

# JavaScript

AJAX JSON Simple APIs

# Course Developer Setup Resources

<http://brackets.io/> - editor

<https://www.getpostman.com> - testing of endpoints

<http://jsonplaceholder.typicode.com/> - Fake Online REST API for Testing and Prototyping -

<https://github.com/typicode/json-server> - JSON server

<https://nodejs.org> - Download install node js

<https://www.npmjs.com/get-npm> - Node package manager



## TRY IT :

1. Setup your editor and open a browser to live preview. Make sure you can run your code from the HTTP protocol not the file protocol.

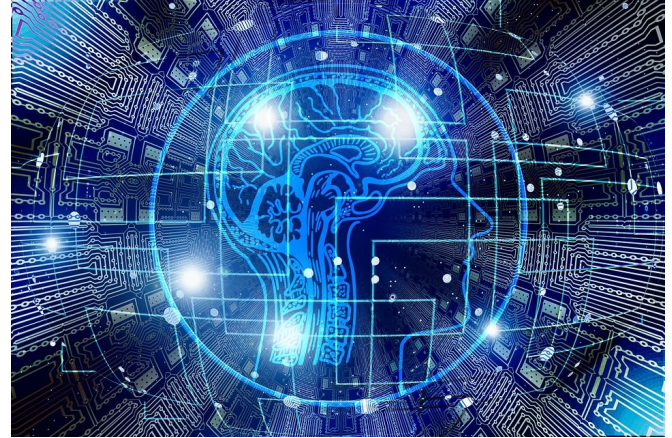
# What is JSON

- Extended from the JavaScript
- Media type is application/json
- Extension is .json
- JSON is a language-independent data format
- JSON JavaScript Object Notation

Always starts and ends with curly brackets { }

Name and value is separated by a colon :

More than one pair is separated by comma ,



# JSON vs XML vs YAML

JSON and XML are human readable formats JSON is faster to write. XML has not arrays. JSON much easier to parse in JavaScript

```
{
  "firstName": "John",
  "lastName": "Smith",
  "age": 25,
  "address": {
    "streetAddress": "21 2nd Street",
    "city": "New York",
    "state": "NY",
    "postalCode": "10021"
  },
  "phoneNumber": [
    {
      "type": "home",
      "number": "212 555-1234"
    },
    {
      "type": "fax",
      "number": "646 555-4567"
    }
  ],
  "gender": {
    "type": "male"
  }
}
```

```
<person>
  <firstName>John</firstName>
  <lastName>Smith</lastName>
  <age>25</age>
  <address>
    <streetAddress>21 2nd Street</streetAddress>
    <city>New York</city>
    <state>NY</state>
    <postalCode>10021</postalCode>
  </address>
  <phoneNumber>
    <type>home</type>
    <number>212 555-1234</number>
  </phoneNumber>
  <phoneNumber>
    <type>fax</type>
    <number>646 555-4567</number>
  </phoneNumber>
  <gender>
    <type>male</type>
  </gender>
</person>
```

```
firstName: John
lastName: Smith
age: 25
address:
  streetAddress: 21 2nd Street
  city: New York
  state: NY
  postalCode: '10021'
phoneNumber:
  - type: home
    number: 212 555-1234
  - type: fax
    number: 646 555-4567
gender:
  type: male
```

# Basics of JSON

key/name value pairs

```
{ "name" : "value" }
```

Objects are comma separated

```
{ "name1" : "value" , "name2" : "value" , "name3" : "value" }
```

Arrays have square brackets with values separated by comma

```
{ "name" : [ { "name" : "value" }, { "name" : "value" } ] }
```

```
{  
  "name": "value"  
}  
  
{  
  "name1": "value",  
  "name2": "value",  
  "name3": "value"  
}  
  
{  
  "name": [{  
    "name": "value"  
  }, {  
    "name": "value"  
  }]  
}
```

# JSON Data

JSON is an open-standard format that uses human-readable text to transmit data objects consisting of attribute value pairs.

<https://jsonlint.com> - Validation of JSON

```
{
  "firstName": "John",
  "lastName": "Smith",
  "isAlive": true,
  "age": 25,
  "address": {
    "streetAddress": "21 2nd Street",
    "city": "New York",
    "state": "NY",
    "postalCode": "10021-3100"
  },
  "phoneNumbers": [
    {
      "type": "home",
      "number": "212 555-1234"
    },
    {
      "type": "office",
      "number": "646 555-4567"
    },
    {
      "type": "mobile",
      "number": "123 456-7890"
    }
  ],
  "children": [],
  "spouse": null
}
```

```
{
  "firstName": "John",
  "lastName": "Smith",
  "isAlive": true,
  "age": 25,
  "address": {
    "streetAddress": "21 2nd Street",
    "city": "New York",
    "state": "NY",
    "postalCode": "10021-3100"
  },
  "phoneNumbers": [
    {
      "type": "home",
      "number": "212 555-1234"
    },
    {
      "type": "mobile",
      "number": "123 456-7890"
    }
  ],
  "children": [],
  "spouse": null
}
```

# JSON lint makes more readable

<http://myjson.com> - Storage of JSON

<https://randomuser.me> - JSON Data API

<https://randomuser.me/api/?results=3>



## TRY IT :

1. Open the randomuser.me api, get the content paste it in jsonlint.com
2. Open <http://myjson.com> place the jsonlint content into the textarea input field.
3. Create a new myjson.com bin with the JSON content. Copy the path of the bin you will need it.

# Object

Group of values represented by name of key.

**name**:*value* pairs

Value can be any data type in JavaScript - number, string, boolean, array even functions also know as methods within the object. Arrays are in fact a type of object.





