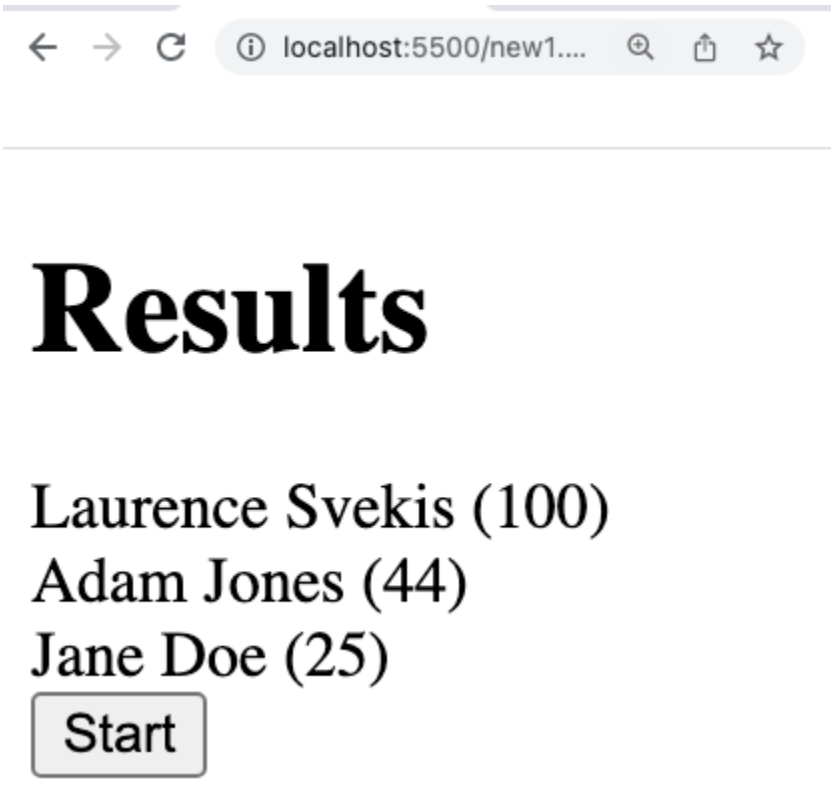
# JavaScript Coding Examples Source Code



## Make an AJAX request to local JSON file



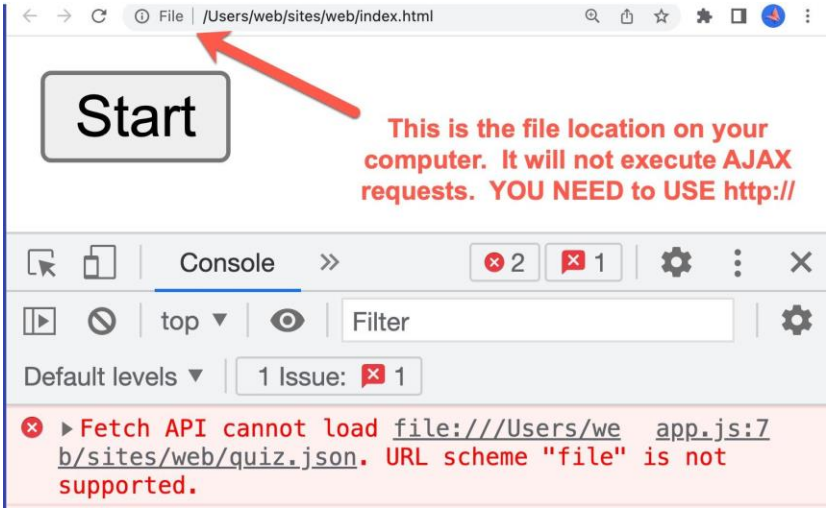

**You must use the http protocol to do an AJAX request to the json file.**

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 the URL that comes after it.

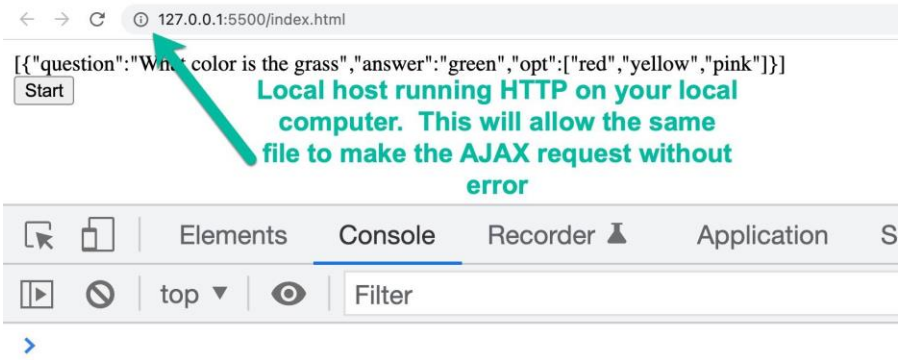


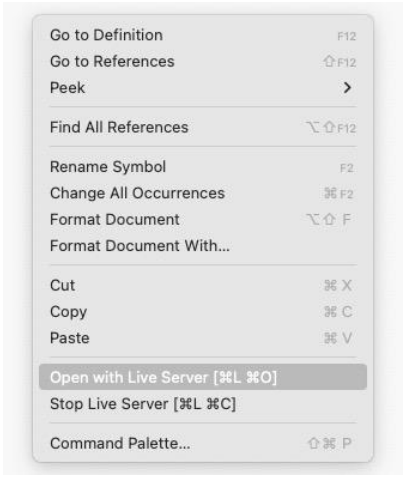

http:// is running the same file that did not work in the file protocol. The local host will include the local IP address http://127.0.0.1:5500/index.html or the http://localhost:5500/index.html are both valid paths to the local server running.

