AJAX BASIC

(Asynchronous JavaScript And Xml)

DANNY YEN/THE SHORTCUT

Asynchronous JavaScript And XML

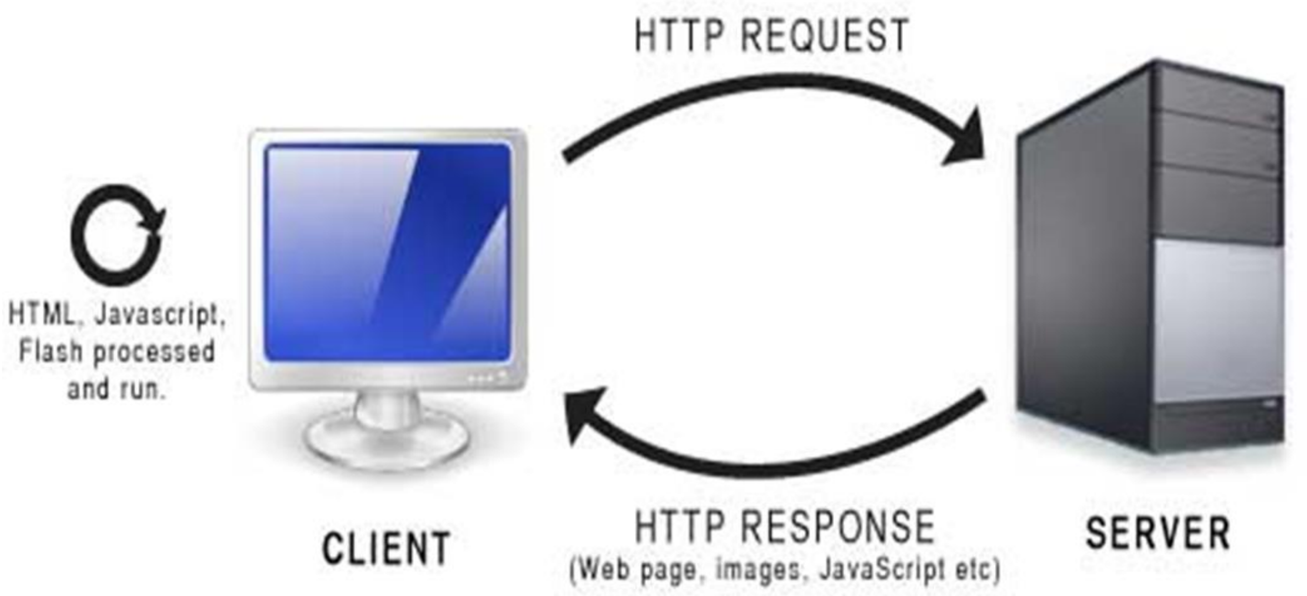
is not a programming language,

uses a browser to build an

XMLHttpRequest object (to request data from a web server) and

JavaScript and HTML DOM (to display or use the data).