# Objects can hold a collection of related data

```
var car = {};  
car.color = 'red';  
car.topSpeed = 300;  
car.model = 'mustang';  
car.company = 'ford';  
car.year = 2012;  
car.price = 50000;
```



Objects can contain many values

name:values pairs in JavaScript objects are called properties:

# Data Types in JSON value can be any of these

**Number** - double- precision floating-point can be digits, positive or negative, decimal fractions, exponents...

`{"name":10}`

**String** - double-quoted Unicode with backslash escaping

`{"name":"Hello world"}`

**Boolean** - true or false `{"name":true}`

**Array** - ordered sequence of values uses square brackets. Values are each separated by a comma. Indexing starts with 0. `{"name": [{"name1": 1}, "hello", "world"]}`

**Object** - unordered collection with key:value pairs. Wrapped with curly brackets `{}`. Colons to separate key and name. Keys should be strings and have unique names. `{"name": {"name1": 1,"name2": 1}}`

**Null** - just empty `{"name": null}`



# object literal

Object which uses curly brackets: {}

Using object literals is the more common and preferred method.



## TRY IT :

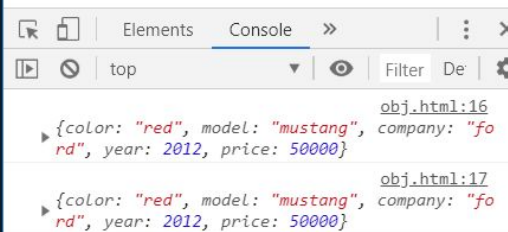
1. Open Your editor
2. Create an object
3. Log it to the console using `console.log(myCar);`

```
const myCar = {  
  "color": "red"  
  , "model": "mustang"  
  , "company": "ford"  
  , "year": 2012  
  , "price": 50000  
};  
console.log(myCar);
```

# JSON and JavaScript Objects

JSON needs quotes for the names to be valid whereas JavaScript objects do not.

```
const myCar1 = {  
  color: "red"  
  , model: "mustang"  
  , company: "ford"  
  , year: 2012  
  , price: 50000  
};  
const myCar2 = {  
  "color": "red"  
  , "model": "mustang"  
  , "company": "ford"  
  , "year": 2012  
  , "price": 50000  
};  
console.log(myCar1);  
console.log(myCar2);
```



```
const myCar1 = {  
  color: "red"  
  , model: "mustang"  
  , company: "ford"  
  , year: 2012  
  , price: 50000  
};  
const myCar2 = {  
  "color": "red"  
  , "model": "mustang"  
  , "company": "ford"  
  , "year": 2012  
  , "price": 50000  
};  
console.log(myCar1);  
console.log(myCar2);
```

# Difference: JSON & JavaScript Object

JSON all *keys* must be quoted, object literals it is not necessary:

```
{ "foo": "bar" }
```

```
var o = { foo: "bar" };
```

JSON has double quotes while JavaScript can use single or doubles

JavaScript can include functions which is not available in JSON.



# JavaScript Objects

Get values of objects. Use the variable name of the object and then return values using the name. You can use dot notation or bracket notation. All the same even double or single quotes. Bracket notation is dynamic.

```
};  
console.log(myCar1.model);  
console.log(myCar1['model']);  
console.log(myCar2.model);  
console.log(myCar2["model"]);
```

mustang

mustang

mustang

mustang

## TRY IT :

1. Open Your editor
2. Create an object
3. Get the properties of the object, return the values in the console. Try the various ways to return the results.
4. Set a variable to the value of the name and then use the variable within the bracket.

```
console.log(myCar1.model);  
console.log(myCar1['model']);  
console.log(myCar2.model);  
console.log(myCar2["model"]);  
const temp = 'color';  
console.log(myCar2[temp]);
```

# JavaScript Output to Web Page

Using JavaScript output the contents of the object values on a website.

Create the HTML tags, select using JavaScript and output content into them.

## TRY IT :

1. Open Your editor
2. Create an object
3. Select the elements as objects.
4. Update `textContent` of element with new object data.

```
<div id="output1"></div>
<div id="output2"></div>
<script>
  const output1 = document.getElementById('output1');
  const output2 = document.getElementById('output2');
  console.dir(output1);
  const myObj = {
    "firstName": "Mike"
    , "lastName": "Smith"
    , "age": 30
  };
  console.log(myObj);
  const name = 'Name';
  output1.innerHTML = myObj.firstName;
  output2.innerHTML = myObj['last' + name];
</script>
```

# Array of items

## Better way

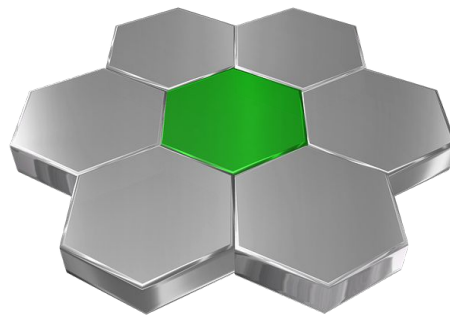
```
var cars = {"car":["Blue","black"]}  
console.log(cars)
```

Now we can add more details to each item :)

```
var myJSON = {"car1" : {"color":"black"}, "car2" : {"color" : "blue" }};  
console.log(myJSON)
```

Even more details as much as we want!!!

```
var myJSON = {"car1" : {"color":"black", "model":"Mustang"}, "car2" :  
{"color" : "blue", "model":"F150" }};  
console.log(myJSON)
```





# JavaScript Object Arrays

Values can be arrays or other objects.  
Can have a name or not. Check in the JSONLint you will find both are valid JSON.

