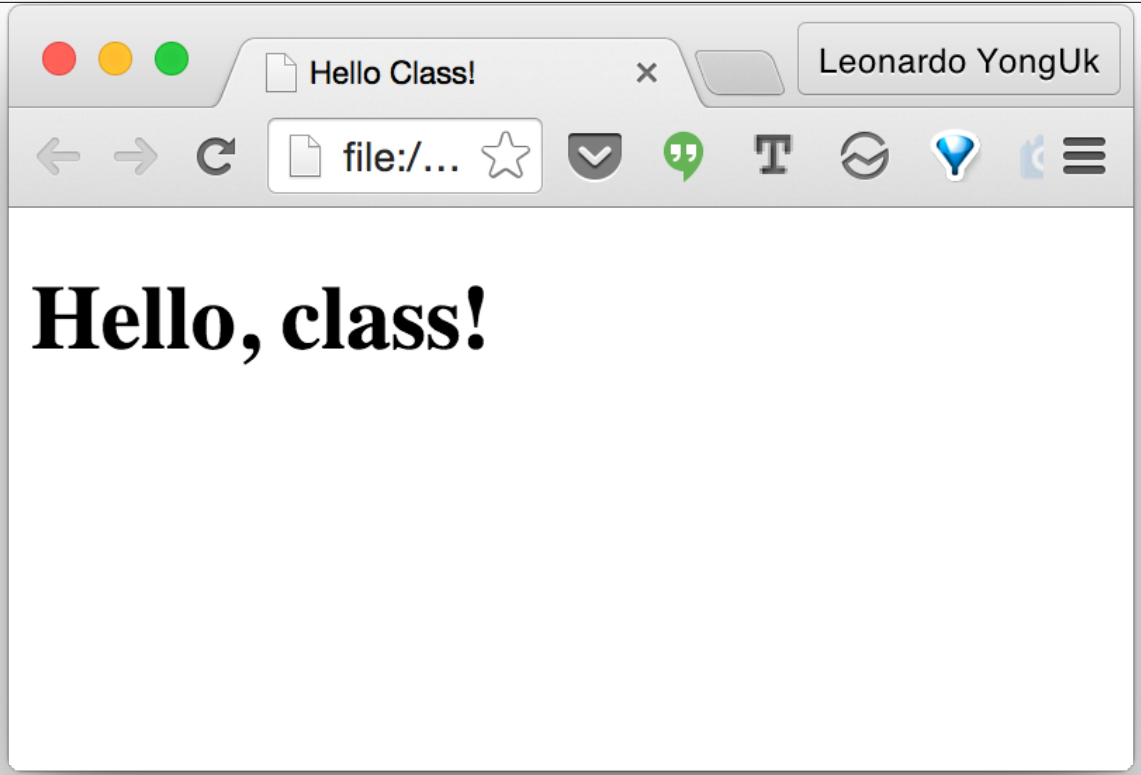
1.4 Hello, class


```
<script type="text/babel">
  var HelloClass = React.createClass({
    render: function() {
      return (
        <h1>Hello, class!</h1>
      );
    }
  });
  React.render(
    <HelloClass />,
    document.getElementById('example')
  );
</script>
```

HelloClass란 이름으로 컴포넌트 생성.
클래스명은 대문자로 시작.

JSX를 리턴하는 함수를 render에

작성한 HelloClass 객체를 화면에 표시.
React.render에 사용할 클래스는 하나만.

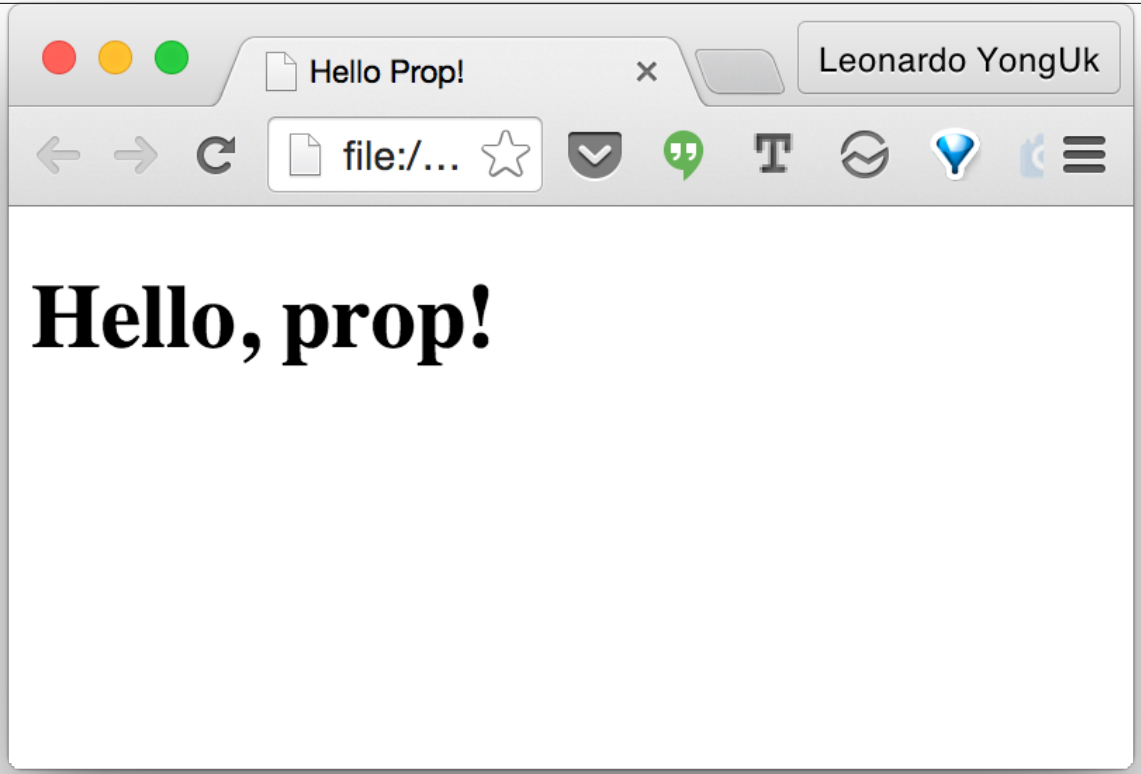

Hello Class!
Leonardo YongUk
file:/...
Hello, class!

1.5 Hello, prop


```
<script type="text/babel">
  var HelloClass = React.createClass({
    render: function() {
      return (
        <h1>Hello, {this.props.someone}!</h1>
      );
    }
  });
  React.render(
    <HelloClass someone="prop"/>,
    document.getElementById('example')
  );
</script>
```

프로퍼티 조회는 this.props에서
프로퍼티는 사용 시점에 정의되며
읽기 전용.

someone 프로퍼티 설정


Hello Prop!
Leonardo YongUk
file:/...
Hello, prop!

1.6 Count, state

DEVIEW
2015

```
<script type="text/babel">
var Counter = React.createClass({
  getInitialState: function() {
    return { number: this.props.number };
  },
  handleClick: function() {
    this.setState({ number: this.state.number + 1 });
  },
  render: function() {
    return (
      <p onClick={this.handleClick}>
        DEVIEW, {this.state.number}!
      </p>
    );
  }
});
React.render(
  <Counter number={2015} />,
  document.getElementById('example')
);
</script>
```

1.6 Count, state

DEVIEW
2015

```
<script type="text/babel">
var Counter = React.createClass({
  getInitialState: function() {
    return { number: this.props.number };
  },
  handleClick: function() {
    this.setState({ number: this.state.number + 1 });
  },
  render: function() {
    return (
      <p onClick={this.handleClick}>
        DEVIEW, {this.state.number}!
      </p>
    );
  }
});
```

state의 초기값은 getInitialState에서
props.number 의 값으로 설정한다.

click 이벤트 처리할 콜백.
상태 값 변경은 this.setState에서

onClick 처리 this.handleClick에서
this.state.number로 설정된 상태 사용.

