

AJAX

by Anastasiia Derkach

soft**serve**

WHAT IS AJAX?

Asynchronous JavaScript And XML

! AJAX is not a programming language. It is a methodology on using several web technologies together, in an effort to close the gap between the usability and interactivity of a desktop application and the ever demanding web application.

- Asynchronous Javascript and XML is a client side techniques that combines a set of known technologies in order to create faster and more user friendly web pages.
- AJAX provides an ability to communicate with the server asynchronously.

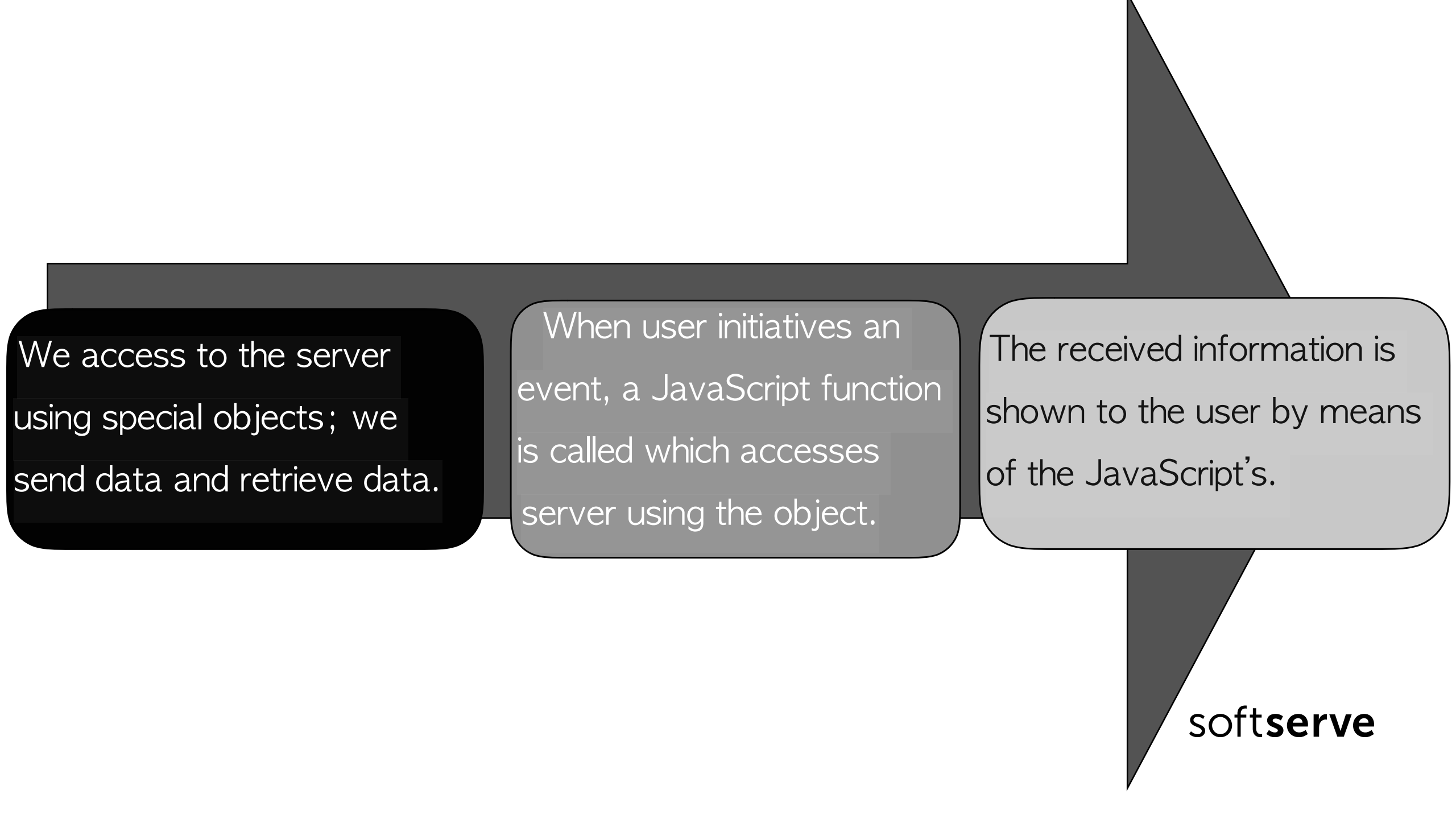
AJAX HISTORY

The term AJAX is coined on February 18, 2005, by Jesse James Garret in a short essay published a few days after Google released its Maps application.

