

## React Study 5강

### 목 차

1. 리액트 비동기처리
2. 리액트 프로미스
3. 리액트 어싱크
4. 유튜브 API
5. 퍼블리싱

2018. 11. 22

# 1. 리액트 비동기처리

---

- 비동기처리  
특정코드의 연산이 끝날때까지 코드의 실행을 멈추지 않고  
다음코드를 먼저 실행하는 자바스크립트 특성 입니다.
- 왜 이렇게 필요한가?  
자바스크립트의 경우 대부분이 클라이언트 코드이므로 서버에 요청을 하는 경우  
이런 요청이 많은데 요청이 다 끝날때까지 기다리면  
서버측에서 요청을 주지않으면 다른 작업을 못하는 상황이 발생
- 비동기코드의 대표적인 경우가 setTimeout
- 리액트 컴포넌트 하나 만들어 보면서 감을 잡죠
- 라이프사이클인
- constructor에 1부터 9 까지 로그 찍어봅니다.

```
class App extends Component {  
  constructor(props) {  
    super(props)  
    //동기코드1  
    for (var i = 0; i < 10; i++) {  
      console.log(i);  
    }  
    console.log('done');  
  }  
}
```

# 1. 리액트 비동기처리

- 컴포넌트윌마운트 라이프사이클 api에 비동기코드 만들어봅니다.
- 코드로 보면 hello -> bye -> hello again
- 실제 로그는

	App.js:2
done	App.js:11
Hello	App.js:17
Hello Again	App.js:21

```
componentWillMount() {  
  // 비동기코드2  
  console.log('Hello');  
  setTimeout(function () {  
    console.log('Bye');  
  }, 3000);  
  console.log('Hello Again');  
}
```

- 컴포넌트디드마운트 라이프사이클에도 비동기코드 한번
- 0.01초 뒤에 수행되므로
- 먼저 done 찍고
- 콜백으로 가보면 이미 for문은 맨끝임
- 결국 10만찍고 나오네요.

- 비동기처리가 무엇인지 감을 대충은..

```
componentDidMount() {  
  // 비동기코드3  
  for (var i = 0; i < 10; i++) {  
    setTimeout(function() {  
      console.log(i);  
    }, 10);  
  }  
  console.log('done');  
}
```

# 리액트

---

- 잠시쉬어갑시다.

CALLBACKS, PROMISES, GENERATORS

## CALLBACKS - CALLBACK HELL

```
var function1 = function (callback) {  
  function2(function(error, result) {  
    if (error) { callback(error); return; }  
  
    function3(result, function(error, result) {  
      if (error) { callback(error); return; }  
  
      function4(result, function(error, result) {  
        if (error) { callback(error); return; }  
  
        function5(result, function(error, result) {  
          // ...  
        });  
      });  
    });  
  });  
};
```

## 2. 리액트 프로미스

---

- 비동기처리의 일부분인 콜백으로 인해 코드의 복잡도가 높아지는 것을 해결
- 콜백에서 에러처리가 잘 되지 않는 부분 개선을 위해 프로미스
- 프로미스 생성, 실행은 `new Promise( resolve, reject) {...}`로 생성

결국 비동기함수를 만들어서 사용해야 할 때  
프로미스 객체를 리턴하게 만들어서 사용하면  
콜백헬을 방지하고 에러처리를 수월하게 할 수 있다.

누군가 만들어놓은 대부분의 비동기함수는  
프로미스 객체를 리턴하게 만들어 뒀으므로,  
프로미스를 실행하면 된다.

```
// 콜백에서 에러를 발생시켰는데
try {
  setTimeout( () => { throw 'Error'; }, 1000);
} catch (e) {
  console.log('에러를 캐치하지 못한다.');
  console.log(e);
}
```

## 2. 리액트 프로미스 연습

- 실제 비동기처리는 외부API호출하는 부분에서 많이 사용됨.
- 리액트가 클라이언트이므로 대부분 렌더링이 끝나고
- 초기화면같은것을 보여주고 나서 Loading중 이란 알림 메시지를 주고
- 외부에서 구현된 내역을 호출하고 외부응답이 종료되면
- 콜백에서 응답종료시 할 작업을 지정합니다.

