CSIT128:
Introduction to Web Technology

JSON

Joseph Tonien
School of Computing and Information Technology
University of Wollongong
2019

JavaScript Object Notation (JSON)

- In most web applications, XML and JSON are used to store or transport data
- JSON is "self-describing" and easy to understand

This is an example of a JSON describing a student object:

```
{
  "fullname": "John Smith",
  "studentNumber": "U1234567",
  "age": 20,
  "csMajor": true
}
```

JSON

- Data is in name/value pairs
- Data is separated by commas
- Curly braces hold objects

```
{
  "fullname": "John Smith",
  "studentNumber": "U1234567",
  "age": 20,
  "csMajor": true
}
```

JSON

Square brackets hold arrays

```
[
  {
    "firstName":"John",
    "lastName":"Smith"
  },
  {
    "firstName":"Kate",
    "lastName":"Williams"
  }
]
```

JSON

- Curly braces hold objects
- Square brackets hold arrays

```
{
  "firstName":"John",
  "lastName":"Smith",
  "subjectList":[
    {
      "code":"MATH101",
      "title":"Algebra"
    },
    {
      "code":"CSIT122",
      "title":"C programming"
    }
  ]
}
```

JSON

Translate from Javascript object to JSON string

```
objJSON = JSON.stringify(obj);
```

Translate from JSON string to javascript object

```
obj = JSON.parse(objJSON);
```

JSON

OBJECT

```
{
  fullname: "John Smith",
  studentNumber: "U1234567",
  age: 20,
  csMajor: true
}
```

JSON.stringify

JSON.parse

JSON

```
{
  "fullname": "John Smith",
  "studentNumber": "U1234567",
  "age": 20,
  "csMajor": true
}
```

JSON.stringify

The **`JSON.stringify`** method converts a JavaScript value to a JSON string.

```
Syntax:  JSON.stringify(jsvalue, replacer, space)
```

- `jsvalue:` the javascript value to convert to a JSON string.
- `replacer` (Optional): selecting/filtering which properties of the object to be included in the JSON string. If the `replacer` is null or not provided, all properties of the object are included in the resulting JSON string.
- `space` (Optional): use for indentation, specifying white spaces in the output JSON string for readability purposes.

JSON.stringify function demo

Enter information to construct a student object:

Full name: John Smith

Student number: U1234567

Age: 20

CompSci major: ☐

Click View buttons to see JSON string of the student object.

View `JSON.stringify(studentObj)`

```
{"fullname":"John Smith","studentNumber":"U1234567","age":20,"csMajor":false}
```

View `JSON.stringify(studentObj, null, 2)`

```
{
  "fullname": "John Smith",
  "studentNumber": "U1234567",
  "age": 20,
  "csMajor": false
}
```

View `JSON.stringify(studentObj, ["studentNumber", "csMajor"]);`

```
{"studentNumber":"U1234567","csMajor":false}
```

View `JSON.stringify(studentObj, ["studentNumber", "csMajor"], 2)`

```
{
    "studentNumber": "U1234567",
    "csMajor": false
}
```

JSON.stringify

```
var studentObj = {
  fullname: "John Smith",
  studentNumber: "U1234567",
  age: 20,
  csMajor: false
};
```



```
JSON.stringify(studentObj)
```

```
{"fullname":"John Smith","studentNumber":"U1234567","age":20,
"csMajor":false}
```

output JSON sticks together
make it hard to read

JSON.stringify

```
var studentObj = {
  fullname: "John Smith",
  studentNumber: "U1234567",
  age: 20,
  csMajor: false
};
```

```
JSON.stringify(studentObj, null, 2)
```

using 2 spaces indentation

```
{
  "fullname": "John Smith",
  "studentNumber": "U1234567",
  "age": 20,
  "csMajor": false
}
```

JSON.stringify

```
var studentObj = {
  fullname: "John Smith",
  studentNumber: "U1234567",
  age: 20,
  csMajor: false
};
```

```
JSON.stringify(studentObj, ["studentNumber", "csMajor"])
```

only output the student number
and compsci major

```
{"studentNumber":"U1234567","csMajor":false}
```