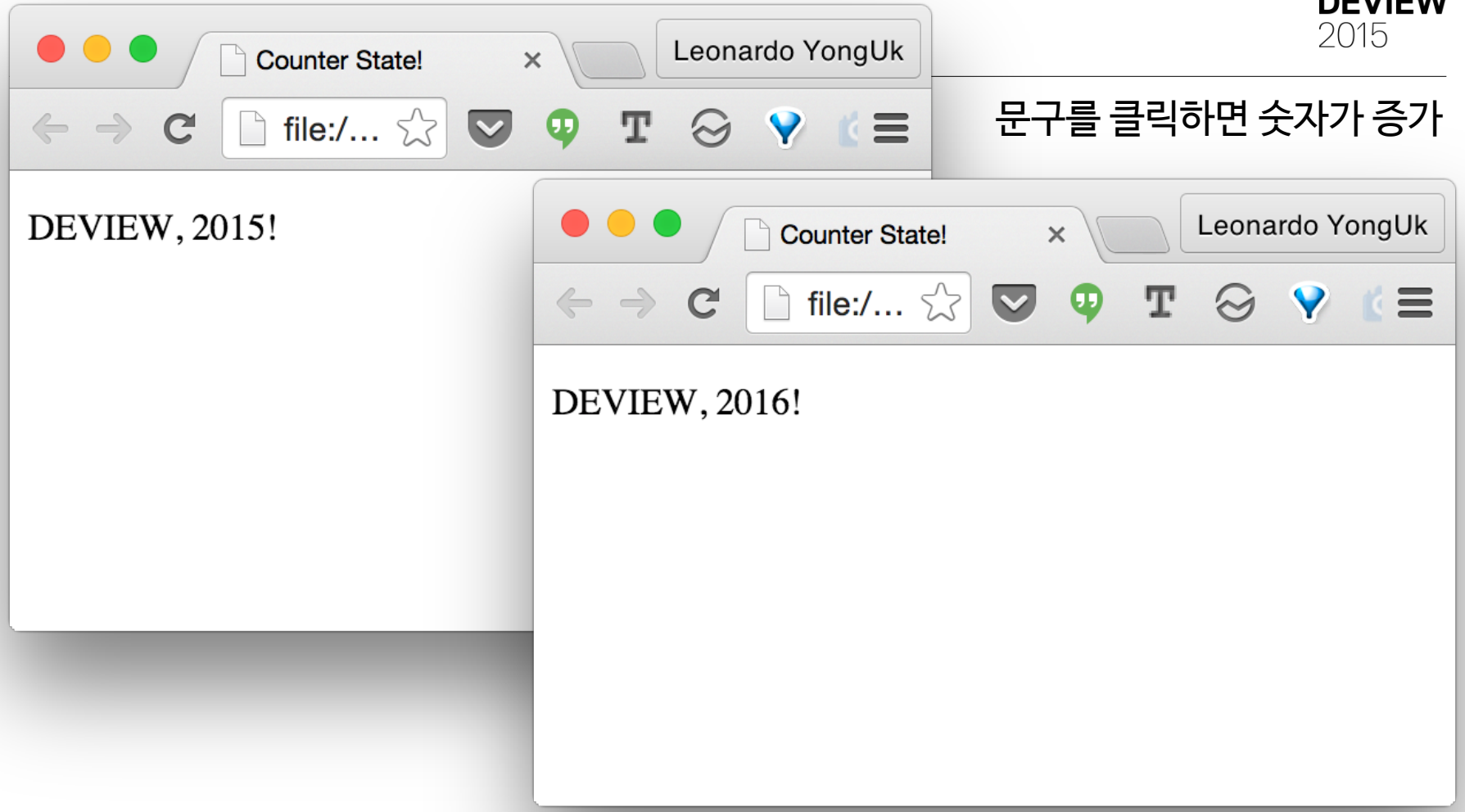
1.6 Count, state


```
React.render(
  <Counter number={2015} />,
  document.getElementById('example')
);
</script>
```

this.props.number의 값을 2015로 설정한다.

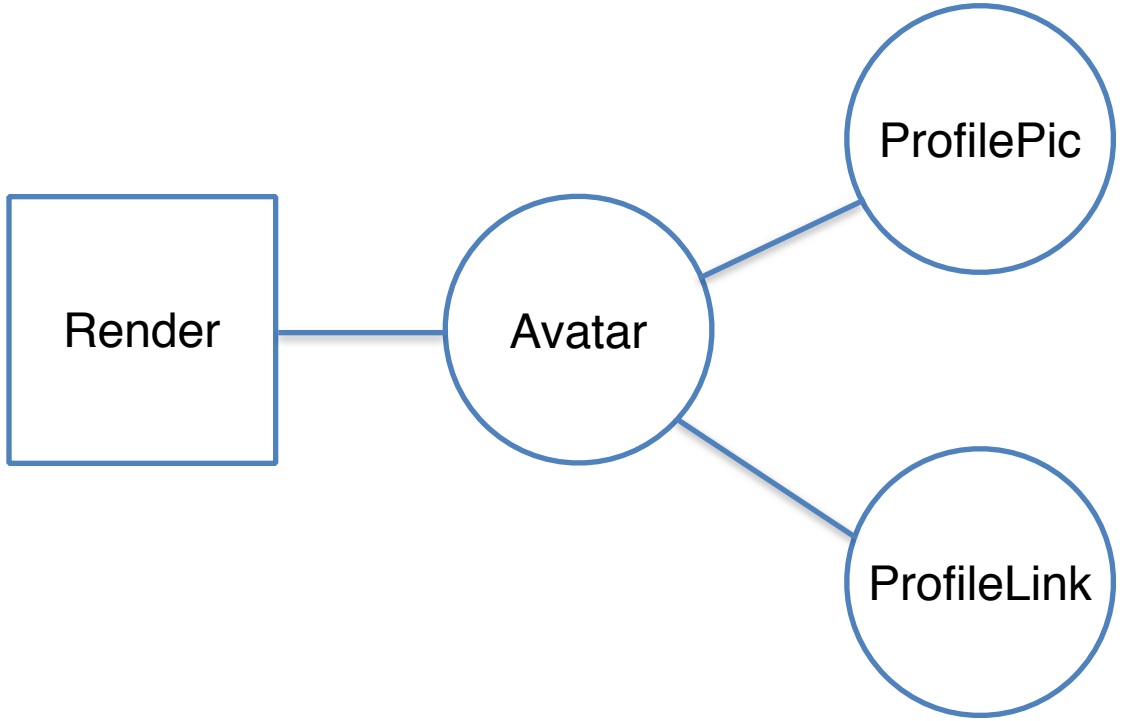
값을 제대로 전달하기 위해 {2015}로 설정하였다. “2015”로 전달하면 문자열로 전달된다.

문구를 클릭하면 숫자가 증가



1.7 Multiple Components

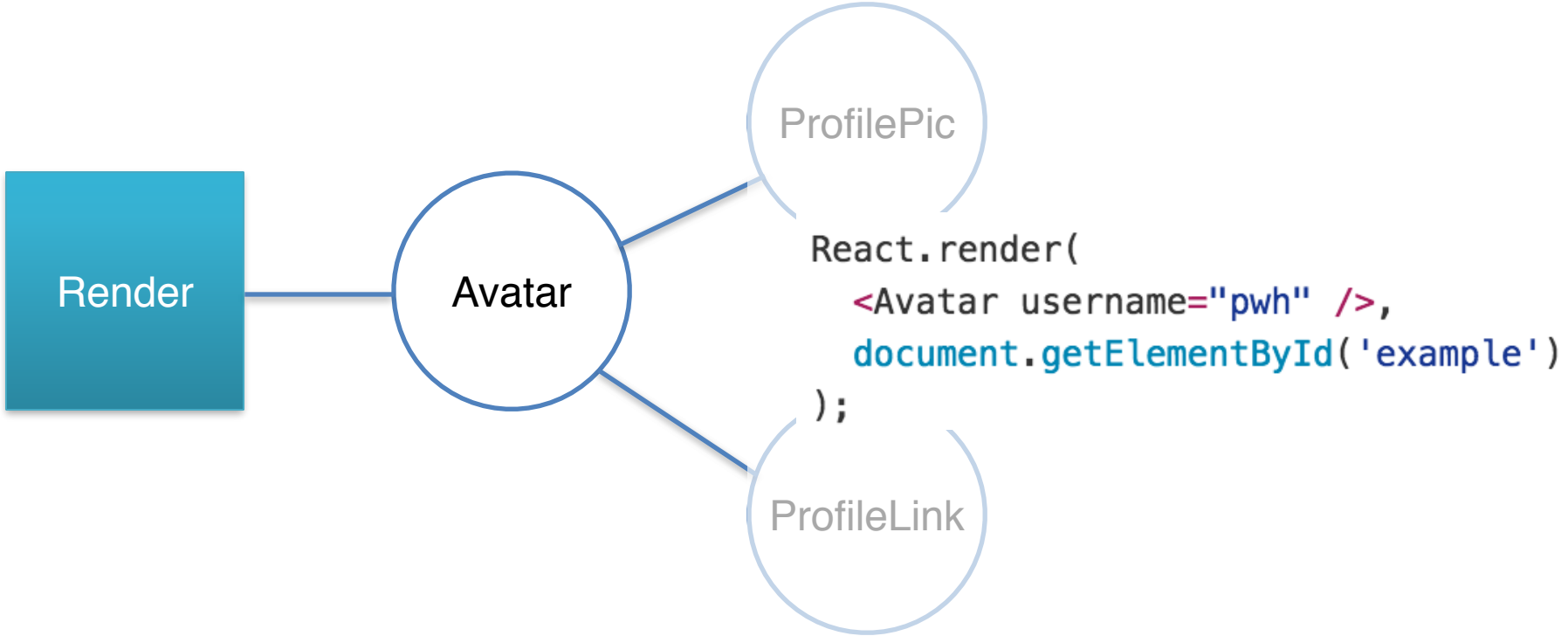
DEVIEW
2015




```
var Avatar = React.creat
  render: function() {
    return (
      <div>
        <ProfilePic user
        <ProfileLink use
      </div>
    );
  }
});

var ProfilePic = React.c
  render: function() {
    return (
      <img src={'https:/
    );
  }
});

var ProfileLink = React.
  render: function() {
    return (
      <a href={'https://
        {this.props.user
      </a>
    );
  }
});

React.render(
  <Avatar username="pwh"
  document.getElementBy
);
```

