Internet Technologies

AJAX and JSON

Mahesh Kumar
[maheshkumar@andc.du.ac.in]

Course Web Page
[www.mkbhandari.com/mkwiki]

Outline

1. What is AJAX?
2. Why use AJAX?
3. How AJAX Works?
4. Handling AJAX Requests & Responses
5. Data Formats
6. JSON
7. Working with JSON Data

What is AJAX?

- AJAX (**A**synchronous **J**avaScript **A**nd **X**ML) is a **technique** for loading data into part of a page without having to refresh the entire page.
  - → *improves the user experience*
  - → *process of sending and receiving data on the fly without the page reload.*
- AJAX is **not** a programming language, it just uses a combination of:
  1. *A browser built-in **XMLHttpRequest** object (to request data from a web server)*
  2. ***JavaScript** and **HTML DOM** (to display or use the data)*
- AJAX allows web pages to be updated **asynchronously** by exchanging data with a web server behind the scenes.

What is AJAX?

- Examples:

1. **Live search** (or **autocomplete**) commonly uses Ajax.

2. **Online shopping**, adding items to shopping cart and updating it without leaving the page.

3. **Websites with user-generated content** may allow you to display your information on your on websites.

4. **Registering for a website** a script may check whether your username is available before you have completed the rest of the form.

Why use AJAX?

- Ajax uses an **asynchronous processing model**.
  - → *user can do other things while the web browser is waiting for the data to load, speeding up the user experience.*

1. **Using AJAX while pages are loading**

   - → *When a browser comes across a **<script>** tag, it will typically stop processing the rest of the page until it has **loaded** and **processed** that script. This is known as a **synchronous processing model.***

   - → *With Ajax, the browser can request some data from a server and - once that data has been requested - continue to load the rest of the page and process the user's interactions with the page. It is known as an **asynchronous (or non-blocking) processing model.***

Why use AJAX?

- Ajax uses an **asynchronous processing model**.
  - → *user can do other things while the web browser is waiting for the data to load, speeding up the user experience.*

2 Using AJAX when pages have loaded

- → *Once a page has loaded, if you want to update what the user sees in the browser window, typically you would refresh the entire page. This means that the user has to wait for a whole new page to download and be rendered by the browser.*

- → *With AJAX, if you only want to update a part of the page, you can just update the content of one element. This is done by intercepting an event (such as the user clicking on a link or submitting a form) and requesting the new content from the server using an asynchronous request.*

Why use AJAX?

- Ajax uses an **asynchronous processing model**.
  - → *user can do other things while the web browser is waiting for the data to load, speeding up the user experience.*