JSON.stringify

```
var studentObj = {
  fullname: "John Smith",
  studentNumber: "U1234567",
  age: 20,
  csMajor: false
};
```

```
JSON.stringify(studentObj, ["studentNumber", "csMajor"], 2)
```

only output the student number and compsci major, using 2 spaces indentation

```
{
  "studentNumber": "U1234567",
  "csMajor": false
}
```

Example 1: `JSON.stringify`

```
function showObjectJSON(){
  //create a student object
  var studentObj = {};
  studentObj.fullname = "John Smith";
  studentObj.studentNumber = "U1234567";
  studentObj.age = 20;
  studentObj.csMajor = true;

  //get JSON string from the javascript object
  var studentJSON = JSON.stringify(studentObj);

  //print the JSON string to the console
  console.log(studentJSON);
}

<button onClick="showObjectJSON()">
Click here to see JSON string
</button>
```

Example 2: `JSON.parse`

```
function showObject(){
 //JSON string
 var studentJSON = '{"fullname":"John Smith","studentNumber":
"U1234567","age":20,"csMajor":true}';

 //get javascript object from JSON string
 var studentObj = JSON.parse(studentJSON);

 //print the object to the console
 console.log(studentObj);
 console.log("Full name is " + studentObj.fullname);
}


<button onClick="showObject()">
Click here to see object from JSON
</button>
```

Example 3: `JSON.stringify`

```
function showArrayJSON(){
  var user1 = {};
  user1.firstName = "John";
  user1.lastName = "Smith";

  var user2 = {};
  user2.firstName = "Kate";
  user2.lastName = "Williams";

  //create an array of user objects
  var userList = [user1, user2];

  //get JSON string from the javascript array
  var userListJSON = JSON.stringify(userList);

  //print the JSON string to the console
  console.log(userListJSON);
}
```

```
<button onClick="showArrayJSON()">
Click here to see JSON string
</button>
```

Example 4: `JSON.parse`

```
function showArray(){
 //JSON string
 var userListJSON = '[{"firstName":"John","lastName":"Smith"},
                    {"firstName":"Kate","lastName":"Williams"}]';

 //get javascript array from JSON string
 var userList = JSON.parse(userListJSON);

 //print the object to the console
 console.log(userList);
 console.log("There are " + userList.length + " users");
}
```

```
<button onClick="showArray()">
Click here to see array from JSON
</button>
```

