

# CLIENT SIDE WEB DEVELOPMENT LESSON 7: AJAX

# Índice

2

- Introduction
- Load data on demand
- Passing data to the server

# 1. Introduction

3

- Ajax → Asynchronous JavaScript and XML
- Ajax solution includes the following technologies:
  - **JavaScript:** This is used to capture interactions with the user or other browser-related events and to interpret the data from the server and present it on the page.
  - **XMLHttpRequest:** This allows requests to be made to the server without interrupting other browser tasks.
  - **Textual data:** The server provides data in a format such as XML, HTML, or JSON.

# 1. Introduction

4

- An asynchronous request can be fired at any time (before or after a page has loaded) and the response to an asynchronous request often includes HTML that can be dynamically inserted into a page.
- **Facebook** uses a lot of asynchronous requests so that the page almost never needs to refresh for users to see new content:
  - *When a user scrolls down in a users account, new stories get inserted into the page which never needs to refresh to show new content.*

# 1. Introduction

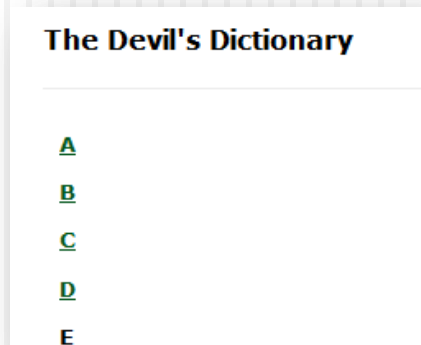
5

- Ajax will let us:
  1. Load data from the server without a page refresh
  2. Send data from JavaScript in the browser back to the server
  3. Interpret data in a variety of formats, including HTML, XML, and JSON
  4. Provide feedback to the user about the status of an Ajax request
- Ajax-jQuery documentation [here](#).

## 2. Load data on demand

6

- Exercise 1: You are going to build a page that displays entries from a dictionary, grouped by the starting letter of the dictionary entry.
  1. Download Exercise1.rar
  2. If you open *index.html*, you should find something like this:



We are going to use different links in order to load dynamically information.

## 2. Load data on demand

7

### 3. Appending HTML:

- Ajax applications are often no more than a request for a chunk of HTML. This technique, Asynchronous HTTP and HTML (AJAX), is almost trivial to implement with jQuery.
- If you check inside the downloaded file, you have a *a.html*.
- Note that *a.html* is not a true HTML document; it contains no `<html>`, `<head>`, or `<body>`, all of which are normally required.
- We usually call such a file a *partial* or *fragment*; its only purpose is to be inserted into another HTML document.
- In order to do so, add this code inside *a71.js* file:

```
1 $(document).ready(function() {  
2     $('#letter-a a').click(function(event) {  
3         event.preventDefault();  
4         $('#dictionary').load('a.html');  
5     });  
6 }
```



## 2. Load data on demand

8

### The Devil's Dictionary

[A](#)

[B](#)

[C](#)

[D](#)

[E](#)

[Eavesdrop](#)

[Edible](#)

[Education](#)

[Eloquence](#)

[Elysium](#)

[Emancipation](#)

[Emotion](#)

[Envelope](#)

[Envy](#)

[Epitaph](#)

[Evangelist](#)

[F](#)

#### **ABDICATION** *n.*

An act whereby a sovereign attests his sense of the high temperature of the throne.

Poor Isabella's Dead, whose abdication  
Set all tongues wagging in the Spanish nation.  
For that performance 'twere unfair to scold her:  
She wisely left a throne too hot to hold her.  
To History she'll be no royal riddle —  
Merely a plain parched pea that jumped the griddle.

G.J.

#### **ABSOLUTE** *adj.*

Independent, irresponsible. An absolute monarchy is one in which the sovereign does absolute monarchies are left, most of them having been replaced by limited monarchies; curtailed, and by republics, which are governed by chance.

#### **ACKNOWLEDGE** *v.t.*

To confess. Acknowledgement of one another's faults is the highest duty imposed by o

- Note that the HTML is now styled, whereas before it was plain (a.html) → This is due to the CSS rules in the main document; as soon as the new HTML snippet is inserted, the rules apply to its elements as well.