In the year 2006, the W3C announces the release of the first draft which made AJAX an official web standard.

SIMPLE PROCESSING

AJAX is based on JavaScript, and the main functionality is to access the web server inside the JavaScript code.

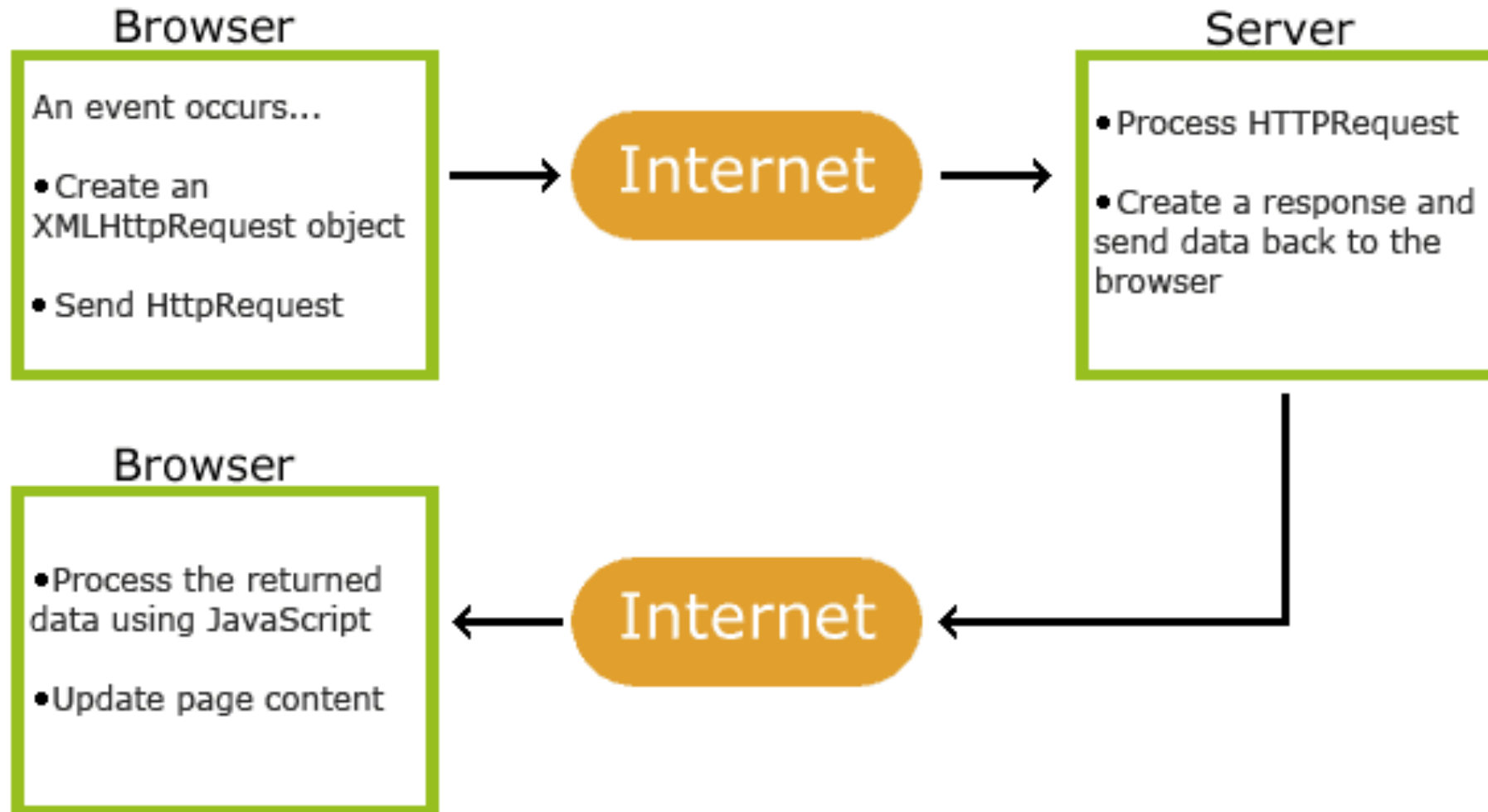


We access to the server using special objects; we send data and retrieve data.

When user initiatives an event, a JavaScript function is called which accesses server using the object.