## TRY IT :

1. Open Your editor
2. Create an object with multiple items
3. Output the values in the console.

```
let val = 0;
const myObj1 = {
  "people": [{
    "firstName": "Mike"
    , "lastName": "Smith"
    , "age": 30
  }, {
    "firstName": "John"
    , "lastName": "Jones"
    , "age": 40
  }]
};
var myObj2 = [{
  "firstName": "Mike"
  , "lastName": "Smith"
  , "age": 30
}, {
  "firstName": "John"
  , "lastName": "Jones"
  , "age": 40
}];

console.log(myObj1.people);
console.log(myObj2);
console.log(myObj1.people[val]);
console.log(myObj2[val]);
val++;
console.log(myObj1.people[val]);
console.log(myObj2[val]);
```

# JavaScript Looping

JavaScript makes it easy to loop through data of an object.

## TRY IT :

1. Open Your editor
2. Create an object with multiple items
3. Output the ALL values in the console.

```
var myObj = {  
  "people": [ {  
    "firstName": "Mike"  
    , "lastName": "Smith"  
    , "age": 30  
  }, {  
    "firstName": "John"  
    , "lastName": "Jones"  
    , "age": 40  
  }]  
  , "places": [ {  
    "location": "Toronto"  
    , "lat": 87  
    , "long": 140  
  }, {  
    "location": "New York"  
    , "lat": 67  
    , "long": 110  
  }]  
};  
for (let i = 0; i < myObj.people.length; i++) {  
  console.log(myObj.people[i].firstName + " " +  
myObj.people[i].lastName);  
}  
myObj.people.forEach(function (item) {  
  console.log(item.firstName + " " + item.lastName);  
})
```

# JavaScript MORE Looping

JavaScript makes it easy to loop through data of an object.

## TRY IT :

1. Open Your editor
2. Use for ... in.. to output content into console.
3. Try keys and entries output to console.
4. Try map to build new object, and output to console.
5. Try with places from the object as well.

```
for (let prop in myObj.people) {  
  console.log(prop);  
  console.log(myObj.people[prop].firstName + " " +  
myObj.people[prop].lastName);  
}  
console.log(Object.keys(myObj));  
console.log(Object.entries(myObj));  
console.log(Object.entries(myObj)[0]);  
const myObj2 = myObj.people.map(function (key, index) {  
  console.log(key, index);  
  return key;  
});  
console.log(myObj2);
```

# JavaScript JSON Parse and Stringify

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

```
Laurence
```

```
15
```

```
16
```

```
{"first": "Laurence", "count": 16}
```

```
const json = '{"first": "Laurence", "count": 15}';  
const obj = JSON.parse(json);  
console.log(obj.first);  
console.log(obj.count);  
obj.count++;  
console.log(obj.count);  
const str = JSON.stringify(obj);  
console.log(str);
```

## TRY IT :

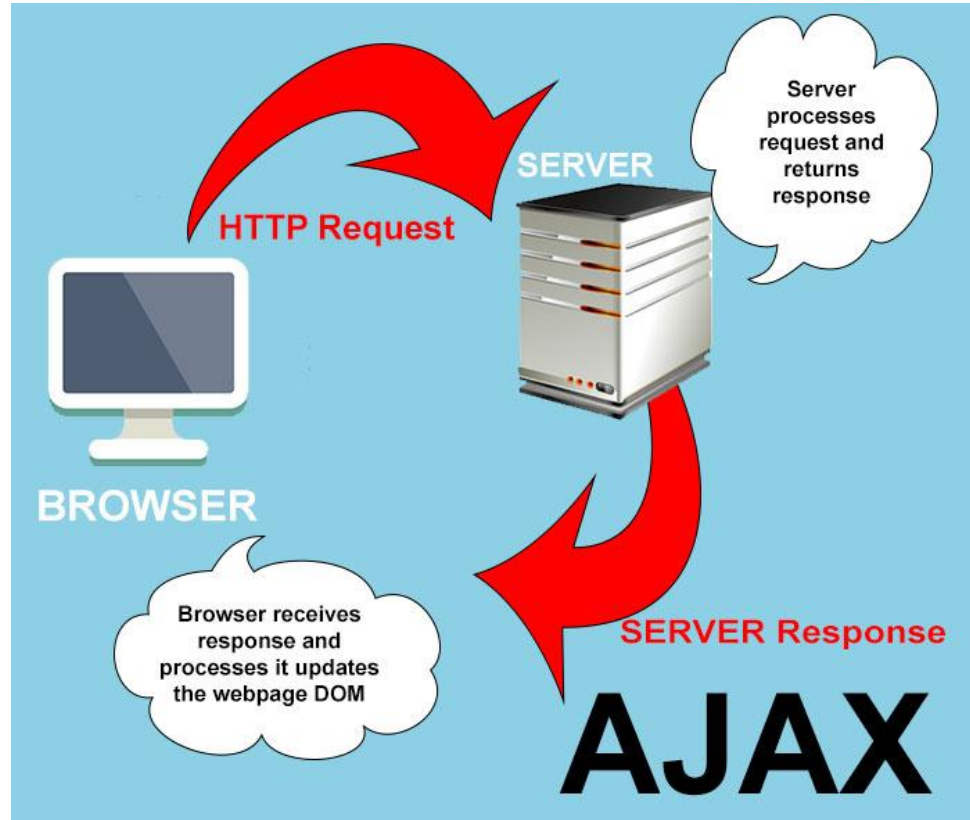
1. Open Your editor
2. Convert a string into an object
3. Output object values into the console
4. Convert an object into a string
5. Output new string into the console.

