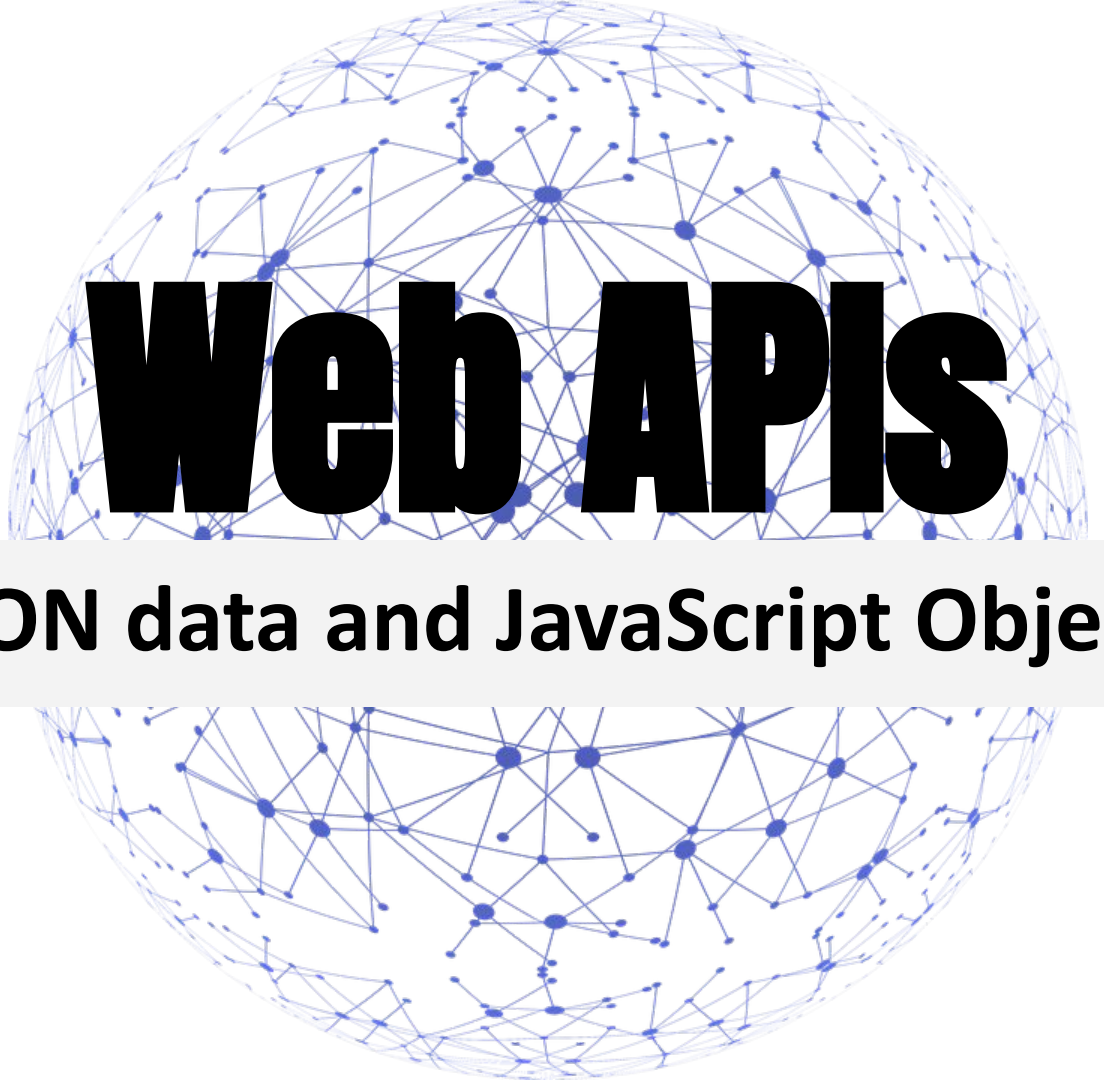


Web APIs

JSON data and JavaScript Objects

INSTRUCTOR:

LAURENCE SVEKIS

- Over 300 courses in technology and web applications.
- 20 years of JavaScript web programming experience
- 800,000+ students across multiple platforms
- Digital instructor since 2002

READY TO HELP YOU LEARN and ANSWER ANY questions you may have.

