12월 14일 (화)

1. Codecademy - Async JavaScript and HTTP Requests - APIs and HTTP Requests

Article 3. Introduction to Web APIs

- **What are APIs ?**
- **What can APIs do ?**
- **How do APIs work ?**
- **Summary**

출처 : https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Client-side_web_APIs/Introduction

Article 4. What Is JSON ?

- **Introduction**
- **What is JSON ?**
- **Common Uses of JSON**
- **JSON Syntax**
- **JSON Data Types**

- **Introduction**

데이터가 넘치는 세상에서 데이터가 어떻게 작동하는지 아는 것은 훨씬 더 중요해졌다. 프로그래머로서, 우리는 우리가 선택한 모든 언어의 채워진 데이터 구조를 다른 언어와 플랫폼에서 인식하고 읽을 수 있는 형식으로

전송할 수 있어야 한다. 다행히 그러한 데이터 교환 포맷이 존재한다.

- **What is JSON ?**

JSON (JavaScript Object Notation) 은 데이터를 저장하고 교환하기 위한 인기있는 언어 독립적 표준 형식이다. 정보 및 통신 시스템을 표준화하기 위해 1961년에 설립된 ECMA International 에서 채택한 JSON은 모든 프로그래밍 언어 간에 데이터를 저장하고 전송하는 것을 용이하게 하는 표준 형식이 되었다.

- **Common Uses of JSON**

JSON은 웹 브라우저와 같은 클라이언트와 서버 간의 웹 어플리케이션에서 데이터 전송을 용이하게 하는데 많이 사용한다. 데이터 전송이 발생하는 일반적인 예는 웹 form 을 입력할 때 발생한다. form 데이터는 HTML에서 JavaScript 객체로, JSON 객체로 변환되고, 처리를 위해 원격 웹 서버에 보내진다. 일반적으로 데이터를 공유할 때 주고받는 정보는 JSON 형식이다. 즉, 객체형식으로 주고 받는다.

인기 있는 웹 API는 다음과 같다.

- 구글지도
- 구글 인증 2.0 인증
- 페이스북 소셜 그래프 API
- 스포티파이 뮤직 웹 API
- LinkedIn 이력서 API

- **JSON Syntax**

JSON은 JavaScript 프로그래밍 언어에서 파생되었기 때문에 모양이 JavaScript 객체와 유사하다.

아래 예시를 보자

```
{
  "student": {
    "name": "Rumaisa Mahoney",
    "age": 30,
    "fullTime": true,
    "languages": [ "JavaScript", "HTML", "CSS" ],
    "GPA": 3.9,
    "favoriteSubject": null
  }
}
```

- 중괄호로 객체를 구성한다.
- 대괄호로 배열을 구성한다.

- 데이터는 콜론(:) 으로 구분된 이름-값 쌍으로 저장된다.
- 모든 이름-값 쌍은 쉼표(,)로 다른 쌍과 구분한다.
- 후행쉼표는 금지한다. (마지막 이름-값 쌍 뒤에 쉼표를 붙이는 것)
- 프로퍼티 이름은 무조건 쌍따옴표(" ")로 써야한다. single - quoted(' ') 는 허용하지 않는다.

- **JSON Data Types**

데이터 유형은 다음 중 하나여야 한다.

- string (쌍따옴표 사용)
- number (정수 or 소수)
- object (이름- 값 쌍)
- array
- boolean
- null

JSON은 모든 데이터 유형을 다루지 않는다. 날짜와 같이 JSON 으로 표시도지 않는 유형은 문자열로 저장하고 언어별 데이터 구조로 변환할 수 있다. ISO 8601 에서 국제적으로 인정되는 날짜 형식은 다음과 같다.

```
"2014-01-01T23:28:56.782Z"
```

위 형식은 사실 문자열 그대로 읽기 힘들고 사용하기 힘든 형태이다. 편리하게도 모든 프로그래밍 언어는 위 스트링을 훨씬 더 읽기 쉽고, 사용하기 쉬운 형식으로 아래 예시와 같이 전환해주는 빌트인 JSON 기능을 가지고 있다.

```
Wed Jan 01 2014 13:28:56 GMT-1000 (Hawaiian Standard Time)
```

Article 5. Working with JSON in JavaScript

Introduction

JavaScript Object Notation의 약자 JSON은 업계 표준으로 받아들여진 언어 독립적인 데이터 형식이다. JavaScript 프로그래밍 언어를 기반으로 하기 때문에 JSON의 구문은 약간의 차이점이 있는 JavaScript 객체와 유사합니다 . 우리는 그들 사이의 미묘한 차이점을 살펴볼 것입니다. 나중에 JSON을 구문 분석하고 콘텐츠를 JavaScript로 추출하는 방법을 배울 것이다. 마지막으로 JavaScript로 JSON 객체를 작성하는 방법을 배울 것이다.