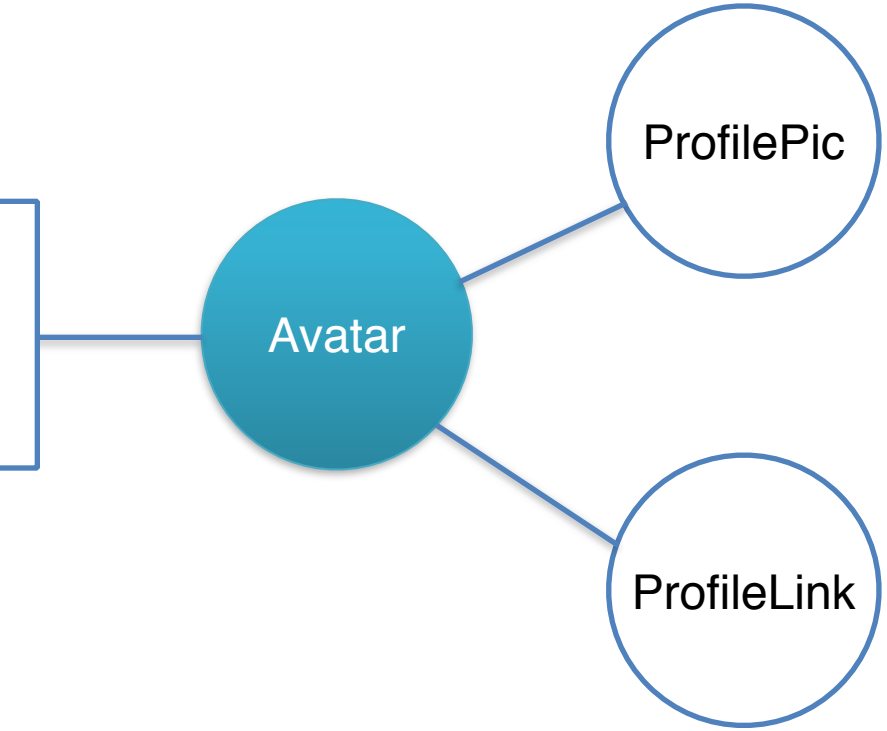
1.7 Multiple Components

DEVIEW 2015



1.7 Multiple Components

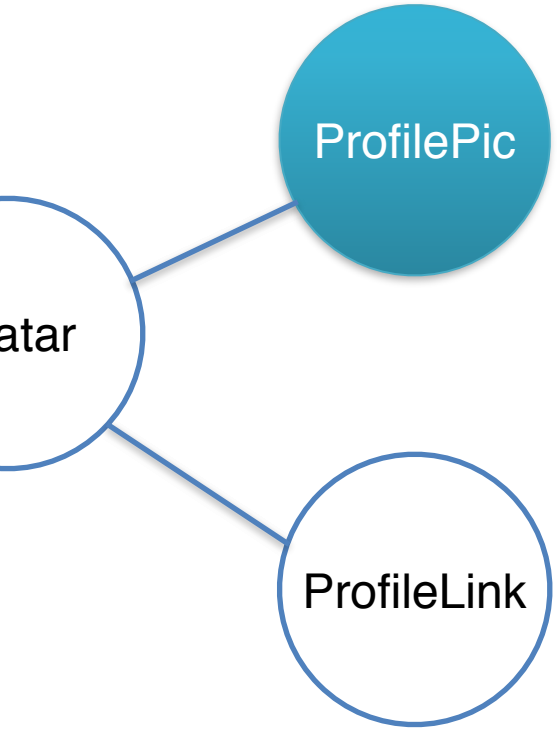
DEVIEW 2015




```
var Avatar = React.createClass({
  render: function() {
    return (
      <div>
        <ProfilePic username={this.props.username} />
        <ProfileLink username={this.props.username} />
      </div>
    );
  }
});
```

1.7 Multiple Components

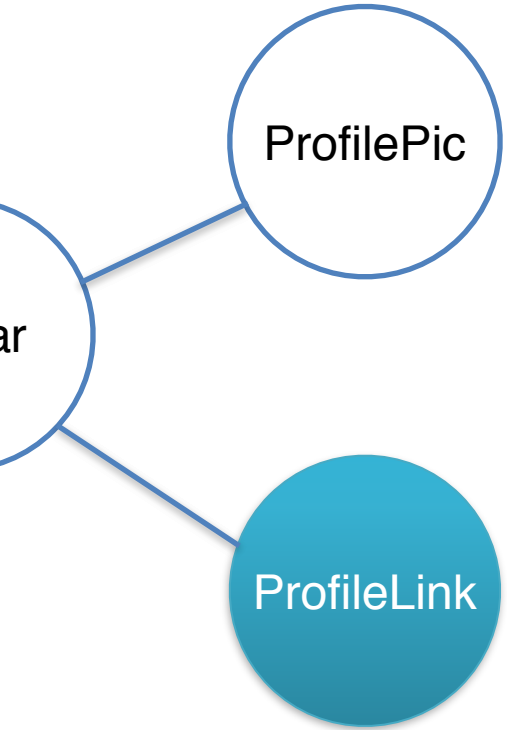
DEVIEW
2015



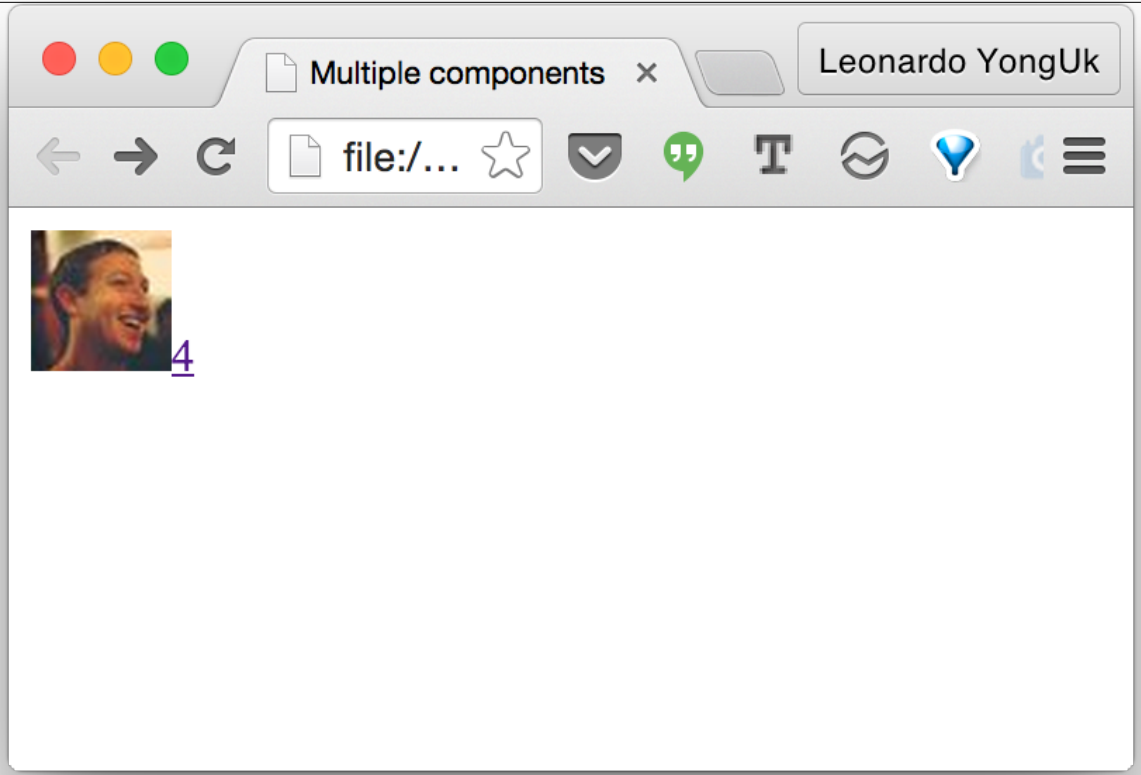

```
var ProfilePic = React.createClass({
  render: function() {
    return (
      <img src={'https://graph.facebook.com/' + this.props.username + '/picture'} />
    );
  }
});
```