## 2. Load data on demand

9

### 4. Retrieving JSON:

- Pulling in fully formed HTML on demand as before, is very convenient, but it means having to transfer a lot of information (HTML structure) along with the actual content.
- There are times when we would rather transfer as little data as possible and process it after it arrives  
→ JSON (**JavaScript Object Notation**)
  - JavaScript objects are just sets of key-value pairs, and can be defined succinctly using curly braces ({}).
  - JavaScript arrays, on the other hand, are defined on the fly with square brackets ([]).

```
{  
  "key": "value",  
  "key 2": [  
    "array",  
    "of",  
    "items"  
  ]  
}
```

## 2. Load data on demand

10

- Inside downloaded files you have *b.json* file. There you have some more dictionary entries:

```
1  [
2    {
3      "term": "BACCHUS",
4      "part": "n.",
5      "definition": "A convenient deity invented by the ancients as an excuse for getting drunk.",
6      "quote": [
7        "Is public worship, then, a sin,",
8        "That for devotions paid to Bacchus",
9        "The lictors dare to run us in,",
10       "And resolutely thump and whack us?"
11     ],
12     "author": "Jorace"
13   },
14   {
```

- To retrieve this data, we'll use the **`$.getJSON()`** method, which fetches the file and processes it. When the data arrives from the server, it is simply a text string in JSON

## 2. Load data on demand

11

- Continuing with a71.js file:

```
9  $('#letter-b a').click(function(event) {  
10      event.preventDefault();  
11      $.getJSON('b.json');  
12  });
```

- Visually it will have no effect. However, if we check firebug Network tab, you will find the corresponding call and JSON response:

The screenshot shows the Firebug Network tab with two requests: 'GET a/1.js' and 'GET b.json'. The 'GET b.json' request is selected, showing a 200 OK status and a JSON response. The response is displayed in the 'JSON' tab, showing an array of 6 objects, each containing 'term', 'part', and 'definition' fields.

Encabezados	Respuesta	JSON	Caché
Ordenar alfabéticamente			
0		Object { term="BACCHUS", part="n.", definition="A convenient deity inven...cuse for getting drunk.", más... }	
1		Object { term="BACKBITE", part="v.t.", definition="To speak of a man as you...when he can't find you." }	
2		Object { term="BEARD", part="n.", definition="The hair that is commonl...om of shaving the head." }	
3		Object { term="BEGGAR", part="n.", definition="One who has relied on th...istance of his friends." }	
4		Object { term="BELLADONNA", part="n.", definition="In Italian a beautiful l...ity of the two tongues." }	
5		Object { term="BIGAMY", part="n.", definition="A mistake in taste for w...ishment called trigamy." }	
6		Object { term="BORE", part="n.", definition="A person who talks when you wish him to listen." }	

## 2. Load data on demand

12

- What we need now, is to process the JSON response:
  - The `$.getJSON()` function takes a second argument, which is a function to be called when the load is complete.
  - That callback function also takes an argument, which is filled with the resulting data. So, we can write:

```
$('#letter-b a').click(function(event) {  
    event.preventDefault();  
    $.getJSON('b.json', function(data) {  
        // Unfinished code  
    });  
});
```

## 2. Load data on demand

13

- Let's modify previous code in order to fill the corresponding *dictionary* div fill with some HTML:

```
$('#letter-b a').click(function(event) {  
    event.preventDefault();  
    $.getJSON('b.json', function(data) {  
        var html = '';  
        $.each(data, function(entryIndex, entry) {  
            html += '<div class="entry">';  
            html += '<h3 class="term">' + entry.term + '</h3>';  
            html += '<div class="part">' + entry.part + '</div>';  
            html += '<div class="definition">';  
            html += entry.definition;  
            html += '</div>';  
            html += '</div>';  
        });  
        $('#dictionary').html(html);  
    });  
});
```

**BACCHUS** *n.*

A convenient deity invented by the ancients as an excuse for getting drunk.

**BACKBITE** *v.t.*

To speak of a man as you find him when he can't find you.

**BEARD** *n.*

The hair that is commonly cut off by those who justly execrate the absurd Chinese custom of shaving the head.