JSON Object vs. JavaScript Object

```
{
  "person": {
    "name": "Kate",
    "age": 30,
    "hobbies": [ "reading", "writing", "cooking", "tennis" ]
  }
}
```

```
{
  person: {
    name: 'Kate',
    age: 30,
    hobbies: [ 'reading', 'writing', 'cooking', 'tennis' ]
  }
}
```

- **Reading a JSON String**

JSON 파일을 자바스크립트로 읽는 법을 알아보자.

자바스크립트에서는 JSON 클래스안에 있는 .parse( ) 메서드로 JSON 객체를 javascript 객체로 변환해준다. 이를 변환하면 JSON 객체를 자바스크립트 객체처럼 사용할 수 있는 것이다.

```
const jsonData = '{ "book": { "name": "JSON Primer", "price": 29.99, "inStock": true, "rating": null } }';

const jsObject = JSON.parse(jsonData);

console.log(jsObject);

//-------------------This will print out jsObject as follows:

{
  book: { name: 'JSON Primer', price: 29.99, inStock: true, rating: null }
}
```

자바스크립트 객체로 변환한 JSON 객체는 다음과 같이 접근하여 사용할 수 있다. 자바스크립트 객체에 접근하는 방법과 같다.

```
// Using the dot notation
const book = jsObject.book;
console.log(book);
console.log(book.name, book.price, book.inStock);

// Using the bracket notation
const book2 = jsObject['book'];
console.log(book2);
console.log(book2["name"], book2["price"], book2["inStock"]);

//------------Both ways of accessing the book property return the same output:


{ name: 'JSON Primer', price: 29.99, inStock: true, rating: null }
JSON Primer 29.99 true
```

- **Exercise: Reading a JSON String**

```
const jsonData = '{ "parent": {
"name": "Sally", "age": 45,
"children" : [ { "name": "Kim",
"age": 3 }, { "name": "Lee", "age": 1
} ] } }';

const jsObject = JSON.parse(jsonData);

console.log(jsObject.parent.children);
```

```
[ { name: 'Kim', age: 3 }, {
name: 'Lee', age: 1 } ]
```

- **Writing a JSON String**

우리는 웹을 통해 데이터를 어딘가로 보내기 전에, 자바스크립트 객체에 담긴 데이터를 JSON 스트링으로 바꿔야 한다. 자바스크립트에서 우리는 빌트인 JSON 클래스 메서드인 JSON.stringify( ) 메서드를 자바스크립트 객체를 JSON 스트링으로 변환하기 위해 사용할 것이다.

```
const jsObject = { book: 'JSON Primer', price: 29.99, inStock: true, rating: null };
const jsonData = JSON.stringify(jsObject);
console.log(jsonData);

//------------This will display the following output:

{ "book": "JSON Primer", "price": 29.99, "inStock": true, "rating": null }
```

- **Exercise: Writing a JSON String**

개발자는 변수에 JSON 문자열 형식의 일부 데이터를 받습니다 `jsonData`. 그러나 의 내용이 `jsonData`완전히 정확하지는 않습니다. `age`상위 속성 의 값은 `35`대신 이어야 합니다 `45`. 의 내용을 `jsonData`직접 변경하지 않고 `age`값을 업데이트한 다음 콘솔에 올바른 값으로 새 JSON 문자열을 기록합니다.

이 문제를 해결하기 위한 단계별 가이드는 다음과 같습니다.

1. `jsonData`를 사용하여 JavaScript 객체로 변환 `JSON.parse()`하고 `const`변수 로 저장합니다(예: `jsObject`.
2. 점, `.key`또는 대괄호, `['key']`표기법을 사용하여 `parent`속성 `jsObject`다음에 속성에 액세스하고 `age`해당 값을 에서 `45`로 변경 합니다 `35`.
3. 를 `jsObject`사용하여 JSON 문자열로 다시 변환 `JSON.stringify()`하고 다른 const 변수로 저장합니다(예: `jsObjectToJson`.
4. `jsObjectToJson`콘솔에 문자열을 기록합니다 .

```
const jsonData = '{"parent":{"name":"Sally","age":45,"children":[{"name":"Kim","age":3},{"name":"Lee","age":1}]}}';


const jsObject = JSON.parse(jsonData);

jsObject.parent.age = 35;

const jsObjectToJson = JSON.stringify(jsObject);

console.log(jsObjectToJson);
```

```
{"parent":{"name":"Sally","age":35,"children":[{"name":"Kim","age":3},{"name":"Lee","age":1}]}}
```

- **Review**

JSON 과 JavaScript 객체 문법의 차이점을 비교하였다.