1.7 Multiple Components

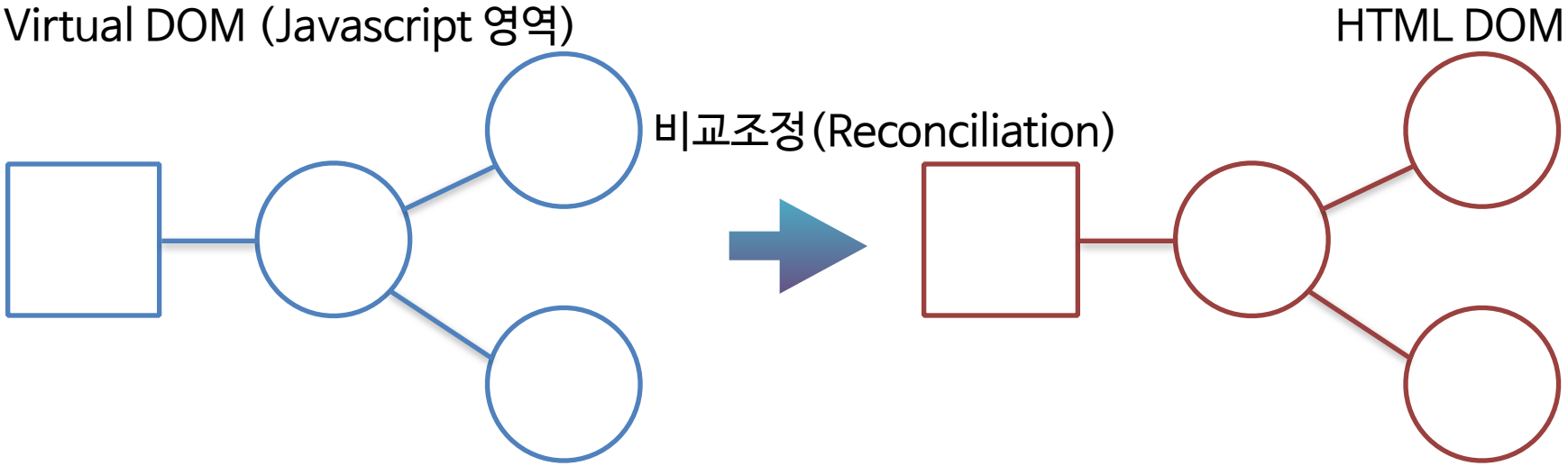
DEVIEW
2015




```
var ProfileLink = React.createClass({
  render: function() {
    return (
      <a href={'https://www.facebook.com/' + this.props.username}>
        {this.props.username}
      </a>
    );
  }
});
```


Multiple components
Leonardo YongUk
file:/...
4

1.8 Virtual DOM



상태(state)가 바뀌면 항상 Virtual DOM은 그려진다.
Virtual DOM이 변경되어도 실질적인 변경이 있을 때만 DOM에 적용된다.
개발자도 상태를 신경쓸 필요가 거의 없고 성능의 감소도 적다.

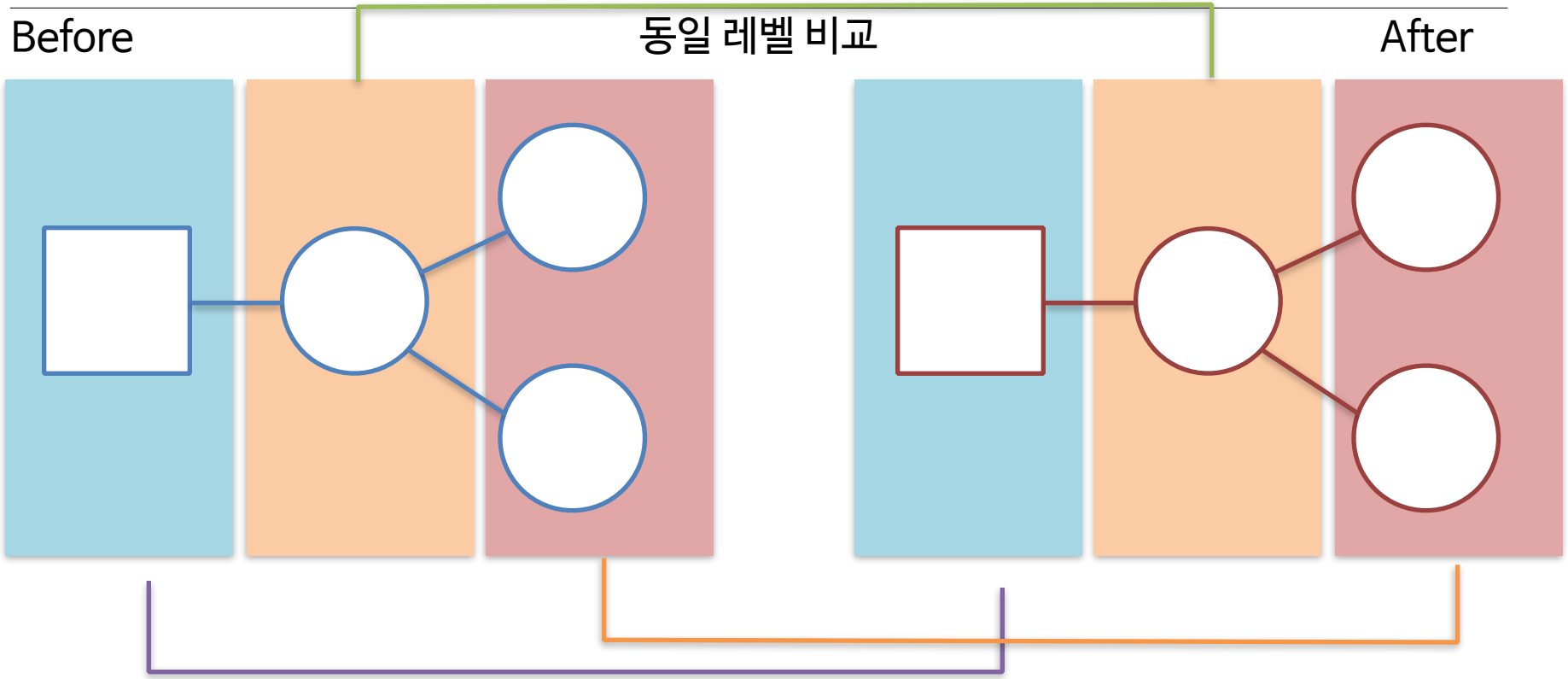
1.8 Virtual DOM

비교조정은 아직 효율적인 알고리즘을 발견하지 못했다. O(n^3)
http://grfia.dlsi.ua.es/ml/algorithms/references/editsurvey_bille.pdf

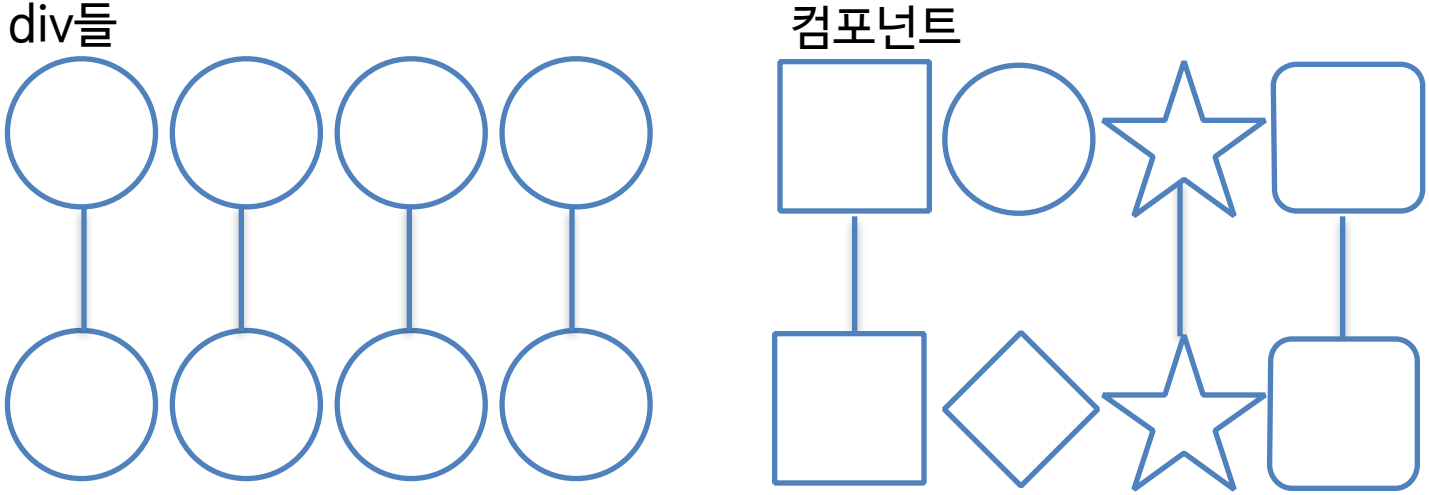
React는 휴리스틱을 동원하여 현실적인 효율을 높인다.