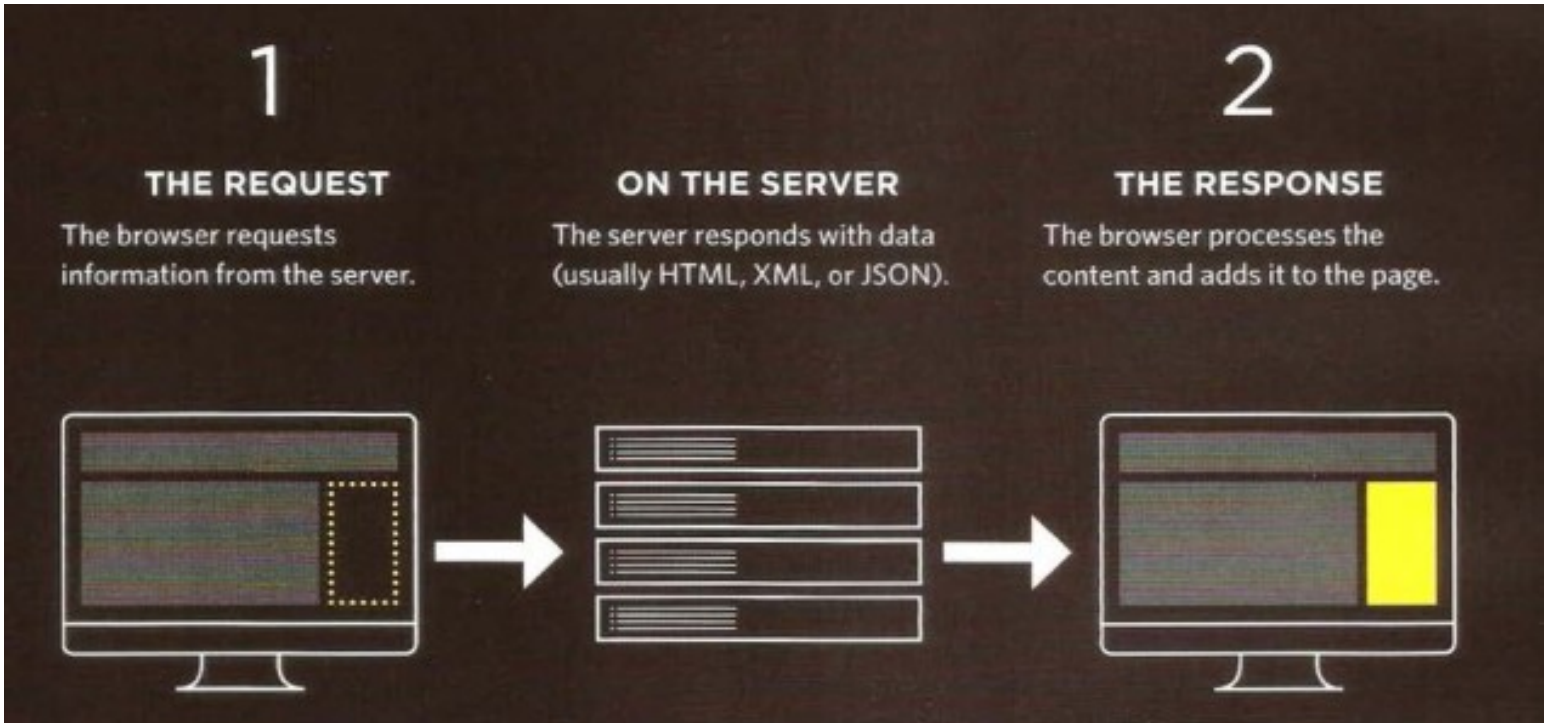
2 **Using AJAX when pages have loaded**

- → *While that data is loading, the user can continue to interact with the rest of the page. Then, once the server has responded, a special Ajax event will trigger another part of the script that reads the new data from the server and updates just that one part of the page.*

- → *Because you do not have to refresh the whole page, the data will load faster and the user can still use the rest of the page while they are waiting.*

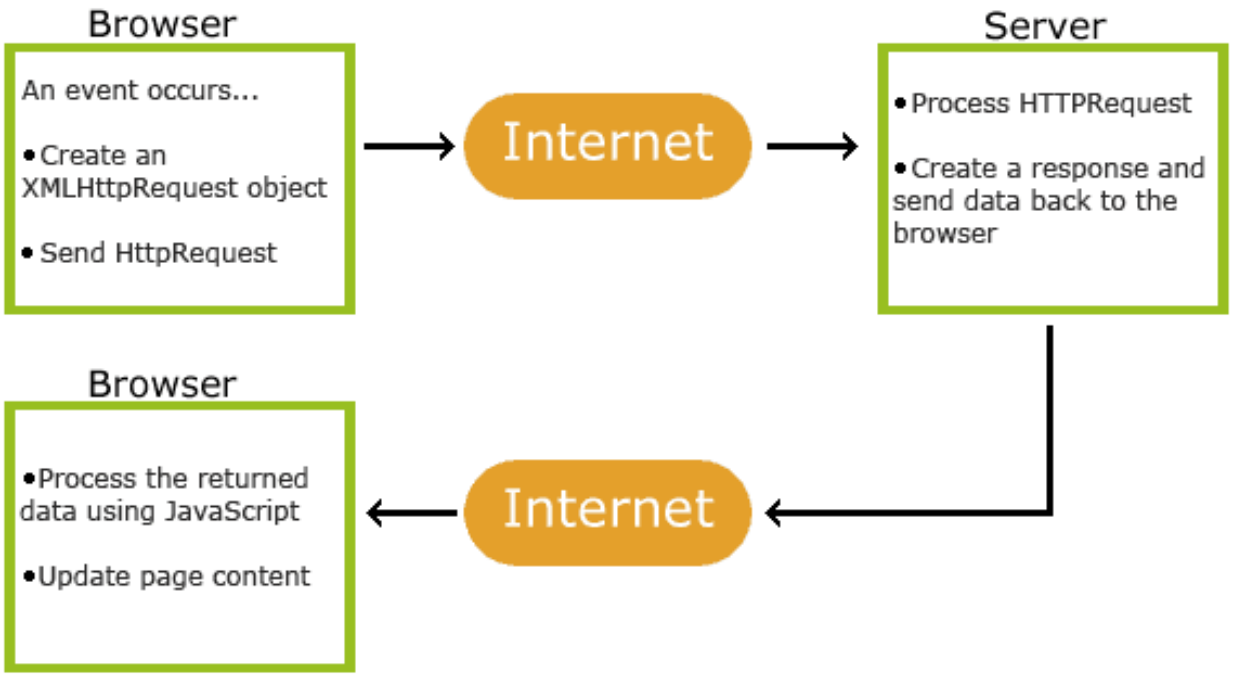
How AJAX Works?