JSON string 을 JavaScript 객체로 전환시키는 법 - JSON.parse(JsonData)

JavaScript 객체를 JSON string 으로 전환시키는 법 - JSON.stringify(jsObject)

2. Codecademy- Async JavaScript and HTTP Requests - Learn JavaScript : Requests

Lesson 1. Requests -1-

- Introduction to Requests
- HTTP Requests
- XHR GET Requests -1-
- XHR GET Requests -2-
- XHR GET Requests -3-
- XHR GET Requests -4-
- XHR POST Requests -1-
- XHR POST Requests -2-
- XHR POST Requests -3-
- Review

- **Introduction to Requests**

폼을 입력 후 제출하게 되면 서버로 데이터가 전송된다. 이러한 전송을 위한 요청은 HTTP 요청으로 이루어진다. HTTP 요청에는 대표적으로 4가지가 가장 많이 쓰이는데 GET, POST, PUT, DELETE 로 이루어져있다. GET 요청은 서버로부터 데이터를 얻을 때 사용하며, POST 는 서버로 데이터를 보낼 때 사용한다. 이번 레슨에서 GET, POST 에 대해서 배울 것이다.

이번 레슨에서는 자바스크립트의 XHR 객체를 사용하여 어떻게 GET 과 POST 요청이 만들어지는지 배워볼 것이다. 우리는 GET 요청을 위해서 Datamuse API 를 사용할 것이며 POST 요청을 위해서 Rebrandly URL Shortener API 를 사용할 것이다.

API 연결 : https://www.codecademy.com/articles/rebrandly-signup

- **HTTP Requests**

자바스크립트의 가장 큰 자산중 하나는 논블로킹 프로퍼티 또는 비동기적 언어라는 것이다.

신문 웹사이트와 같은 사이트는 이러한 논블로킹 속성을 활용하여 사용자에게 더 나은 경험을 제공한다. 일반적으로 큰 용량의 이미지를 불러올 때 비동기적으로 텍스트를 먼저 렌더링하고 이미지가 비동기적으로 요청이 완료되면 렌더링된다.

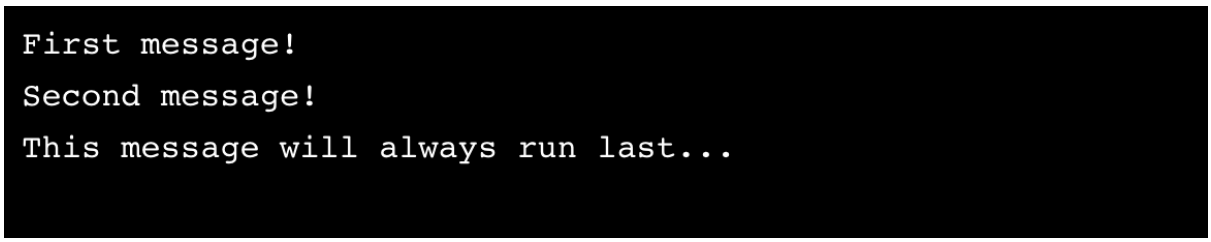
자바스크립트는 이벤트 루프를 사용하여 비동기 함수 호출을 처리한다. 프로그램이 실행될 때 함수 호출이 만들어지고 스택에 추가된다. 서버가 응답할 때까지 기다려야 하는 요청을 만든 다음 별도의 큐(대기열)로 보내지는 함수이다. 스택이 지워지면 큐의 기능이 실행된다.

웹 개발자는 이벤트 루프를 사용하여 함수를 호출할 시기와 비동기 이벤트를 처리하는 방법을 결정하여 보다 부드러운 브라우징 경험을 만든다. 비동기 자바스크립트 및 XML 또는 AJAX 라는 기술 시스템을 살펴보자

출처: 이벤트 루프 https://developer.mozilla.org/en-US/docs/Web/JavaScript/EventLoop

실습 예제

```
console.log('First message!');
setTimeout(() => {
   console.log('This message will always run last...');
}, 2500);
console.log('Second message!');
```




```
console.log('First message!');
setTimeout(() => {
   console.log('This message will always run last...');
```

```
}, 0);
console.log('Second message!');
```

```
First message!
Second message!
This message will always run last...
```

- **XHR GET Requests -1-**

AJAX(Asynchronous JavaScript And XML)는 초기 페이지 로드후에 요청을 생성하도록 한다. AJAX는 XML 형식의 데이터에만 사용되었다. 하지만 이제는 다양한 형식의 요청을 만드는 데 사용할 수 있다.

마찬가지로 XML의 이름을 따서 명명된 XMLHttpRequest(XHR) API는 여러 종류의 요청을 수행하고 다른 형식의 데이터를 지원하는 데 사용할 수 있다.

GET을 사용하여 소스에서 데이터를 검색한다는 것을 기억해라. XHR GET요청을 만드는 방법을 보려면 아래 다이어그램을 보자.

- **XHR GET Requests -2-**

XHR의 GET 요청을 직접 작성해보자.

