Web Application Development

2019

AJAX

Asynchronous JavaScript and XML

AJAX

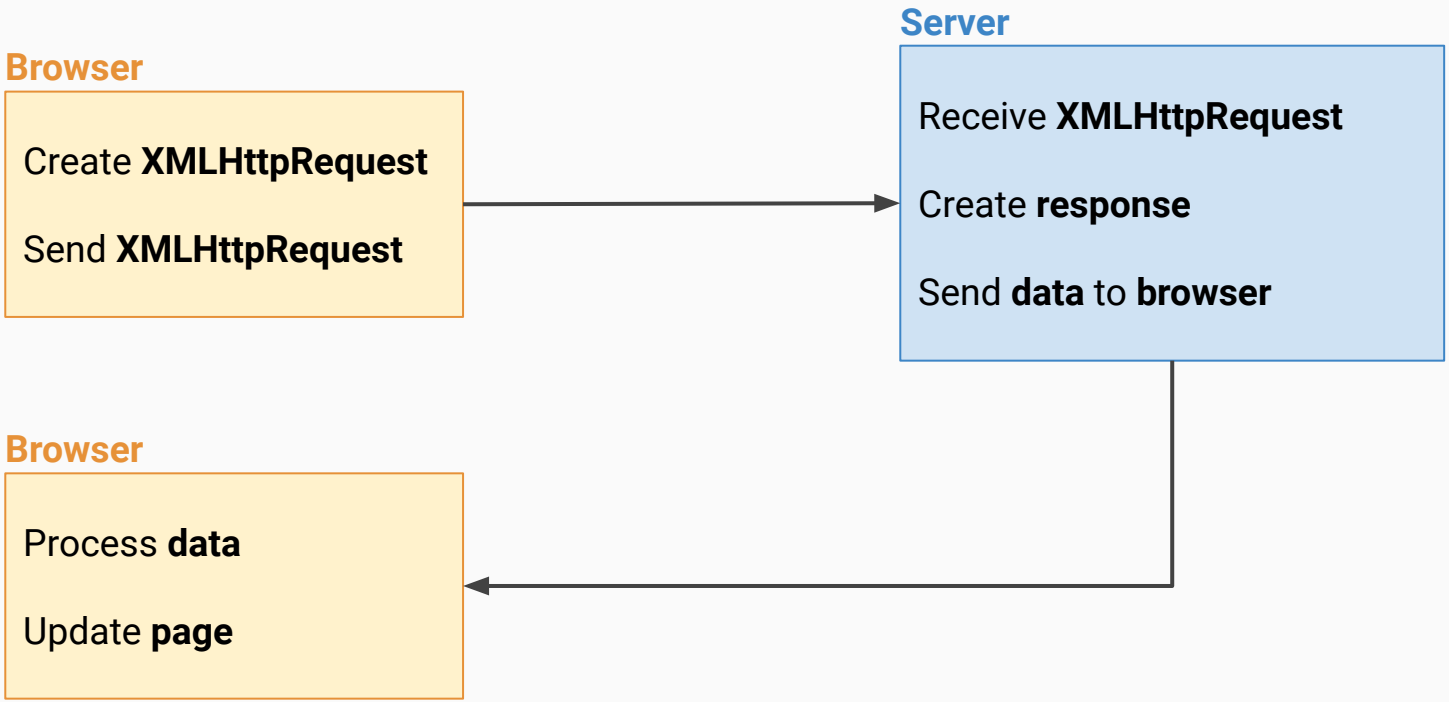
Asynchronous JavaScript + XML, while not a technology in itself, is a term coined in 2005 by Jesse James Garrett, that describes a "new" approach to using a number of existing technologies together, including HTML or XHTML, Cascading Style Sheets, JavaScript, The Document Object Model, XML, XSLT, and most importantly the XMLHttpRequest object.

When these technologies are combined in the Ajax model, web applications are able to make quick, incremental updates to the user interface without reloading the entire browser page.[1]

https://developer.mozilla.org/en-US/docs/Web/Guide/AJAX

AJAX is great, you can:

- **Read data from a web server -** after a web page has loaded
- **Update a web page without reloading the page**
- **Send data to a web server -** in the background


Browser
Create **XMLHttpRequest**
Send **XMLHttpRequest**
Server
Receive **XMLHttpRequest**
Create **response**
Send **data** to **browser**
Browser
Process **data**
Update **page**

XMLHttpRequest Object

All modern browsers support the XMLHttpRequest object.