https://www.w3schools.com/whatis/whatis_ajax.asp

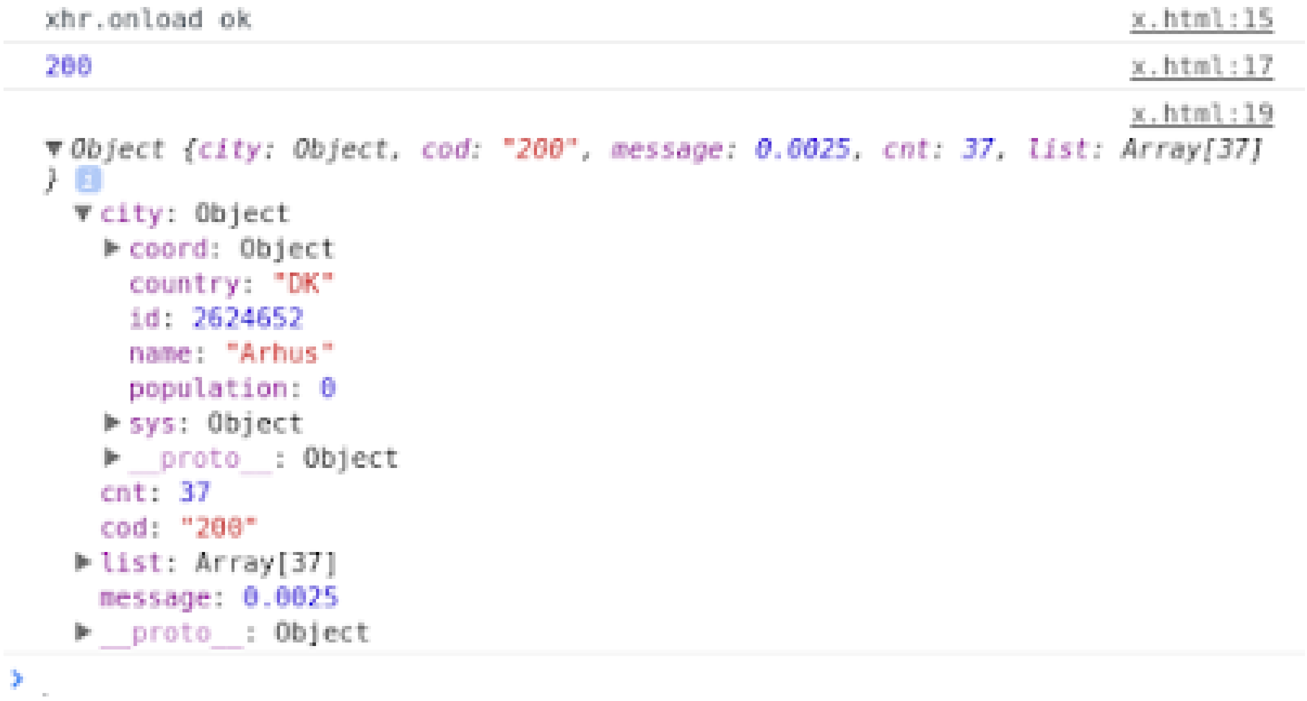

HTTP REQUEST
HTML, Javascript, Flash processed and run.
CLIENT
HTTP RESPONSE
(Web page, images, JavaScript etc)
SERVER

XMLHttpRequest (xhr)

Uses XMLHttpRequest (XHR) objects to interact with servers and retrieves data from a URL.

The object returns in JSON format and is used with JavaScript Runtime to display in the browser.

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



```
xhr.onload ok                                                          x.html:15
200                                                                    x.html:17
                                                                       x.html:19
▼Object {city: Object, cod: "200", message: 0.0025, cnt: 37, list: Array[37]
}
  ▼city: Object
    ▶coord: Object
     country: "DK"
     id: 2624652
     name: "Arhus"
     population: 0
    ▶sys: Object
    ▶__proto__: Object
   cnt: 37
   cod: "200"
  ▶list: Array[37]
   message: 0.0025
  ▶__proto__: Object
>
```

XMLHttpRequest.onreadystatechange

An EventHandler that is called whenever the readyState attribute changes.

XMLHttpRequest.readyState

Returns an unsigned short, the state of the request.

XMLHttpRequest.response

Returns an ArrayBuffer, Blob, Document, JavaScript object, or a DOMString, depending on the value of XMLHttpRequest.responseType, that contains the response entity body.

XMLHttpRequest.responseText

Returns a DOMString that contains the response to the request as text, or null if the request was unsuccessful or has not yet been sent.

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

- After the **open** method
  the **readyState** property of the XMLHttpRequest object should be assigned a value of 1 (OPENED).

- After the **send** method the HTTP
  the **readyState** property of the XMLHttpRequest object should be assigned a value of 2 (HEADERS_RECEIVED).

- Once the HTTP response content begins to load, the **readyState** property of the XMLHttpRequest object should be assigned a value of 3 (LOADING).

- Once the HTTP response content has finished loading,
  the **readyState** property of the XMLHttpRequest object should be assigned a value of 4 (DONE).