1.8 Virtual DOM

DEVIEW
2015



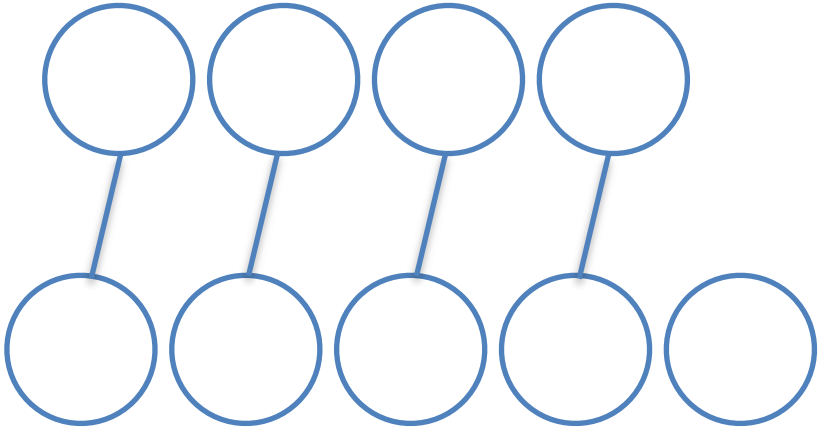
1.8 Virtual DOM



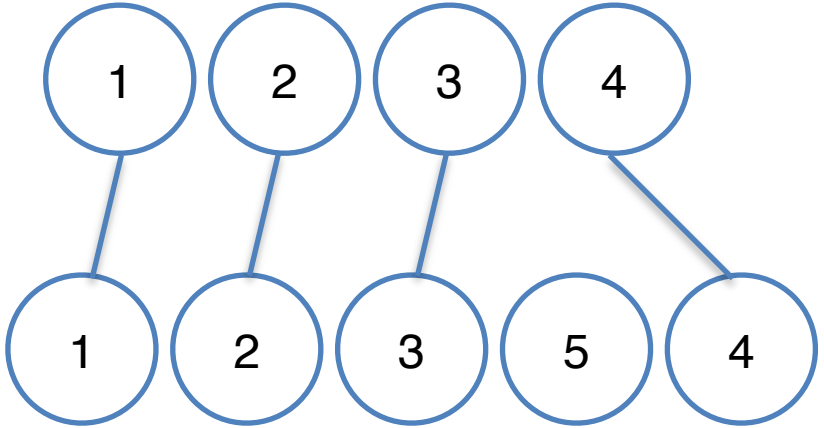
같은 콤퍼넌트에 대해서 재사용을 시도하고 다른 콤포넌트는 지우고 생성한다.
다른 콤퍼넌트는 다른 트리를 만들어 낼 것으로 예상한다.

1.8 Virtual DOM

키가 없는 경우



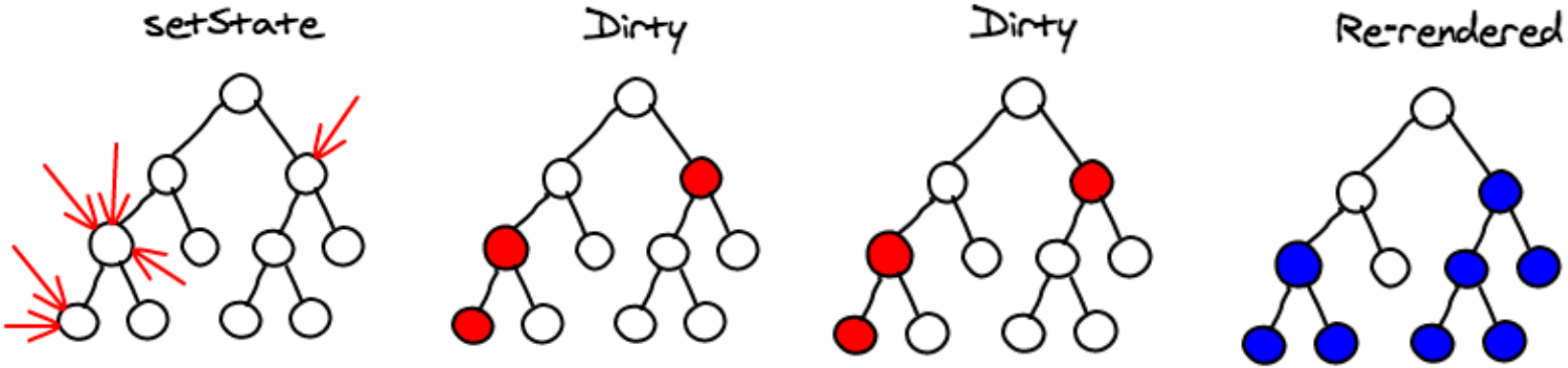
키가 있는 경우



리스트의 경우 키(key)가 없는 콤퍼넌트는 첫번째 요소부터 재사용을 한다.
이 경우 state의 값은 날라가고 소유주(상위 클래스)가 값을 되살릴 의무가 있다.
키가 있는 경우에는 재사용에 앞서 순서를 조절하여 활용한다.

1.8 Virtual DOM

DEVIEW
2015



https://facebook.github.io/react/docs/reconciliation-ko-KR.html

http://calendar.perfplanet.com/2013/diff/

1.9 Life Cycle

DOM에 붙을 예정

DOM에 붙었음

프로퍼티 받을 것임

업데이트 해야하나?

컴포넌트 업데이트 예정

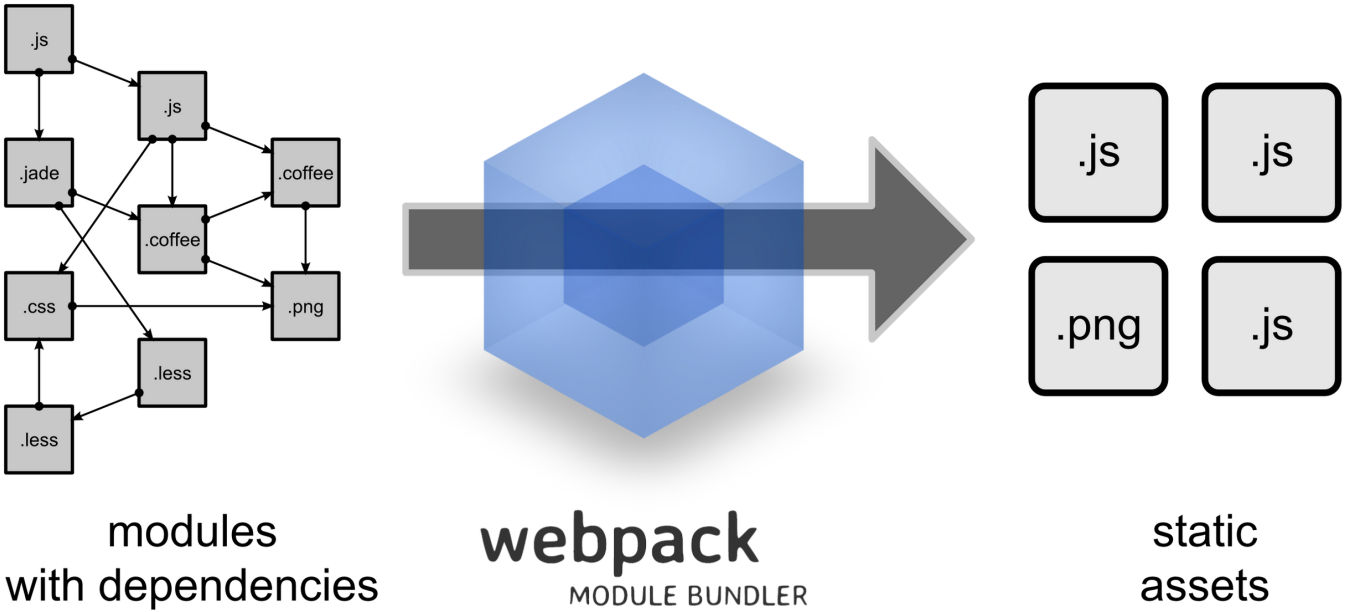
컴포넌트 업데이트

DOM에서 떨어질 예정

2.
웹팩,핫 모듈 대체, 유니버설 렌더링

2.1 Webpack이 뭐에요?

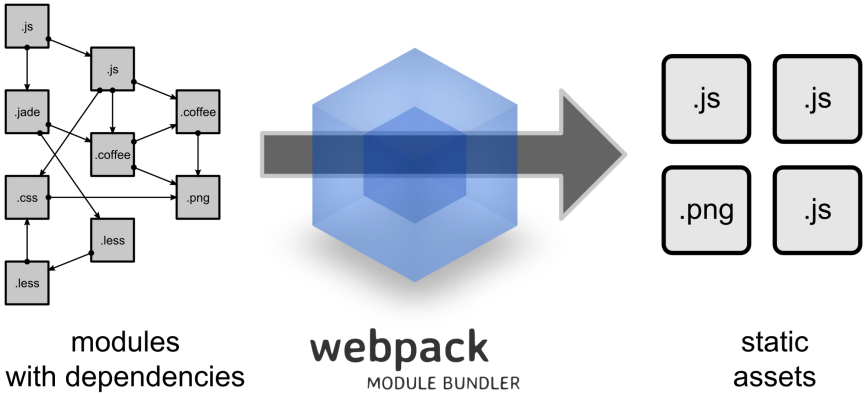
DEVIEW
2015



modules
with dependencies

static
assets

2.1 Webpack이 뭐에요?



의존성을 타고 로더를 적용해 스태틱 에셋으로 변환.
- CommonJs 모델과 AMD모두 지원

다양한 로더를 통해 확장성을 지원.
- Babel을 통한 ES6, 7의 지원. SCSS, Less 등의 지원. 인라인 파일.
- 핫 모듈 대체를 이용하여 실시간으로 기능을 갱신.