위 코드예시를 그대로 작성한 것

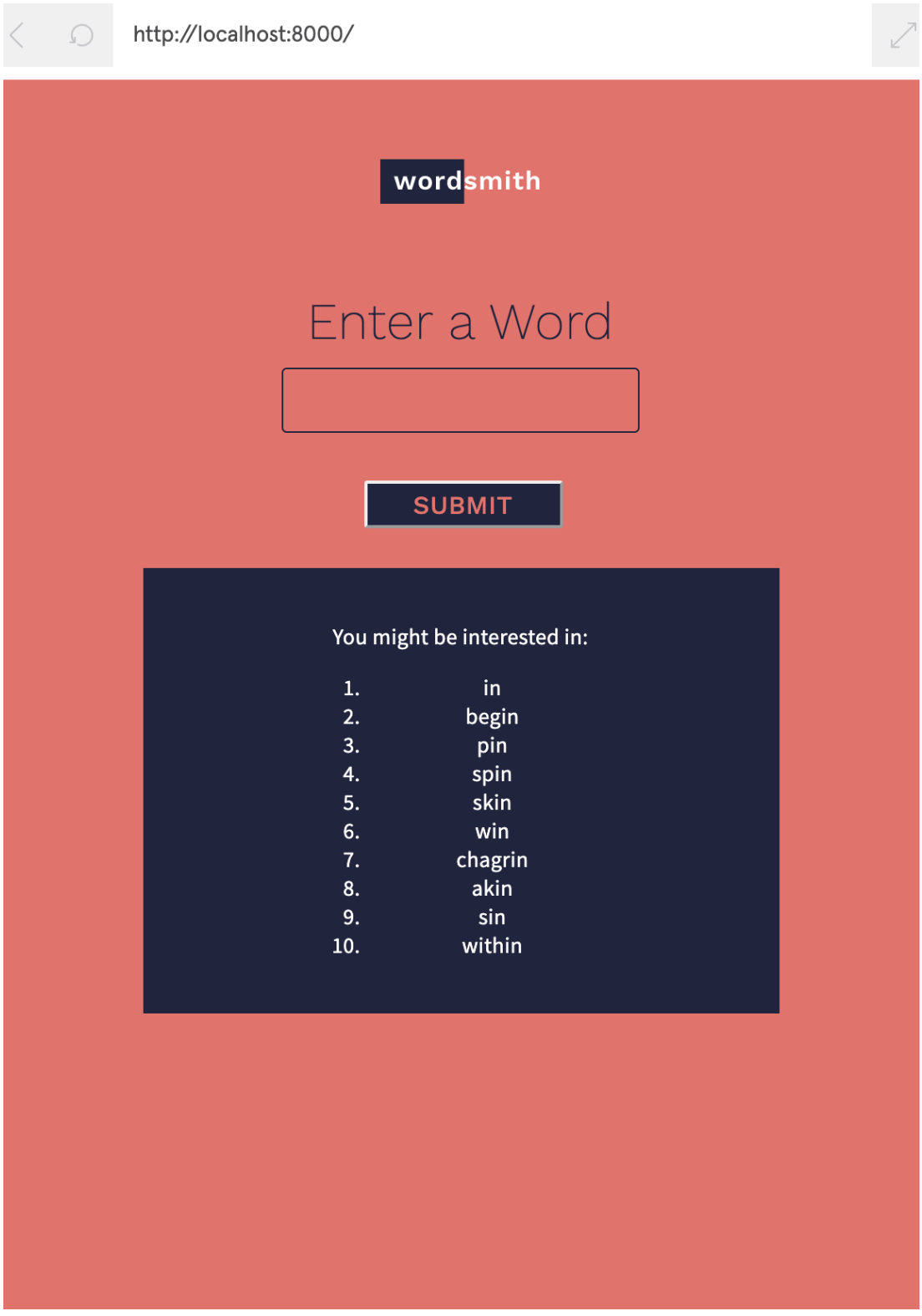
```
const xhr = new XMLHttpRequest();
const url = 'https://api-to-call.com/endpoint';

xhr.responseType = 'json';
xhr.onreadystatechange = () => {
  if (xhr.readyState === XMLHttpRequest.DONE) {
    return xhr.response;
  }
};

xhr.open('GET', url);
xhr.send();
```