Course instructor : Laurence Svekis

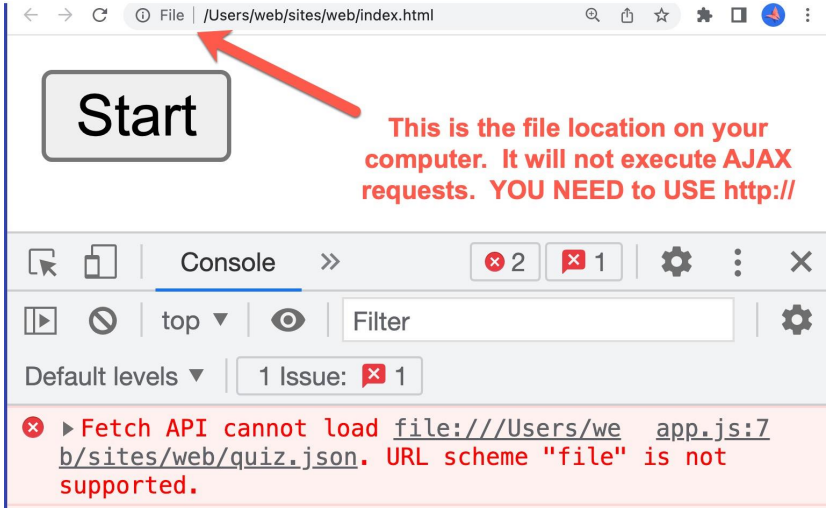
HTTP vs File protocol

Suggested editor is https://code.visualstudio.com/
Set up Local server within Visual Studio Code Use of live server is https://marketplace.visualstudio.com/items?itemName=ritwickdey.LiveServer

Do not use the file: protocol to make AJAX requests with JavaScript!

file:// is a request for a local file.
http:// is using HTTP protocol to request a file from the web or from your local computer.

Files requested via file:// get opened from local drive. A file requested via http:// get opened via HTTP request to URL that comes after it.