- When using AJAX, the browser **requests** information from a web server. It then **processes** the server's response and **shows** it within the page.



How AJAX Works?

- When using AJAX, the browser **requests** information from a web server. It then **processes** the server's response and **shows** it within the page.



How AJAX works **[2]**

How AJAX Works?

1. *An **event** occurs in a web page (the **page is loaded**, a **button is clicked**).*
2. *An **XMLHttpRequest** object is created by JavaScript.*
3. *The **XMLHttpRequest** object **sends a request** to a web server*
4. *The **server processes the request** using Server-side technologies like **ASP.net, PHP, JSP, NodeJS, or Ruby.***
5. *The **server sends a response** back to the web page in data formats like **HTML, XML,** or **JSON,** which the browser ultimately turns into **HTML.***
6. *The response is read by JavaScript*
7. *Finally JavaSricpt will **process the data and incorporate it into the part of the page**(without affecting the rest of page). Proper action (like page update) is performed by JavaScript.*

Handling AJAX Requests & Responses

- To create an Ajax request, browsers use the **XMLHttpRequest object**.
- When the server responds to the browser's request, the same **XMLHttpRequest object** will process the result.
- ***The Request***

```
① var xhr = new XMLHttpRequest();
② xhr.open('GET', 'data/test.json', true);
③ xhr.send('search=arduino');
```

Handling AJAX Requests & Responses

1. An instance of the XMLHttpRequest object is created using object constructor notation and stores the object in a variable name **xhr**.

2. The XMLHttpRequest object's **open()** method prepares the request. It has three parameters:
   - → *The HTTP method (GET or POST)*
   - → *The **URL** of the page that will handle your request*
   - → *A Boolean indicating if it should be asynchronous*

3. The **send()** method is the one that sends the prepared request to the server. Extra information can be passed to the server in the parentheses. If no extra information is sent, you may see the keyword null used (although it is not strictly needed)

Handling AJAX Requests & Responses

- ***The Response***

```
① xhr.onload = function() {
②   if (xhr.status === 200) {
      // Code to process the results from the server
    }
  }
```

Handling AJAX Requests & Responses

1. When the browser has received and loaded a response from the server, the **onload** event will fire.

   → *This will trigger a function (here, it is an anonymous fu nction).*

2. The function checks the status property of the object. This is used to make sure the server's response was okay.

   → *When a server responds to any request, it should send back a status message, to indicate if it completed the request. The values can be:*

200	*The server has responded and all is ok*
304	*Not modified*
404	*Page not found*
500	*Internal error on the server*

Data Formats

- The response to an Ajax request usually comes in one of **three formats:**

1. HTML
2. XML
3. JSON

Data Formats - HTML

- When you want to update a section of a web page, it is the simplest way to get data into a page.

BENEFITS

- It is easy to write, request, and display.
- The data sent from the server goes straight into the page. There's no need for the browser to process it (as with the other two methods).

DRAWBACKS

- The server must produce the HTML in a format that is ready for use on your page.
- It is not well-suited for use in applications other than web browsers. It does not have good **data portability**.
- The request must come from the same domain* (see below).