- **XHR GET Requests -3-**

실습 - API 활용하여 가져오기




```
//main.js

// Information to reach API
```

```
const url = 'https://api.datamuse.com/words?';
const queryParams = 'rel_rhy=';

// Selecting page elements
const inputField = document.querySelector('#input');
const submit = document.querySelector('#submit');
const responseField = document.querySelector('#responseField');

// AJAX function
const getSuggestions = () => {
  const wordQuery = inputField.value;
  const endpoint = `${url}${queryParams}${wordQuery}`;

  const xhr = new XMLHttpRequest();
  xhr.responseType = 'json';

  xhr.onreadystatechange = () => {
    if(xhr.readyState === XMLHttpRequest.DONE) {
      renderResponse(xhr.response)
    }
  }

  xhr.open('GET', endpoint); // 새로운 요청을 생성한다.
  xhr.send(); // 생성한 요청을 서버에 보낸다.

}

// Clear previous results and display results to webpage
const displaySuggestions = (event) => {
  event.preventDefault();
  while(responseField.firstChild){
    responseField.removeChild(responseField.firstChild);
  };
  getSuggestions();
}

submit.addEventListener('click', displaySuggestions);
```

```
//helperFunctions.js


// Formats response to look presentable on webpage
const renderResponse = (res) => {
  // Handles if res is falsey
  if(!res){
    console.log(res.status);
  }
  // In case res comes back as a blank array
  if(!res.length){
    responseField.innerHTML = "<p>Try again!</p><p>There were no suggestions found!</p>";
    return;
  }

  // Creates an empty array to contain the HTML strings
  let wordList = [];
  // Loops through the response and caps off at 10
  for(let i = 0; i < Math.min(res.length, 10); i++){
    // creating a list of words
    wordList.push(`<li>${res[i].word}</li>`);
  }
  // Joins the array of HTML strings into one string
  wordList = wordList.join("");
```

```
  // Manipulates responseField to render the modified response
  responseField.innerHTML = `<p>You might be interested in:</p><ol>${wordList}</ol>`;
  return
}

// Renders response before it is modified
const renderRawResponse = (res) => {
  // Takes the first 10 words from res
  let trimmedResponse = res.slice(0, 10);
  // Manipulates responseField to render the unformatted response
  responseField.innerHTML = `<text>${JSON.stringify(trimmedResponse)}</text>`;
}
```

- **XHR GET Requests -4- (query string 사용)**

이전 연습에서 우리는 Datamuse API 에 비슷한 라임의 단어를 찾기 위해 GET 요청을 만들었다. 이번 연습에서 우리는 주제를 설정하고 query string 을 사용하여 입력된 단어를 설명하는 형용사를 찾기 위한 요청을 생성할 것이다.

쿼리 스트링은 요청과 함께 보내지는 추가적인 정보를 포함한다. Datamuse API는 요청 URL에 부착된 쿼리 스트링으로 더 특정한 데이터를 얻을 수 있도록 해준다.

query string (쿼리 스트링)은 '?'을 사용하는 URL 로부터 분리되어진다. '?' 뒤에 '='로 결합된 키 밸류쌍인 파라미터를 생성할 수 있다. 예시는 아래와 같다.

```
'https://api.datamuse.com/words?key=value'
```

만약 추가적인 파라미터를 추가하고 싶다면, 파라미터를 분리하기 위해 '&' 를 사용해야할 것이다. 아래 예시이다.