# AJAX API requests

- No refresh
- Can request external data from a server
- Can receive external data from a server
- Can Send Data to a Server

Typically when a user goes to a website the user has to wait for the server to respond with the data. This is not the case as with AJAX we have the option to load the data when the user is already on the page making the user really happy.

APIs are made up of requests and responses



# What is AJAX

- **Not a programming language** - group of technologies
- Combination of technology within the browser XHR object + JavaScript to connect the request data and the Document Object Model DOM
- Not only for XML but can be used and commonly used JSON and text data

AJAX set of Web development techniques using many Web technologies on the client side to create **asynchronous Web applications**.

With Ajax, Web applications can **send and retrieve data** from a server asynchronously (in the background) without interfering with the display and behavior of the existing page



# Meet the Players of AJAX

**JavaScript** - front end scripting language provides functionality for web pages. XMLHttpRequest or fetch to end point.

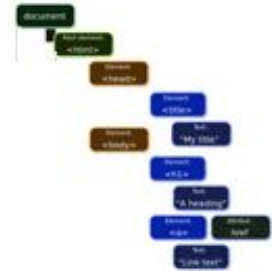
**DOM** - Document Object Model lets JavaScript update the page.

**Server Side** - server side language to process request and output result. Can connect with databases, add logic, and any typically server functionality. Returns response.

**JavaScript Object Notation** or **JSON** - transmit data objects consisting of attribute–value pairs and array data types



{ DOM }



# xHR request

The XMLHttpRequest.readyState property returns the state an XMLHttpRequest client is in. An XHR client exists in one of the following states:

- 0 UNSENT Client has been created. open() not called yet.
- 1 OPENED open() has been called.
- 2 HEADERS\_RECEIVED send() has been called, and headers and status are available.
- 3 LOADING Downloading; responseText holds partial data.
- 4 DONE The operation is complete.

<https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest/readyState>

<https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest/open>

## TRY IT :

1. Open Your editor
2. Make the connection to  
<https://api.myjson.com/bins/1lzpg>

```
var xhr = new XMLHttpRequest();
console.log('UNSENT', xhr.readyState);
xhr.open('GET', 'https://api.myjson.com/bins/1lzpg', true);
console.log('OPENED', xhr.readyState);
xhr.onprogress = function () {
    console.log('LOADING', xhr.readyState);
};
xhr.onload = function () {
    console.log('DONE', xhr.readyState);
};
xhr.send(null);
```

```
var xhr2 = new XMLHttpRequest();
xhr2.open('get', 'https://api.myjson.com/bins/1lzpg');
xhr2.onreadystatechange = function () {
    if (xhr2.readyState === 4) {
        if (xhr2.status === 200) {
            alert(xhr2.responseText);
        }
        else {
            alert('Error: ' + xhr2.status);
        }
    }
};
xhr2.send(null);
```



# JavaScript xHR Request AJAX

Get data from random user website using xHR request from JavaScript. Output it to the webpage.

## TRY IT :

1. Open Your editor
2. Create page elements select using JavaScript
3. Create a new XMLHttpRequest, using GET to connect to the URL.
4. Output the random first name to the output element.
5. Output other data to the second output2 element.

```
<div id="output1">People List  
  <br> </div>  
<div id="output2"></div>  
<script>  
  var output1 = document.getElementById('output1');  
  var output2 = document.getElementById('output2');  
  var xHR = new XMLHttpRequest();  
  var url = "https://randomuser.me/api/?results=1";  
  xHR.onreadystatechange = function () {  
    console.log(xHR.readyState);  
    if (this.readyState == 4 && this.status == 200) {  
      var myObj = JSON.parse(xHR.responseText);  
      console.log(myObj.results[0].name.first);  
      output1.innerHTML = myObj.results[0].name.first;  
    }  
  }  
  xHR.open("GET", url, true);  
  xHR.send();  
  console.log(xHR);  
</script>
```

# JavaScript More Results Looping

Try with more results.

## TRY IT :

1. Use for loop to get 10 results from randomuser
2. Output the results into the output1 element.



```
<div id="output1">People List</div>
<script>
  const output1 = document.getElementById('output1');
  const xHR = new XMLHttpRequest();
  const url = "https://randomuser.me/api/?results=10";
  xHR.onreadystatechange = function () {
    if (this.readyState == 4 && this.status == 200) {
      const myObj = JSON.parse(xHR.responseText);
      console.log(myObj.results);
      for (let x = 0; x < myObj.results.length; x++) {
        let tempName = myObj.results[x].name;
        output1.innerHTML += `${tempName.first}
${tempName.last} <br>`;
      }
    }
  }
  xHR.open("GET", url, true);
  xHR.send();
</script>
```

# JavaScript fetch

The Fetch API provides an interface for fetching resources (including across the network). It will seem familiar to anyone who has used XMLHttpRequest, but the new API provides a more powerful and flexible feature set.

[https://developer.mozilla.org/en-US/docs/Web/API/Fetch\\_API](https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API)

## TRY IT :

1. Connect to the randomuser API using fetch
2. Get data and update the webpage

```
<div id="output1"></div>
<script>
  const output1 = document.getElementById('output1');
  const url = "https://randomuser.me/api/?results=1";
  fetchData();

  function fetchData() {
    fetch(url).then(function (rep) {
      return rep.json()
    }).then(function (data) {
      console.log(data.results[0].name);
      let temp = data.results[0].name;
      output1.textContent = `${temp.first} ${temp.last}`;
    })
  }
</script>
```

# JavaScript Map

The `map()` method creates a new array with the results of calling a provided function on every element in the calling array.

