# CLIENT SIDE WEB DEVELOPMENT LESSON 7: AJAX

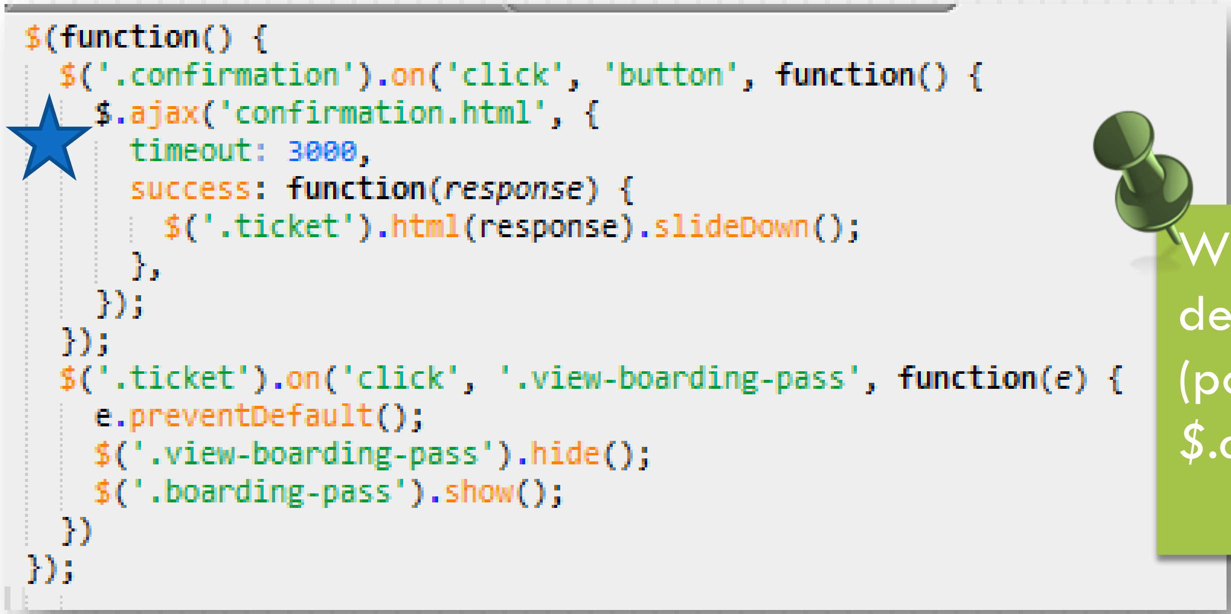
CFGS - DAW 2

# Index

# 4. Requests and APIs

- Previous *$.get()* and *$.post()*, both are easier ways of using [*$.ajax()*]() jQuery methods.
- <u>Exercise 2</u>: *$.ajax call Success and Error Handling*

1. Fill this code inside a72.js



```
$(function() {
  $('.confirmation').on('click', 'button', function() {
    $.ajax('confirmation.html', {
      timeout: 3000,
      success: function(response) {
        $('.ticket').html(response).slideDown();
      },
    });
  });
  $('.ticket').on('click', '.view-boarding-pass', function(e) {
    e.preventDefault();
    $('.view-boarding-pass').hide();
    $('.boarding-pass').show();
  })
});
```

Which is the default method (post or get) for *$.ajax()* method?

# 4. Requests and APIs

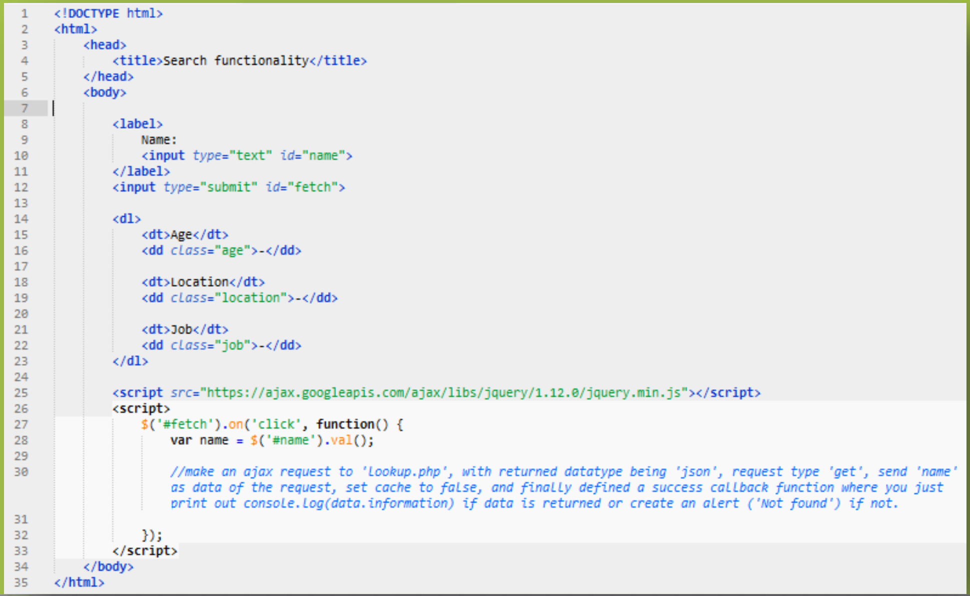
A.7.4. How would you get control of a possible error doing the ajax call? Check the API documentation in order to modify the previous code and launch an alert with the Error message.

# 4. Requests and APIs

- <u>Exercise 3</u>: *Search functionality for a site:*

1. Our goal will be to send the name, make a lookup by the name, output the result in a JSON format and output then in the corresponding list.

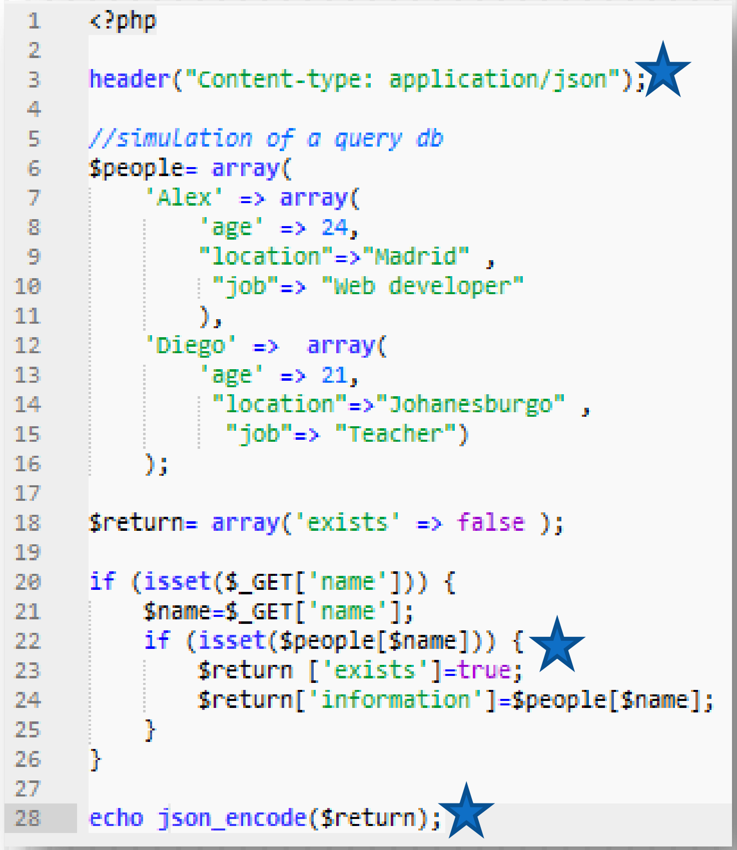
A.7.5. Fill the code inside script of Exercise3.html following the given requirements in order to make the ajax call:



```
<!DOCTYPE html>
<html>
    <head>
        <title>Search functionality</title>
    </head>
    <body>

        <label>
            Name:
            <input type="text" id="name">
        </label>
        <input type="submit" id="fetch">

        <dl>
            <dt>Age</dt>
            <dd class="age">-</dd>

            <dt>Location</dt>
            <dd class="location">-</dd>

            <dt>Job</dt>
            <dd class="job">-</dd>
        </dl>

        <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.0/jquery.min.js"></script>
        <script>
            $('#fetch').on('click', function() {
                var name = $('#name').val();

                //make an ajax request to 'lookup.php', with returned datatype being 'json', request type 'get', send 'name'
                as data of the request, set cache to false, and finally defined a success callback function where you just
                print out console.log(data.information) if data is returned or create an alert ('Not found') if not.

            });
        </script>
    </body>
</html>
```

# 4. Requests and APIs

2. Now, we are going to make the lookup functionality simulating DB query ('hardcoded information array')

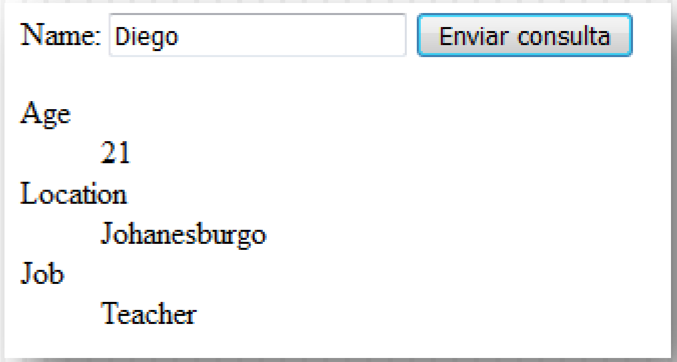


```php
<?php

header("Content-type: application/json");

//simulation of a query db
$people= array(
    'Alex' => array(
        'age' => 24,
        "location"=>"Madrid" ,
         "job"=> "Web developer"
        ),
    'Diego' =>  array(
        'age' => 21,
         "location"=>"Johanesburgo" ,
          "job"=> "Teacher")
    );

$return= array('exists' => false );

if (isset($_GET['name'])) {
    $name=$_GET['name'];
    if (isset($people[$name])) {
        $return ['exists']=true;
        $return['information']=$people[$name];
    }
}

echo json_encode($return);
```

## 4. Requests and APIs

3. Before continuing, test the application. You should get some information in JSON format as a result of the lookup.

A.7.6. Now that you have obtained the response data in JSON format… change the success callback in order to print out the corresponding information inside the definition list.