The XMLHttpRequest object can be used to exchange data with a server behind the scenes. This means that it is possible to update parts of a web page, without reloading the whole page.

```
let xhttp = new XMLHttpRequest();
```

```
if (window.XMLHttpRequest) {
    // code for modern browsers
    xmlhttp = new XMLHttpRequest();
 } else {
    // code for old IE browsers
    xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
}
```

XMLHttpRequest Request

With the XMLHttpRequest object you can define a function to be executed when the request receives an answer.

The function is defined in the **onreadystatechange** property of the XMLHttpResponse object:

```
1.  let xhttp = new XMLHttpRequest();
2.  xhttp.onreadystatechange = function() {
3.    if (this.readyState == 4 && this.status == 200) {
4.       document.getElementById("demo").innerHTML =
    this.responseText;
5.    }
6.  };
7.  xhttp.open("GET", "ajax_info.txt", true);
8.  xhttp.send();
```

Access Across Domains

- For security reasons, modern browsers do not allow access across domains.
- This means that both the web page and the XML file it tries to load, must be located on the same server.

Name is Misleading

AJAX is a misleading name. AJAX applications might use XML to transport data, but it is equally common to transport data as plain text or JSON text.

JSON

JavaScript Object Notation

JSON

> *JavaScript Object Notation* (**JSON**) is a data-interchange format. Although not a strict subset, JSON closely resembles a subset of JavaScript syntax. Though many programming languages support JSON, JSON is especially useful for JavaScript-based apps, including websites and browser extensions..[1]

https://developer.mozilla.org/en-US/docs/Glossary/JSON

JSON

When exchanging data between a browser and a server, the data can only be **text**.

JSON is text, and we can convert any JavaScript object into JSON, and send JSON to the server.

We can also convert any JSON received from the server into JavaScript objects.

JSON Syntax

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

- JSON is a syntax for serializing:
  - objects
  - arrays
  - numbers
  - strings
  - booleans
  - null
- It is based upon JavaScript syntax but is distinct from it: some JavaScript is not JSON.

JSON Data Types

- a string
- a number
- an object (JSON object)
- an array
- a boolean
- null

JSON values cannot be one of the following data types:

- a function
- a date
- undefined

JSON Strings

Strings in JSON must be written in double quotes.

```
1.   { "name":"John" }
```

JSON Numbers

Numbers in JSON must be an integer or a floating point.

```
1.   { "age":30 }
```

JSON Booleans

Values in JSON can be true/false.

```
1.   { "sale":true }
```

JSON null

Values in JSON can be null.

```
1.   { "middlename":null }
```

JSON Objects

JSON objects are written inside curly braces.

Just like in JavaScript, objects can contain multiple name/value pairs:

```
{
    "employee": {
        "name":"John",
        "age":30,
        "city":"New York"
    }
}
```

JSON Arrays

JSON arrays are written inside square brackets.

Just like in JavaScript, an array can contain objects:

```
{
    "employees":[
        {"firstName":"John", "lastName":"Doe"},
        {"firstName":"Anna", "lastName":"Smith"},
        {"firstName":"Peter", "lastName":"Jones"}
    ]
}
```

JSON vs XML

- Both JSON and XML are "self describing"
- Both JSON and XML are hierarchical
- Both JSON and XML can be parsed and used by lots of programming languages
- Both JSON and XML can be fetched with an XMLHttpRequest

- JSON doesn't use end tag
- JSON is shorter
- JSON is quicker to read and write
- JSON can use arrays

JSON.parse()

A common use of JSON is to exchange data to/from a web server.

When receiving data from a web server, the data is always a string.

Parse the data with JSON.parse(), and the data becomes a JavaScript object.

```
1.   let obj = JSON.parse('{ "name":"John", "age":30,
     "city":"New York"}');
```

JSON.stringify()

A common use of JSON is to exchange data to/from a web server.

When sending data to a web server, the data has to be a string.

Convert a JavaScript object into a string with JSON.stringify().

```
1.   let obj = { name: "John", age: 30, city: "New York" };
2.   let myJSON = JSON.stringify(obj);
```

References

https://developer.mozilla.org/en-US/

https://www.w3schools.com/js

Questions?

Next: AJAX / JSON