BApp Dev Lab 1

Frontend Programming Part 1

React, React.js, ReactJS

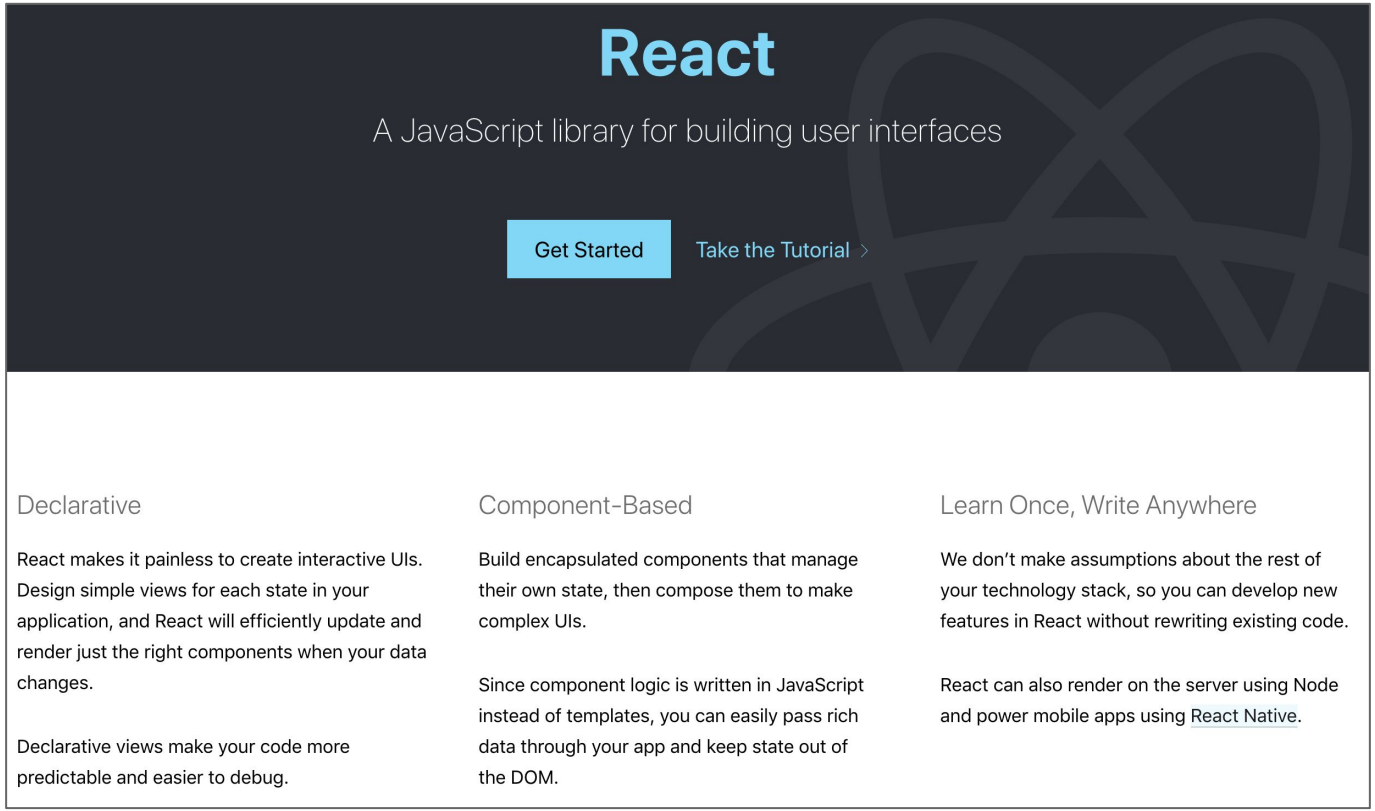
리액트 (자바스크립트 라이브러리)

위키백과, 우리 모두의 백과사전.

컴퓨팅에서 **리액트**(React, **React.js** 또는 **ReactJS**)는 자바스크립트 라이브러리의 하나로서[2] 사용자 인터페이스를 만들기 위해 사용된다. 페이스북과 개별 개발자 및 기업들 공동체에 의해 유지보수된다.[3][4][5]

리액트는 싱글 페이지나 모바일 애플리케이션의 개발 시 토대로 사용될 수 있다. 복잡한 리액트 애플리케이션들은 상태 관리, 라우팅, API와의 통신을 위한 추가 라이브러리의 사용이 일반적으로 요구된다.[6][7]

Reactjs.org


React
A JavaScript library for building user interfaces
Get Started
Take the Tutorial
Declarative
React makes it painless to create interactive UIs. Design simple views for each state in your application, and React will efficiently update and render just the right components when your data changes.
Declarative views make your code more predictable and easier to debug.
Component-Based
Build encapsulated components that manage their own state, then compose them to make complex UIs.
Since component logic is written in JavaScript instead of templates, you can easily pass rich data through your app and keep state out of the DOM.
Learn Once, Write Anywhere
We don't make assumptions about the rest of your technology stack, so you can develop new features in React without rewriting existing code.
React can also render on the server using Node and power mobile apps using React Native.