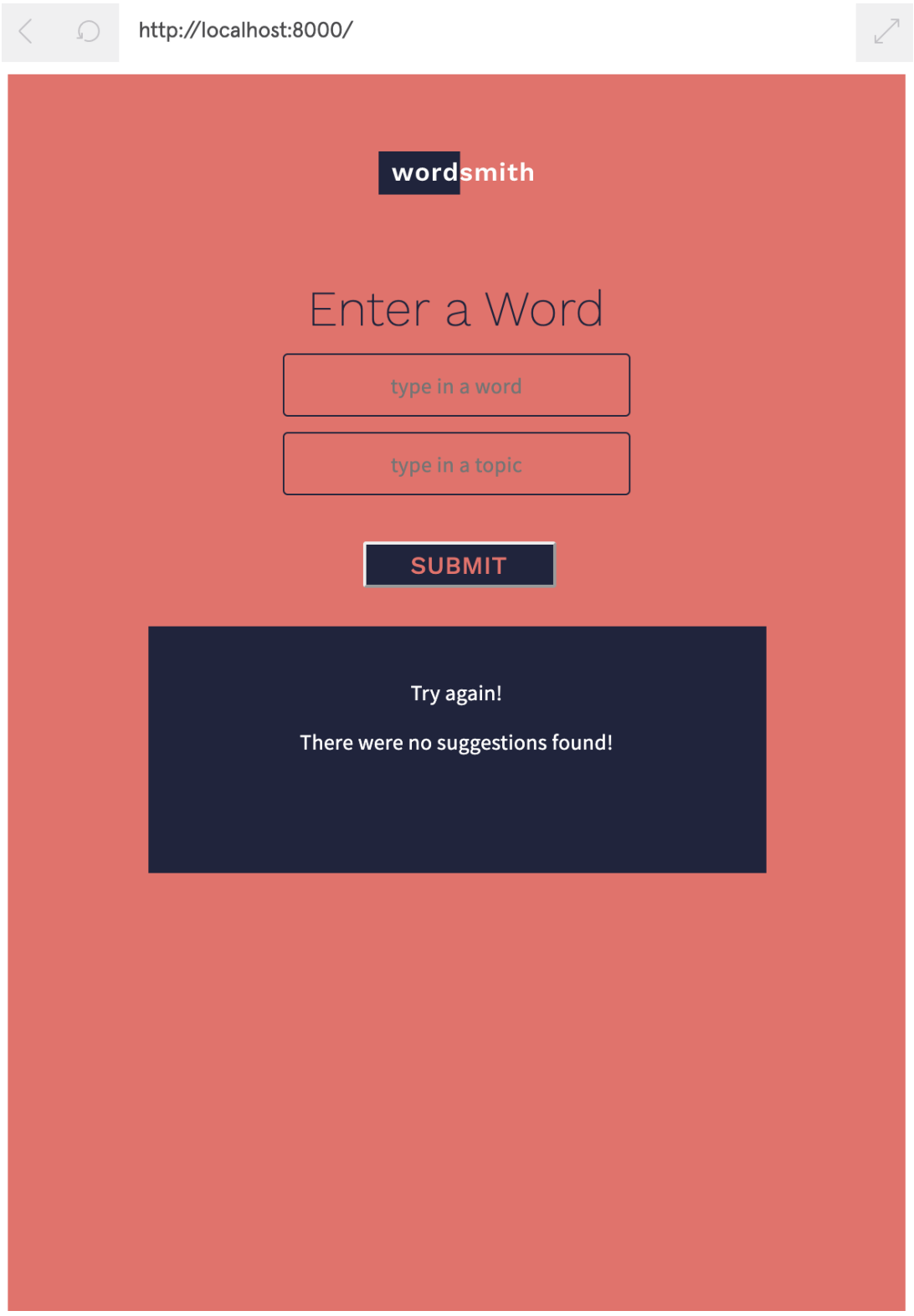
```
'https://api.datamuse.com/words?key=value&anotherKey=anotherValue'
```

실습 예제

```
// Information to reach API
const url = 'https://api.datamuse.com/words?';
const queryParams = 'rel_jjb=';
const additionalParams = '&topics=';   //추가적인 파라미터를 설정

// Selecting page elements
const inputField = document.querySelector('#input');
const topicField = document.querySelector('#topic');   // 새로운 토픽필드
const submit = document.querySelector('#submit');
const responseField = document.querySelector('#responseField');

// AJAX function
const getSuggestions = () => {
  const wordQuery = inputField.value;
  const topicQuery = topicField.value;
  const endpoint = `${url}${queryParams}${wordQuery}
                    ${additionalParams}${topicQuery}`; // 쿼리 스트링을 추가하였다.
```



- **XHR POST Requests -1-**

GET 요청과 POST 요청의 주요 차이점은 POST 요청은 요청을 통해 추가 정보를 보내야 한다는 것입니다.
이 추가 정보는 post 요청의 body 안에 보내진다.

- **XHR POST Requests -2-**

위 POST 요청 예시를 다시 한 번 작성해보자.

```
const xhr = new XMLHttpRequest();
const url = 'https://api-to-call.com/endpoint';
const data = JSON.stringify({id: '200'});   // 보낼 데이터를 설정, JSON 형식으로 변경

xhr.responseType = 'json';
// onreadystatechange 는 xhr의 상태가 변경할 때 불려지는 이벤트 핸들러를 포함한다.
xhr.onreadystatechange = () => {
```