http://webpack.github.io/

2.2 웹팩, 스태틱 버전 패키지 설치

npm install webpack -g	웹팩
npm install babel-core —save-dev	바벨 번역기 (이제 dev에서만 필요합니다.)
npm install babel-loader —save-dev	바벨 로더 (웹팩에서 바벨을 수행하기 위함)
npm install react —save	리액트

2.3 웹팩, 설정 파일(webpack.config.js)

DEVIEW 2015

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

module.exports = {
  entry: './hello.jsx',
  output: {
    path: __dirname + '/build',
    filename: 'hello.js'
  },
  module: {
    loaders: [
      {
        test: /.jsx?$/,
        loader: 'babel',
        exclude: /node_modules/
      }
    ]
  }
};
```

entry의 hello.jsx 파일을 타고 들어갑니다.

출력될 파일은 현재디렉토리/build/hello.js

.jsx? 로 끝나는 파일을 찾아
/node_modules/이 포함된 것은 빼고
babel 로더(jsx 처리, ES6)를
적용합니다.

2.4 웹팩, hello.jsx

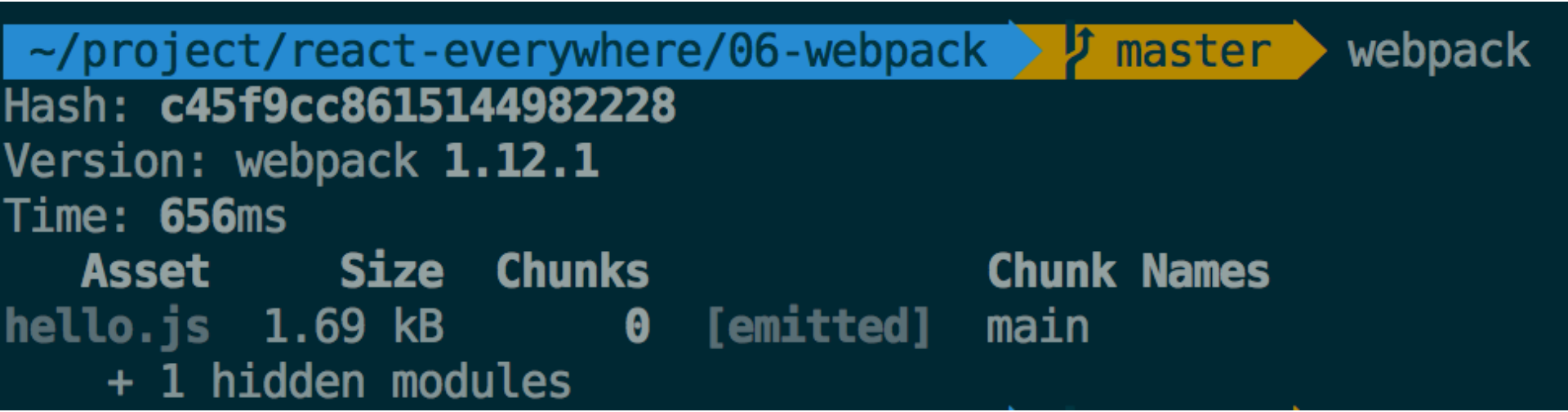
DEVIEW
2015

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

var HelloClass = React.createClass({
  render: function() {
    return (
      <h1>Hello, class!</h1>
    );
  }
});
React.render(
  <HelloClass />,
  document.getElementById('example')
);
```

별도의 JSX 파일에서는 CommonJs의 방식으로 react를 참조해야 합니다.

document를 참조할 때는 HTML에서 head가 아닌 body에서 참조해야한다.

2.3 웹팩, 수행


~/project/react-everywhere/06-webpack
master
webpack
Hash: c45f9cc8615144982228
Version: webpack 1.12.1
Time: 656ms
Asset Size Chunks Chunk Names
hello.js 1.69 kB 0 [emitted] main
+ 1 hidden modules

2.3 hello.jsx (ES6)

DEVIEW 2015

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

var HelloClass = React.createClass({
  render: function() {
    return (
      <h1>Hello, class!</h1>
    );
  }
});
React.render(
  <HelloClass />,
  document.getElementById('example')
);
```

```
import React from 'react';

class HelloClass extends React.Compone
  render() {
    return <h1>Hello, class!</h1>
  }
}

React.render(
  <HelloClass />,
  document.getElementById('example')
);
```

2.3 hello.jsx (ES6)


```
import React from 'react';

class HelloClass extends React.Component {
  render() {
    return <h1>Hello, class!</h1>
  }
}