React 자습서 (https://ko.reactjs.org/tutorial/tutorial.html)

자습서: React 시작하기

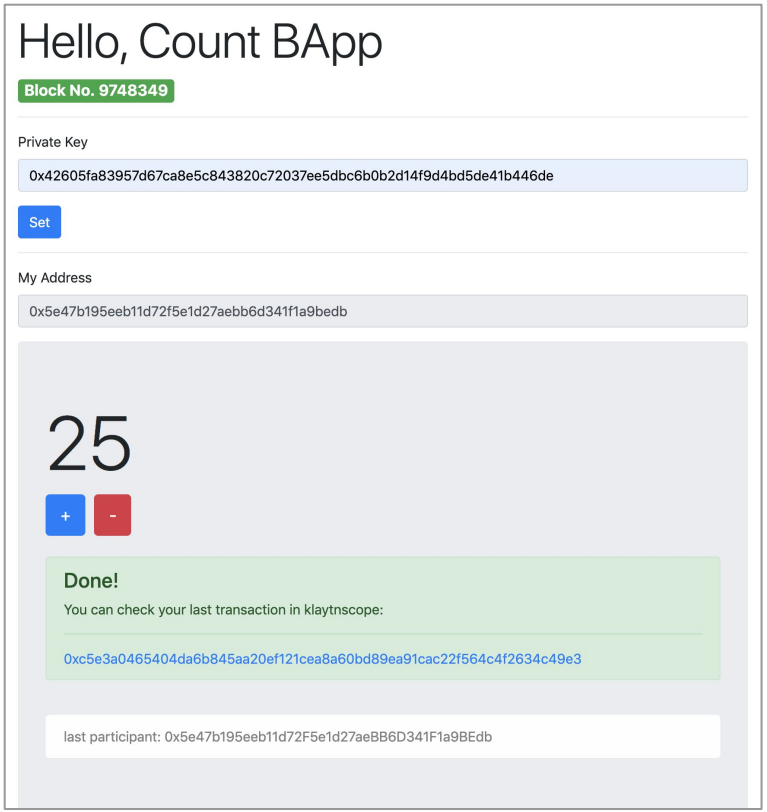
이 자습서는 React에 대한 어떤 지식도 가정하지 않습니다.

자습서를 시작하기 전에

우리는 이 자습서에서 작은 게임을 만들겁니다. **게임을 만들고 싶지 않아서 자습서를 건너뛰고 싶을 수 있습니다. 그래도 한번 해보세요!** 자습서를 통해 React 앱을 만드는 기본적인 사항들을 배웁니다. 이것을 마스터하면 React를 더 깊게 이해할 수 있습니다.

자습서는 집에서 각자 봅니다 ~~(과제아님)~~

Count BApp Reference (https://countbapp.herokuapp.com)



- Count BApp 예시 (실행가능)
- React로 개발
- CSS를 사용한 스타일 적용은 옵션
- 코드는 다음 저장소를 참조: https://github.com/w3kim/countbapp

개발 순서

1. 리액트 프로젝트 생성하기
2. 첫번째 컴포넌트 만들기
3. 컴포넌트 추출하기
4. Klaytn과 통신하는 컴포넌트 만들기
5. 비밀키 입력폼 만들기
6. Count 컨트랙트 호출하기

리액트 + Klaytn 프로젝트 생성하기

```
$ npx create-react-app count

$ cd count

$ npm install caver-js

$ rm -f src/*
```

Hello, World!

- VS Code로 src/index.js 파일을 생성 후 다음과 같이 작성

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

- 터미널에서 count 디렉토리 위치로 이동 후 다음 명령어를 실행

```
$ npm start
```

*ko.reactjs.org/docs에서 발췌했습니다

엘리먼트

엘리먼트는 React 앱의 가장 작은 단위입니다.

엘리먼트는 화면에 표시할 내용을 기술합니다.

```
const element = <h1>Hello, world</h1>;
```

브라우저 DOM 엘리먼트와 달리 React 엘리먼트는 일반 객체이며(plain object) 쉽게 생성할 수 있습니다.
React DOM은 React 엘리먼트와 일치하도록 DOM을 업데이트합니다.