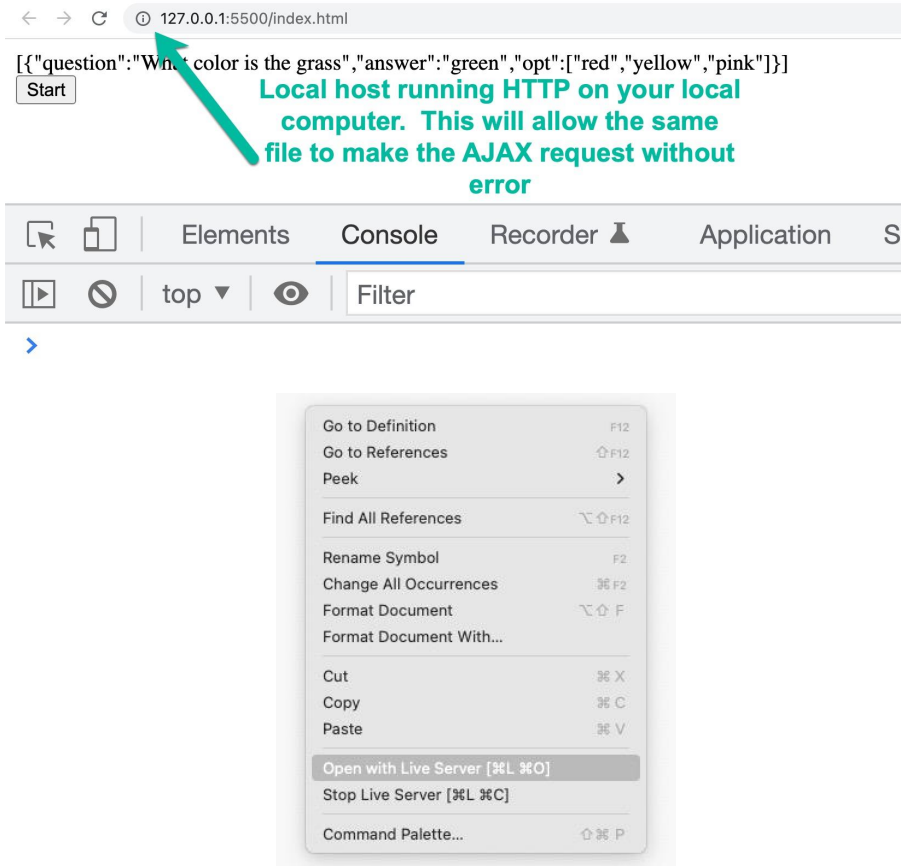
HTTP vs File protocol

http:// is running the same file that did not work in the file protocol. The local host will include the local IP address http://127.0.0.1:5500/index.html or the http://localhost:5500/index.html are both valid paths to the local server running.

In visual studio code right click in the open space or run the live server from the shortcut once its installed.

Open the live server on the html file, your workspace will need an index file which will be the default location where the local host opens.

If you see the file: in the URL bar you need to setup a local host to make AJAX requests.



Sample Code from JSON data

The below code included the html file, a JavaScript file named app.js and a json file names quiz.json.

This is an example of an AJAX request that outputs a local JSON data content.

```
[{
   "question": "What color is the grass",
   "answer": "green",
   "opt": [
       "red", "yellow", "pink"
   ]
}]
```

```
<!DOCTYPE html>
<html><head><title>JavaScript Course</title></head>
<body>
   <div class="container"></div>
   <button id="btn">Start</button>
   <script src="app.js"></script>
</body></html>
```

```
const url = 'quiz.json';
document.querySelector('#btn').onclick = loadData;
function loadData() {
   fetch(url)
       .then(res => res.json())
       .then(data => {
           document.querySelector('.container').innerText
= JSON.stringify(data);
       })
}
```

JavaScript Object

An object is a collection of related data and/or functionality.

- Functions can be contained in JavaScript objects they are referred to as methods within the object.
- JavaScript objects names don't need quotes, can be single, double or none.
- Values can be Strings, NUmbers, Booleans, Arrays, Objects

Create an object setting a variable name and assigning the {} to the variable.
Object names can hold values of other objects and arrays
Can go multiple levels deep, as many as needed.

Dot notation : The object name (person) acts as the namespace, then a dot, then the item you want to access.
Bracket notation : Similar format to arrays, instead of using an index number to select an item you are using the name associated with each member's value.

CHALLENGE :
Output the friend, country into the html.

SOLUTION :

```
<div class='output'></div>
document.querySelector('.output').textContent =
friend.location.country;
```

```
const friend = {
      name: 'Laurence Svekis'
      , age: 40
      , gender: 'male'
};
console.log(friend);

**************************************************
const friend = {
      name: {
            first: 'Laurence'
            , last: 'Svekis'
      }
      , age: 40
      , gender: 'male'
      , location: {
            city: 'Toronto'
            , country: 'Canada'
            , address: '100 Main st.'
      }
};
console.log(friend);
let val1 = friend.name;
let val2 = friend['name'];
console.log(val1);
console.log(val2);
```

JavaScript Array

An Array can hold multiple values
Arrays cannot use strings as element indexes but must use integers.
Arrays are zero based, first index value is always 0;

Array values can be strings, numbers, booleans, arrays or objects.

Can add/update/change the value of the array selecting it by the array index value.

In JavaScript there are many array methods that help manipulate arrays.
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array#

CHALLENGE :
Output the name John into the html.