In visual studio code right click in the open space or run the live server from the shortcut once it's 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 set up a local host to make AJAX requests.




Exercise :

1. Create a JSON file with objects contained in an array.
2. Create an HTML file with a button that will be made clickable and an element where you can output content into.
3. In the JavaScript file, select the page elements
4. Select the path to the JSON file as a variable named url.
5. Add an event listener to the button click , making a fetch request to the url.
6. Once a response is received from the JSON file, return it as JSON into a JavaScript object with json()
7. Create a function named addData() that will loop through an array of results, and create a string value of HTML code to output on the page, containing the property values from the objects in the JSON file array.

HTML

```
<!DOCTYPE html>
<html><head><title>JavaScript Course</title></head>
<body>
  <div class="container"></div>
  <button id="btn">Start</button>
  <script src="js1.js"></script>
</body></html>
```

JavaScript

```
const btn = document.querySelector('#btn');
const output = document.querySelector('.container');
const url = 'temp1.json';
//console.log(btn);
btn.onclick = ()=>{
  //console.log('clicked');
  fetch(url)
  .then(res => res.json())
  .then(data => {
     addData(data);
  })
}

function addData(data){
  let html = '<h1>Results</h1>';
  data.forEach(person=>{
     console.log(person);
     html += `<div>${person.first} ${person.last} (${person.id})</div>`;
  })
  output.innerHTML = html;
```

```
}
```

JSON

```
[
  {
     "first" : "Laurence",
     "last" : "Svekis",
     "id" : 100
  },
  {
     "first" : "Adam",
     "last" : "Jones",
     "id" : 44
  },
  {
     "first" : "Jane",
     "last" : "Doe",
     "id" : 25
  }
]
```

# Fun with Functions

Using a condition if true to execute a function.

**Exercise**

1. Create a function that will run depending on a condition.  In the condition nest the function invoking statement
2. Create a separate statement that will include the condition and the function within one statement using the && operator.
3. Create page elements that can be interacted with.
4. Add a counter value that increments when the button is clicked.
5. Create a function that when invoked adds a new page element.

6. Using the % modulus, check for a remainder for the counter, if divisible by 3 output the value in red, if divisible by 2 output the value in blue.
7. Try the code with both a condition and then using just the one statement and && operator
8. Add some arguments to the element that gets created adding it into the code.

```
const output = document.createElement('div');
const message = document.createElement('div');
message.textContent = 'Output';
const btn = document.createElement('button');
btn.textContent = 'Click me';
output.append(message);
output.append(btn);
document.body.prepend(output);
let counter = 0;
let val1 = 5;

(true) && (true) && (true)&& fun();

(4>1)&&(10<20)&&fun();

if((4>1)&&(10<20)){
  fun();
}

function fun(){
  val1++;
  console.log(val1);
}

btn.onclick = ()=>{
```

```
  counter++;
  message.textContent = counter;
  (counter%3 == 0) && adder(`${counter} : ${counter%3}`,'red');
  (counter%2 == 0) && adder(`${counter} : ${counter%2}`,'blue');
  //(ele) && updater(ele,'red');
  /*
  if(counter%3 == 0){
     const ele = adder(`${counter} : ${counter%3}`);
  }
  */
}

function updater(ele,cl){
  ele.style.color = cl;
}



function adder(html,cl){
  const div = document.createElement('div');
  updater(div,cl)
  div.textContent = html;
  return output.appendChild(div);
}
```

# How to get unique values within your array

```
app1.js:14
(16) ['Laurence', 'Linda', 'Jane', 'Jack', 'Jack', 'Laurence', 'Laurence', 'Laurence',
▶ 'Laurence', 'Laurence', 'Laurence', 'Laurence', 'Laurence', 'Laurence', 'Laurence', 'Mi
ke']
▶ (6) ['Laurence', 'Linda', 'Jane', 'Jack', 'Mike', 'New']   app1.js:15
▶ (4) ['Laurence', 'Linda', 'Jane', 'Jack']                  app1.js:16
```

Exercise :

1. Create an array and add values, include some duplicate values
2. Using the new Set() return the original array with only unique values, creating a new array
3. Assign one array to the original array, notice that they are connected. Any updates to one will update the other as well.
4. Assign a new array to the people1 array, using the new Set method returning only unique values from the original people array
5. Using the filter method to return unique items by checking for a match of the index value, duplicates will not match the index value since the indexOf returns the index value of the first matching result.

```
const people = ['Laurence','Linda','Jane','Jack','Jack'];
for(let i=0;i<10;i++){
  people.push('Laurence');
}
let people1 = people;
console.log(people);
const newPeople = [... new Set(people)];
console.log(newPeople);
people1.push('Mike');
console.log(people);
people1 =  [... new Set(people)];
console.log(people1);
people1.push('New');
console.log(people);
console.log(people1);
console.log(newPeople);
```

```
const people2 = people.filter((item,i)=> people.indexOf(item) === i);
console.log(people2);
```

# Clean and remove empty falsy values from array

Exercise:

1. create an array, adding in empty and falsy values
2. Using filter with argument Boolean, also try with String
3. Use filter to create a new cleaned array of empty values.

```
const people = ['Laurence',0,null,undefined,false,,,,'John','Lisa'];
people[20] = 'Jane';
console.log(people);
const arr2 = people.filter(Boolean);
const arr3 = people.filter((item)=>{
  return item != null;
})
const arr4 = people.filter(String);
console.log(arr2);
console.log(arr3);
console.log(arr4);
```