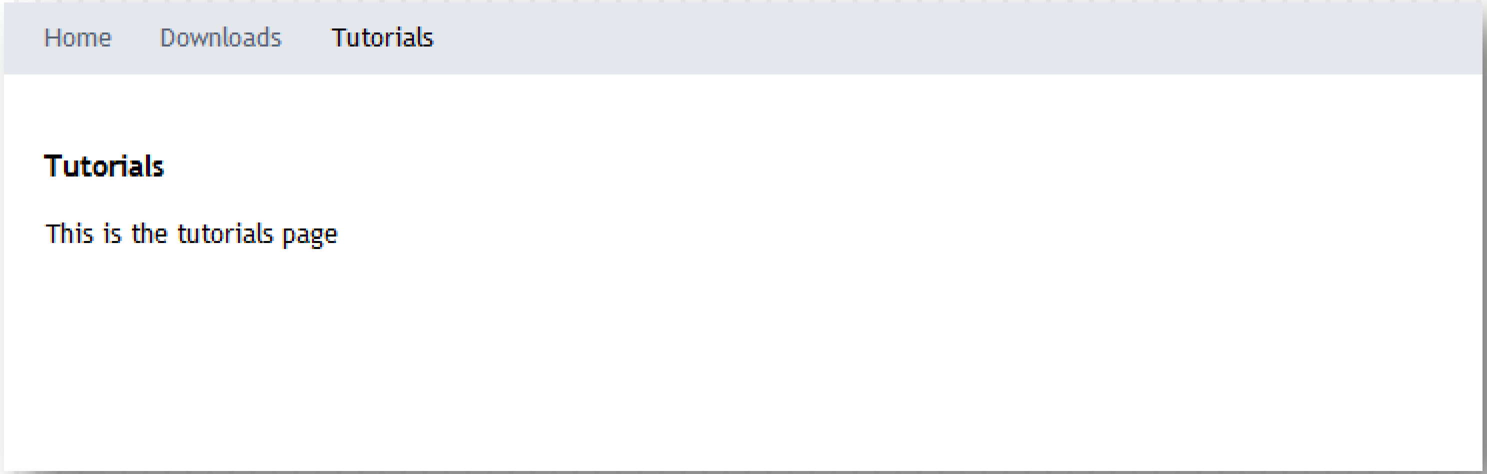
Name: Diego [Enviar consulta]

Age
    21
Location
    Johanesburgo
Job
    Teacher

# 4. Requests and APIs

- Exercise 4: *Loading content:*

1. Our goal will be load the main content in a Layout using ajax.

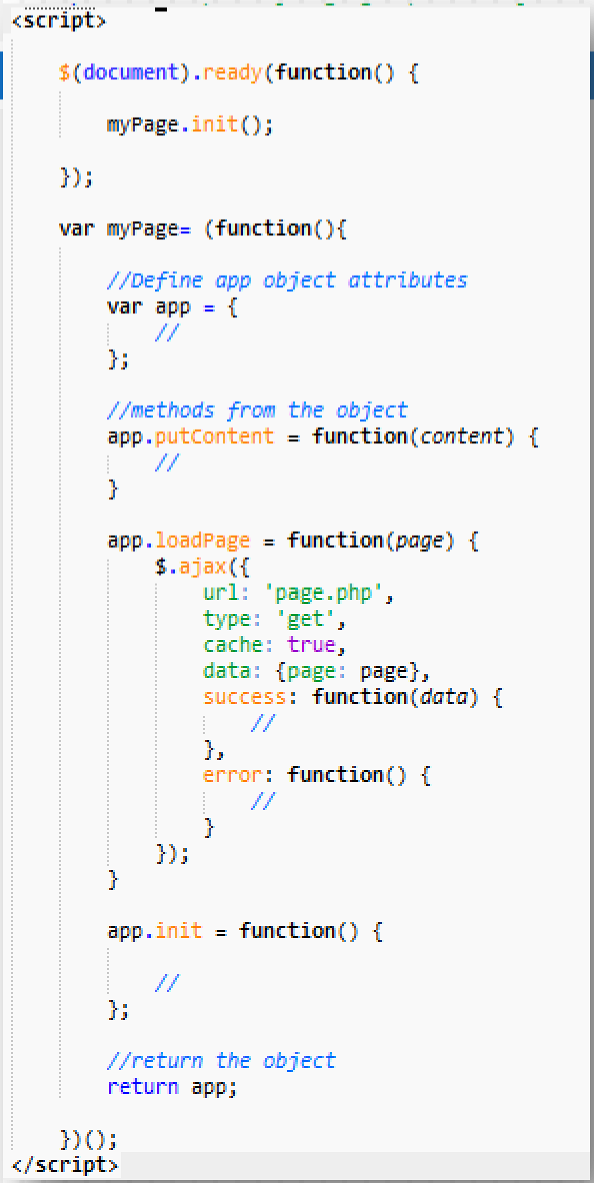


Tutorials

This is the tutorials page

# 4. Requests and APIs

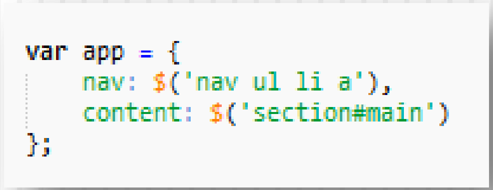
2. In this exercise, you will see how js or jQuery can also be defined in an Object Oriented way.
3. Inside <script> tags:
   - When “the document is ready” you will call to *init* function from the *myPage* object.
   - Inside the definition of *myPage* object you will define *app* object:



```
<script>

    $(document).ready(function() {

        myPage.init();

    });

    var myPage= (function(){

        //Define app object attributes
        var app = {
            //
        };

        //methods from the object
        app.putContent = function(content) {
            //
        }

        app.loadPage = function(page) {
            $.ajax({
                url: 'page.php',
                type: 'get',
                cache: true,
                data: {page: page},
                success: function(data) {
                    //
                },
                error: function() {
                    //
                }
            });
        }

        app.init = function() {

            //
        };

        //return the object
        return app;

    })();
</script>
```

# 4. Requests and APIs

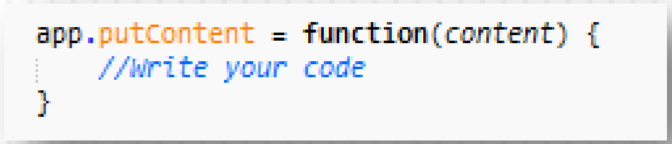
- Our app object will have defined two attributes:
  - One for the links from the menu
  - Second the main section where the content is going to be loaded
  - In an OO way, this is defined like this:



```
var app = {
    nav: $('nav ul li a'),
    content: $('section#main')
};
```

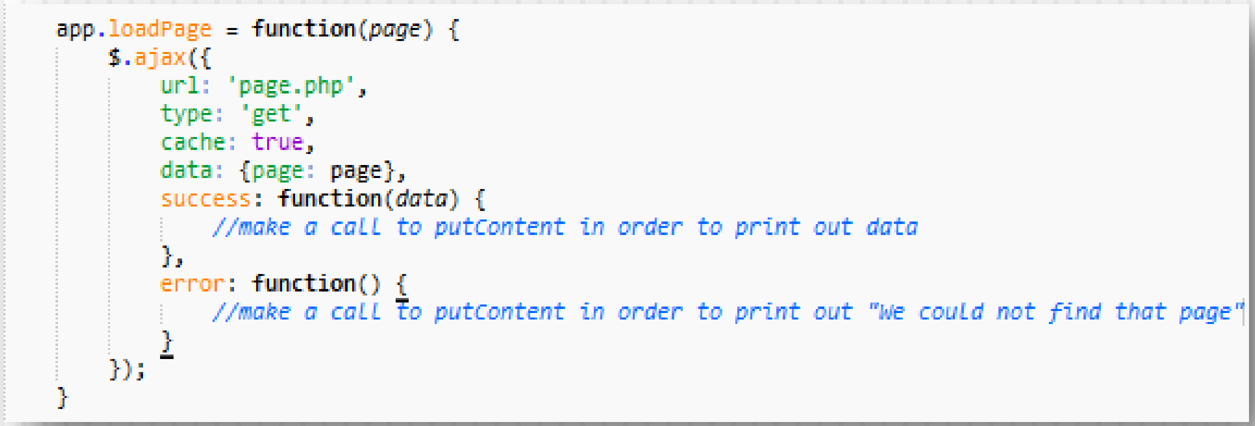
# 4. Requests and APIs

- *app* will have defined 3 methods:
  - *putContent*: Its goal will be to put inside content property the content past as parameter to the method:



```
app.putContent = function(content) {
    //Write your code
}
```