React.render(
  <HelloClass />,
  document.getElementById('example')
);
```

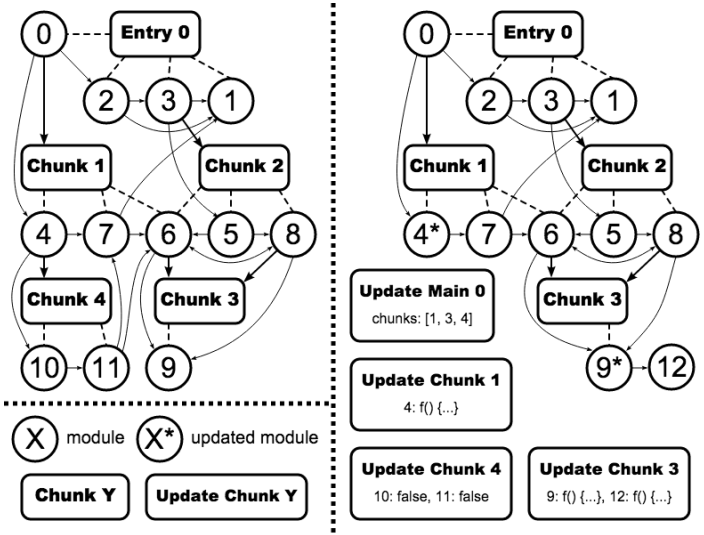
ES6 import

상속기반의 인터페이스

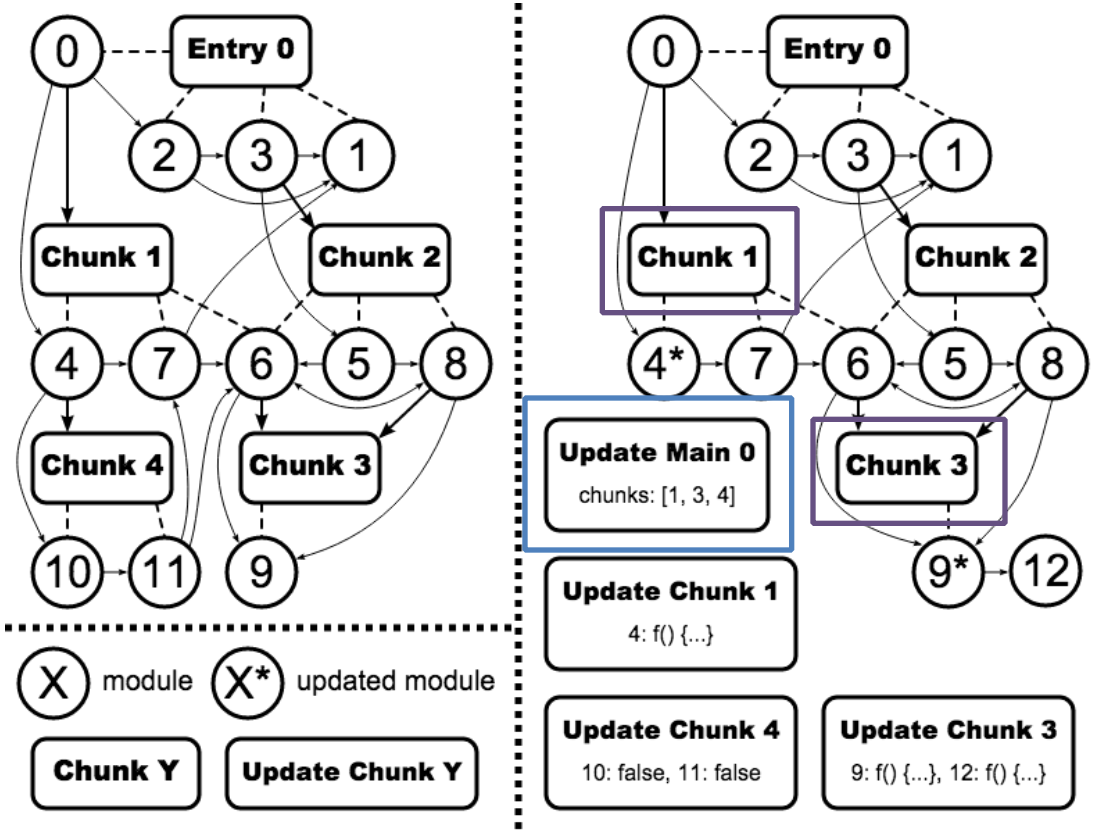
2.4 Hot Module Replacement

동적으로 코드 모듈을 교체하는 기법.

Hot Module Replacement를 지원하는 모듈은 업데이트를 가지고 있다.
- 업데이트 메니페스트 (json)
- 하나 이상의 업데이트 청크(chunks) (js)



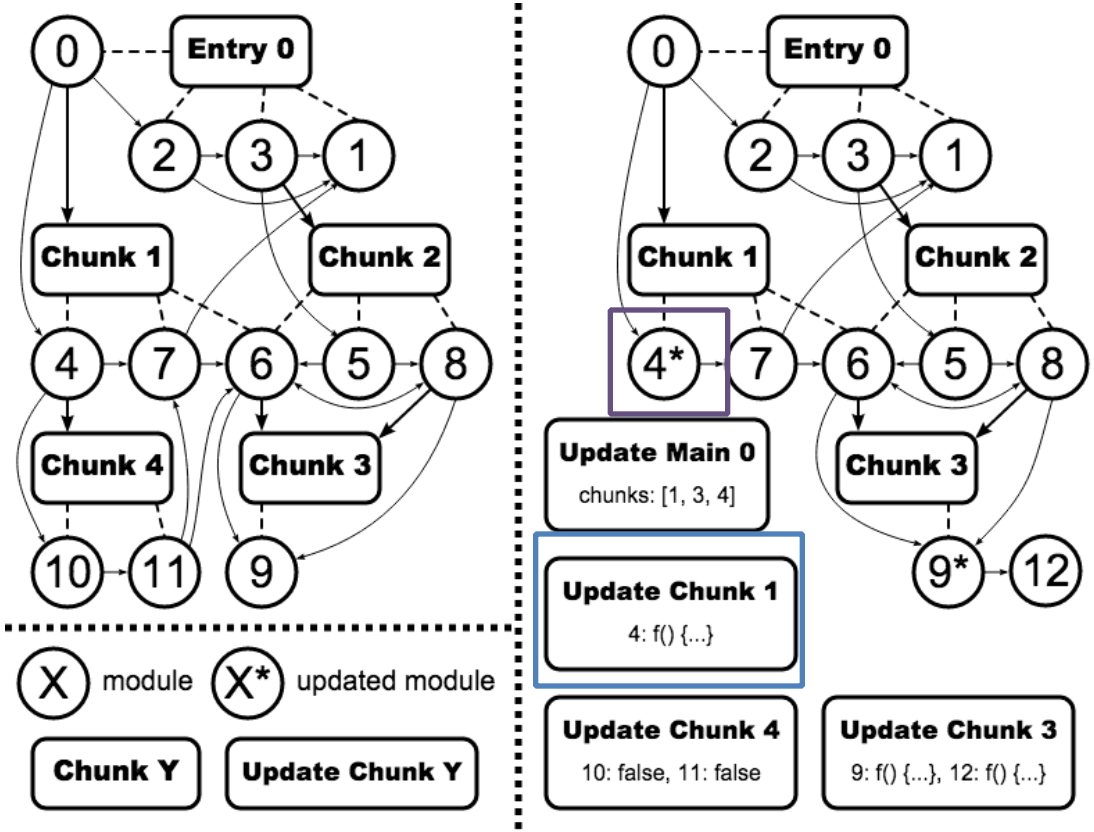
2.4 Hot Module Replacement



청크 1, 3, 4를 갱신하는
업데이트 메인 0

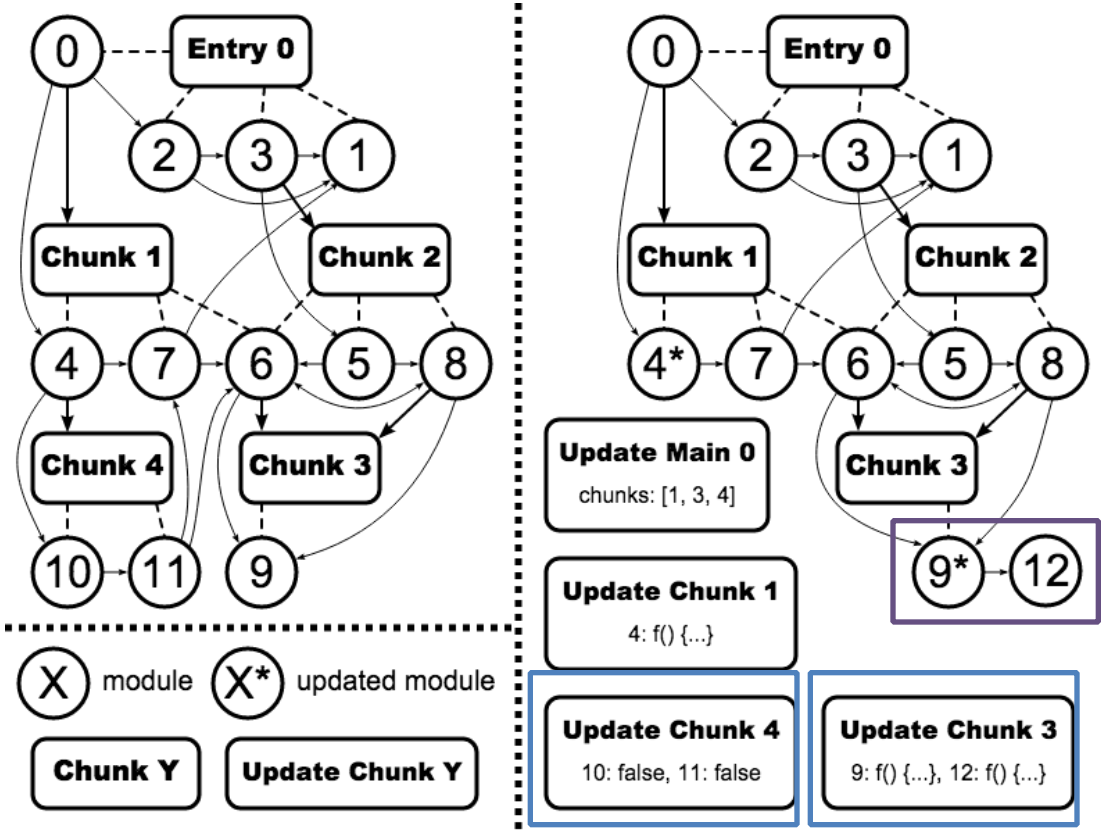
2.4 Hot Module Replacement

DEVIEW
2015



모듈 4를 교체하는 청크 1

2.4 Hot Module Replacement



모듈 9를 교체하고 12를 추가하는 청크 3

청크 4는 모듈 10과 11을 제거했다.

2.4 Hot Module Replacement

DEVIEW
2015

- 모듈 단위로 라이브리로드가 가능.
- 프로덕션 레벨에서 사용 가능 (이라고 설명하고 있지만 WebpackDevServer에서만 봤음.)
- 코드 스플리팅을 통해 필요한 부분만 다운로드.
- 부분적으로 적용이 가능하고 HMR 코드를 비활성화하면 관련 코드 제거.

이미 만들어진 HMR을 씁시다. (eg. React HMR, Redux HMR)

제작에 관심이 있는 분:

- example: http://webpack.github.io/example-app/
- API: http://webpack.github.io/docs/hot-module-replacement.html

2.4 Hot Module Replacement

DEVIEW 2015