SOLUTION :
```
<div class='output'></div>
document.querySelector('.output').textContent = friendList[1].name;
```

```
const friends = ['Laurence', 'John', 'Steve', 'Linda', 'Jane'];
console.log(friends);
console.log(friends[2]);

**************************************************

const friend1 = {
      name: 'Laurence'
};
const friend2 = {
      name: 'John'
};
const friendList = [friend1, friend2];
friendList[10] = {
      name: 'Jane'
}; //will add empty items
console.log(friendList);
console.log(friendList.length);
```

Iterate Array Contents

The design of objects and arrays is to hold lots of content. You can loop through the content in a number of ways using JavaScript.
You can loop through the data in the array using a number of methods in JavaScript. Arrays need the index to find the value associated with it. If objects are contained within you should structure them the same way so it is easier to check the values contained.

CHALLENGE :

Use the array to create a **new array** with objects using the value as name coming from the list of values containing the array. const friends = ['Laurence', 'John', 'Steve', 'Linda', 'Jane'];
Loop the new array contents and output the name values into the console.

SOLUTION :

```
const friends = ['Laurence', 'John', 'Steve', 'Linda', 'Jane'];
const newArray = [];
friends.forEach(function (item) {
      let temp = {
            name: item
      };
      newArray.push(temp);
});
console.log(newArray);

newArray.forEach(function(item){
      console.log(item.name);
})
```

```
const friend1 = { name: 'Laurence' };
const friend2 = { name: 'John'};
const friend3 = { name: 'Jane'};
const friendList = [friend1, friend2];
friendList.push(friend3);
console.log(friendList);
console.log(friendList.length);

for (let x = 0; x < friendList.length; x++) {
      console.log(friendList[x].name);
}

friendList.forEach(function (item, index, array) {
     // console.log(item, index);
      console.log(item.name);
});

for (ind in friendList) {
       console.log(friendList[ind].name);
}
```

Iterate Object Contents

Objects have length so using a for loop is possible. There is also Object.entries which can get the key and the value from the object.
Keep data structured the same so that you can easily determine where the values are located.

CHALLENGE :
Give all the friends the same last name of Smith.

SOLUTION :

```
for (const [key, value] of Object.entries(friends)) {
      friends[key].last = 'Smith';
}
console.log(friends);
```

```
const friend1 = {
      name: 'Laurence'
};
const friend2 = {
      name: 'John'
};
const friend3 = {
      name: 'Jane'
};
const friends = [friend1, friend2, friend3];
console.log(friends);
console.log(friends.length);
for (const key in friends) {
      let value = friends[key];
      console.log(value.name);
}
console.log(Object.entries(friends));
for (const [key, value] of Object.entries(friends)) {
      console.log(key);
      console.log(value);
      console.log(value.name);
}
```

JSON parse and Stringify

The **JSON.stringify()** method converts a JavaScript object or value to a JSON string

The **JSON.parse()** method parses a JSON string, constructing the JavaScript value or object described by the string

CHALLENGE :

Create object from friends string, update the object with a new value. Loop new object content and output on separate lines within your html page.

SOLUTION :

```
const str = '[{"name":"Laurence"},{"name":"John"},{"name":"Jane"}]';
const friends2 = JSON.parse(str);
friends2.push({
      name: 'Mike'
});
let html = '';
friends2.forEach(function (val) {
      html += val.name + '<br>';
})
document.querySelector('.output').innerHTML = html;
```

```
const friend1 = {
      name: 'Laurence'
};
const friend2 = {
      name: 'John'
};
const friend3 = {
      name: 'Jane'
};
const friends = [friend1, friend2, friend3];
document.querySelector('.output').textContent =
JSON.stringify(friends);
const str =
'[{"name":"Laurence"},{"name":"John"},{"name":"Jane"}]';
const friends2 = JSON.parse(str);
friends2.push({
      name: 'Mike'
});
console.log(friends2);
```

JSON Content Files

The JSON object contains methods for parsing JavaScript Object Notation (JSON) and converting values to JSON.
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/JSON

- Objects and Arrays: Property names must be double-quoted strings; trailing commas are forbidden.
- Numbers: Leading zeros are prohibited; a decimal point must be followed by at least one digit. NaN and Infinity are unsupported.