[https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global\\_Objects/Array/map](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/map)

## TRY IT :

1. Use `fetch` and `map` to create a new object
2. Output into the HTML using the new object

```
<div id="output1"></div>
<script>
  const output1 = document.getElementById('output1');
  const url = "https://randomuser.me/api/?results=1";
  fetchData();
  function fetchData() {
    fetch(url).then(function (rep) {
      return rep.json()
    }).then(function (data) {
      let temp = data.results[0].name;
      let person = data.results.map(function (el) {
        return el.name;
      })
      output1.textContent = `${person[0].first}
${person[0].last}`;
      console.log(person[0]);
    })
  }
</script>
```

# JavaScript Fetch More Results Looping

Try with more results now with fetch.

## TRY IT :

1. Use for loop to get 10 results from randomuser
2. Output the results into the output1 element.



```
<div id="output1"></div>
<script>
  const output1 = document.getElementById('output1');
  const url = "https://randomuser.me/api/?results=10";
  fetchData();

  function fetchData() {
    fetch(url).then(function (rep) {
      return rep.json()
    }).then(function (data) {
      let people = data.results.map(function (el) {
        return el.name;
      })
      people.forEach(function (person) {
        output1.innerHTML += `${person.first}
${person.last} <br>`;
      })
    })
  }
</script>
```

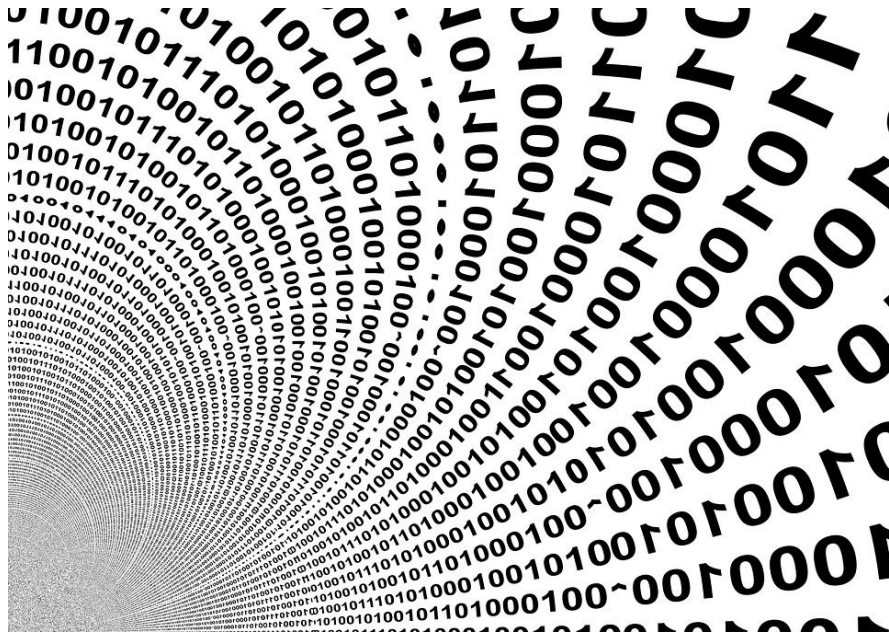
# Server JSON

<https://my-json-server.typicode.com/>

We need a backend server to simulate the server handling of the request.

JSON Server helps you to setup a REST API with CRUD operations very fast.

CRUD - create, read, update, and delete are the four basic functions of persistent storage



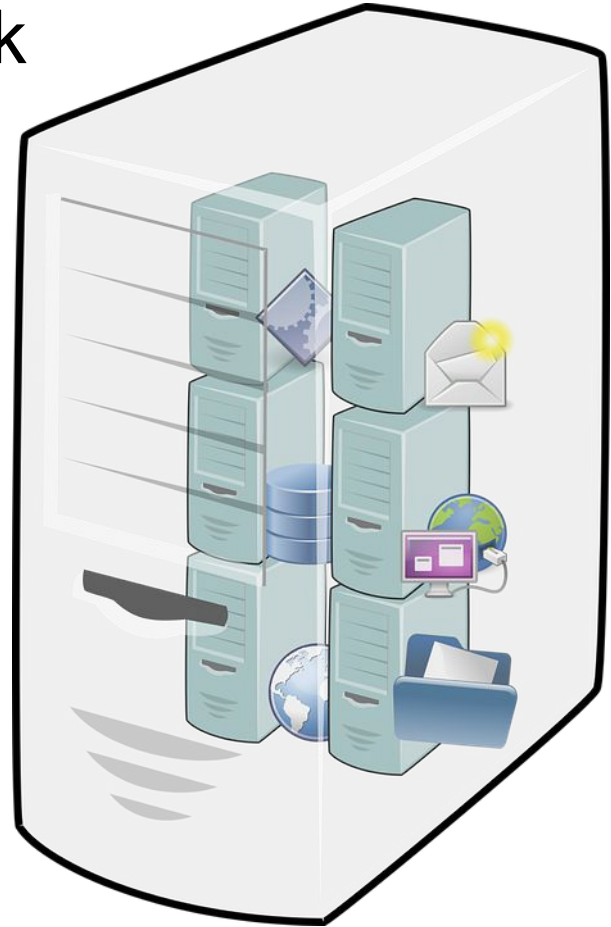
# What is Node and how does it work

JavaScript on the server.

- run JavaScript on the server
- open source server environment
- can connect to a database
- Create, open, read write, delete files on your server
- Generate page content

Node.js runs single-threaded, asynchronously programming, which is very memory efficient.

<https://nodejs.org/en/about/>



# Node.js

Node files have a js extension

Download and install node at

<https://nodejs.org>

Terminal - for command line interface

Node package manager - share code and reuse existing code. Faster development.