## 2. Load data on demand

14

- Another change has to be made in order to process quote entries if they exist in the response JSON data:

```
$('#letter-b a').click(function(event) {
    event.preventDefault();
    $.getJSON('b.json', function(data) {
        var html = '';
        $.each(data, function(entryIndex, entry) {
            html += '<div class="entry">';
            html += '<h3 class="term">' + entry.term + '</h3>';
            html += '<div class="part">' + entry.part + '</div>';
            html += '<div class="definition">';
            html += entry.definition;
            if (entry.quote) {
                html += '<div class="quote">';
                $.each(entry.quote, function(lineIndex, line) {
                    html += '<div class="quote-line">' + line + '</div>';
                });
                if (entry.author) {
                    html += '<div class="quote-author">' + entry.author + '</div>';
                }
                html += '</div>';
            }
            html += '</div>';
        });
        $('#dictionary').html(html);
    });
});
```

## 2. Load data on demand

15

### **BACCHUS** *n.*

A convenient deity invented by the ancients as an excuse for getting drunk.

Is public worship, then, a sin,  
That for devotions paid to Bacchus  
The lictors dare to run us in,  
And resolutely thump and whack us?

**Jorace**

### **BACKBITE** *v.t.*

To speak of a man as you find him when he can't find you.

## 2. Load data on demand

16

### 5. Executing a script:

- Occasionally, we don't want to retrieve all the JavaScript when the page is first loaded.
- We might not know what scripts will be necessary until some user interaction occurs (“lazy loading”).
- Pulling in a script is about as simple as loading an HTML fragment. In this case, we use the `$.getScript()` function:

```
$('#letter-c a').click(function(event) {  
    event.preventDefault();  
    $.getScript('c.js');  
});
```



## 2. Load data on demand

17

### 6. Loading an XML document:

- Check that you have *d.xml* file in order to load it.
- We'll start our function in a familiar manner, using , it's `$.get()` function:

```
$('#letter-d a').click(function(event) {  
    event.preventDefault();  
    $.get('d.xml', function(data) {  
        // Unfinished code  
    });  
});
```

## 2. Load data on demand

18

- In order to transverse the XML, we can use the usual *.find()* (and other methods) just as we would on HTML:

```
$(document).ready(function() {
    $('#letter-d a').click(function(event) {
        event.preventDefault();
        $.get('d.xml', function(data) {
            $('#dictionary').empty();
            ★ $(data).find('entry').each(function() {
                var $entry = $(this);
                var html = '<div class="entry">';
                html += '<h3 class="term">' + $entry.attr('term'); ★
                html += '</h3>';
                html += '<div class="part">' + $entry.attr('part');
                html += '</div>';
                html += '<div class="definition">';
                html += $entry.find('definition').text();
                var $quote = $entry.find('quote');
                if ($quote.length) {
                    html += '<div class="quote">';
                    $quote.find('line').each(function() {
                        html += '<div class="quote-line">';
                        html += $(this).text() + '</div>';
                    });
                    if ($quote.attr('author')) {
                        html += '<div class="quote-author">';
                        html += $quote.attr('author') + '</div>';
                    }
                    html += '</div>';
                }
                html += '</div>';
                html += '</div>';
                $('#dictionary').append($(html));
            });
        });
    });
});
```

## 2. Load data on demand

19



A.7.1. We have seen different data format loading. Can you imagine when to use each of them?

# 3. Passing data to the server

20

## □ Let's continue with previous exercise:

### 1. Performing a GET request:

- To illustrate the communication between client (JavaScript) and server (PHP), we'll write a script that only sends one dictionary entry to the browser on each request. The entry chosen will depend on a parameter sent from the browser.
- In this case, you will have a server script `e.php`:
  - Since the data is a part of the script here, the code to retrieve it is quite straightforward. In a production version of this example, the data would probably be stored in a database and loaded on demand.

# 3. Passing data to the server

21

- If you just try the code, clicking in any 'e' link, the server will return a new HTML file and will be directly presented:

## **EAVESDROP**

v.i.

Secretly to overhear a catalogue of the crimes and vices of another or yourself.

A lady with one of her ears applied

To an open keyhole heard, inside,

Two female gossips in converse free —

The subject engaging them was she.

"I think," said one, "and my husband thinks

That she's a prying, inquisitive minx!"

As soon as no more of it she could hear

The lady, indignant, removed her ear.

"I will not stay," she said, with a pout,

"To hear my character lied about!"


Gopete Sherany

# 3. Passing data to the server

22

- Now we need to get our JavaScript code to call the PHP script with the right parameters:
- We could do this with the normal `.load()` mechanism, appending the query string right to the URL and fetching data with addresses such as `e.php?term=eavesdrop` directly.
- Instead, we can make jQuery to construct the query string based on an object we provide to the `$.get()` function:

```
$('#letter-e a').click(function(event) {  
    event.preventDefault();  
    var requestData = {term: $(this).text()};  
    $.get('e.php', requestData, function(data) {  
        // Unfinished code  
    });  
});
```



- The second parameter allows to supply an object containing keys and values that become part of the query string, in this case, the key is always `term`, but the value is taken from the text of each link.