More complex json still JavaScript Object
https://jsonlint.com/ to read the data structure easier.

Link to json content in source - don't forget the double quotes as this is json and not a JavaScript Object

You can use a json file extension and link to source, same as having javascript file with JSON data assigned to variable data.

CHALLENGE :
Create a complex JSON object using both arrays and values.
Check and validate it in the jsonlint

SOLUTION :
Paste in https://jsonlint.com/

```
const json = {
      "books": [
            {
                  "title": "Learn to Code"
                  , "author": "John Smith"
                  , "isbn": "324-23243"
        }
            , {
                  "title": "The Adventures JSON"
                  , "author": "Jason Jones"
                  , "isbn": "3324-2-444"
        }
            , {
                  "title": "New Objects"
                  , "author": "Jane Doe"
                  , "isbn": "2343-234-2433"
        }
    ]
};
console.log(json);
json.books.forEach(function (val) {
      console.log(val);
})

*****************************************************
<script src="data.json"></script>
<script> console.log(json); </script>
```

Web APIs

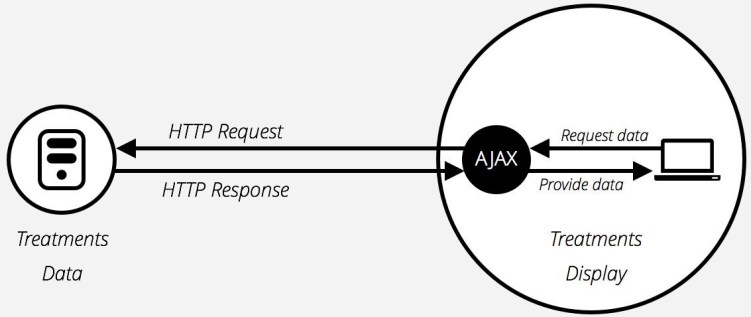
Application Programming Interfaces (APIs) are constructs made available in programming languages to allow developers to create complex functionality more easily. They abstract more complex code away from you, providing some easier syntax to use in its place.
https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Client-side_web_APIs/Introduction

APIs that fetch data from the server to update small sections of a webpage on their own are very commonly used. This seemingly small detail has had a huge impact on the performance and behaviour of sites — if you just need to update a stock listing or list of available new stories, doing it instantly without having to reload the whole entire page from the server can make the site or app feel much more responsive and "snappy". APIs that make this possible include XMLHttpRequest and the Fetch API. You may also come across the term Ajax, which describes this technique

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

CHALLENGE :
Connect to https://www.discoveryvip.com/shared/json.php?f=peopledata1 get the data.
Output the data into the console.

```
const u =
'https://www.discoveryvip.com/shared/json.php?f=peopledata1';
fetch(u).then(function (resp) {
      return resp.json()
}).then(function (data) {
      console.log(data);
}).catch(function () {
      // catch any errors
});
```



Fetch API

The **Fetch API** provides an interface for fetching resources. It will seem familiar to anyone who has used **XMLHttpRequest**, but the new API provides a more powerful and flexible feature set.
Use **XMLHttpRequest (XHR)** objects to interact with servers. You can retrieve data from a URL without having to do a full page refresh. This enables a Web page to update just part of a page without disrupting what the user is doing. XMLHttpRequest is used heavily in AJAX programming.
https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest
Fetch is json ready, The json() method of the Body mixin takes a Response stream and reads it to completion. It returns a promise that resolves with the result of parsing the body text as JSON.