The received information is shown to the user by means of the JavaScript's.

softserve



COMPONENTS OF AJAX

- Data interchange and manipulation using XML or JSON
- Asynchronous data retrieval using XMLHttpRequest
- Dynamic display and interaction using DOM

XML

The **EX**tensible **M**arkup **L**anguage

XML was designed to describe data and to focus on what data is (unlike HTML which was designed to display data and to focus on how data looks)

It is general-purpose markup language for creating special-purpose markup languages carry data.

XMLHttpRequest

The XMLHttpRequest object allows client-side JavaScript to make HTTP request to the server without reloading pages in the browser.

This JavaScript object was originally introduced in Internet Explorer 5 by Microsoft and it is the enabling technology that allows asynchronous requests

XMLHttpRequest

By performing screen updates on the client, you have a great amount of flexibility when it comes to creating your Web site:

1. Eliminate page refreshes every time there is user input;
2. Edit data directly in place, without requiring the user to navigate to a new page to edit the data;
3. Increase site performance by reducing the amount of data downloaded from the server

AJAX METHODS

open()

open(,method', ,url', boolean)

1. To send a request to server

2. Method - GET or POST

```
xhttp.open('POST', '/api/todo', true);
```

3. Url - address of the target file

4. Boolean - to donate whether the request is synchronous or asynchronous. If true, asynchronous.

softserve

setRequestHeader()

setRequestHeader(header, value)

1. The name of the header whose value is to be set
2. **Value** - The value to set as the body of the header.

```
xhttp.setRequestHeader('Content-Type', 'application/json');
```

overrideMimeType()

overrideMimeType(mimeType)

1. Specifies a MIME type other than the one provided by the server to be used instead when interpreting the data being transferred in a request.
2. **MimeType** - DOMString specifying the MIME type to use instead of the one specified by the server.

```
xhr.overrideMimeType("text/plain");
```

softserve

send()

send(content)

1. To send data to server for processing

2. **Content** - may be string or DOM type of document

```
xhttp.send(JSON.stringify({ item: data }));
```

softserve

getResponseHeader()

getResponseHeader(headerName)

1. Returns the string containing the text of a particular header's value;
2. headerName - A `ByteString` indicating the name of the header you want to return the text value of.

getAllResponseHeaders()

getAllResponseHeaders()

1. Returns all the response headers

abort()

abort()

1.Method aborts the request if it has already been sent

```
xhr.abort();
```

softserve

AJAX PROPERTIES

XMLHttpRequest.onreadystatechange

Open event handler for an event that fires at every state change.

```
XMLHttpRequest.onreadystatechange = callback
```

XMLHttpRequest.readyState

Open used to identify the state of the request. Possible values
0-4

Value	State	Description
0	UNSENT	Client has been created. <code>open ()</code> not called yet.
1	OPENED	<code>open ()</code> has been called.
2	HEADERS_RECEIVED	<code>send ()</code> has been called, and headers and status are available.
3	LOADING	Downloading; <code>responseText</code> holds partial data.
4	DONE	The operation is complete.

XMLHttpRequest.status

HTTP Status Codes



softserve

XMLHttpRequest.statusText

Text return by server. By default - ,OK‘

XMLHttpRequest.responseText

The string data returned by the server process

XMLHttpRequest.responseText

It is an enumerated string value specifying the type of data contained in the response.

XMLHttpRequest.responseURL

The property returns the serialized URL of the response or the empty string if the URL is null

```
console.log(xhr.responseURL); // http://example.com/test
```

EXAMPLE

```
function sendAjax(data) {  
  const xhttp = new XMLHttpRequest();  
  xhttp.onreadystatechange = function() {  
    if(xhttp.readyState == 4 && xhttp.status == 200) {  
      handleResponse(xhttp.responseText);  
    } else {  
      console.log('Something wrong')  
    }  
  }  
  xhttp.open('POST', '/api/todo', true);  
  xhttp.setRequestHeader('Content-Type', 'application/json');  
  xhttp.send(JSON.stringify({item: data}))  
}
```

softserve

FETCH

soft**serve**

The **Fetch API** is a modern alternative to **XMLHttpRequest**.

```
fetch(  
  'http://domain/service',  
  { method: 'GET' }  
)  
  .then( response => response.json() )  
  .then( json => console.log(json) )  
  .catch( error => console.error('error:', error) );
```

softserve

RESOURCES USED

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