  - *loadPage*: This will make an ajax call to *page.php,* with *get* method, an passing the value of *page* parameter. You will have to fill the missing code:



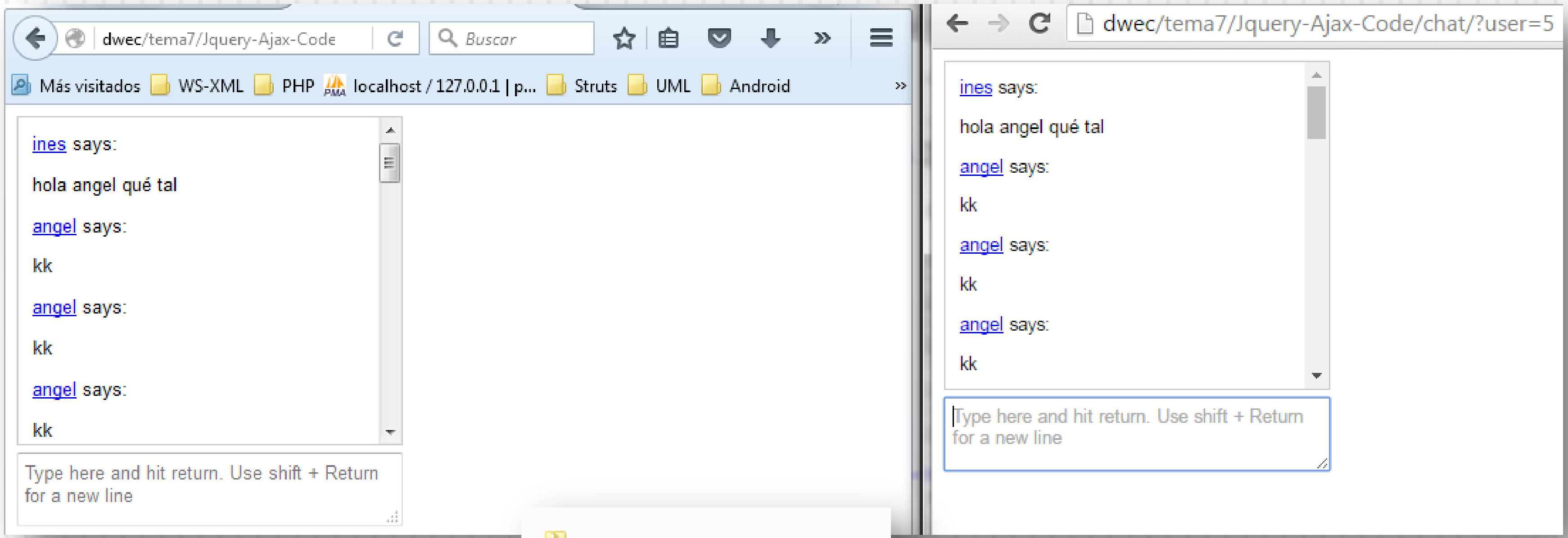
```
app.loadPage = function(page) {
    $.ajax({
        url: 'page.php',
        type: 'get',
        cache: true,
        data: {page: page},
        success: function(data) {
            //make a call to putContent in order to print out data
        },
        error: function() {
            //make a call to putContent in order to print out "We could not find that page"
        }
    });
}
```

# 4. Requests and APIs

- *init*: This function will be called with the page loading.
  1. Make a call to *app.loadPage* in order to load ‘home’ at the beginning.
  2. Write an event handler for the ‘click’ event of the nav property of the app object. Inside the function executed with the event:
     1. Save in a page variable the value of ‘page’ data
     2. Make a call to *app.loadPage* method passing the created page variable.

# 4. Requests and APIs

- Exercise 5: *Now you'll learn how to use AJAX to easily implement an online chat:*


dwec/tema7/Jquery-Ajax-Code
Buscar
Más visitados
WS-XML
PHP
localhost / 127.0.0.1 | p...
Struts
UML
Android
dwec/tema7/Jquery-Ajax-Code/chat/?user=5

ines says:

hola angel qué tal

angel says:

kk

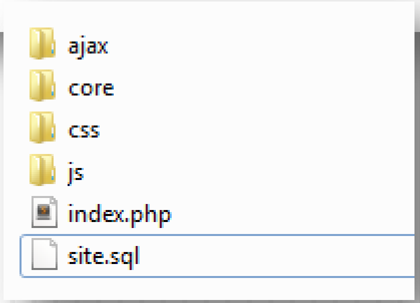
angel says:

kk

angel says:

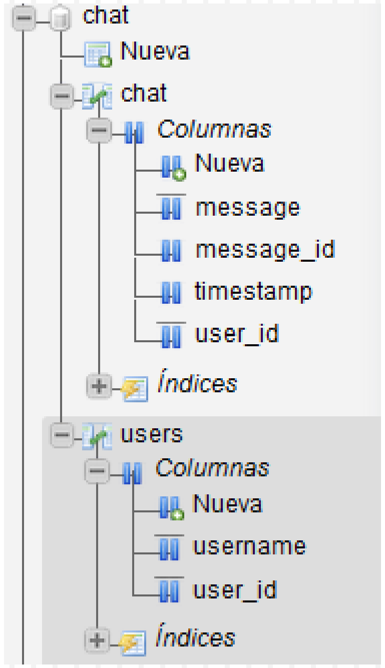
kk

Type here and hit return. Use shift + Return for a new line



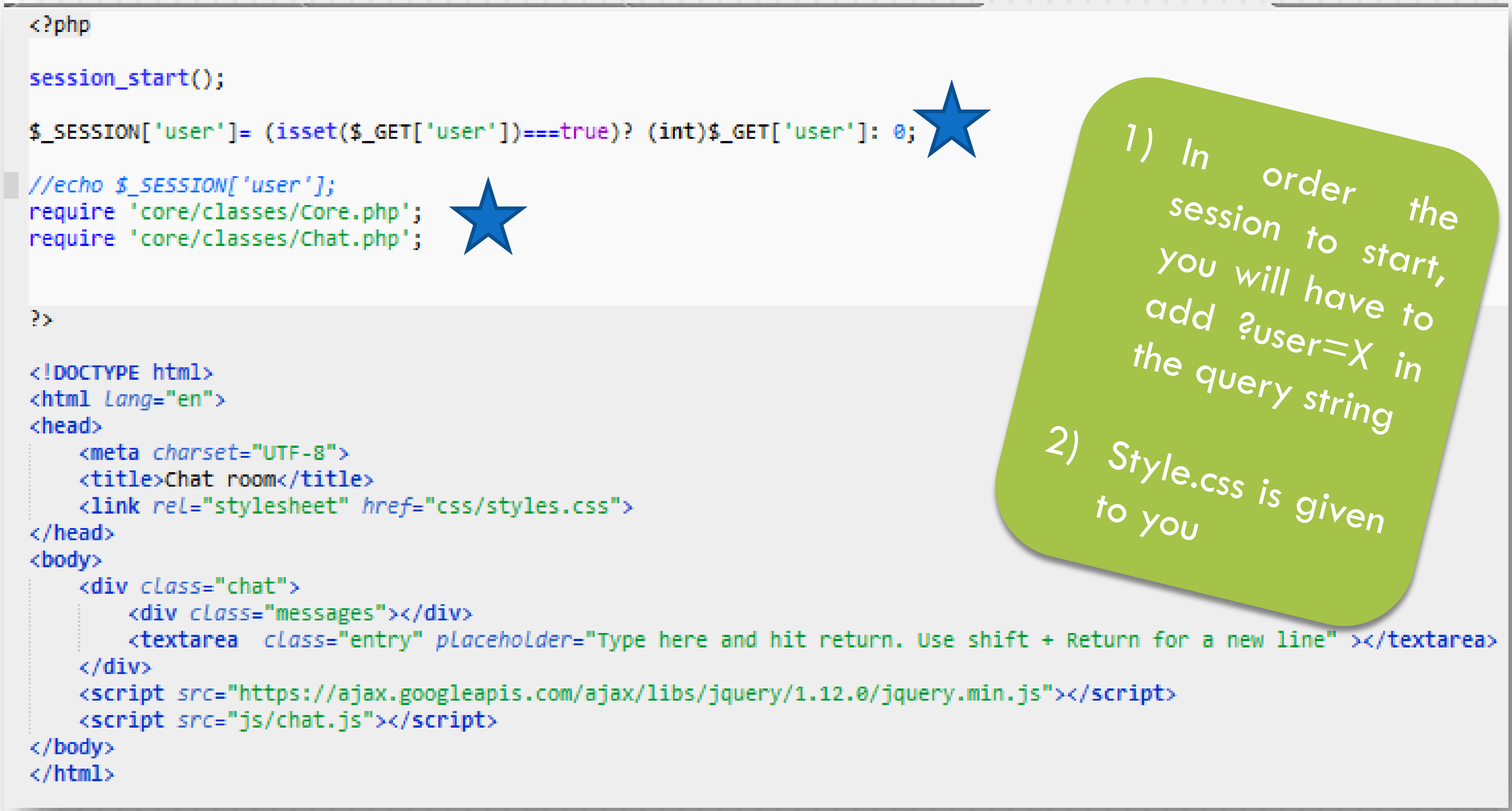
# 4. Requests and APIs

1. The first thing will be to create our data model. Go into phpMyAdmin and launch site.sql script given. That will create a table for the users and another one for keeping the messages.
2. Create two users using directly phpMyAdmin console.