Data Formats - XML

- XML looks similar to HTML, but the tag names are different because they describe the data that they contain. The syntax is also more strict than HTML.

BENEFITS

- It is a flexible data format and can represent complex structures.
- It works well with different platforms and applications.
- It is processed using the same DOM methods as HTML.

DRAWBACKS

- It is considered a verbose language because the tags add a lot of extra characters to the data being sent.
- The request must come from the same domain as the rest of the page* (see below).
- It can require a lot of code to process the result.

Data Formats - JSON

- JavaScript Object Notation (JSON) uses a similar syntax to object literal notation in order to represent data.

BENEFITS

- It can be called from any domain (see JSON-P/CORS).
- It is more concise (less verbose) than HTML/XML.
- It is commonly used with JavaScript (and is gaining wider use across web applications).

DRAWBACKS

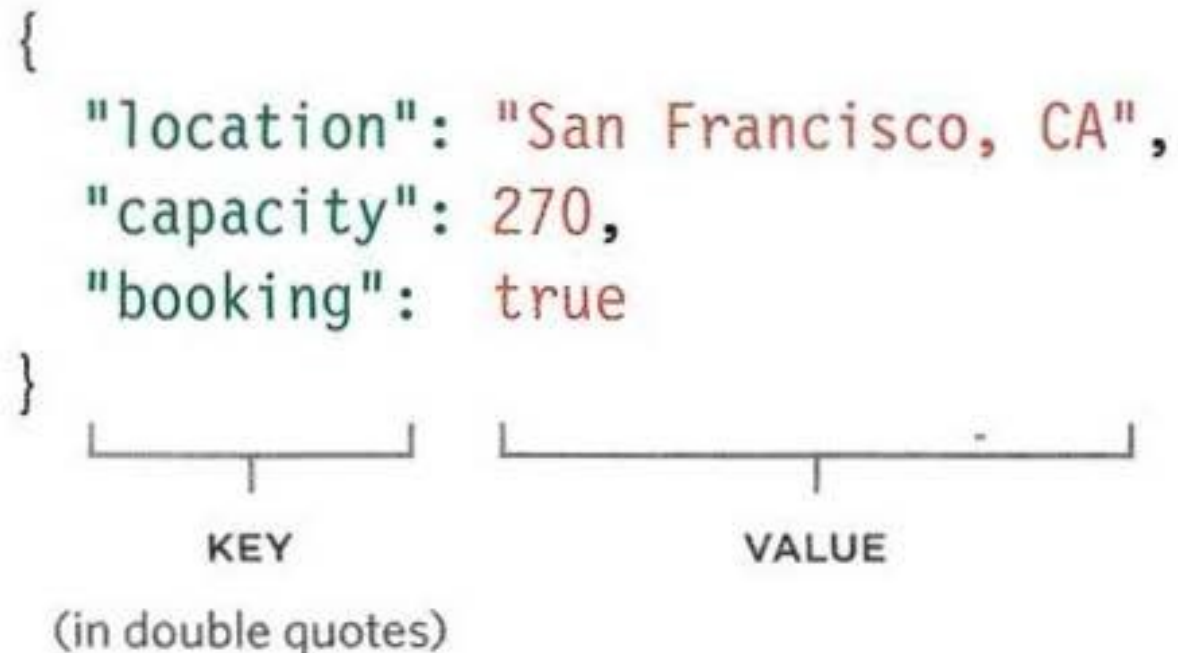
- The syntax is not forgiving. A missed quote, comma, or colon can break the file.
- Because it is JavaScript, it can contain malicious content (see XSS on p228). Therefore, you should only use JSON that has been produced by trusted sources.

Introduction to JSON

- JSON (JavaScript Object Notation JSON) is a lightweight **data-interchange format.**
- Popular and agreed upon format for sending, receiving, and storing data.
- It looks very similar to object literal syntax, but it is not an object.
  - → *JSON data looks like the object literal notation, however, it is just **plain text data** (not an object).*
  - → *HTML is also just plain text, and the browser converts it into DOM objects.*
  - → *You cannot transfer the actual objects over a network. Rather, you send text which is converted into objects by the browser.*
  - → *Code for reading and generating JSON exists in many programming languages.*

Introduction to JSON

- JSON Example 1:



Introduction to JSON

KEYS

In JSON, the key should be placed in **double quotes** (not single quotes).