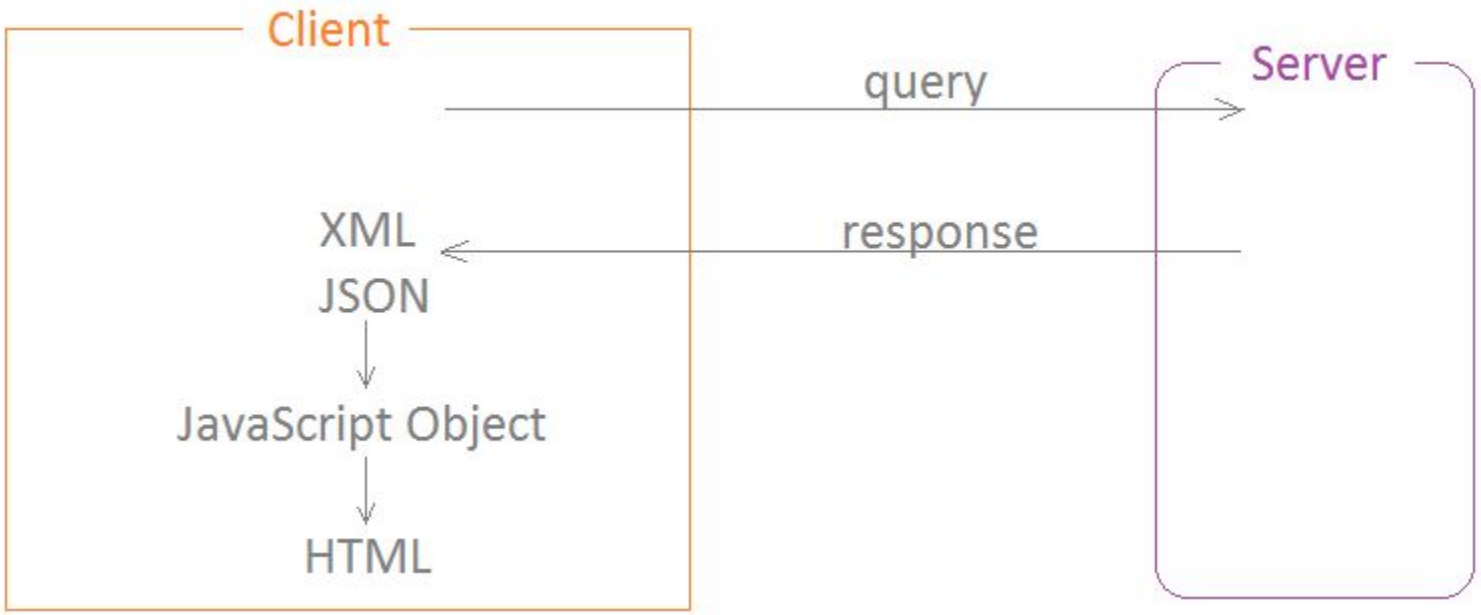
Example 5: `JSON.stringify`

```
function showObjectJSON(){
  var studentObj = {}; //create a student object
  studentObj.firstName = "John";
  studentObj.lastName = "Smith";
  studentObj.subjectList = []; //empty array to hold subjects

  var subjectObj1 = {};
  subjectObj1.code = "MATH101";
  subjectObj1.title = "Algebra";
  //add subject into array
  studentObj.subjectList.push(subjectObj1);

  var subjectObj2 = {};
  subjectObj2.code = "CSIT122";
  subjectObj2.title = "C programming";
  //add subject into array
  studentObj.subjectList.push(subjectObj2);

  //get JSON string from obj and print it on console
  var studentJSON = JSON.stringify(studentObj, null, 2);
  console.log(studentJSON);
}
```

Example 5: `JSON.stringify`

```
{
  "firstName":"John",
  "lastName":"Smith",
  "subjectList":[
    {
      "code":"MATH101",
      "title":"Algebra"
    },
    {
      "code":"CSIT122",
      "title":"C programming"
    }
  ]
}
```

AJAX: Review



Writing AJAX/JSON application:

- **Step 1**: Make the query
- **Step 2**: Get the response JSON
- **Step 3**: Parse the JSON response into a JavaScript object
- **Step 4**: Display the JavaScript object in a HTML page

In AJAX/JSON application, Step 1 and Step 4 are the same as in AJAX/XML application. Only Step 2 and Step 3 are different.

AJAX: Step 2 - Get the response

```
// handler for the readyState change

function readyStateChangeHandler(xhttp){
  if (xhttp.readyState == 4){

    if(xhttp.status == 200){

      handleStatusSuccess(xhttp);

    }else{

      handleStatusFailure(xhttp);

    }

  }

}
```

readyState = 4 means DONE

status = 200 means OK

status is not OK

In the callback function, we know the request is successful when `XMLHttpRequest.readyState` **= 4** and `XMLHttpRequest.status` **= 200**.

AJAX: Step 2 - Get the response

If `XMLHttpRequest.readyState = 4` and `XMLHttpRequest.status = 200` then we know the request is successful. It is time to get the response and process it.

`XMLHttpRequest.responseText:`
Use this for JSON format response.
Returns a DOMString that contains the response to the request as text.

```
// XMLHttpRequest success
function handleStatusSuccess(xhttp){
  // for JSON response format
  var jsonText = xhttp.responseText;

  // parse the json into an object
  var obj = JSON.parse(jsonText);

  // display the object on the page
  display(obj);
}
```

This is different

This is different

AJAX: Step 3 - Parse the JSON response into JS object

In AJAX/XML application, we have to write a function to parse the XML response into a JavaScript object.

