Programmable Web Project

Exercise 4

RESTful clients

Learning outcomes (I)

- ~~Students understand what a Web API is and learn different Web API architectures.~~
- ~~Students understand the concept of hypermedia and how it can be used to build Web APIs.~~
- ~~Students are able to design and implement a Web API following REST architectural style principles using existing web frameworks.~~

Learning outcomes (II)

- ~~Students are able to write unit and functional tests to inspect their APIS.~~
- ~~Students are able to document their Web APIs using adequate software tools.~~
- **Students are able to implement simple software applications that make use of the APIs.**

RESTFUL CLIENTS

- Just one entry point
- No need to memorize URLs
- No need to memorize workflow -> @controls
- No need to memorize resource attributes:
  - Not even to create/modify resources
- Link relations tells me which are the actions that I can perform in certain states
  - Buttons and links to activate
  - Forms to visualize

PYTHON CLIENTS

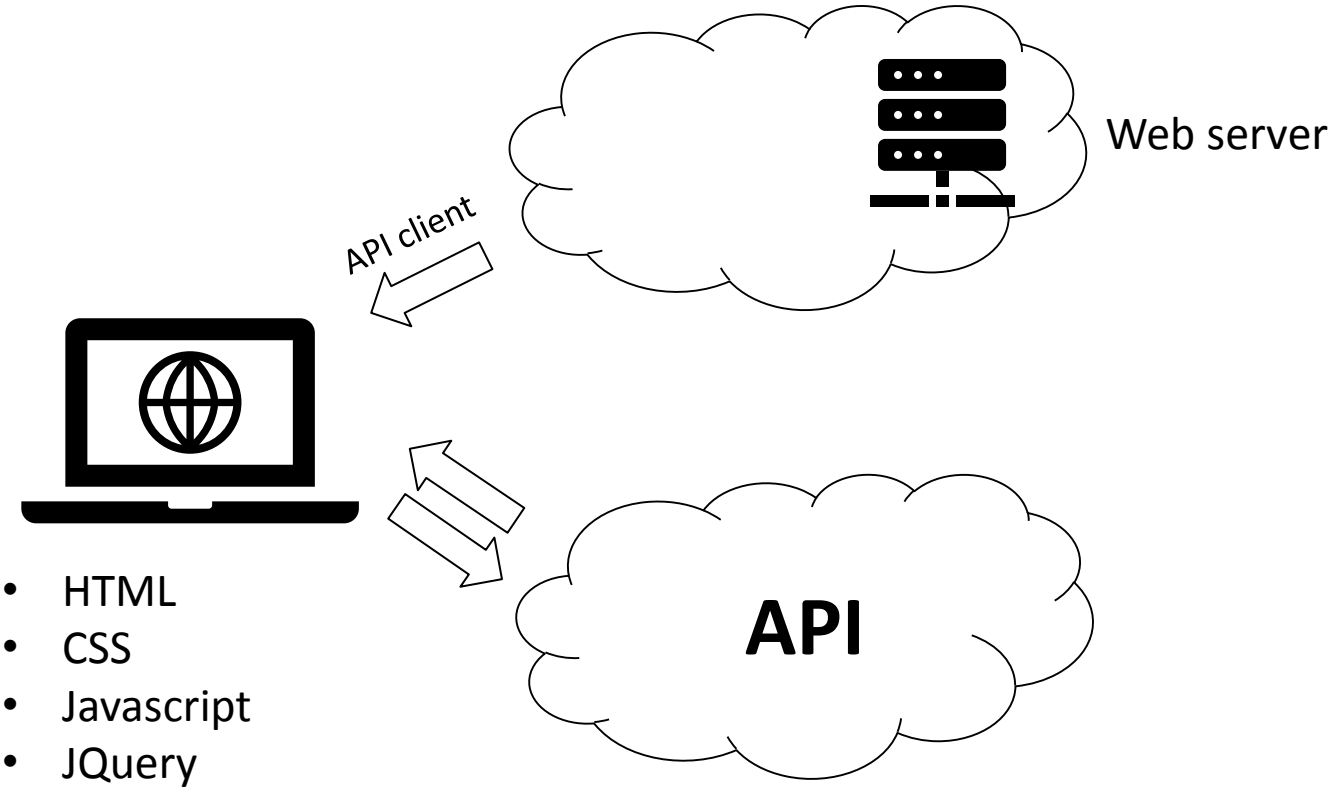
REQUESTS LIBRARY

```
import requests
import json

SERVER_URL = "http://myserver.com"
resp = requests.get(SERVER_URL + "/api/artists/")
body = resp.json()

data = {"name": "Mono", "location": "JP"}
resp = reqests.post(SERVER_URL + "/api/artists/", data=json.dumps(data))
Resource_url = resp.headers["Location"]
```