CHALLENGE :
Get data from an external file and iterate through the results outputting only the titles into the HTML page.

SOLUTION :

```
fetch('json.json').then(function (response) {
      return response.json();
}).then(function (data) {
      let html = '';
      data.books.forEach(function (val) {
            document.querySelector('.output').innerHTML  +=
val.title + '<br>';
      })
});
```

```
{
      "books": [{
            "title": "Learn to Code",
            "author": "John Smith",
            "isbn": "324-23243"
      }, {
            "title": "The Adventures JSON",
            "author": "Jason Jones",
            "isbn": "3324-2-444"
      }, {
            "title": "New Objects",
            "author": "Jane Doe",
            "isbn": "2343-234-2433"
      }]
}
*****************************************************
let req = new XMLHttpRequest();
req.open("GET", "json.json");
req.responseType = 'json';
req.onload = function () {
      console.log(req.response);
}
req.send();
*****************************************************
fetch('json.json').then(function (response) {
      return response.json();
}).then(function (data) {
      console.log(data);
});
```

Fetch Arrow function expressions

An arrow function expression is a syntactically compact alternative to a regular function expression

Looks shorter does the same thing, although Arrow function expressions are ill suited as methods, and they cannot be used as constructors. For fetch this is not relevant.

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Functions/Arrow_functions

```
const url = 'json.json';
fetch(url).then(res => res.json()).then(json =>
console.log(json));
```

CHALLENGE :

Replace previous solution with arrow function expressions.

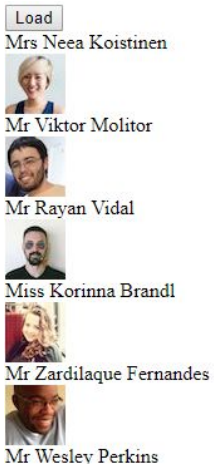
SOLUTION :

```
const url = 'json.json';
fetch(url).then(res => res.json()).then(data => {
      let html = '';
      data.books.forEach(function (val) {
            document.querySelector('.output').innerHTML += val.title
+ '<br>';
      })
});
```

Random User API

Get random users with names, with images and full user info. Great to practice outputting content to your website.
https://randomuser.me/api/?results=10

CHALLENGE :
Get user info from the API.
Output the users into the page
Add images and data about the user construct nice output of the data.
Set up a event listener that can trigger getting new entries.



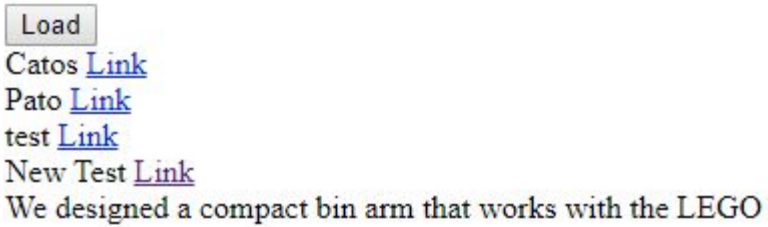

```
const url = 'https://randomuser.me/api/?results=10';
const btn = document.querySelector('button');
const output = document.querySelector('.output');
btn.addEventListener('click', getData);

function getData() {
      output.innerHTML = "";
      fetch(url).then(function (response) {
            return response.json();
      }).then(function (data) {
            console.log(data);
            data.results.forEach(function (val) {
                  let div = document.createElement('div');
                  div.textContent = `${val.name.title}
${val.name.first}  ${val.name.last}`;
                  let ima = document.createElement('img');
                  ima.setAttribute('src', val.picture.thumbnail);
                  ima.style.display = 'block';
                  div.appendChild(ima);
                  output.appendChild(div);
            })
      });
}
```