```
module: {
  loaders: [
    {
      test: /\.js$/,
      exclude: /node_modules/,
      loaders: ["react-hot", "babel-loader"],
    },
    {
      test: /\.html$/,
      loader: "file?name=[name].[ext]",
    }
  ]
}
```

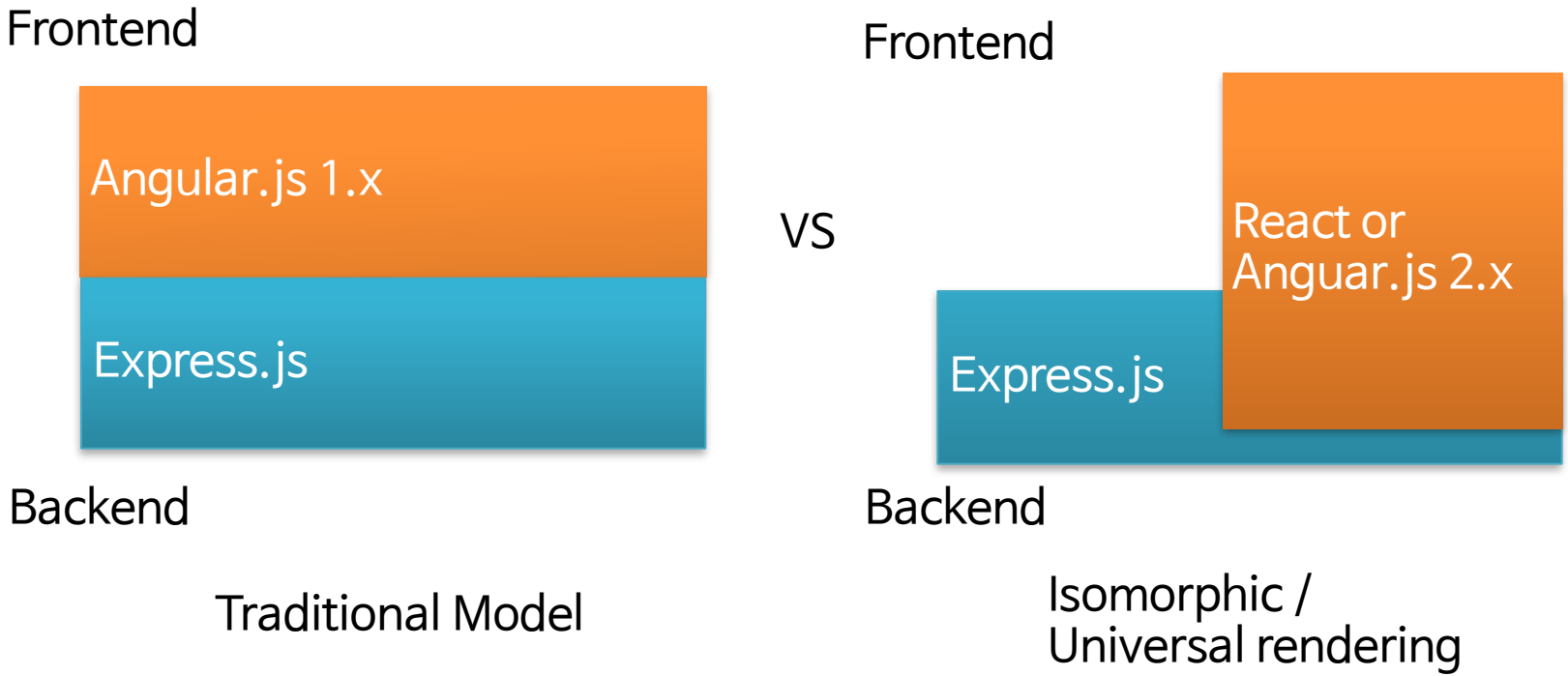
Thanks, Dan Abramov

2.4 Hot Module Replacement

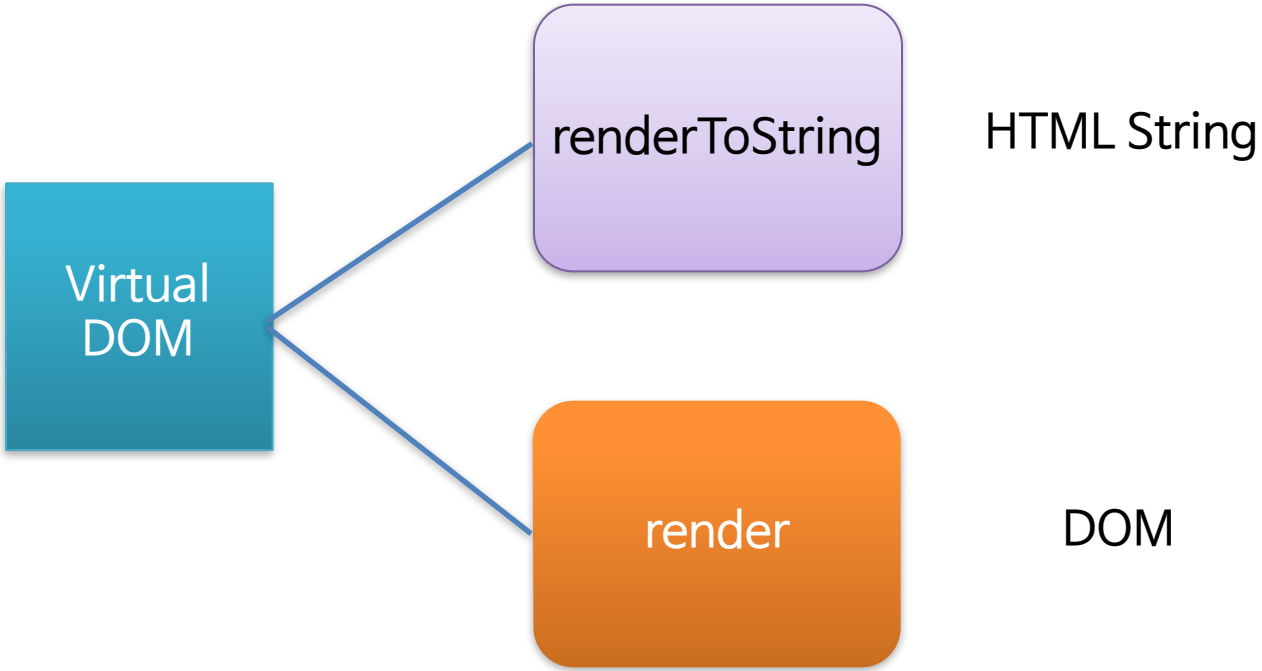
npm start

```
"scripts": {
  "start": "webpack-dev-server --hot --inline"
},
```

2.5 Universal Rendering



2.5 Universal Rendering



3.
리액트 네이티브, 일렉트론 (데모)

4
다음은?

4.1 react.careers

Search for React.js jobs at Airbnb

Search for React.js jobs at BBC

Search for React.js jobs at Codecademy

Search for React.js jobs at Conversocial

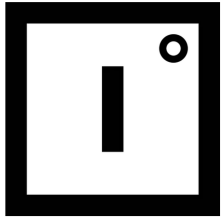
Search for React.js jobs at eBay

Search for React.js jobs at Facebook

Search for React.js jobs at Flipboard

Search for React.js jobs at Hipchat

Search for React.js jobs at Holiday Extras



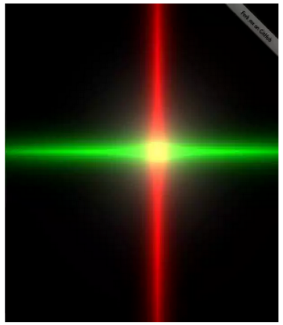
Search for React.js jobs at Improbable.io

Search for React.js jobs at Instagram

Search for React.js jobs at Iodine

4.2 react.rocks

83 Animation Examples

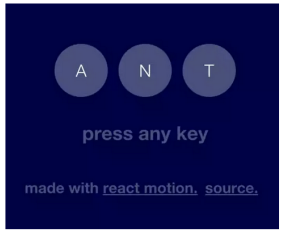


GL react Spring: Dynamic WebGL shader with react-motion spring. Very fluid.

```
<Spinner spinnerName='three-bounce'/>
<Spinner spinnerName='double-bounce'/>
<Spinner spinnerName='rotating-plane'/>
<Spinner spinnerName='wave'/>
<Spinner spinnerName='wandering-cubes'/>
```

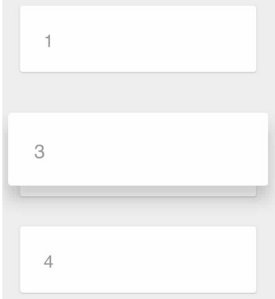
React spinkit: A collection of loading indicators animated with CSS

SVG Watch: Smooth analog watch face. SVG.



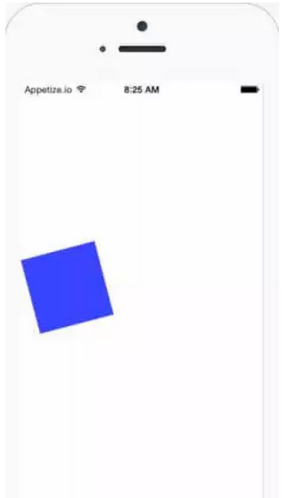
React motion TransitionSpring: Super-smooth interruptible spring-based animation with react-motion

Endangered Birds: Bird database with smooth animations using addEventListener(transitionend). Chart.js radar charts.



React motion reorder list: Smooth list reordering with react-motion springs. Dragged element comes 'forward' with scaling and stronger box-shadow.

Shape interpolation: React-motion + React-canvas, interpolating between shapes



4.3 참고

- React 한글 버전 문서:
  http://reactkr.github.io/react/docs/getting-started-ko-KR.html

- Flux 한글 버전 문서:
  http://haruair.github.io/flux/docs/overview.html#content

- Redux 한글버전 문서:
  http://dobbit.github.io/redux/

- 페이스북 그룹 React Korea와 Reactist

4.3 참고

- Unidirectional User Interface Architectures
  http://staltz.com/unidirectional-user-interface-architectures.html

- React Router
  https://rackt.github.io/react-router/

Q&A

Thank You