In AJAX/JSON application, the parsing is done by an easy function call:

```
// parse the json into an object
var obj = JSON.parse(jsonText);
```

```
{
  "queryLocation": "Wollongong",
  "forecast": "Mostly Cloudy",
  "temperature": {
    "degree": "21",
    "scale": "C"
  },
  "humidity": "66%",
  "windSpeed": "18 km/h"
}
```

JSON.parse

```
weatherObj{
  queryLocation: "Wollongong",
  forecast: "Mostly Cloudy",
  temperatureDegree: "21",
  temperatureScale: "C",
  humidity: "66%",
  windSpeed: "18 km/h",
}
```

AJAX/JSON Example:

Hello World

AJAX/JSON Example: Hello World

The purpose of this example is

- to show how to make AJAX call,
- to see how the callback function get executed, and
- to debug the values of
  - `XMLHttpRequest.readyState`
  - `XMLHttpRequest.status`
  - `XMLHttpRequest.responseText`

In the server, we have a file called `hello.json`. We will use AJAX to get this file.

We will print out on the console the values of `XMLHttpRequest.readyState`, `XMLHttpRequest.status` and `XMLHttpRequest.responseText`.

AJAX/JSON Example: Hello World

```
<button onClick="makeAjaxQuery()">
Get JSON file using AJAX
</button>
```

When the user clicks the button, make an Ajax call to get the json file `hello.json`.

```
function makeAjaxQuery(){
  // create an XMLHttpRequest
  var xhttp = new XMLHttpRequest();

  // create a handler for the readyState change
  xhttp.onreadystatechange = function() {
    readyStateChangeHandler(xhttp);
  };

  // get JSON file by making async call
  xhttp.open("GET", "hello.json", true);
  xhttp.send();
}
```

AJAX/JSON Example: Hello World

```
// handler for the readyState change
function readyStateChangeHandler(xhttp){

  console.log("Callback function readyStateChangeHandler is executed");

  // print the status and readyState on the console
  console.log("readyState = " + xhttp.readyState);
  console.log("status = " + xhttp.status);

  // print the responseText on the console
  console.log("responseText:");
  console.log(xhttp.responseText);
}
```

Let’s have a look at the console to see the execution of this callback function.

AJAX/JSON Example: Hello World

We can see on the console the state of the request changing and it **triggers the execution** of the callback function.

Elements Console Sources Network Perfo

top Filter

```
Callback function readyStateChangeHandler is executed
readyState = 1
status = 0
responseText:

Callback function readyStateChangeHandler is executed
readyState = 2
status = 200
responseText:

Callback function readyStateChangeHandler is executed
readyState = 3
status = 200
responseText:
{
  "message":"Hello World!"
}
Callback function readyStateChangeHandler is executed
readyState = 4
status = 200
responseText:
{
  "message":"Hello World!"
}
```

`XMLHttpRequest.readyState`

Returns an `unsigned short`, the state of the request:

Value	State	Description
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.

When state = 3 (LOADING) and state = 4 (DONE) we also see the response JSON

AJAX/JSON Example: Hello World - file not found case

```
<button onClick="makeAjaxQuery()">
Get JSON file using AJAX
</button>
```

Now modify the code so that when the user clicks on the button, make an Ajax call to get the json file `frog.json`. This JSON file is missing and we expect an error.

```
function makeAjaxQuery(){
  // create an XMLHttpRequest
  var xhttp = new XMLHttpRequest();

  // create a handler for the readyState change
  xhttp.onreadystatechange = function() {
    readyStateChangeHandler(xhttp);
  };

  // get JSON file by making async call
  xhttp.open("GET", "frog.json", true);
  xhttp.send();
}
```

AJAX/JSON Example: Hello World - file not found case

We can see on the console the state of the request changing and it triggers the execution of the callback function.