- 비트코인 외부 API를 호출해봅니다.
- 컴포넌트디드마운트 API내에서 호출한후
- 스테이트에 할당해보시죠.

```
constructor(props) {  
  super(props)  
  this.state = {  
    data : []  
  }  
}
```

```
fetch('https://api.coinmarketcap.com/v1/ticker/?limit=10')  
  .then(res => res.json())  
  .then(json => this.setState({ data: json }));  
  
console.log("외부API data **> " + this.state.data.length )
```

← → ↻ <https://api.coinmarketcap.com/v1/ticker/?limit=10>

```
[  
  {  
    "id": "bitcoin",  
    "name": "Bitcoin",  
    "symbol": "BTC",  
    "rank": "1",  
    "price_usd": "4628.20097461",  
    "price_btc": "1.0",  
    "24h_volume_usd": "7565311821.42",  
    "market_cap_usd": "80469775949.0",  
    "available_supply": "17386837.0",  
    "total_supply": "17386837.0",  
    "max_supply": "21000000.0",  
    "percent_change_1h": "0.58",  
    "percent_change_24h": "3.13",  
    "percent_change_7d": "-26.2",  
    "last_updated": "1542799332"  
  },  
  {  
    "id": "ripple",  
    "name": "XRP",  
    "symbol": "XRP",  
    "rank": "2",  
    "price_usd": "0.453628",  
    "price_btc": "0.00003657",  
    "24h_volume_usd": "178602254.46",  
    "market_cap_usd": "3007987432.0",  
    "available_supply": "40327341704.0",  
    "total_supply": "1000000000.0",  
    "max_supply": "1000000000.0",  
    "percent_change_1h": "0.24",  
    "percent_change_24h": "-0.76",  
    "percent_change_7d": "-9.89",  
    "last_updated": "1542799388"  
  },  
  {  
    "id": "ethereum",  
    "name": "Ethereum",  
    "symbol": "ETH",  
    "rank": "3",  
    "price_usd": "178.0039",  
    "price_btc": "0.00403273",  
    "24h_volume_usd": "178602254.46",  
    "market_cap_usd": "3007987432.0",  
    "available_supply": "40327341704.0",  
    "total_supply": "1000000000.0",  
    "max_supply": "1000000000.0",  
    "percent_change_1h": "0.24",  
    "percent_change_24h": "-0.76",  
    "percent_change_7d": "-9.89",  
    "last_updated": "1542799388"  
  },  
  {  
    "id": "cardano",  
    "name": "Cardano",  
    "symbol": "ADA",  
    "rank": "4",  
    "price_usd": "0.1780039",  
    "price_btc": "0.0000403273",  
    "24h_volume_usd": "178602254.46",  
    "market_cap_usd": "3007987432.0",  
    "available_supply": "40327341704.0",  
    "total_supply": "1000000000.0",  
    "max_supply": "1000000000.0",  
    "percent_change_1h": "0.24",  
    "percent_change_24h": "-0.76",  
    "percent_change_7d": "-9.89",  
    "last_updated": "1542799388"  
  },  
  {  
    "id": "dogecoin",  
    "name": "Dogecoin",  
    "symbol": "DOGE",  
    "rank": "5",  
    "price_usd": "0.00001780039",  
    "price_btc": "0.000000403273",  
    "24h_volume_usd": "178602254.46",  
    "market_cap_usd": "3007987432.0",  
    "available_supply": "40327341704.0",  
    "total_supply": "1000000000.0",  
    "max_supply": "1000000000.0",  
    "percent_change_1h": "0.24",  
    "percent_change_24h": "-0.76",  
    "percent_change_7d": "-9.89",  
    "last_updated": "1542799388"  
  },  
  {  
    "id": "dash",  
    "name": "Dash",  
    "symbol": "DASH",  
    "rank": "6",  
    "price_usd": "17.80039",  
    "price_btc": "0.00403273",  
    "24h_volume_usd": "178602254.46",  
    "market_cap_usd": "3007987432.0",  
    "available_supply": "40327341704.0",  
    "total_supply": "1000000000.0",  
    "max_supply": "1000000000.0",  
    "percent_change_1h": "0.24",  
    "percent_change_24h": "-0.76",  
    "percent_change_7d": "-9.89",  
    "last_updated": "1542799388"  
  },  
  {  
    "id": "monero",  
    "name": "Monero",  
    "symbol": "XMR",  
    "rank": "7",  
    "price_usd": "178.0039",  
    "price_btc": "0.00403273",  
    "24h_volume_usd": "178602254.46",  
    "market_cap_usd": "3007987432.0",  
    "available_supply": "40327341704.0",  
    "total_supply": "1000000000.0",  
    "max_supply": "1000000000.0",  
    "percent_change_1h": "0.24",  
    "percent_change_24h": "-0.76",  
    "percent_change_7d": "-9.89",  
    "last_updated": "1542799388"  
  },  
  {  
    "id": "zcash",  
    "name": "Zcash",  
    "symbol": "ZEC",  
    "rank": "8",  
    "price_usd": "17.80039",  
    "price_btc": "0.00403273",  
    "24h_volume_usd": "178602254.46",  
    "market_cap_usd": "3007987432.0",  
    "available_supply": "40327341704.0",  
    "total_supply": "1000000000.0",  
    "max_supply": "1000000000.0",  
    "percent_change_1h": "0.24",  
    "percent_change_24h": "-0.76",  
    "percent_change_7d": "-9.89",  
    "last_updated": "1542799388"  
  },  
  {  
    "id": "bitfury",  
    "name": "Bitfury",  
    "symbol": "BF",  
    "rank": "9",  
    "price_usd": "17.80039",  
    "price_btc": "0.00403273",  
    "24h_volume_usd": "178602254.46",  
    "market_cap_usd": "3007987432.0",  
    "available_supply": "40327341704.0",  
    "total_supply": "1000000000.0",  
    "max_supply": "1000000000.0",  
    "percent_change_1h": "0.24",  
    "percent_change_24h": "-0.76",  
    "percent_change_7d": "-9.89",  
    "last_updated": "1542799388"  
  },  
  {  
    "id": "bitfury",  
    "name": "Bitfury",  
    "symbol": "BF",  
    "rank": "10",  
    "price_usd": "17.80039",  
    "price_btc": "0.00403273",  
    "24h_volume_usd": "178602254.46",  
    "market_cap_usd": "3007987432.0",  
    "available_supply": "40327341704.0",  
    "total_supply": "1000000000.0",  
    "max_supply": "1000000000.0",  
    "percent_change_1h": "0.24",  
    "percent_change_24h": "-0.76",  
    "percent_change_7d": "-9.89",  
    "last_updated": "1542799388"  
  }  
]
```