Using sessions

```
import json
import requests
SERVER_URL = "http://myserver.com"
with requests.Session() as s:
          s.headers.update({"Accept": "application/vnd.mason+json"})
          resp = s.get(SERVER_URL + "/api/artists/")
```

CLIENTS EMBEDDED IN WEBSITE



- HTML
- CSS
- Javascript
- JQuery

Same-origin policy

- Due to security issues Javascript has limitations on where it can send HTTP requests
- The same-origin-policy states that:
  - Javascript have access to the properties of windows and documents that have the same origin (host and port)
  - XMLHttpRequest object can send HTTP requests to documents that have the same origin as the current web page.
    - You cannot execute AJAX running a Web page from the file system.
- Cross origin requests are permitted if the server sends adequate CORS (Cross-Origin Resource Sharing headers)

HTML

HTML

```
<!DOCTYPE html>
<html>
    <head>
        <title>Hello World!!!</title>
        <link rel="stylesheet" type="text/css" href="forum.css">
        <script type="text/javascript" src="jquery.js"></script>
    </head>
    <body>
        <p>
            "Hello World"
        </p>
        <img src="/images/helloworld.png"/>
    </body>
</html>
```

HTML anchor, lists and paragraphs

<p> -> The content is a paragraph or text. Can contain also other content such as images or lists.

<image src="*link*"> -> The content is an image

<ul> / **<ol>** -> Defines a list of elements (unordered / ordered)

<li> -> Defines each one of the elements in the list

<a href="*link*"> -> Defines hyperlinks to redirect the user to other document or to a different section of the document. The href attribute defines the URI of the target resource while the content is the text/image that user sees.

HTML forms

`<form>` -> Tool to collect data from user.

Fields:

- text-fields, checkboxes, radio-boxes, comboxes....
- Each one of this fields are defined by `<input>`, `<textarea>`, `<select>` elements.
  - http://www.htmldog.com/guides/htmlbeginner/forms/

- A form must contain two attributes:
  - action: The target resource who process the form data
  - method: The method sent in the HTTP request.
    - Only GET and POST accepted. For the rest use AJAX

HTML containers

`<div>` -> Block-line element.

`<span>` -> Inline element.

- Use heavily to provide style.
  - Excess of use is not recommendable for creating Web page -> NO SEMANTIC
  - HOWEVER, it is imprescindible when you are designing UI for other type of applications.

CSS

CSS

- Cascading Style Sheets is a style language used to describe the presentation of a document written in a markup language.
  - It is based in a set of property and values:
    - Property: The feature that we want to give style
    - Value: How the feature must be handled by the browser engine.
  - Selectors defines the HTML elements modified by that properties. They can be nested.
    - Element selector
    - Class selector (starting with a .)
    - Id selector (starting with a #)

```
1  body {
2      margin: 0 auto;
3      width: 940px;
4      font: 13px/22px Helvetica, Arial, sans-serif;
5      background: #f0f0f0;
6  }
```

```
#sidebar ul li a {
        display: block;
        margin-right: 20px;
        width: 140px;
        font-size: 14px;
}

#sidebar ul li.selected a {
    color: #fff;
}
```

JAVASCRIPT

Javascript principles (I)

- **Dynamic-typed and object-oriented language.**
- Javascript types:
  - Primitive types: number, string, boolean, null and undefined
  - Objects: collection of properties (named values)
    - *Arrays* are objects that represented an ordered collection of values.
    - *Functions* are object that has executable code associated.
      - Javascript treats them as regular objects -> can be assigned to variables.
- Operators: quite close to Python operators

Javascript principles(II)

- Variables must be declared using the `let` keyword:

```
let sum;
sum = 5+3;
```

  - **Global variables** -> Those defined in the global object
  - **Block variables ->** Variables defined inside a block

- Statements end with ";"
- Statements are grouped in blocks.
  - Blocks are defined using {}

Javascript principles(III)

- Defining functions:

```
let square = function (x) {return x*x}

function square (x) {return x*x}
```

  – Javascript supports nesting functions and closures.