Since the json file `frog.json` does not exists in the server we get the status=**404** NOT FOUND

Elements Console Sources Network Performance

top Filter

```
Callback function readyStateChangeHandler is executed
readyState = 1
status = 0
responseText:

▸GET frog.json 404 (Not Found)
Callback function readyStateChangeHandler is executed
readyState = 2
status = 404
responseText:

Callback function readyStateChangeHandler is executed
readyState = 3
status = 404
responseText:
...
<title>404 - Page Not Found - Directory Services @ UOW</title>
...
Callback function readyStateChangeHandler is executed
readyState = 4
status = 404
responseText:
...
<title>404 - Page Not Found - Directory Services @ UOW</title>
...
```

`XMLHttpRequest.readyState`

Returns an `unsigned short`, the state of the request:

Value	State	Description
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.

A sample AJAX/JSON program

A sample AJAX/JSON program

This is the main function:

Step 1: Make the query

```
function makeAjaxQuery(){
  // create an XMLHttpRequest
  var xhttp = new XMLHttpRequest();

  // create a handler for the readyState change
  xhttp.onreadystatechange = function() {
    readyStateChangeHandler(xhttp);
  };

  // making query by async call
  xhttp.open("GET", "url-to-query-the-server", true);
  xhttp.send();
}


// handler for the readyState change
function readyStateChangeHandler(xhttp){ … }
```

A sample AJAX/JSON program

This is the callback function:

```
// handler for the readyState change
function readyStateChangeHandler(xhttp){
  if (xhttp.readyState == 4){
    // readyState = 4 means DONE
    if(xhttp.status == 200){
      // status = 200 means OK
      handleStatusSuccess(xhttp);
    }else{
      // status is NOT OK
      handleStatusFailure(xhttp);
    }
  }
}

// XMLHttpRequest failed
function handleStatusFailure(xhttp){ … }

// XMLHttpRequest success
function handleStatusSuccess(xhttp){ … }
```

A sample AJAX/JSON program

```
// XMLHttpRequest success
function handleStatusSuccess(xhttp){

  var jsonText = xhttp.responseText;

  // parse the json into an object
  var obj = JSON.parse(jsonText);

  // display the object on the page
  display(obj);
}
```

Step 2: *Get the response JSON*

Step 3: *Parse the JSON response into a JavaScript object*

Step 4: Display the object in a HTML page

A sample AJAX/JSON program

```
// parse the json into an object
var obj = JSON.parse(jsonText);
```

Step 3: *Parse the JSON response into a JavaScript object.*

Note that this step is done by an easy function call JSON.parse()

```
// display the object on the page
function display(obj){
  // construct HTML code to display the object
  ...
}
```

Step 4: Display the object in a HTML page

The main job the AJAX/JSON program is to write the function: **`display`**

AJAX/JSON Example:

Weather Forecast

This example emulates an application where a server allows the user to retrieve current weather forecast for a queried location.

Get Weather JSON

Wollongong

Mostly Cloudy

21 °C

Humidity: 66%
Wind speed: 18 km/h

AJAX/JSON Example: Weather Forecast

The purpose of this example is

- to show how to distinguish between a failed request and a successful request
- when the request is failed, display an error message
- when the request is successfully then display the weather information:
    1. parse the JSON response to a JavaScript weather object;
    2. display the weather object on the web page.

AJAX/JSON Example: Weather Forecast

Get Weather JSON

Wollongong

Mostly Cloudy

21 °C

Humidity: 66%
Wind speed: 18 km/h

```
<button onClick="makeAjaxQueryWeather()">
Get Weather JSON
</button>

<br /><br />

<div id="display">
</div>
```

```
function makeAjaxQueryWeather(){
  // create an XMLHttpRequest
  var xhttp = new XMLHttpRequest();

  // create a handler for the readyState change
  xhttp.onreadystatechange = function() {
    readyStateChangeHandler(xhttp);
  };

  // get JSON file by making async call
  xhttp.open("GET", "weather.json", true);
  xhttp.send();
}
```

AJAX/JSON Example: Weather Forecast

```
// handler for the readyState change

function readyStateChangeHandler(xhttp){
  if (xhttp.readyState == 4){
    // readyState = 4 means DONE
    if(xhttp.status == 200){
      // status = 200 means OK
      handleStatusSuccess(xhttp);
    }else{
      // status is NOT OK
      handleStatusFailure(xhttp);
    }
  }
}

function handleStatusFailure(xhttp){ … }

function handleStatusSuccess(xhttp){ … }
```