<https://www.npmjs.com/>

Install <https://www.npmjs.com/get-npm>

You will also need an editor - brackets.io

| LTS<br>Recommended For Most Users  | Current<br>Latest Features   |   |
|--|--|---|
| <br>Windows Installer<br><small>node-v10.15.3-x64.msi</small> | <br>macOS Installer<br><small>node-v10.15.3.pkg</small> | <br>Source Code<br><small>node-v10.15.3.tar.gz</small> |

Windows Installer (.msi)

Windows Binary (.zip)

macOS Installer (.pkg)

macOS Binary (.tar.gz)

Linux Binaries (x64)

Linux Binaries (ARM)

Source Code

|                      |        |       |
|----------------------|--------|-------|
| 32-bit               | 64-bit |       |
| 32-bit               | 64-bit |       |
| 64-bit               |        |       |
| 64-bit               |        |       |
| 64-bit               |        |       |
| ARMv6                | ARMv7  | ARMv8 |
| node-v10.15.3.tar.gz |        |       |

## Additional Platforms

SmartOS Binaries

Docker Image

Linux on Power Systems

Linux on System z

AIX on Power Systems

|                               |
|-------------------------------|
| 64-bit                        |
| Official Node.js Docker Image |
| 64-bit                        |
| 64-bit                        |
| 64-bit                        |



# Windows Terminal

Windows - <https://cmder.net/> or use the command prompt terminal

**Launch the Command Prompt** - use the Run window or press the Win + R keys on your keyboard. Then, type cmd and press Enter.

List current directory of files - **dir**

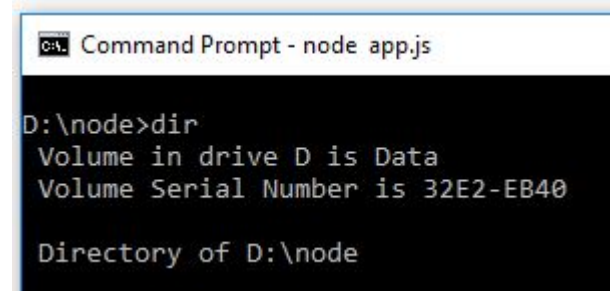
Change directory to D drive - **cd D:**

Change directory down one level - **cd..**

Change directory to folder by name - **cd folder**

Make new folder - **mkdir folderName**

Get help - **help**



```
C:\> Command Prompt - node app.js

D:\node>dir
Volume in drive D is Data
Volume Serial Number is 32E2-EB40

Directory of D:\node
```

# Mac Terminal

Open Terminal by pressing Command+Space or select terminal in the applications list.

List current directory of files - **ls**

Change directory to D drive - **cd D:**

Change directory down one level - **cd..**

Change directory to folder by name - **cd  
folder**

Make new folder - **mkdir folderName**

Get help - **help**

# Command Line Launch

One node is installed check to see version installed. Type **node -v**

Open your editor and create a js file that contains **console.log('Hello World');** save it as test.js

In the terminal type **node test.js** and watch for a result. What gets returned?

NPM - check if its installed **npm -v** latest version install **npm install npm@latest -g**

```
D:\node>node -v  
v6.11.3
```

```
test.js
```

```
1 console.log('Hello World');
```

```
D:\node>node test.js  
Hello World
```

```
D:\node>npm -v  
6.9.0
```

# First Node File

Once node is installed check to see version installed. Type **node -v**

Open your editor and create a js file that contains **console.log('Hello World');** save it as test.js

In the terminal type **node test.js** and watch for a result. What gets returned?

## TRY IT :

1. Setup node and run a js file with simple console message.

```
D:\node>node -v  
v6.11.3
```

tester.js

```
1 console.log('hello');
```

Command Prompt

```
D:\node>node tester.js  
hello
```

```
D:\node>
```

# Setting REST API With JSON Server

-g - installs the application globally

More info

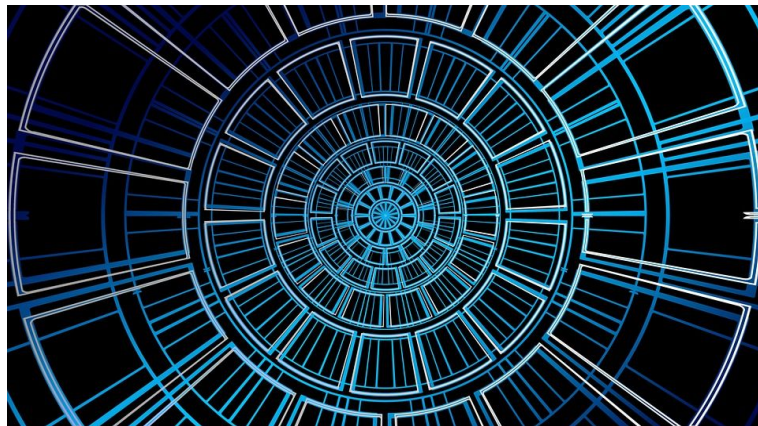
<https://github.com/typicode/json-server>

```
json-server --watch db.json
```

```
npm install -g json-server
```

## TRY IT :

1. Install JSON server **npm install -g json-server**



# Create a database

<https://github.com/typicode/json-server>

You can use Ctrl+C to stop the node application.

JSON Server

Congrats!