- Invoking functions:

```
let result= square(2);
```

  – A function can be invoked with less arguments than declared parameters => the rest are of type ”undefined”.

Object literals

- Easiest way of crating an object is using Object Literal notation.

```
var point = {x:0, y:0}
var book = {title:"RESTful", retrieveIndex:function(){...}}
```

- Querying and Setting properties using the . or [] operators.

```
point.x = 1;          point ["x"] = 1;
let x= point.x        let x = point ["x"]
```

  - The [] operator permits dynamic generation.

- Functions belonging to objects are methods.

- **`this`** keyword is a reference to the object owning a method.

DOM

DOM in browsers.

- The browser parses the HTML file into a DOM tree.
- Each Javascript Global object has associated a Document object.
- Javascript implements the DOM interface, so we can traverse and modify the different elements and properties of the Document.
- Some methods:
  - **`Document.getElementsByTagName(tag)`**
  - **`Document.getElementsByName(name)`**
  - **`Document.getElementsById(name)`**
  - **`Document.getElementsByClassName(name)`** **`->HTML5`**
  - **`Document.querySelectorAll(cssselector)`**

DOM in browsers.

- Attributes are accessed usin the . syntax:

```
var link = document.getElementById(”mylink”)
var destination = link.href
```

- Element content can be accessed:
  - Using innerHTML property.
  - Using the property textContent (only work with text nodes)
- Elements can be created using `document.createElement()` or `document.createTextNode()`
- Elements can be appended using `Element.appendNode()`
- Elements can be deleted using `Element.removeChild()` or `Element.replaceChild()`
- Element can be styled using the `Element.style` attributes.
  - Same as CSS properties.

Events and handlers (I)

- Javascript is not usually used synchronously but asynchronously:
  - Programs register event handlers functions
  - Those functions are invoked asynchronously when the event occurs
- Each event has:
  - A name = the general type of event that has happened (”click”, ”readystatechange”)
  - A target = the object on which the event ocurred (usually a DOM Node Element).
- In HTML the events propagates up the document tree (bubbling), so a parent can handle also an event captured in a child.
  - The propagation is stopped if a handler return false.
- The method **`.addEventListener (event, handler`**) permits associate multiples handler to the same event:

```
button.addEventListener (”click”, ”clickButtonhandler”)
```

Events and handlers (II)

- Handling events process:
  - Write a handler function.
  - Assign the handler to the property of the target in charge of processing an event. Usually is the name of the event prefixed with on.
  - When an event occurs the event handler is called.
    - Receives the event object as argument;
      - Event object contains details about event: type, current target, target (the target which originally dispatched the event and other properties and methods (for example to stop propagation).
    - this keyword is the target.

NETWORKING AND DEBUGGING

AJAX (I)

- Asynchronous Javascript and XML.
- Technology that uses scripted HTTP to start data exchange with a server without reloading the document, having the user experience of a Desktop application.
  - The generation and sent of an HTTP AJAX requests is triggered by an event handler (e.g pressing a button)
  - The HTTP request is not blocking. Javascript continues execution
  - An event handler process the response when it is received by the browser.
- In modern browsers AJAX is implemented using the XMLHttpRequest object as transport.