AJAX/JSON Example: Weather Forecast

When the request is failed, display an error message

```
// XMLHttpRequest failed
function handleStatusFailure(xhttp){

  // display error message

  var displayDiv = document.getElementById("display");

  displayDiv.innerHTML = "XMLHttpRequest failed: status " + xhttp.status;
}
```

AJAX/JSON Example: Weather Forecast

When the request is successful

```
// XMLHttpRequest success
function handleStatusSuccess(xhttp){

  var jsonText = xhttp.responseText;

  // parse the json into an object
  var weatherObj = JSON.parse(jsonText);

  // display the object on the page
  displayWeather(weatherObj);
}
```

Get the response JSON

Parse the JSON response into a JavaScript object

Display the object in a HTML page

AJAX/JSON Example: Weather Forecast

```
// parse the json into an object
var weatherObj = JSON.parse(jsonText);
```

What is the `weatherObj` look like?

```
{
  "queryLocation": "Wollongong",
  "forecast": "Mostly Cloudy",
  "temperature": {
    "degree": "21",
    "scale": "C"
  },
  "humidity": "66%",
  "windSpeed": "18 km/h"
}
```

→

```
weatherObj {
  queryLocation: "Wollongong",
  forecast: "Mostly Cloudy",
  temperature: {
    degree: "21",
    scale: "C"
  },
  humidity: "66%",
  windSpeed: "18 km/h"
}
```

parse the JSON response into a JavaScript object

AJAX/JSON Example: Weather Forecast

```
// display the weather object on the page
function displayWeather(weatherObj){
   ...
}
```

```
weatherObj {
  queryLocation: "Wollongong",
  forecast: "Mostly Cloudy",
  temperature
    degree: "21",
    scale: "C"
  },
  humidity: "66%",
  windSpeed: "18 km/h"
}
```

Wollongong

Mostly Cloudy

21 °C

Humidity: 66%
Wind speed: 18 km/h

*We need to construct the following **HTML code** to display the weather information*

```
<h1>Wollongong</h1>
<font size='5' color='gray'>Mostly Cloudy</font>
<br /><br />

<font size='7'>21</font>
&deg; C
<br /><br />

<i>Humidity: 66%</i>
<br />

<i>Wind speed: 18 km/h</i>
```

AJAX/JSON Example: Weather Forecast

```
// display the weather object on the page
function displayWeather(weatherObj){
   ...
}
```

```
weatherObj {
  queryLocation: "Wollongong",
  forecast: "Mostly Cloudy",
  temperature
    degree: "21",
    scale: "C"
  },
  humidity: "66%",
  windSpeed: "18 km/h"
}
```

Wollongong

Mostly Cloudy

21 °C

Humidity: 66%
Wind speed: 18 km/h

Q: How to we get the query location?

A:
weatherObj.queryLocation

```
<h1>Wollongong</h1>
<font size='5' color='gray'>Mostly Cloudy</font>
<br /><br />

<font size='7'>21</font>
&deg; C
<br /><br />

<i>Humidity: 66%</i>
<br />

<i>Wind speed: 18 km/h</i>
```

AJAX/JSON Example: Weather Forecast

```
// display the weather object on the page
function displayWeather(weatherObj){
   ...
}
```

```
weatherObj {
  queryLocation: "Wollongong",
  forecast: "Mostly Cloudy",
  temperature: {
    degree: "21",
    scale: "C"
  },
  humidity: "66%",
  windSpeed: "18 km/h"
}
```

Wollongong

Mostly Cloudy

21 °C

Humidity: 66%
Wind speed: 18 km/h

```
<h1>Wollongong</h1>
<font size='5' color='gray'>Mostly Cloudy</font>
<br /><br />

<font size='7'>21</font>
&deg; C
<br /><br />

<i>Humidity: 66%</i>
<br />

<i>Wind speed: 18 km/h</i>
```

Q: How to we get the temperature scale?