## 2. 리액트 프로미스 연습

- 렌더링도 비트코인 관련해서 해봅시다.
- 배열을 map함수로

```
return (  
  <div className="App">  
    <h3>비동기 처리 </h3>  
    <ul>  
      {this.state.data.map(bitcoin => (  
        <li>  
          {bitcoin.name}: {bitcoin.price_usd}  
        </li>  
      ))}  
    </ul>  
  </div>  
);
```

← → ↻ ⓘ localhost:3000

### 비동기 처리

- Bitcoin: 4603.34237859
- XRP: 0.4457919742
- Ethereum: 136.285156575

Bitcoin Cash: 241.721539475  
Stellar: 0.2029129996  
EOS: 3.789414897  
Litecoin: 34.3993426967  
Tether: 0.9833428978  
Cardano: 0.0474639131  
Monero: 68.0427928311

## 2. 리액트 프로미스

---

- 프로미스는 말 그대로 약속이다
  - 지금은 없으니깐 이따 준다는 약속
  - 좀더 생각해보면 지금은 없는데 이상없으면 이따가 주고 없으면 알려줄께
  
  - 프로미스는 상태가 있다
  - 펜딩 : 아직 약속을 수행중인상태
  - 풀필드 : 약속이 지켜진 상태
  - 리젝티드 : 약속이 못지켜진 상태
  - 세틀드 : 약속이 결론이 난 상태(지켜졌든 안지켜졌든)
-



## 리액트

---

– 잠시쉬어갑니다.

### 3. 어썬크 어웨이트

---

- 프로미스가 코드가 못생겼다는 비판이 일면서 새로운 대안으로..
- 함수명 앞에 `async`를 붙이면 이는 비동기처리함수
- `await`는 `async`함수내에서 사용
- 
- 한번해보죠
- 외부API 연습용 url
- <https://jsonplaceholder.typicode.com/posts/>
- 사용자정의 함수를 생성합니다.

```
async testExternalApi() {  
  let response = await fetch('https://jsonplaceholder.typicode.com/posts/');  
  let data = await response.json();  
  this.setState({ externalData: data })  
}
```

- `componentDidMount()`에서 호출합니다.
  - rendering 해줍니다.
-

### 3. Async await 자세히

---

- 콜백함수를 사용하기 위해서는 `async`와 `await` 키워드를 사용합니다.

```
1  async function foo(){  
2      await someAsyncFunction(){...}  
3      await anotherAsyncFunction(){...}  
4  }
```

함수 이름 앞에 `async` 키워드를 그리고 호출할 비동기 함수 앞에 `await` 키워드를 사용합니다.