You're successfully running JSON Server

🎉🎉🎉🎉🎉

Resources

/people 3x

To access and modify resources, you can use any HTTP method

GET POST PUT PATCH DELETE OPTIONS

```
{
  "people": [
    {
      "id": 1
      , "first": "Laurence"
      , "last": "Svekis"
    }
    , {
      "id": 2
      , "first": "Jane"
      , "last": "Doe"
    }
    , {
      "id": 1
      , "first": "John"
      , "last": "Smith"
    }
  ]
}
```

```
json-server --watch db.json
```

## TRY IT :

1. Setup a file called db.json
2. Open your browser go to <http://localhost:3000/people>
3. Open to <http://localhost:3000/> to see resources
4. <http://localhost:3000/people/2>
5. Filter <http://localhost:3000/people?id=1>

# Endpoints are created

GET /people

GET /people/{id}

POST /people

PUT /people/{id}

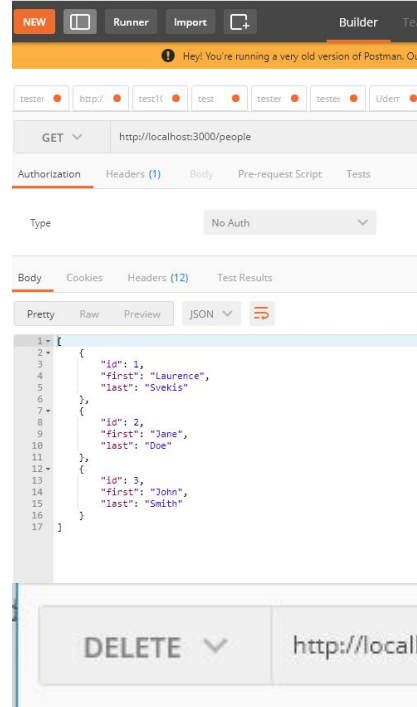
PATCH /people/{id}

DELETE /people/{id}

<http://jsonplaceholder.typicode.com/>

Get postman to test endpoints

<https://chrome.google.com/webstore/detail/postman/fhbjgbfijnjbdgggehcdcbncdddomop/>



## TRY IT :

1. Open Postman app in chrome.
2. In the address bar using GET paste `http://localhost:3000/people/1`
3. Try with delete
4. Check your db.json file. (should remove the selected item by id)

# Post to Database

## POST /people

Simulate form submission

### TRY IT :

1. Select x-www-form-urlencoded add keys first and last.  
Create some values for the keys
2. Select POST and hit send
3. Check you db.json file

POST ▼ http://localhost:3000/people/ Params Send ▼

Authorization Headers (2) **Body** ● Pre-request Script Tests

☐ form-data ☒ x-www-form-urlencoded ☐ raw ☐ binary

|                                     | Key     | Value    | Description |
|-------------------------------------|---------|----------|-------------|
| <input checked="" type="checkbox"/> | first   | laurence |             |
| <input checked="" type="checkbox"/> | last    | svekis   |             |
|                                     | New key | Value    | Description |

```
    "first": "John",  
    "last": "Smith"  
  },  
  {  
    "first": "laurence",  
    "last": "svekis",  
    "id": 4  
  },  
  {  
    "first": "laurence",  
    "last": "svekis",  
    "id": 5  
  }  
]
```



# Put to Database

## PUT /people/5

Simulate form item update

### TRY IT :

1. Select x-www-form-urlencoded add keys first and last. Create some values for the keys. Change the path to an existing id value.
2. Select PUT and hit send
3. Check you db.json file

PUT ▼ http://localhost:3000/people/5 Params Send

Authorization Headers (2) **Body** Pre-request Script Tests

☐ form-data ☒ x-www-form-urlencoded ☐ raw ☐ binary

| Key                                       | Value      | Description |
|---|------------|-------------|
| <input checked="" type="checkbox"/> first | New        |             |
| <input checked="" type="checkbox"/> last  | LastNameer |             |
| New key                                   | Value      | Description |

```
{
  "first": "laurence",
  "last": "svekis",
  "id": 4
},
{
  "first": "New",
  "last": "LastNameer",
  "id": 5
}
```

# Put to Database

## PUT /people/5

Simulate form item update

### TRY IT :

1. Select x-www-form-urlencoded add keys first and last. Create some values for the keys. Change the path to an existing id value.
2. Select PUT and hit send
3. Check you db.json file

PUT ▼ http://localhost:3000/people/5 Params Send

Authorization Headers (2) **Body** Pre-request Script Tests

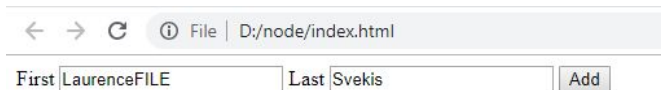
☐ form-data ☒ x-www-form-urlencoded ☐ raw ☐ binary

| Key                                       | Value      | Description |
|---|------------|-------------|
| <input checked="" type="checkbox"/> first | New        |             |
| <input checked="" type="checkbox"/> last  | LastNameer |             |
| New key                                   | Value      | Description |

```
{
  "first": "laurence",
  "last": "svekis",
  "id": 4
},
{
  "first": "New",
  "last": "LastNameer",
  "id": 5
}
```

# Post new Item from web page

Add a new entry to the database from index.html You can open the file directly in a browser or launch it as live preview in Brackets editor.



First Laurence Last Svekis Add

## TRY IT :

1. Create HTML and JavaScript code to submit form contents to the json db file.
2. Return response on new submission.

```
First <input type="text" name="first" value="Laurence"> Last <input
type="text" name="last" value="Svekis"> <button>Add</button>
<script>
  const btn = document.querySelector("button");
  const first = document.querySelector("input[name=first]");
  const last = document.querySelector("input[name=last]");
  btn.addEventListener('click', adder);
  function adder() {
    fetch("http://localhost:3000/people/", {
      method: "POST"
      , body: JSON.stringify({
        first: first.value
        , last: last.value
      })
      , headers: {
        "Content-Type": "application/json"
      }
    }).then(function (res) {
      return res.text();
    }).then(function (res) {
      return console.log(res);
    })
  }
</script>
```

# Get contents

The `forEach()` method executes a provided function once for each array element.