# 4. Requests and APIs

3. This is going to be the index.php file:



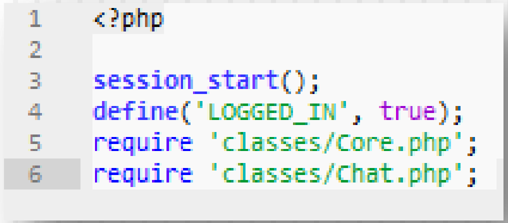
```
<?php

session_start();

$_SESSION['user']= (isset($_GET['user'])===true)? (int)$_GET['user']: 0;

//echo $_SESSION['user'];
require 'core/classes/Core.php';
require 'core/classes/Chat.php';


?>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Chat room</title>
    <link rel="stylesheet" href="css/styles.css">
</head>
<body>
    <div class="chat">
        <div class="messages"></div>
        <textarea  class="entry" placeholder="Type here and hit return. Use shift + Return for a new line" ></textarea>
    </div>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.0/jquery.min.js"></script>
    <script src="js/chat.js"></script>
</body>
</html>
```

1) In order the session to start, you will have to add ?user=X in the query string

2) Style.css is given to you

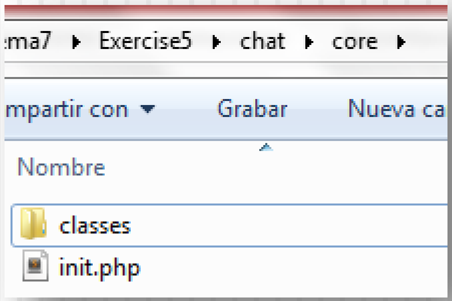
# 4. Requests and APIs

4. Include this code inside core/init.php like this:



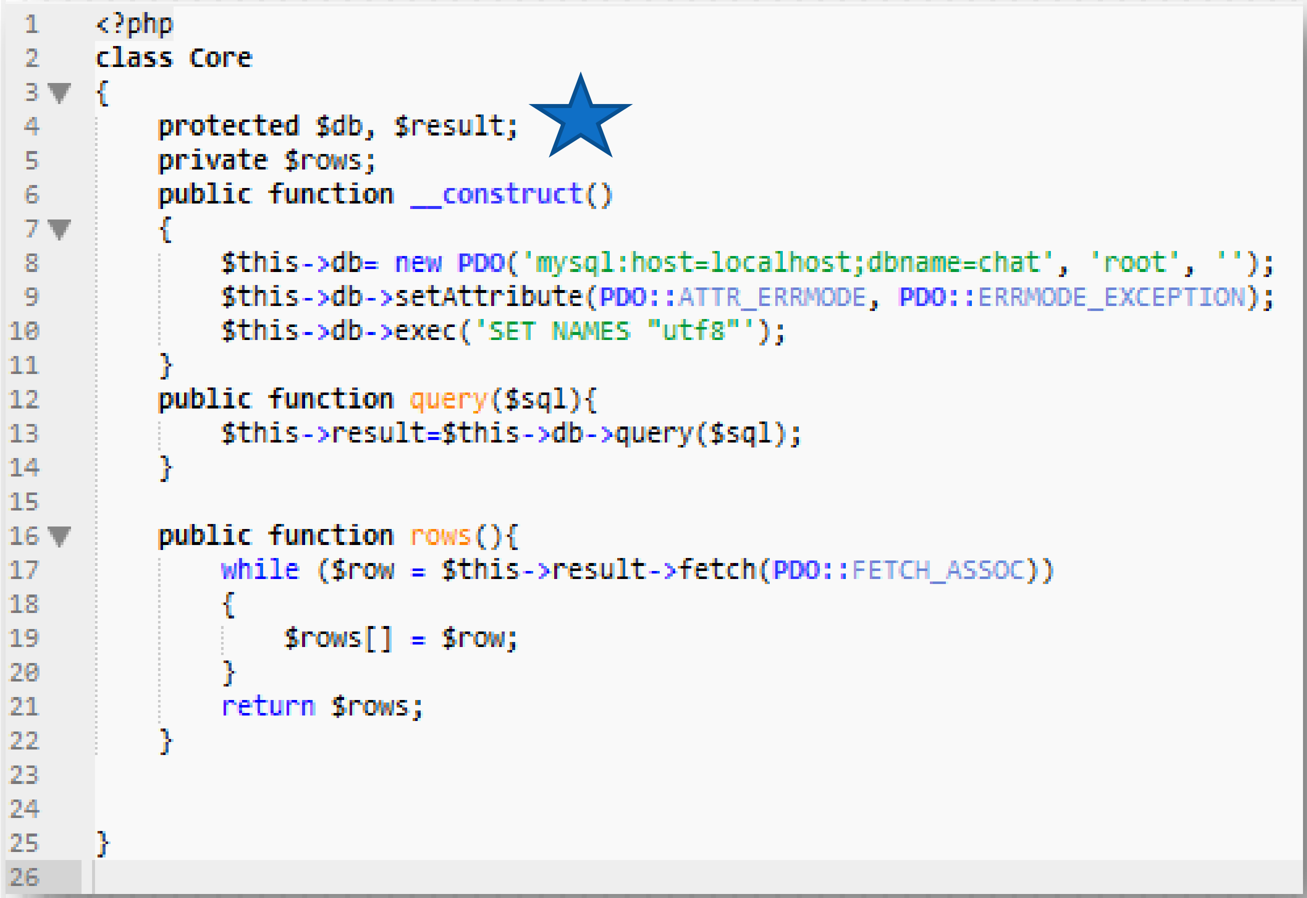
```php
<?php

session_start();
define('LOGGED_IN', true);
require 'classes/Core.php';
require 'classes/Chat.php';
```

- Every ajax call will initialize or check if there is a session.
- There is a constant defined (LOGGED_IN)
- Require Core and Chat Classes. You will see later <u>on how</u> Chat extends Core.



# 4. Requests and APIs

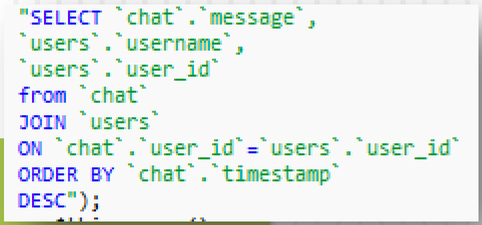
5. Create a class called Core inside core/classes/Core.php:



```php
<?php
class Core
{
    protected $db, $result;
    private $rows;
    public function __construct()
    {
        $this->db= new PDO('mysql:host=localhost;dbname=chat', 'root', '');
        $this->db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
        $this->db->exec('SET NAMES "utf8"');
    }
    public function query($sql){
        $this->result=$this->db->query($sql);
    }

    public function rows(){
        while ($row = $this->result->fetch(PDO::FETCH_ASSOC))
        {
            $rows[] = $row;
        }
        return $rows;
    }


}

```

# 4. Requests and APIs

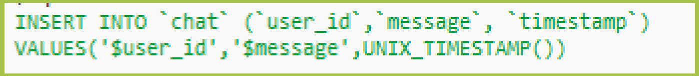
6. Create a class called Chat in core/classes/Chat.php and complete its functionality:



```
"SELECT `chat`.`message`,
`users`.`username`,
`users`.`user_id`
from `chat`
JOIN `users`
ON `chat`.`user_id`=`users`.`user_id`
ORDER BY `chat`.`timestamp`
DESC");
```

A.7.5. Taking into account that it extends Core class:

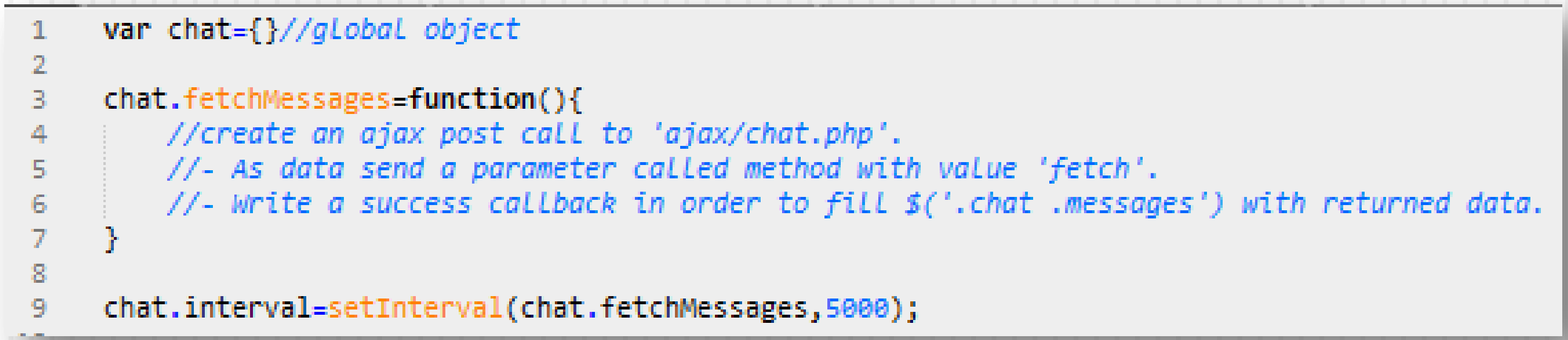
- Create a method called *fetchMessages()* where you have to query the DB and return the corresponding rows.
- Create a method called *throwMessage ($user_id, $message)*, where you have to insert the corresponding message into DB. *Hint: In order to insert the timestamp, use UNIX_TIMESTAMP() function.*



```
INSERT INTO `chat` (`user_id`,`message`, `timestamp`)
VALUES('$user_id','$message',UNIX_TIMESTAMP())
```

# 4. Requests and APIs

7. Inside js/chat.js you will create a timer that will refresh the message container of your page:

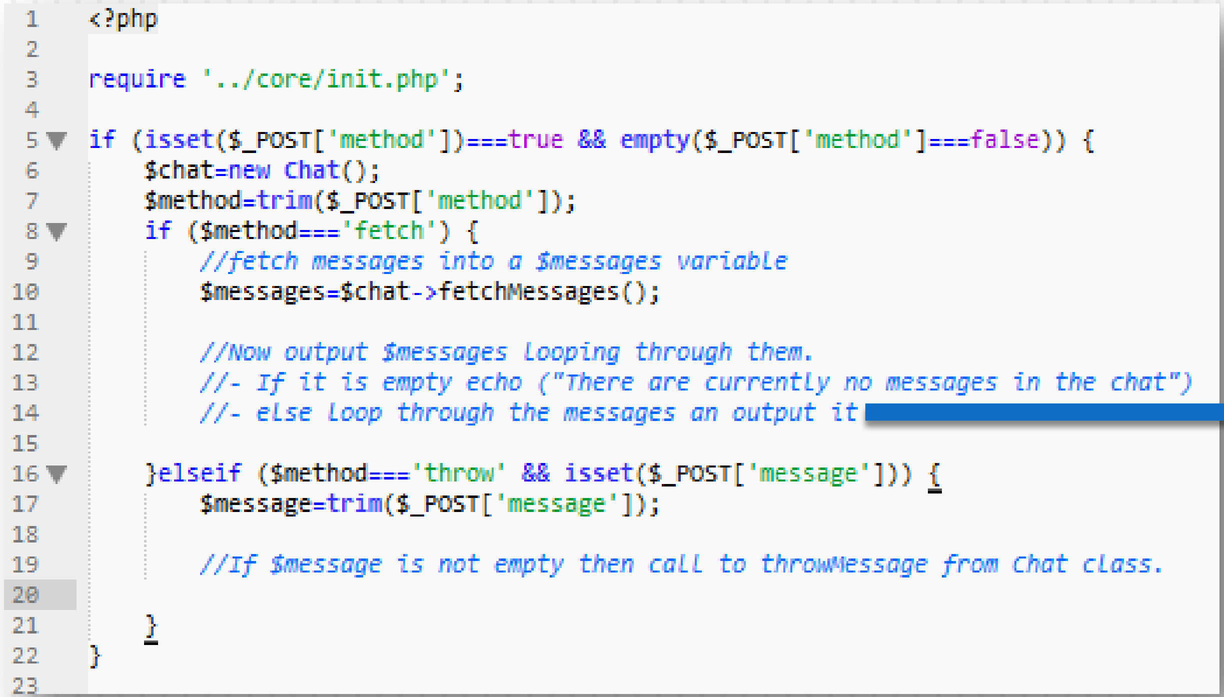


```
var chat={}//global object

chat.fetchMessages=function(){
    //create an ajax post call to 'ajax/chat.php'.
    //- As data send a parameter called method with value 'fetch'.
    //- Write a success callback in order to fill $('.chat .messages') with returned data.
}

chat.interval=setInterval(chat.fetchMessages,5000);
```

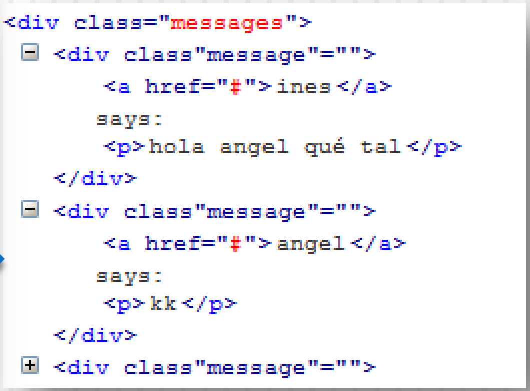
- In order to test it, open *chat.php* and write some *echo "Hello";* in order to see if the message container changes.
- Check Network tab in order to see calls every 5 sec.

# 4. Requests and APIs

8. Now you are going to complete ajax/*chat.php* file:



```php
<?php

require '../core/init.php';

if (isset($_POST['method'])===true && empty($_POST['method']===false)) {
    $chat=new Chat();
    $method=trim($_POST['method']);
    if ($method==='fetch') {
        //fetch messages into a $messages variable
        $messages=$chat->fetchMessages();

        //Now output $messages looping through them.
        //- If it is empty echo ("There are currently no messages in the chat")
        //- else loop through the messages an output it

    }elseif ($method==='throw' && isset($_POST['message'])) {
        $message=trim($_POST['message']);

        //If $message is not empty then call to throwMessage from Chat class.

    }
}
```

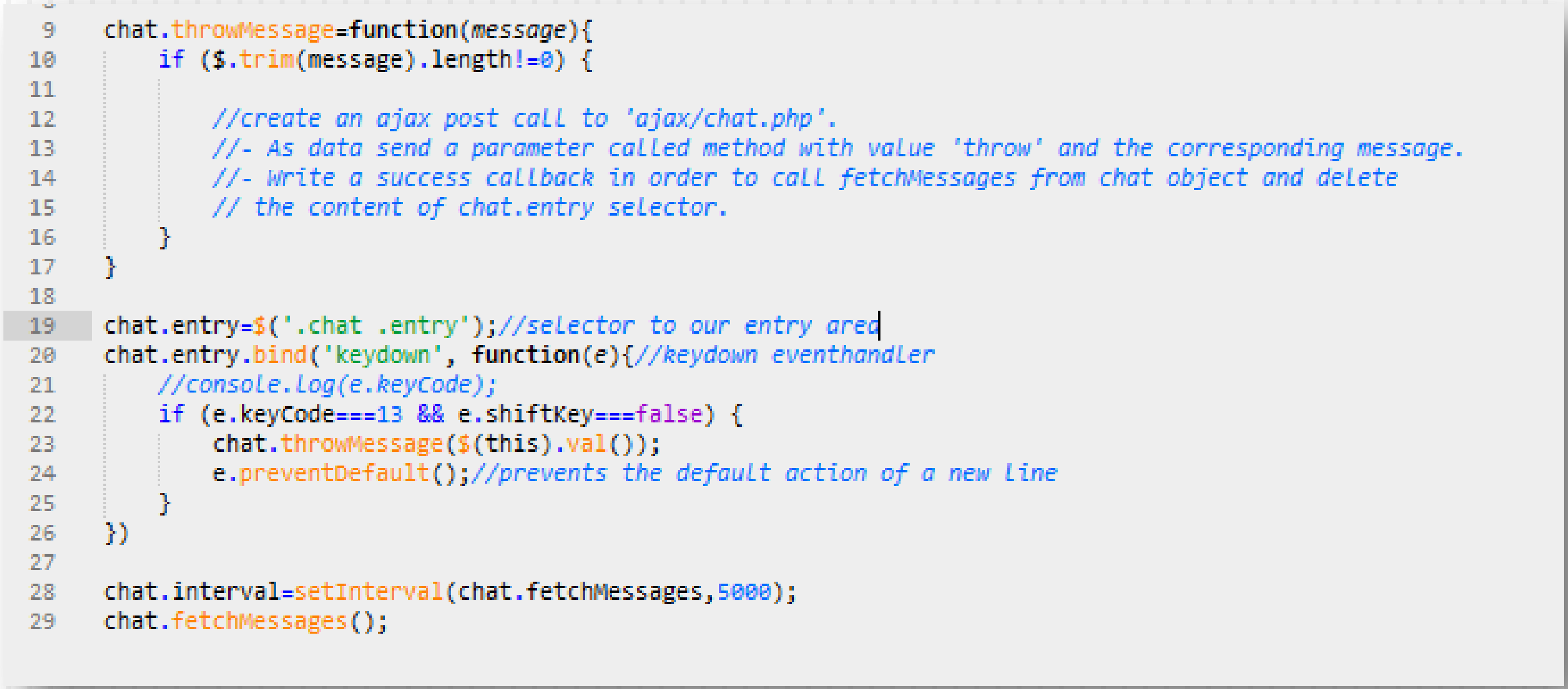


```html
<div class="messages">
 ⊟ <div class"message"="">
        <a href="#">ines</a>
       says:
        <p>hola angel qué tal</p>
   </div>
 ⊟ <div class"message"="">
        <a href="#">angel</a>
       says:
        <p>kk</p>
   </div>
 ⊞ <div class"message"="">
```

- In order to test it, create some messages directly in the database.