A:

weatherObj.temperature.scale

AJAX/JSON Example: Weather Forecast

```
// display the weather object on the page
function displayWeather(weatherObj){
  // construct HTML code to display weather information
  var html = "<h1>" + weatherObj.queryLocation + "</h1>";

  html = html + "<font size='5' color='gray'>" +  weatherObj.forecast + "</font>";
  html = html + "<br /><br />";

  html = html + "<font size='7'>" +  weatherObj.temperature.degree + "</font>";
  html = html + "&deg;" +  weatherObj.temperature.scale;
  html = html + "<br /><br />";

  html = html + "<i>Humidity: " +  weatherObj.humidity + "</i>";
  html = html + "<br />";

  html = html + "<i>Wind speed: " +  weatherObj.windSpeed + "</i>";

  // show the constructed HTML code in the display div
  var displayDiv = document.getElementById("display");
  displayDiv.innerHTML = html;
}
```

AJAX/JSON Example:

Stock Market

This example emulates an application where a server allows the user to retrieve stock market information.

AJAX/JSON Example: Stock Market

Assume that there is a JSON file, called `market.json`. Write HTML and JavaScript codes that do the following:

There is a button “Click here to view Stock Market Activity”. When the user clicks on this button, make an Ajax call to get the stock information from the json file and display them in a table.

Click here to view Stock Market Activity

Stock Market Activity 24/02/2015 11:30:00

Stock	Value	Change	Net / %
NASDAQ	4725.64	-37.58▼	0.79%
NASDAQ-100 (NDX)	4312.01	-29.38▼	0.68%
Pre-Market (NDX)	4316.29	-25.1▼	0.58%
After Hours (NDX)	4320.61	8.6▲	0.2%
DJIA	17651.26	-99.65▼	0.56%
S&P 500	2051.12	-12.25▼	0.59%
Russell 2000	1113.13	-8.62▼	0.77%

AJAX/JSON Example: Stock Market

This is the content of the JSON file `market.json`


```
{
  "queryTime": "24/02/2015 11:30:00",
  "stockList": [
    {
      "name": "NASDAQ",
      "value": 4725.64,
      "change": -37.58,
      "netpct": 0.79
    },
    {
      "name": "NASDAQ-100 (NDX)",
      "value": 4312.01,
      "change": -29.38,
      "netpct": 0.68
    },
    ....
    {
      "name": "Russell 2000",
      "value": 1113.13,
      "change": -8.62,
      "netpct": 0.77
    }
  ]
}
```

Version 0 - plain display

```
marketObj {
  queryTime: "24/02/2015 11:30:00",
  stockList: [
    {
      name: "NASDAQ",
      value: 4725.64,
      change: -37.58,
      netpct: 0.79
    },
    {
      name: "NASDAQ-100 (NDX)",
      value: 4312.01,
      change: -29.38,
      netpct: 0.68
    },
    ....
    {
      name: "Russell 2000",
      value: 1113.13,
      change: -8.62,
      netpct: 0.77
    }
  ]
}
```

```
// display the market object on the page
function displayMarket(marketObj){
  // construct HTML code to display market information
  var html = "";

  html += "queryTime: " + marketObj.queryTime;
  html += "<br /><br />";

  for(var i=0; i < marketObj.stockList.length; i++){
    var stockObj = marketObj.stockList[i];

    html += "name: " + stockObj.name;
    html += "<br />";

    html += "value: " + stockObj.value;
    html += "<br />";

    html += "change: " + stockObj.change;
    html += "<br />";

    html += "netpct: " + stockObj.netpct;
    html += "<br /><br />";
  }

  // show the constructed HTML code in the display div
  var displayDiv = document.getElementById("display");
  displayDiv.innerHTML = html;
}
```

queryTime: 24/02/2015 11:30:00

name: NASDAQ
value: 4725.64
change: -37.58
netpct: 0.79

name: NASDAQ-100 (NDX)
value: 4312.01
change: -29.38
netpct: 0.68

...