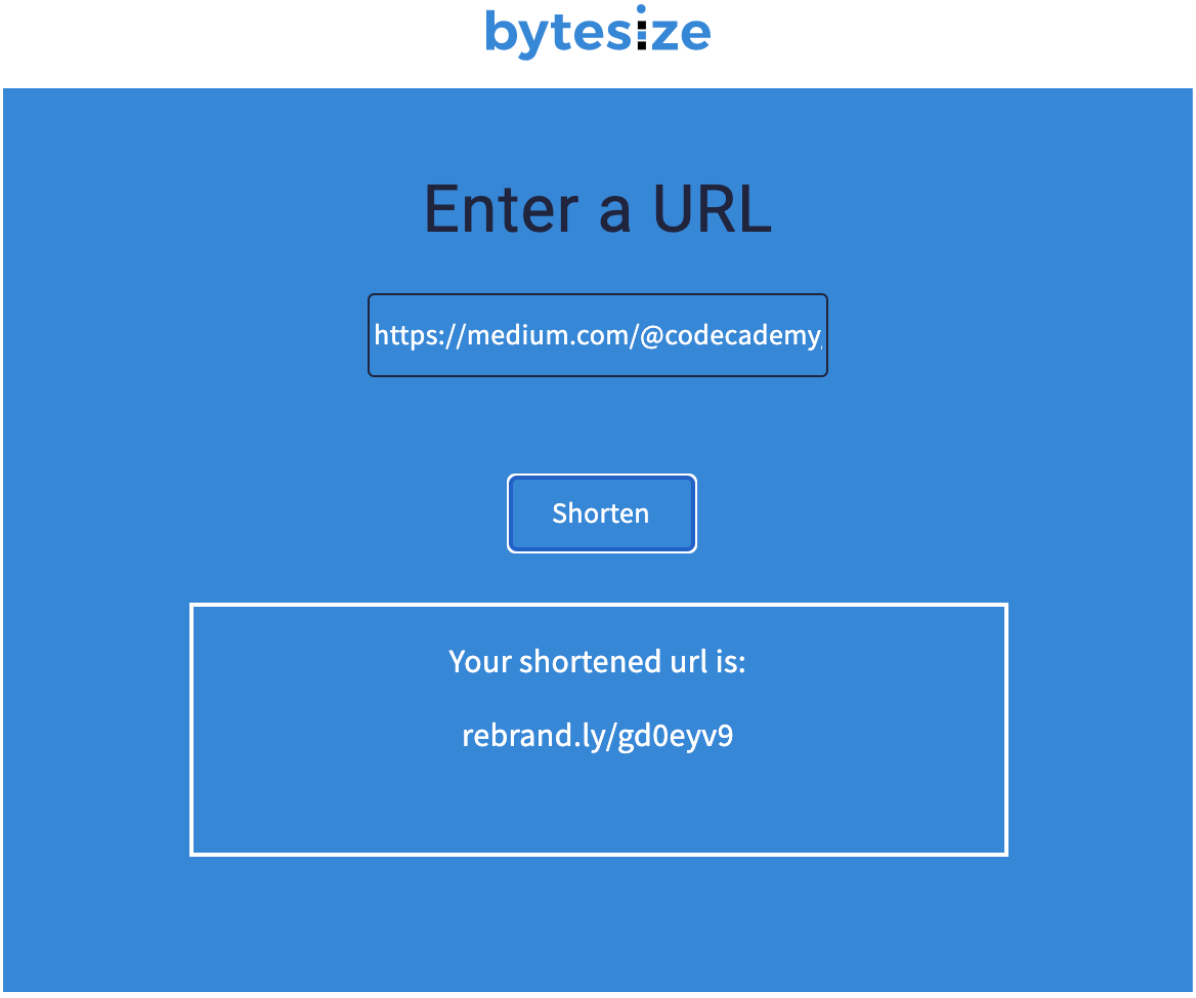
```
  if (xhr.readyState === XMLHttpRequest.DONE) {
    return xhr.response      // response 프로퍼티는 POST 요청으로 부터 반환되는 데이터를 포함한다.
  }

  xhr.open('POST', url); // 새로운 POST요청을 생성한다.
  xhr.send(data); // data 와 함께 서버에 요청을 보낸다.

}
```

- **XHR POST Requests -3-**

주소를 입력하면 짧게 보기좋게 바꾸어주는 API 를 직접 이용해보자.



```
//main.js


// API에 접근하기 위한 정보들
const apiKey = 'd9271a7f20ae43df8c70146075e626a4'; // api키를 입력해준다.
const url = 'https://api.rebrandly.com/v1/links';

// 페이지 요소들
const inputField = document.querySelector('#input');
const shortenButton = document.querySelector('#shorten');
const responseField = document.querySelector('#responseField');

// AJAX 함수
const shortenUrl = () => {
  const urlToShorten = inputField.value;
  const data = JSON.stringify({destination: urlToShorten});
  const xhr = new XMLHttpRequest();

  xhr.responseType = 'json';
  xhr.onreadystatechange = () => {
    if (xhr.readyState === XMLHttpRequest.DONE) {
      renderResponse(xhr.response);
    }
  }

  xhr.open('POST', url);
  // Rebrandly API 에 접근하기 위해, 두개의 키 값 쌍을 가진 헤더가 필요하다. 아래와 같이 설정해준다.
  xhr.setRequestHeader('Content-type', 'application/json');
  xhr.setRequestHeader('apikey', apiKey);
  xhr.send(data);

}


// 페이지를 초기화, AJAX 함수를 불러온다.
const displayShortUrl = (event) => {
  event.preventDefault();
  while(responseField.firstChild){
    responseField.removeChild(responseField.firstChild);
  }
  shortenUrl();
}

shortenButton.addEventListener('click', displayShortUrl);
```