- https://en.wikipedia.org/wiki/XMLHttpRequest

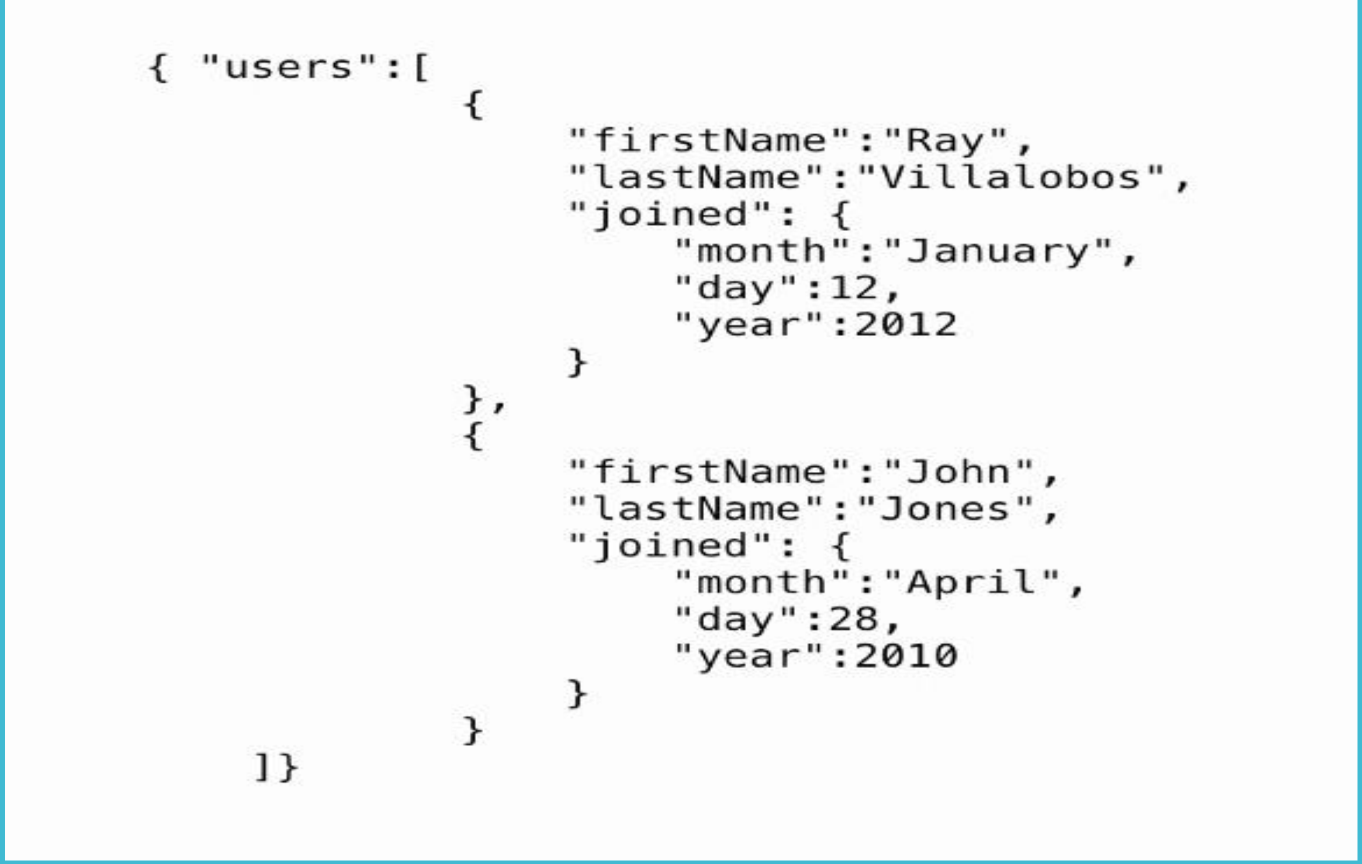
```
var request = new XMLHttpRequest();
request.onreadystatechange = function () {
  var DONE = this.DONE || 4;
  if (this.readyState === DONE){
    alert(this.readyState);
  }
};

request.open('GET', 'somepage.xml', true);


request.setRequestHeader('X-Requested-With', 'XMLHttpRequest');


request.send(null);
```

JSON Format



```
{ "users":[
            {
                "firstName":"Ray",
                "lastName":"Villalobos",
                "joined": {
                    "month":"January",
                    "day":12,
                    "year":2012
                }
            },
            {
                "firstName":"John",
                "lastName":"Jones",
                "joined": {
                    "month":"April",
                    "day":28,
                    "year":2010
                }
            }
    ]}
```

Promises: Fetch JS

The Fetch API provides an interface for fetching resources (including across the network). It is similar XMLHttpRequest, but the new API provides a more powerful and flexible feature set.

The fetch() method takes one mandatory argument, the path to the resource you want to fetch. It returns a Promise that resolves to the Response to that request, whether it is successful or not. You can also optionally pass in an init options object as the second argument (see Request).

https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API

```
fetch(url)
  .then((resp) => resp.json())
  .then(function(data) {
    let authors = data.results;
    return authors.map(function(author) {
      let li = createNode('li'),
          img = createNode('img'),
          span = createNode('span');
      img.src = author.picture.medium;
      span.innerHTML = `${author.name.first} ${author.name.last}`;
      append(li, img);
      append(li, span);
      append(ul, li);
    })
  })
```