# 4. Requests and APIs

9. Let´s complete *chat.js* in order to throw new messages:

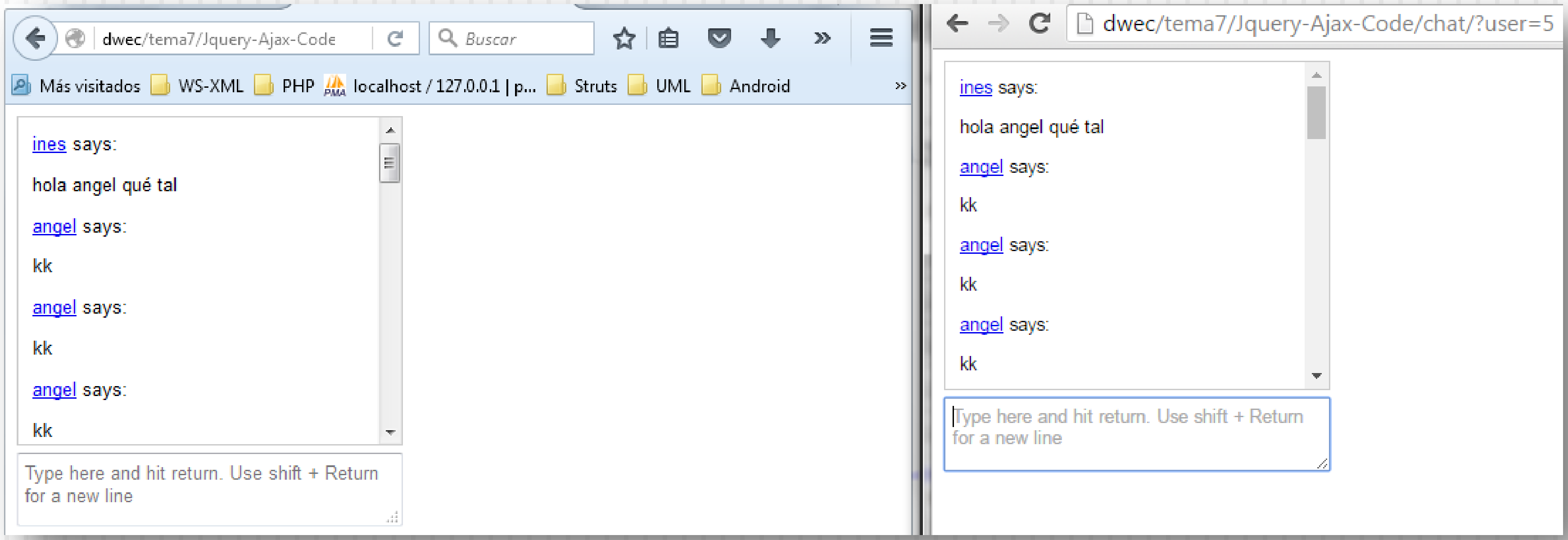


```
chat.throwMessage=function(message){
    if ($.trim(message).length!=0) {

        //create an ajax post call to 'ajax/chat.php'.
        //- As data send a parameter called method with value 'throw' and the corresponding message.
        //- Write a success callback in order to call fetchMessages from chat object and delete
        // the content of chat.entry selector.
    }
}

chat.entry=$('.chat .entry');//selector to our entry area
chat.entry.bind('keydown', function(e){//keydown eventhandler
    //console.log(e.keyCode);
    if (e.keyCode===13 && e.shiftKey===false) {
        chat.throwMessage($(this).val());
        e.preventDefault();//prevents the default action of a new line
    }
})

chat.interval=setInterval(chat.fetchMessages,5000);
chat.fetchMessages();
```

- Check how does *ENTER* and *SHIFT+ENTER* works when you enter new messages.

# 4. Requests and APIs

10. Test the application fully with two different users:


dwec/tema7/Jquery-Ajax-Code
Buscar
Más visitados
WS-XML
PHP
localhost / 127.0.0.1 | p...
Struts
UML
Android
dwec/tema7/Jquery-Ajax-Code/chat/?user=5

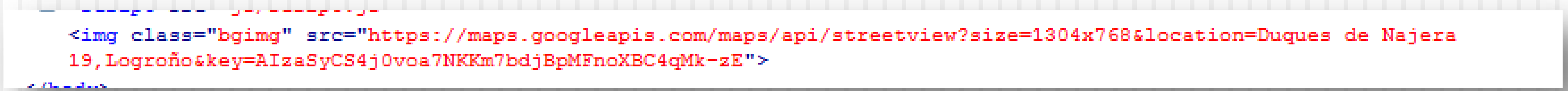
```
ines says:
hola angel qué tal
angel says:
kk
angel says:
kk
angel says:
kk
Type here and hit return. Use shift + Return for a new line
```

```
ines says:
hola angel qué tal
angel says:
kk
angel says:
kk
angel says:
kk
Type here and hit return. Use shift + Return for a new line
```

# 4. Requests and APIs

- <u>Exercise 6</u>: *Use of Google Streetview, Wikipedia and NY times APIs*

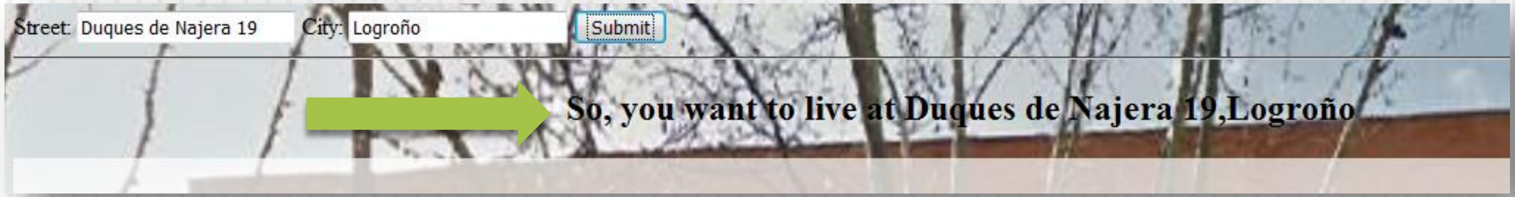
1. In order to appear a Google Street View image appear in the background, all you need to do is to append an <img> with the correct src
   1. The first step will be in our script.js using jQuery to take the information from the corresponding textboxes.
   2. Once we have them, you have to dinamically change src attribute from img, in order to call to Google Streetview API (Google Streetview documentation). Width and Heigth from *Screen.availWidth* and Screen.availHeigth.
      1. Remember to create your own API_KEY!!!
   3. Append the img to the body.



```
<img class="bgimg" src="https://maps.googleapis.com/maps/api/streetview?size=1304x768&location=Duques de Najera
19,Logroño&key=AIzaSyCS4j0voa7NKKm7bdjBpMFnoXBC4qMk-zE">
```

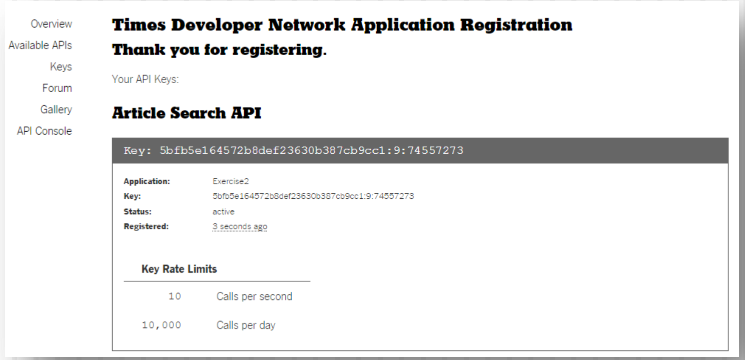
# 4. Requests and APIs

2. After that modify the heading in order to dynamically change its value (#greeting).



So, you want to live at Duques de Najera 19,Logroño

3. Now create your own *NYT API key*, following the corresponding instructions (NYT API key), in order to get access to *Article Search API*.