Reddit Post API

Get reddit content
https://www.reddit.com/r/test/top/.json?limit=5

CHALLENGE :
Get data from the reddit JSON api
Search posts for test as content
Output the title and the hyperlink to the original post.




```
const url = 'https://www.reddit.com/r/test/top/.json?limit=5';
const btn = document.querySelector('button');
const output = document.querySelector('.output');
btn.addEventListener('click', getData);

function getData() {
      output.innerHTML = "";
      fetch(url).then(function (response) {
            return response.json();
      }).then(function (data) {
            console.log(data);
            data.data.children.forEach(function (val) {
                  console.log(val.data);
                  let div = document.createElement('div');
                  div.innerHTML = `${val.data.title} <a
href=${val.data.url} target='_blank'>Link</a>`;
                  output.appendChild(div);
            })
      }).catch(function () {
            console.log("error");
      });
}
```

GitHub Repo API

https://api.github.com/search/repositories?q=tetris&sort=stars&order=desc

CHALLENGE :

Get data from the Github JSON api

Output latest repos about Tetris, get the name and create the link to the URL with the description.


```
const url =
'https://api.github.com/search/repositories?q=tetris&sort=stars&or
der=desc';
const btn = document.querySelector('button');
const output = document.querySelector('.output');
btn.addEventListener('click', getData);

function getData() {
      output.innerHTML = "";
      fetch(url).then(function (response) {
            return response.json();
      }).then(function (data) {
            console.log(data);
            data.items.forEach(function (val) {
                  console.log(val);
                  let div = document.createElement('div');
                  div.innerHTML = `${val.name} <a href=${val.url}
target='_blank'>${val.description}</a><br>`;
                  output.appendChild(div);
            })
      }).catch(function () {
            console.log("error");
      });
}
```

StackOverflow API

https://api.stackexchange.com/2.2/questions?order=desc&sort=activity&tagged=javascript&site=stackoverflow
https://api.stackexchange.com/docs - Create your own results

CHALLENGE :

Get data from the StackOverflow JSON api

Output latest posts that are tagged 'JavaScript' with titles, question id and links to the posts.


```
const url =
'https://api.stackexchange.com/2.2/questions?order=desc&sort=activ
ity&tagged=javascript&site=stackoverflow';
const btn = document.querySelector('button');
const output = document.querySelector('.output');
btn.addEventListener('click', getData);

function getData() {
      output.innerHTML = "";
      fetch(url).then(function (response) {
            return response.json();
      }).then(function (data) {
            console.log(data);
            data.items.forEach(function (val) {
                  console.log(val);
       let div = document.createElement('div');
                  div.innerHTML = `${val.question_id} <a
href=${val.link} target='_blank'>${val.title}</a><br>`;
                  output.appendChild(div);
            })
      }).catch(function () {
            console.log("error");
      });
}
```

POST data

Send Data to an API, endpoint is at https://script.google.com/macros/s/AKfycbwIrTmXmMCtgC-1VVitPdm9cnqj7tLZV8m4BvR2wKDrm9cvxBw/exec and is expecting 2 values name and message.

CHALLENGE :

Connect to the API send data to the endpoint using post.
Output the response of the post in the page output area.
Next lesson will show you how to retrieve the results.

```
<input name="name" value="Laurence">
<br> Message :
<input name="message" value="Hello World">
<br>
<button>Send</button>
```

```
const url =
'https://script.google.com/macros/s/AKfycbwIrTmXmMCtgC-1VVitPdm9cn
qj7tLZV8m4BvR2wKDrm9cvxBw/exec';
const btn = document.querySelector('button');
const output = document.querySelector('.output');
btn.addEventListener('click', getData);

function getData() {
      let val1 =
document.querySelector('input[name="name"]').value || 'Unknown';
            let val2 =
document.querySelector('input[name="message"]').value ||
'Message';
      let arr = [val1 ,val2];
      let formData = new FormData();
      formData.append('data', JSON.stringify(arr));
      fetch(url, {
            method: 'POST'
            , body: formData
      }).then(function (response) {
            return response.json()
      }).then(function (data) {
            console.log(data);
      })
}
```