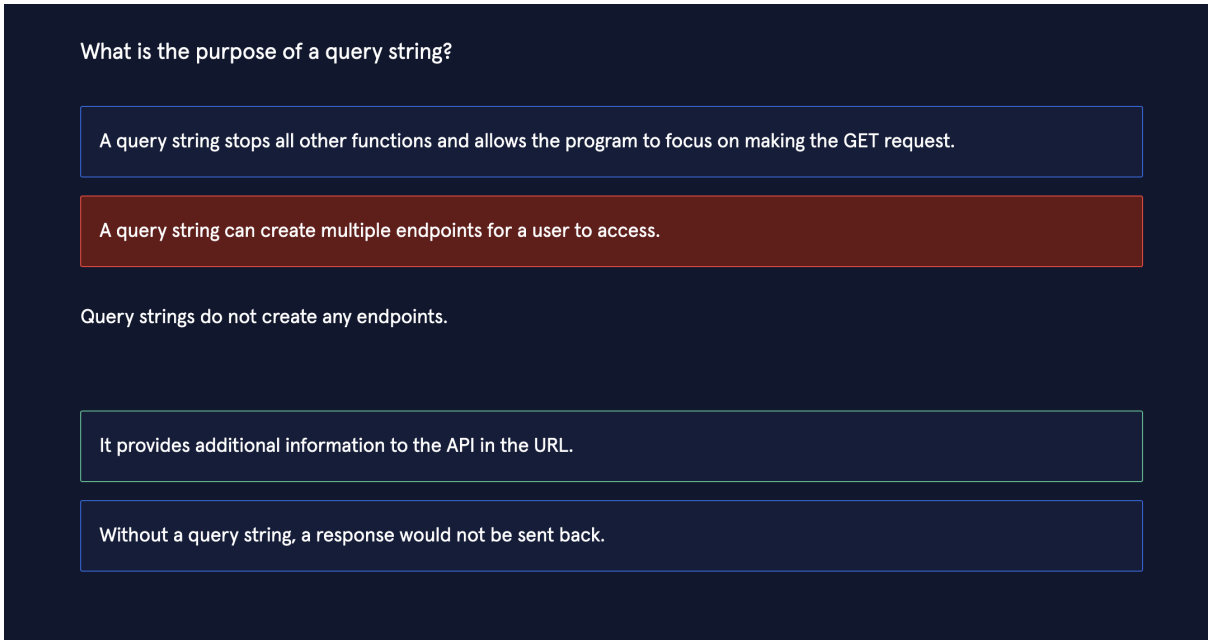
```
//helperFunctions.js


// Manipulates responseField to render a formatted and appropriate message
const renderResponse = (res) => {
  // Displays either message depending on results
  if(res.errors){
    responseField.innerHTML = "<p>Sorry, couldn't format your URL.</p><p>Try again.</p>";
  } else {
    responseField.innerHTML = `<p>Your shortened url is: </p><p> ${res.shortUrl} </p>`;
  }
}
```

```
// Manipulates responseField to render an unformatted response
const renderRawResponse = (res) => {
  // Displays either message depending on results
  if(res.errors){
    responseField.innerHTML = "<p>Sorry, couldn't format your URL.</p><p>Try again.</p>";
  } else {
    // Adds line breaks for JSON
    let structuredRes = JSON.stringify(res).replace(/,/g, ", \n");
    structuredRes = `<pre>${structuredRes}</pre>`;
    responseField.innerHTML = `${structuredRes}`;
  }
}
```

- **Review**
- 자바스크립트는 비동기적 능력때문에 웹의 언어라고 볼 수 있다. AJAX는 자바스크립트의 비동기적 능력을 이용할 수 있는 툴이다.
- 많은 HTTP 요청 메서드가 있는데 이 중 두 가지는 GET 과 POST 이다.
- GET 요청은 오직 다른 소스로부터 정보를 요청한다.
- POST 메서드는 요청뿐만 아니라, 새로운 데이터를 다른 소스에 제공할 수 있다.
- GET 요청은 바닐라 자바스크립트와 XMLHttpRequest 객체를 사용하여 쓰여진다.
- POST 요청도 마찬가지로 바닐라 자바스크립트와 XMLHttpRequest 객체를 사용하여 쓰여진다.
- GET, POST 요청을 쓸 때, new 키워드와 XHR 객체, responseType 의 세팅, response 객체를 다루는 이벤트함수, opening 과 sending 요청을 요구한다.
- URL endpoint 에 쿼리 스트링을 추가하기 위해 '?' 와 파라미터를 포함할 수 있다.
- 추가적인 파라미터를 제공하기 위해선, '&'과 키와 밸류가 '=' 로 이루어진 쌍을 사용해야한다.
- 요청을 올바르게 작성하는 방법과 올바로 구현하는 방법을 결정하려면 작업 중인 API의 설명서를 주의 깊게 읽어야 합니다.

퀴즈



오늘의 단어

- inundate : 넘치다, 범람하다 ; 쇄도하다, 듬뿍받다.
- glimpse : 흘끗보다. 잠깐 봄, 잠깐 쳐다보다.
- By this point : 이 시점에서