**Times Developer Network Application Registration**

**Thank you for registering.**

Your API Keys:

**Article Search API**

```
Key: 5bfb5e164572b8def23630b387cb9cc1:9:74557273
```

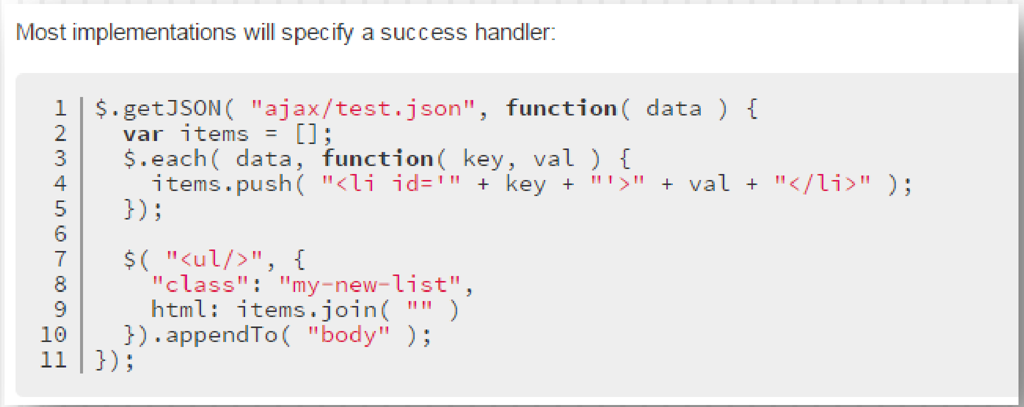
| | |
|---|---|
| **Application:** | Exercise2 |
| **Key:** | 5bfb5e164572b8def23630b387cb9cc1:9:74557273 |
| **Status:** | active |
| **Registered:** | 3 seconds ago |

**Key Rate Limits**

| | |
|---|---|
| 10 | Calls per second |
| 10,000 | Calls per day |

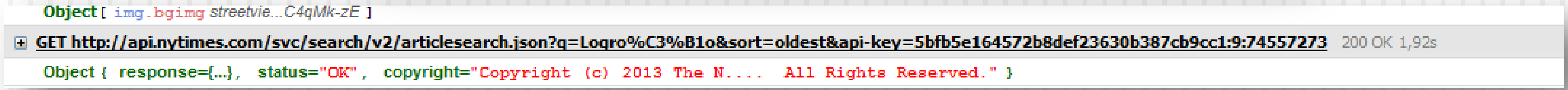
# 4. Requests and APIs

4. Check Article request API documentation in order to know how to conform a new request:
   1. In this case you will have to use $.getJSON() jQuery function in order to make the ajax query and get the information.
   2. In $.getJSON() documentation, check this example in order to know how to make the call and process the information. I recommend to make a *console.log* of the *data* obtained first.



Most implementations will specify a success handler:

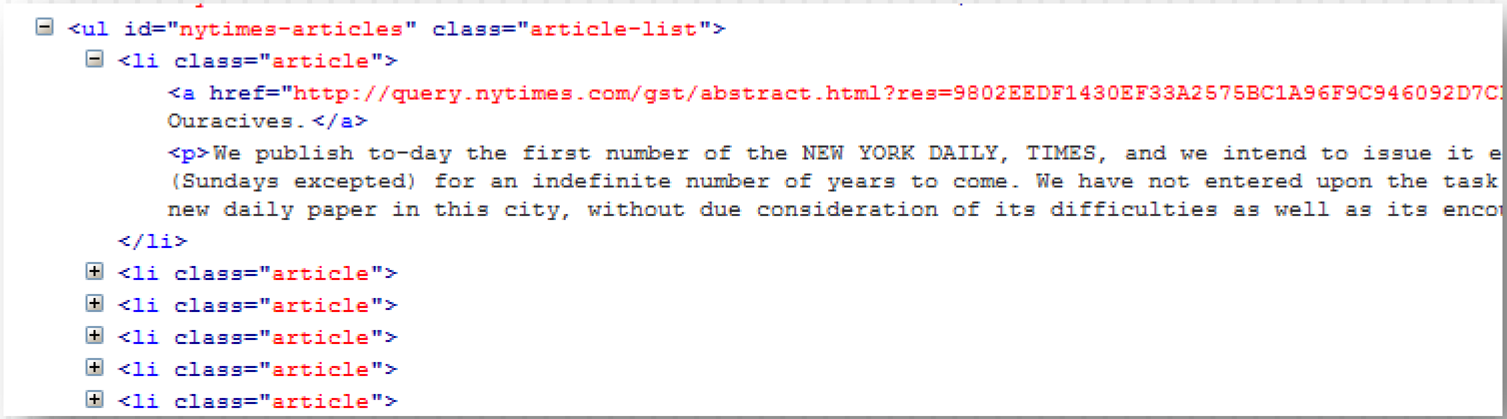
```
$.getJSON( "ajax/test.json", function( data ) {
  var items = [];
  $.each( data, function( key, val ) {
    items.push( "<li id='" + key + "'>" + val + "</li>" );
  });

  $( "<ul/>", {
    "class": "my-new-list",
    html: items.join( "" )
  }).appendTo( "body" );
});
```



```
  Object [ img.bgimg streetvie...C4qMk-zE ]
⊞ GET http://api.nytimes.com/svc/search/v2/articlesearch.json?q=Logro%C3%B1o&sort=oldest&api-key=5bfb5e164572b8def23630b387cb9cc1:9:74557273   200 OK 1,92s
  Object { response={...},  status="OK",  copyright="Copyright (c) 2013 The N....  All Rights Reserved." }
```

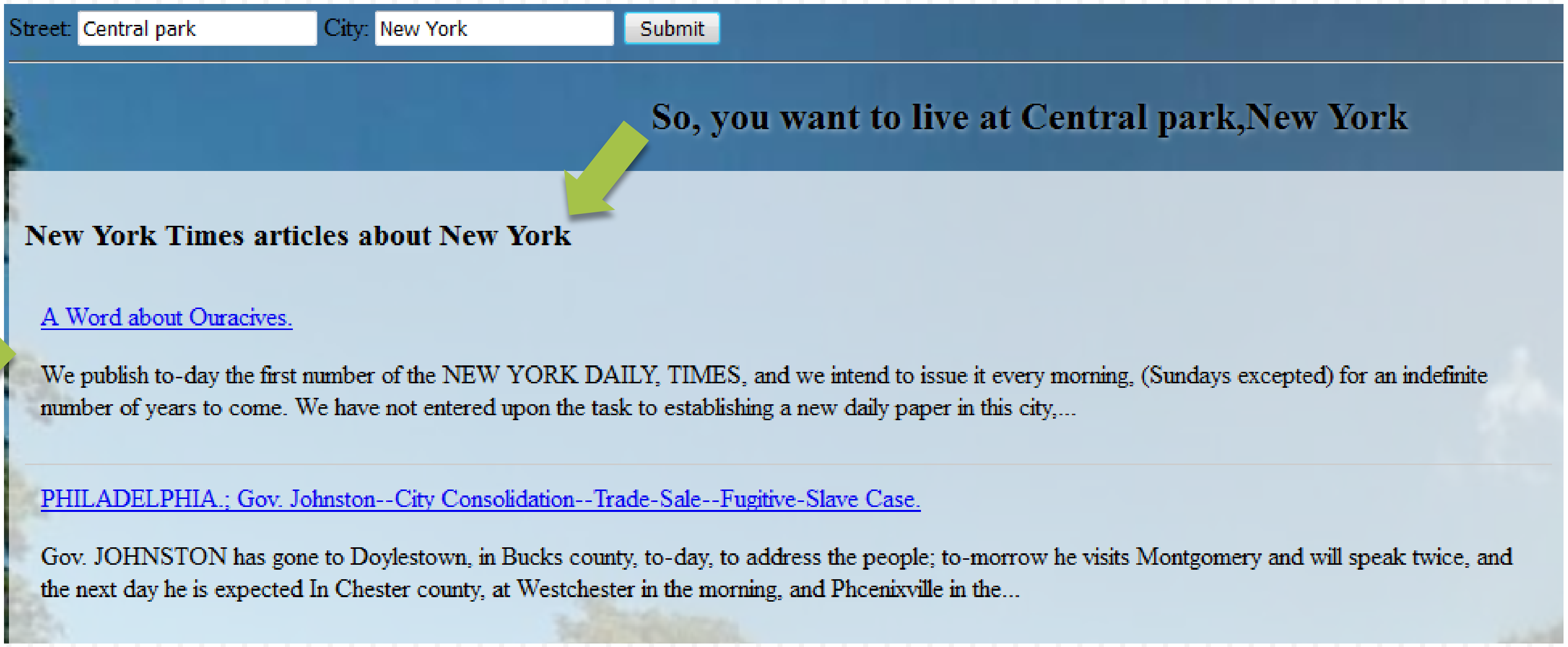
# 4. Requests and APIs

3. Inside the *ul* element with id="nytimes-articles" you will have to create this structure from the response data:



```
⊟ <ul id="nytimes-articles" class="article-list">
    ⊟ <li class="article">
          <a href="http://query.nytimes.com/gst/abstract.html?res=9802EEDF1430EF33A2575BC1A96F9C946092D7C
          Ouracives.</a>
          <p>We publish to-day the first number of the NEW YORK DAILY, TIMES, and we intend to issue it e
          (Sundays excepted) for an indefinite number of years to come. We have not entered upon the task
          new daily paper in this city, without due consideration of its difficulties as well as its enco
      </li>
    ⊞ <li class="article">
    ⊞ <li class="article">
    ⊞ <li class="article">
    ⊞ <li class="article">
    ⊞ <li class="article">
```

# 4. Requests and APIs



So, you want to live at Central park,New York

New York Times articles about New York

A Word about Ouracives.