AJAX (II)

- STEPS:

1. Creation of XMLHttpRequest object:

```
var request = new XMLHttpRequest()
```

SYNTAX IS PLATFORM DEPENDANT

2. Specifiying the request: method, URL.

```
request.open ("METHOD", "URL" = new XMLHttpRequest()
```

   - Add headers: `request.setRequestHeader ("name", "value")`
   - Add response handler: `request.onreadystatychange=handler`

3. Send the request including the body:

```
request.send("body")
```

4. Process the response. When the handler is invoked the request contains:
   - A property named `status` with the status code
   - The response headers accessible through the `request.getResponseHeader()`
   - The response body in textual form in the `responseText` or in Document object in `responseXML` property

JQUERY

JQuery

- Javascript library which simplies DOM manipulation, event handling, styling, AJAX and support animation. It also fixes cross-browser issues.

http://jquery.com/

API: http://api.jquery.com/

JQuery

- **$ represents the JQuery object.**
  - A JQuery object receives as an argument a selector (similar to CSS selector)
  - When the JQuery object is executed it returns a set of JQuery objects, each one wrapping a HTML DOM element that matches the selector:
    - `$("div.red")` returns a list of all the div which has a class "red" inside the document.

      ```
      <div class="red"><p>Here some text</p></div>
      ```
  - Some methods can be called without need of creating a Jquery object first: `$.makeArray()`

JQuery traversing methods

- Accessing parent, child and closest sibling

`.parent(), .next(), .prev()`

```
$("div.red").next()
```

- Accessing ancestors and descendants.
  - All following traversing methods receive a selector as an argument, that filters the set of JQuery objects resulting

`.children()  .find()  .closest ()  .parents()`

```
$("#mainContent").find("div li")
```

- In order to traverse and manipulate the page we must wait until it's ready to be used.

```
$(document).ready(function(){
    // Your jQuery code goes in here
});
```

JQuery modifying elements and styles

- Modify content of element

```
.empty () .remove()
.append("HTMLtext").html("HTMLtext")
```

- Modify style

```
.hide() .show() .addClass() .removeClass() .css()
```

- Accessing and modifying properties / attributes / text

`.val([newval])` => Extracts/modify the value of a form input

`.text([newtext])` => Extracts/modify the value of the text inside an HTML element

`.attr("attributeName"[, attributeValue)]` => Extracts/modify the value of an attribute of an HTML element.

Jquery and AJAX

```
$.ajax({
        url: apiurl, //The URL of the resource
        type: "PUT", //The resource method
        contentType: CONTENT_TYPE, //The mime type of the request body
        data:userData, //The body of the HTTP request
        processData: false, //Do not transform the data in key-value
        dataType:RESPONSE_FORMAT, //The format expected in the
                                  //response : xml or json
        headers: {"Authorization":"admin"},// An object containing
                                           //headers
        success:function(body, status, jqxhr){
        //code to be executed when response is
        //received.
},
        error: function(jqxhr, type, error){
        //code to be executed when response has an
        //error status code or response is malformed
}
        });
```

Debugging Javascript

- All modern browser that support Javascript have a Javascript console which permits accessing to the Global object.
- Furthermore, browser have other tools to:
  - Visualize and modify the DOM
  - Modify style in elements
  - Monitor network (HTTP requests and responses)
- Tools and javascript console:
  - Chrome: Tools>Developer Tools>
  - Firefox: Web Developer
- The way of printing text in the console output is using the `console.log("text")` method.

BOOTSTRAP

Bootstrap

- https://getbootstrap.com/
- Front-end component library to build responsive and fluid layouts, that can be visualized also on mobile phones.
- Use our powerful mobile-first flexbox grid to build layouts of all shapes and sizes thanks to a twelve column system


```
<div class="container">
  <div class="row">
    <div class="col">1 of 3</div>
    <div class="col-6">2 of 3 (wider)</div>
    <div class="col">3 of 3</div>
  </div>
  <div class="row">
    <div class="col">1 of 3</div>
    <div class="col-5">2 of 3 (wider)</div>
    <div class="col">3 of 3</div>
  </div>
</div>
```

FLASK

Web applications with Flask.

- Returning back static files:

```
app = Flask(__name__, static_folder='', static_url_path='')
```

  - static_folder: the folder in the filesystem that contains the Web application files (HTML, CSS and Javascript)
  - static_url_path: The URL associated to the static folder.

Links

- HTML reference: https://developer.mozilla.org/en/docs/Web/HTML/Element
- HTML tutorial: http://htmldog.com/guides/html/
- CSS references: http://htmldog.com/reference/cssproperties/
- Javascript tutorial: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Getting_Started
- JQuery API: http://api.jquery.com/
- JQuery tutorial: http://learn.jquery.com/about-jquery/how-jquery-works/