# 3. Passing data to the server

23



A.7.2. Fill the missing code in previous section, in order to fill the corresponding *dictionary* div.

## **EAVESDROP** *v.i.*

Secretly to overhear a catalogue of the crimes and vices of another or yourself.

A lady with one of her ears applied  
To an open keyhole heard, inside,  
Two female gossips in converse free —  
The subject engaging them was she.  
"I think," said one, "and my husband thinks  
That she's a prying, inquisitive minx!"  
As soon as no more of it she could hear  
The lady, indignant, removed her ear.  
"I will not stay," she said, with a pout,  
"To hear my character lied about!"

**Gopete Sherany**

# 3. Passing data to the server

24

## 2. Performing a POST request:

```
$('#letter-e a').click(function(event) {  
    event.preventDefault();  
    var requestData = {term: $(this).text()};  
    $.post('e.php', requestData, function(data) {  
        $('#dictionary').html(data);  
    });  
});
```

- We can further simplify the code by using the `.load()` method, which uses POST by default:

```
$('#letter-e a').click(function(event) {  
    event.preventDefault();  
    var requestData = {term: $(this).text()};  
    $('#dictionary').load('e.php', requestData);  
});
```

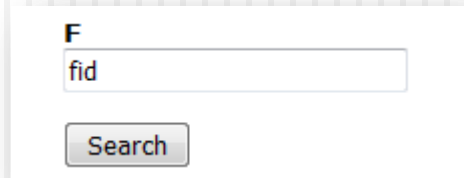


# 3. Passing data to the server

25

## 3. Serializing a form:

- In this case we will use the form inside *index.html*:




A screenshot of a web form. It features a text input field with the label 'F' above it. The input field contains the text 'fid'. Below the input field is a button labeled 'Search'.

- You will find an *f.php* script, similar to previous *e.php*. In this case, we'll return a set of entries from the server by using PHP to search for the supplied term as a substring of a dictionary term.


# 3. Passing data to the server

26

- Inside `a71.js`, add this code in order to make the corresponding call:

```
 $('#letter-f form').submit(function(event) {  
    event.preventDefault();  
    var requestData = {'term': $('#input[name="term"]').val()};  
    $.get('f.php', requestData,  
        function(data) {  
            $('#dictionary').html(data);  
        });  
});
```

- This approach does not scale well as the form becomes more complex... → jQuery offers a shortcut for this: **`$.serialize()`**:

```
$('#letter-f form').submit(function(event) {  
    event.preventDefault();  
    var formValues = $(this).serialize();   
    $.get('f.php', formValues, function(data) {  
        $('#dictionary').html(data);  
    });  
});
```

# 3. Passing data to the server

27

## The Devil's Dictionary

### A

### B

### C

### D

### E

[Eavesdrop](#)

[Edible](#)

[Education](#)

[Eloquence](#)

[Elysium](#)

[Emancipation](#)

[Emotion](#)

[Envelope](#)

[Envy](#)

[Epitaph](#)

[Evangelist](#)

### F

fid

### **FIDDLE** *n.*

An instrument to tickle human ears by friction of a horse's tail on the entrails of a cat.  
To Rome said Nero: "If to smoke you turn  
I shall not cease to fiddle while you burn."  
To Nero Rome replied: "Pray do your worst,  
'Tis my excuse that you were fiddling first."  
Orm Pludge

### **FIDELITY** *n.*

A virtue peculiar to those who are about to be betrayed.

# 3. Passing data to the server

28



A.7.3. Add following functionalities to previous exercise:

- When the page loads, pull the body content of *exercises-content.html* into the content area of the page.
- Inset a new link 'Tetris'. When it is clicked, the application should send a JSONP request to GitHub and retrieve a list of repositories developing the popular Tetris game. Insert the id, name and URL of each repository into the content area of the page. The URL to retrieve the jQuery project's repositories is <https://api.github.com/search/repositories>

```
12014401  
Tetris  
https://api.github.com/repos/PSNB92/Tetris
```

```
3477759  
tetris  
https://api.github.com/repos/tdd-elevator-training/tetris
```

```
797142  
tetris  
https://api.github.com/repos/troglobit/tetris
```

```
12534459  
tetris
```