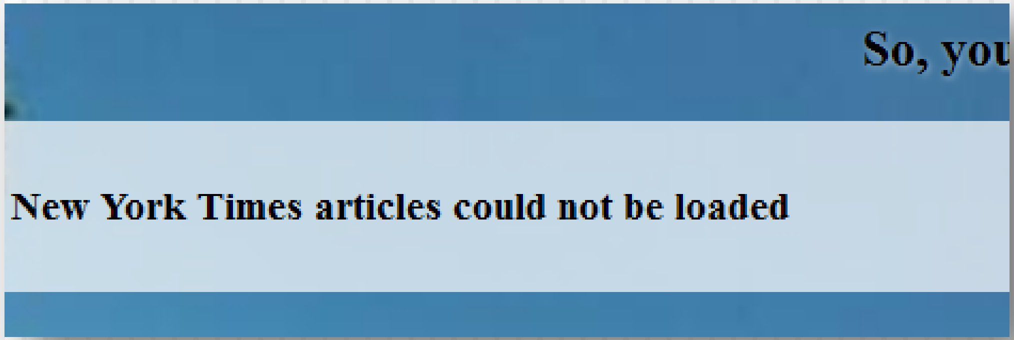
We publish to-day the first number of the NEW YORK DAILY, TIMES, and we intend to issue it every morning, (Sundays excepted) for an indefinite number of years to come. We have not entered upon the task to establishing a new daily paper in this city,...

PHILADELPHIA.; Gov. Johnston--City Consolidation--Trade-Sale--Fugitive-Slave Case.

Gov. JOHNSTON has gone to Doylestown, in Bucks county, to-day, to address the people; to-morrow he visits Montgomery and will speak twice, and the next day he is expected In Chester county, at Westchester in the morning, and Phœnixville in the...

## 4. Requests and APIs

5. Error handling: Sometimes the request can fail.
   1. In order to control that situation we are going to use $.error() method.
   2. You will have to choose the chaining jQuery property in order to use that method.
   3. In case of error, this new message will be displayed (#nytimes-header):

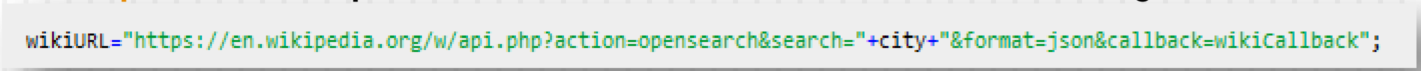


So, you

New York Times articles could not be loaded

# 4. Requests and APIs

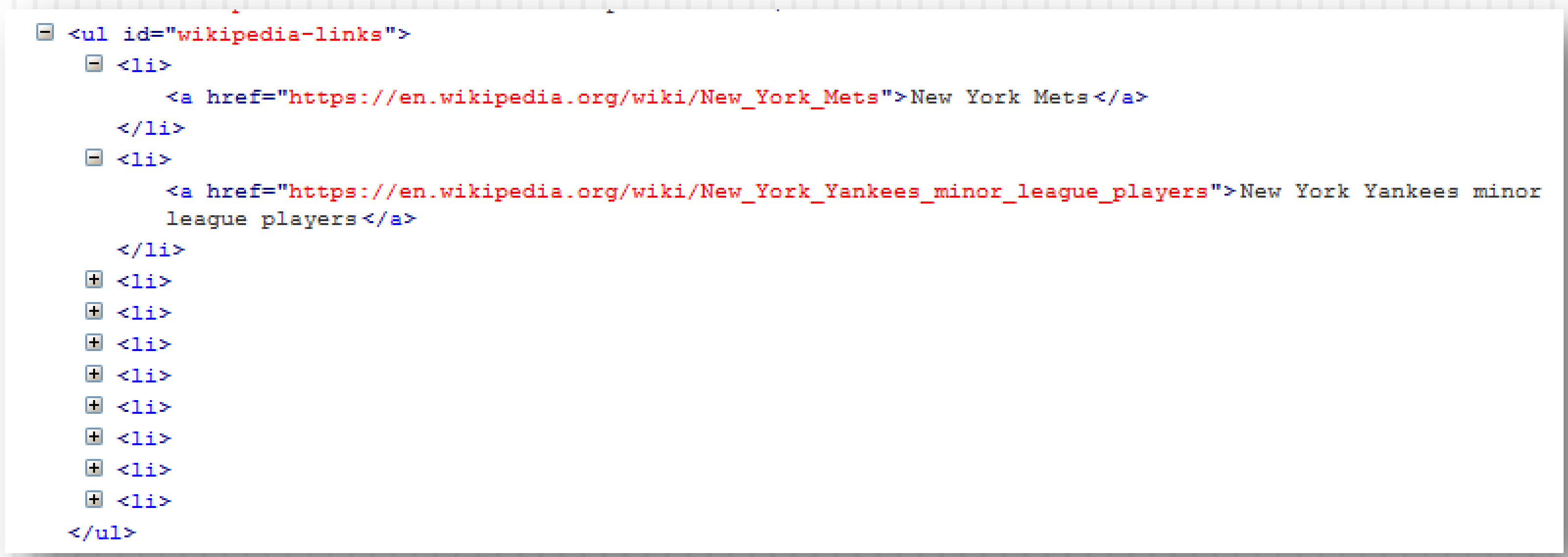
6. Wikipedia API.
    1. From all wikipedia API, you are going to search for items using opensearch protocol. The URL should be something like this:



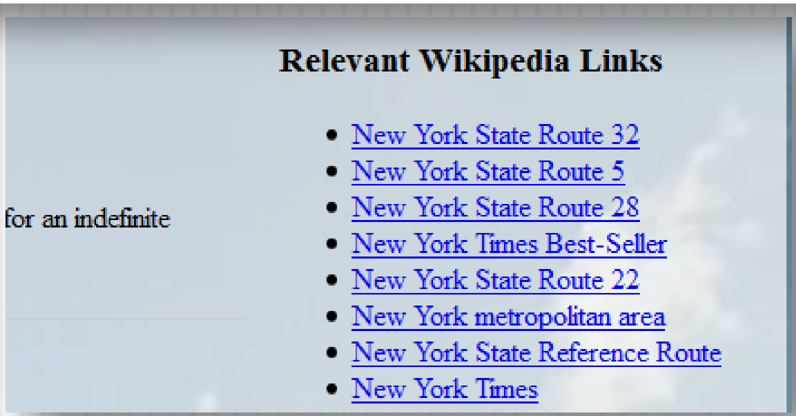
```
wikiURL="https://en.wikipedia.org/w/api.php?action=opensearch&search="+city+"&format=json&callback=wikiCallback";
```

    2. Now, we are going to make the call using $.ajax method instead of $.getJSON. The second one is an abstraction of $.ajax, and would not work in this case because of CORS policy.
        1. In order to solve it, we are going to make a **JSONP** request instead of JSON (that is why we are using $.ajax() instead of $.getJSON()). **Check this tutorial in order to understand the workaround that is made with JSONP.**
        2. With the previous information, you need to send a JSONP request using $.ajax method. As parameters to $.ajax, you will set:
            1. dataType="jsonp"
            2. success (this will have an anonymous function where you will control the data from the response in order to add the corresponding elements into the DOM).

# 4. Requests and APIs



```
<ul id="wikipedia-links">
    <li>
        <a href="https://en.wikipedia.org/wiki/New_York_Mets">New York Mets</a>
    </li>
    <li>
        <a href="https://en.wikipedia.org/wiki/New_York_Yankees_minor_league_players">New York Yankees minor
        league players</a>
    </li>
    <li>
    <li>
    <li>
    <li>
    <li>
    <li>
    <li>
    <li>
</ul>
```



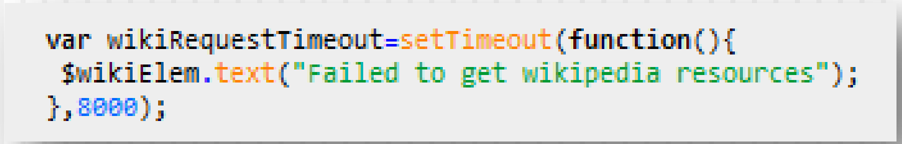
for an indefinite

**Relevant Wikipedia Links**

- New York State Route 32
- New York State Route 5
- New York State Route 28
- New York Times Best-Seller
- New York State Route 22
- New York metropolitan area
- New York State Reference Route
- New York Times

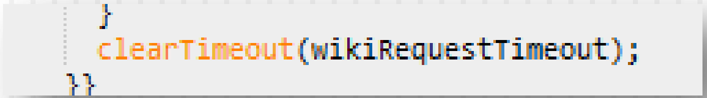
# 4. Requests and APIs

7. JSONP Error Handling.
   1. In order to control the case where something in the request goes wrong:
      1. Before calling $.ajax method, add this code:



```
var wikiRequestTimeout=setTimeout(function(){
 $wikiElem.text("Failed to get wikipedia resources");
},8000);
```

      2. And inside $.ajax method at the end of the success anonymous function:



```
    }
    clearTimeout(wikiRequestTimeout);
}}
```

      3. <u>This way *if nothing clears the timeout, we suppose there is something wrong and that message will be shown to the user.*</u>
      4. You can test it changing the URL to something else, in order the call to fail.