*ko.reactjs.org/docs에서 발췌했습니다

렌더링

HTML 파일 어딘가에 `<div>`가 있다고 가정해 봅시다.

```
<div id="root"></div>
```

이 안에 들어가는 모든 엘리먼트를 React DOM에서 관리하기 때문에 이것을 "루트(root)" DOM 노드라고 부릅니다.

React로 구현된 애플리케이션은 일반적으로 하나의 루트 DOM 노드가 있습니다. React를 기존 앱에 통합하려는 경우 원하는 만큼 많은 수의 독립된 루트 DOM 노드가 있을 수 있습니다.

React 엘리먼트를 루트 DOM 노드에 렌더링하려면 둘 다 `ReactDOM.render()`로 전달하면 됩니다.

```
const element = <h1>Hello, world</h1>;
ReactDOM.render(element, document.getElementById('root'));
```

*ko.reactjs.org/docs에서 발췌했습니다

컴포넌트

- 컴포넌트를 사용하여 재사용할 수 있는 UI를 생성

```
function Welcome(props) {
  return <h1>Hello, {props.name}</h1>;
}
```

```
class Welcome extends React.Component {
  render() {
    return <h1>Hello, {this.props.name}</h1>;
  }
}
```

*ko.reactjs.org/docs에서 발췌했습니다

컴포넌트 렌더링

- 컴포넌트를 엘리먼트로 취급 가능
- 엘리먼트를 DOM 렌더 함수에 넣었던 것과 동일하게 컴포넌트를 렌더가능

```
function Welcome(props) {
  return <h1>Hello, {props.name}</h1>;
}

const element = <Welcome name="Sara" />;
ReactDOM.render(
  element,
  document.getElementById('root')
);
```

*ko.reactjs.org/docs에서 발췌했습니다

App 컴포넌트

- VS Code로 src/index.js 파일을 열어 다음과 같이 수정

```
import React from 'react';
import ReactDOM from 'react-dom';

class App extends React.Component {
    render() {
        return (
            <div className="App">
                <h1>Hello, Count BApp</h1>
            </div>
        );
    }
}

ReactDOM.render(
    <App />,
    document.getElementById('root')
);
```

*ko.reactjs.org/docs에서 발췌했습니다

컴포넌트 추출

- h1 엘리멘트를 추출하여 Greeting 컴포넌트를 생성

```
class Greeting extends React.Component {
    render() {
        return (
            <div><h1>Hello, Count BApp</h1></div>
        );
    }
}

class App extends React.Component {
    render() {
        return (
            <div className="App">
                <Greeting />
            </div>
        );
    }
}

// below unchanged
```

*ko.reactjs.org/docs에서 발췌했습니다

Klaytn과 통신하는 컴포넌트

- 컴포넌트는 별도의 파일로 분리 가능
- index.js가 위치한 디렉토리에 BlockNumber.js라는 파일을 생성

```
import React from 'react'
import Caver from 'caver-js'

const caver = new Caver('https://api.baobab.klaytn.net:8651/')

class BlockNumber extends React.Component {
  state = { currentBlockNumber: '...loading' }

  render() {
    const { currentBlockNumber } = this.state
    return (
      <div>Block No. {currentBlockNumber}</div>
    )
  }
}
export default BlockNumber
```

getBlockNumber

- getBlockNumber 라는 함수를 BlockNumber 컴포넌트에 추가
- Klaytn 블록체인의 가장 최신 블록 번호를 가져오는 것이 목적
- Klaytn과 통신하기 때문에 async를 사용하여 함수를 정의

```
getBlockNumber = async () => {
  const blockNumber = await caver.klay.getBlockNumber()
  this.setState({ currentBlockNumber: blockNumber })
}
```

컴포넌트 라이프사이클

- 모든 컴포넌트는 추가시점, 분리시점에 함수를 동작시킬 수 있음
- 위와 같은 특정 시점을 컴포넌트의 라이프사이클이라 정의
  - 컴포넌트가 DOM에 렌더링 되는 시점을 'Mount' 시점
  - 컴포넌트가 DOM에서 분리되는 시점을 'Unmount' 시점
- 컴포넌트에 componentDidMount와 componentWillUnmount 함수를 추가하여 렌더링, 분리 시점에 컴포넌트가 어떻게 동작해야 하는지 정의

BlockNumber 컴포넌트 라이프사이클

- 다음 상태변수와 함수들을 추가하여 인터벌을 구현
- 인터벌은 특정 코드를 주기적으로 실행할 때 유용

```
intervalId = null

componentDidMount() {
  this.intervalId = setInterval(this.getBlockNumber, 1000)
}

componentWillUnmount() {
  if (this.intervalId) clearInterval(this.intervalId)
}
```