The key (or name) is separated from its value by a colon.

Each key/value pair is separated by a comma. However, note that there is *no* comma after the last key/value pair.

VALUES

The value can be any of the following data types (some of these are demonstrated above; others are shown on the right-hand page):

DATA TYPE	DESCRIPTION
`string`	Text (must be written in quotes)
`number`	Number
`Boolean`	Either `true` or `false`
`array`	Array of values – this can also be an array of objects
`object`	JavaScript object – this can contain child objects or arrays
`null`	This is when the value is empty or missing

Introduction to JSON

- JSON Example 2:’

  Objects and Arrays nested inside each other

```
{
  "events": [
    {
      "location": "San Francisco, CA",
      "date": "May 1",
      "map": "img/map-ca.png"
    },
    {
      "location": "Austin, TX",
      "date": "May 15",
      "map": "img/map-tx.png"
    },
    {
      "location": "New York, NY",
      "date": "May 30",
      "map": "img/map-ny.png"
    }
  ]
}
```

OBJECT ARRAY

Working with JSON Data

- JavaScript's JSON object can turn JSON data into a JavaScript object.
- It can also convert a JavaScript object into a string.

- **JSON.stringify()** converts JavaScript objects into a string, formatted using JSON.
  - → *It allows you to send JavaScript objects from the browser to another application.*

- **JSON.parse()** processes a string containing JSON data.
  - → *It converts the JSON data into a JavaScript objects ready for the browser to use .*

Loading HTML with AJAX

- Whether HTML, XML, or JSON is being returned from the server, the process of setting up the Ajax request and checking whether the file is ready to be worked with is the same. What changes is how you deal with t he data tha t is returned.

JAVASCRIPT c08/js/data-html.js

```
① var xhr = new XMLHttpRequest();                    // Create XMLHttpRequest object

④ xhr.onload = function() {                          // When response has loaded
    // The following conditional check will not work locally - only on a server
⑤   if(xhr.status === 200) {                         // If server status was ok
⑥     document.getElementById('content').innerHTML = xhr.responseText; // Update
    }
  };

② xhr.open('GET', 'data/data.html', true);           // Prepare the request
③ xhr.send(null);                                    // Send the request
```

Loading JSON with AJAX

- When JSON data is sent from a server to a web browser, it is transmitted as a string.

  → *your script must then convert the **string into a JavaScript object** (known as **deserializing an object**) done using the **parse()** method of a built-in object called **JSON** (a global object, so you can use it without creating an instance of it first.)*

  → *Once the string has been parsed, your script can access the data in the object and create HTML using the **innerHTML** property.*

  → *The JSON object also has a method called **stringify()**, which converts objects into a string using JSON notation so it can be sent from the browser back to a server. This is also known as **serializing an object**.*

JAVASCRIPT

c08/js/data-json.js

```
var xhr = new XMLHttpRequest();                    // Create XMLHttpRequest object

xhr.onload = function() {                          // When readystate changes
 if(xhr.status === 200) {                          // If server status was ok
① responseObject = JSON.parse(xhr.responseText);

  // BUILD UP STRING WITH NEW CONTENT (could also use DOM manipulation)
② var newContent = '';
  for (var i = 0; i < responseObject.events.length; i++) {//Loop through object
    newContent += '<div class="event">';
    newContent += '<img src="' + responseObject.events[i].map + '" ';
③   newContent += 'alt="' + responseObject.events[i].location + '" />';
    newContent += '<p><b>' + responseObject.events[i].location + '</b><br>';
    newContent += responseObject.events[i].date + '</p>';
    newContent += '</div>';
  }

  // Update the page with the new content
④ document.getElementById('content').innerHTML = newContent;

 }
};

xhr.open('GET', 'data/data.json', true);           // Prepare the request
xhr.send(null);                                    // Send the request
```

References

1. JavaScript and JQuery – Interactive Front-end Web Development, (Jon Duckett), John Wiley and Sons, Inc.

2. https://www.w3schools.com/xml/ajax_intro.asp