함수 앞에 `async`가 정의되어 있어야만 `await`이 적용된다는 점을 기억해주세요.

위의 코드는 `someAsyncFunction`, `anotherAsyncFunction` 두 함수가 비동기 코드일지라도 `async/await`이 적용되면, 항상 `someAsyncFunction` -> `anotherAsyncFunction` 순서대로 함수가 실행됩니다.

이처럼 비동기 코드를 동기적으로 수행하게 해주는 것이 `async/await`입니다.

이 때 `someAsyncFunction` 과 `anotherAsyncFunction` 함수는 Promise를 리턴해야 합니다.

`async/await`이 Promise 방식을 사용하기 때문입니다.

정말로 Promise 방식을 사용하는지 살펴보도록 하겠습니다.

---

## 리액트

---

– 잠시쉬어갑니다.

## 4. Youtube api 활용 연습

- 구글 개발자 콘솔에서 youtube api 활용을 위한 코드를 받습니다.
- <https://console.cloud.google.com>
- 브라우저 key 복사

The screenshot shows the Google Cloud Platform console interface. The browser address bar displays <https://console.cloud.google.com/apis/credentials?project=fc-chat-test1>. The left sidebar contains a menu with 'API 및 서비스' (APIs & Services) expanded, showing '대시보드' (Dashboard), '라이브러리' (Library), and '사용자 인증 정보' (Credentials) selected. The main content area is titled '사용자 인증 정보' (Credentials) and includes tabs for '사용자 인증 정보' (selected), 'OAuth 동의 화면' (OAuth consent screen), and '도메인 확인' (Domain verification). Below the tabs are buttons for '사용자 인증 정보 만들기' (Create credentials) and '삭제' (Delete). A message states: '사용 설정한 API에 액세스하려면 사용자 인증 정보를 만드세요. 자세한 내용은 API 문서' (To access the APIs you've enabled, create user credentials. For more details, see the API documentation). A table titled 'API 키' (API keys) lists existing keys:

<input type="checkbox"/>	이름	생성일	제한사항	키
<input type="checkbox"/>	⚠ Server key (auto created by Google Service)	2018. 5. 3.	없음	AlzaSyC8Z2TsFTVBK2
<input type="checkbox"/>	⚠ Browser key (auto created by Google Service)	2018. 5. 3.	없음	AlzaSyDcFF9Nln6vbw!

## 4. Youtube api 활용 연습

---

- youtube-api-search API 설치
- npx create-react-app 으로 샘플 프로젝트 만들고
- YTSearch() 함수로 유튜브에서 어떤 항목들로 리턴을 주는지 확인해 본다.
- 유튜브검색 파라미터는 다음과 같다.
- API\_KEY 정보, 검색할단어, 콜백함수

```
YTSearch({ key: API_KEY, term: "크리스마스캐롤" }, data => {  
  console.log(data);  
})
```