## TRY IT :

1. Create HTML and JavaScript code to get all contents of the json db file.
2. Return response on button click and output the data to the HTML page.

```
<div class="pageContent"></div>
<button>GET</button>
<script>
  const btn = document.querySelector('button');
  const pageContent = document.querySelector('.pageContent');
  btn.addEventListener('click', adder);
  function adder() {
    fetch("http://localhost:3000/people/").then(function (response) {
      return response.json();
    }).then(function (data) {
      output(data);
    });
  }
  function output(data) {
    pageContent.innerHTML = "";
    data.forEach(function (el, index) {
      let div = document.createElement('div');
      div.innerHTML = `${el.id} ${el.first} ${el.last}<br>`;
      pageContent.appendChild(div);
      console.log(el)
    });
  }
</script>
```

# Install json server use custom node

npm install json-server --save-dev

```
// server.js
const jsonServer = require('json-server')
const server = jsonServer.create()
const router = jsonServer.router('db.json')
const middlewares = jsonServer.defaults()

server.use(middlewares)
server.use(router)
server.listen(3000, () => {
  console.log('JSON Server is running')
})
```

## TRY IT :

1. Open editor and add the file server.js
2. Install json-server folder `npm install json-server --save-dev`
3. Type `node server.js` into your console.
4. Go to browser and open to url `http://localhost:3000/people`

```
D:\node>node server.js
JSON Server is running
GET /people?id=8 304 10.657 ms - -
GET /people 200 5.615 ms - -
```

# Final Project Add, List and Update Data

Use what you learned in the early lessons to list out all the people from the database. Allow the user to update the people as well as add new ones.

## TRY IT :

1. From the list create input fields that can have a button that updates the field data into the database.
2. Create a function that receives the updated data to send to database

```
function output(data) {
    pageContent.innerHTML = "";
    data.forEach(function (el, index) {
        let div = document.createElement('div');
        div.innerHTML = `${el.id} <input type="text"
value="${el.first}"> <input type="text"
value="${el.last}"><button>Update</button><br>`;
        div.addEventListener('click', function () {
            let temps = div.querySelectorAll('input');
            let updater = div.querySelector('button');
            updater.addEventListener('click', function () {
                updateData(el.id, temps[0].value, temps[1].value)
            })
        })
        pageContent.appendChild(div);
    });
}
```

# Final Project Add, List and Update Data

Put all the pieces together to complete the application.

## TRY IT :

1. Using PUT and the path of the item from the database update the item
2. Ensure it works updating the database

```
function updateData(id, first, last) {  
  console.log(first);  
  fetch("http://localhost:3000/people/" + id, {  
    method: "PUT"  
    , body: JSON.stringify({  
      first: first  
      , last: last  
    })  
    , headers: {  
      "Content-Type": "application/json"  
    }  
  }).then(function (res) {  
    return res.text();  
  }).then(function (res) {  
    return console.log(res);  
  })  
}
```

# Final Project Add, List and Update Data

```
<input type="text" name="first" value="Laurence"> Last  
<input type="text" name="last" value="Svekis">  
<button class="addItem">Add</button>  
<div class="pageContent"></div>  
<button class="listItems">GET</button>  
<script>  
  const btn1 = document.querySelector('.addItem');  
  const btn2 = document.querySelector('.listItems');  
  const pageContent = document.querySelector('.pageContent');  
  const first = document.querySelector('input[name=first]');  
  const last = document.querySelector('input[name=last]');  
  btn1.addEventListener('click', adder);  
  btn2.addEventListener('click', getter);  
  function getter() {  
    fetch('http://localhost:3000/people/').then(function (response)  
{  
      return response.json();  
    }).then(function (data) {  
      output(data);  
    });  
  }  
}
```

```
function output(data) {  
  pageContent.innerHTML = "";  
  data.forEach(function (el, index) {  
    let div = document.createElement('div');  
    div.innerHTML = `${el.id} <input type="text"  
value="${el.first}"> <input type="text"  
value="${el.last}"><button>Update</button><br>`;  
    div.addEventListener('click', function () {  
      let temps = div.querySelectorAll('input');  
      let updater = div.querySelector('button');  
      updater.addEventListener('click', function () {  
        updateData(el.id, temps[0].value, temps[1].value)  
      })  
    })  
    pageContent.appendChild(div);  
  });  
}
```



# Final Project Add, List and Update Data

```
function updateData(id, first, last) {
  console.log(first);
  fetch("http://localhost:3000/people/" + id, {
    method: "PUT"
    , body: JSON.stringify({
      first: first
      , last: last
    })
    , headers: {
      "Content-Type": "application/json"
    }
  }).then(function (res) {
    return res.text();
  }).then(function (res) {
    return console.log(res);
  })
}
```

```
function adder() {
  addDB({
    first: first.value
    , last: last.value
  });
}

function addDB(data) {
  var options = {
    method: 'POST'
    , headers: {
      'Content-Type': 'application/json'
    }
    , body: JSON.stringify(data)
  };
  return fetch('http://localhost:3000/people',
options).then(function (response) {
  getter();
  return response.json();
  });
}
```

# Congratulations on completing the section!

## Thank you

This ebook uses <https://developer.mozilla.org/en-US/docs/Web/JavaScript> as a source for examples. Check out more about JavaScript at MDN.

Find out more about my courses at <http://www.discoveryvip.com/>

**Course instructor : Laurence Svekis -  
providing online training to over  
500,000 students across hundreds of  
courses and many platforms.**