name: Russell 2000
value: 1113.13
change: -8.62
netpct: 0.77

Version 1 - table display

```
// display the object on the page
function displayMarket(marketObj){
   ...
}
```

```
marketObj{
  queryTime: "24/02/2015 11:30:00",
  stockList: [
    {
      name: "NASDAQ",
      value: 4725.64,
      change: -37.58,
      netpct: 0.79
    },
    {
      name: "NASDAQ-100 (NDX)",
      value: 4312.01,
      change: -29.38,
      netpct: 0.68
    }, ...
  ]
}
```

```
<h2>Stock Market Activity  24/02/2015 11:30:00</h2>

<table border='1'>

  <tr> <th>Stock</th> <th>Value</th> <th>Change</th> <th>Net / %</th> </tr>

  <tr>
    <td><b>NASDAQ</b></td>
    <td align='right'> 4725.64</td>
    <td style='color:red' align='right'>
      -37.58
      <img src='stockDown.png' />
    </td>
    <td align='right'> 0.79%</td>
  </tr>

  <tr>
    <td><b>After Hours (NDX)</b></td>
    <td align='right'> 4320.61</td>
    <td style='color:green' align='right'>
      8.6
      <img src='stockUp.png' />
    </td>
    <td align='right'> 0.2%</td>
  </tr>
</table>
```

*We need to construct the following **HTML code** to display the stock market information*

Stock Market Activity 24/02/2015 11:30:00

Index	Value	Change	Net / %
NASDAQ	4725.64	-37.58▼	0.79%
NASDAQ-100 (NDX)	4312.01	-29.38▼	0.68%
Pre-Market (NDX)	4316.29	-25.1▼	0.58%
After Hours (NDX)	4320.61	8.6▲	0.2%
DJIA	17651.26	-99.65▼	0.56%
S&P 500	2051.12	-12.25▼	0.59%
Russell 2000	1113.13	-8.62▼	0.77%

Version 1 - table display

```
// display the market object on the page
function displayMarket(marketObj){
  // construct HTML code to display market information
  var html = "<h2>Stock Market Activity " + marketObj.queryTime + "</h2>";

  html += "<table border='1'>";
  html += "<tr><th>Stock</th><th>Value</th><th>Change</th><th>Net / %</th></tr>";

  for(var i=0; i < marketObj.stockList.length; i++){

    var stockObj = marketObj.stockList[i];

    html += "<tr>";

    html += "<td><b>" + stockObj.name + "</b></td>";
    html += "<td align='right'>" + stockObj.value + "</td>";

    if(stockObj.change < 0){
      html += "<td style='color:red' align='right'>";
      html += stockObj.change;
      html += "<img src='stockDown.png' />";
      html += "</td>";
    }else{
      html += "<td style='color:green' align='right'>";
      html += stockObj.change;
      html += "<img src='stockUp.png' />";
      html += "</td>";
    }

    html += "<td align='right'>" + stockObj.netpct + "%</td>";
    html += "</tr>";
  }

  html += "</table>";

  // show the constructed HTML code in the display div
  var displayDiv = document.getElementById("display");
  displayDiv.innerHTML = html;
}
```

```
marketObj {
  queryTime: "24/02/2015 11:30:00",
  stockList: [
    {
      name: "NASDAQ",
      value: 4725.64,
      change: -37.58,
      netpct: 0.79
    },
    {
      name: "NASDAQ-100 (NDX)",
      value: 4312.01,
      change: -29.38,
      netpct: 0.68
    },
    ....
    {
      name: "Russell 2000",
      value: 1113.13,
      change: -8.62,
      netpct: 0.77
    }
  ]
}
```

Stock Market Activity 24/02/2015 11:30:00

Index	Value	Change	Net / %
NASDAQ	4725.64	-37.58▼	0.79%
NASDAQ-100 (NDX)	4312.01	-29.38▼	0.68%
Pre-Market (NDX)	4316.29	-25.1▼	0.58%
After Hours (NDX)	4320.61	8.6▲	0.2%
DJIA	17651.26	-99.65▼	0.56%
S&P 500	2051.12	-12.25▼	0.59%
Russell 2000	1113.13	-8.62▼	0.77%

References

- `http://www.w3schools.com/json`

- Robert W. Sebesta, *Programming the World Wide Web*, Pearson.