- 데이터확인후 콜백된 결과를 스테이트에 넣어준다.
-

## 4. Youtube api 활용 연습

---

- Props를 활용하여 유튜브비디오 배열을 넘겨
- VideoList.js에서 해당배열을 길이를 찍어보자

## 4. Youtube api 활용 연습

---

- 배열을 Map함수를 활용해서 비디오목록갯수만큼 아이템을 찍어보자
-



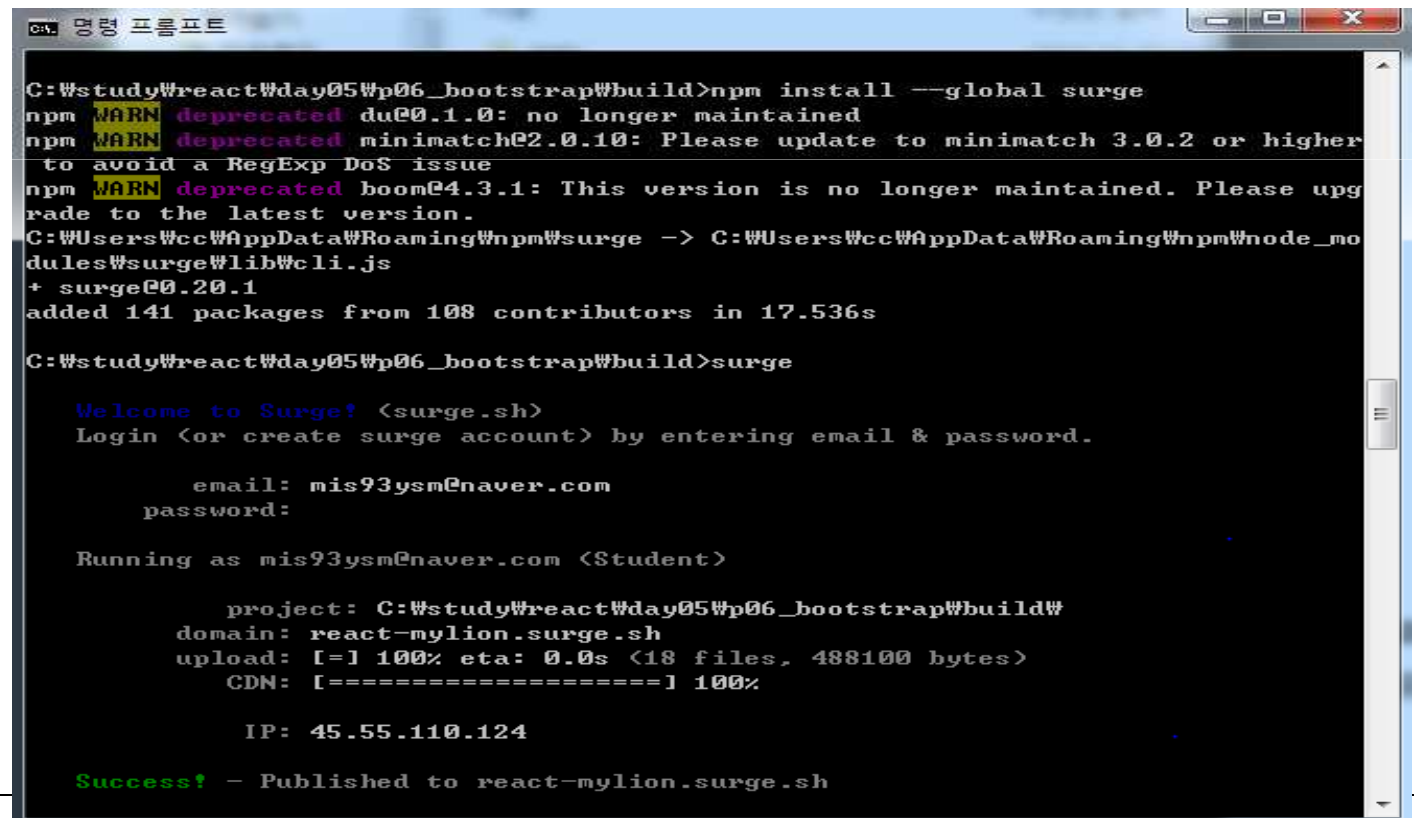
## 4. Youtube api 활용 연습

---

- 실제 비디오 목록에 섬네일, 타이틀, 설명을 찍어보자

## 5. 디플로이 and 퍼블리싱

- 무료 정적파일 서비스인 surge.sh 이용
- 해당 리액트 프로젝트를 build하여 디플로이할 파일 생성
- 디플로이 타겟 폴더로 이동
- `npm -install -global surge`
- `surge`
- 이메일주소와 패스워드로 회원가입후
- 도메인설정후 엔터



```
C:\study\react\day05\p06_bootstrap\build>npm install --global surge
npm WARN deprecated du@0.1.0: no longer maintained
npm WARN deprecated minimatch@2.0.10: Please update to minimatch 3.0.2 or higher
to avoid a RegExp DoS issue
npm WARN deprecated boom@4.3.1: This version is no longer maintained. Please upgrade
to the latest version.
C:\Users\cc\AppData\Roaming\npm\surge -> C:\Users\cc\AppData\Roaming\npm\node_modules\surge\lib\cli.js
+ surge@0.20.1
added 141 packages from 108 contributors in 17.536s

C:\study\react\day05\p06_bootstrap\build>surge

Welcome to Surge! <surge.sh>
Login <or create surge account> by entering email & password.

    email: mis93ysm@naver.com
    password:

Running as mis93ysm@naver.com <Student>

    project: C:\study\react\day05\p06_bootstrap\build\
    domain: react-mylion.surge.sh
    upload: [=] 100% eta: 0.0s <18 files, 488100 bytes>
    CDN: [=====] 100%

    IP: 45.55.110.124

Success! - Published to react-mylion.surge.sh
```

감사합니다.

---