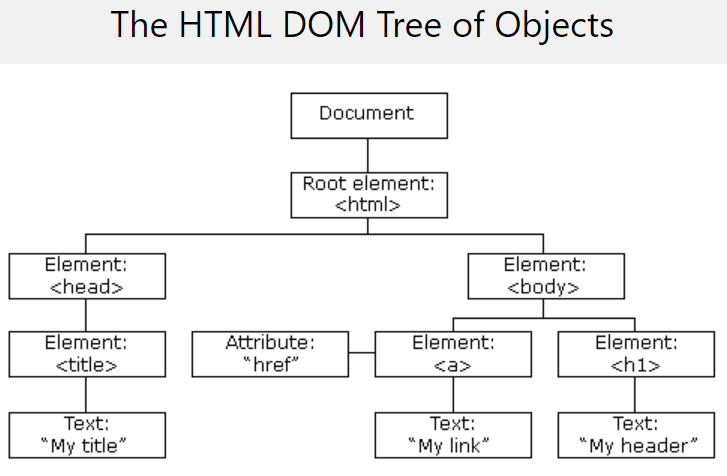
## JavaScript, DOM, AJAX and SVG

**General part**

* **Explain about the Document Object Model, and why it’s (very) relevant for modern Web-development**

[**https://www.w3schools.com/js/js\_htmldom.asp**](https://www.w3schools.com/js/js_htmldom.asp)

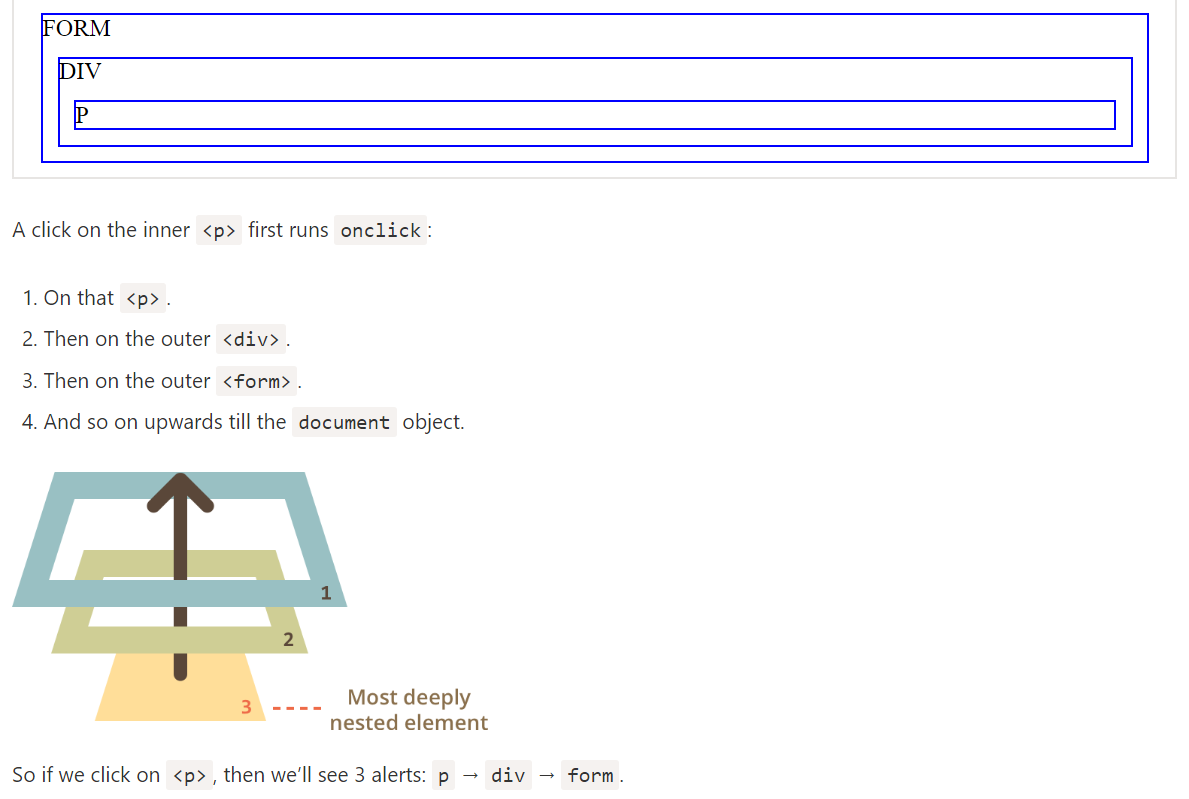
* + When a web page is loaded, the browser creates a Document Object Model (DOM) of the page
  + With the object model, JavaScript can access and change all the elements and attributes of an HTML document also the CSS.
  + With the object model, JavaScript can remove and add new HTML elements and attributes and create dynamic HTML.
  + With the object model, JavaScript react to all existing HTML event in the page
  + With the object model, JavaScript can create ned HTML events in the page



* **Explain (using an example of your own choice) about JavaScript events, and Event Bubbling**

[**https://javascript.info/bubbling-and-capturing**](https://javascript.info/bubbling-and-capturing)

* + Events are things that happen to HTML elements
  + JavaScript can react on these events (onchange, onclick, onmouseover, onmouseout, onkeydown, onload)
  + Event handlers can handle and verify user inputs actions and browser actions.
  + HTML event attributes can execute JavaScript-code directly and call JavaScript functions
  + Event bubbling runs the handler on an element, then on its parent, then all the way up



* **Elaborate on how JSON or XML supports communication between subsystems, even when the subsystems are implemented on diﬀerent platforms.**

[**https://www.w3schools.com/js/js\_json\_xml.asp**](https://www.w3schools.com/js/js_json_xml.asp)

* + JSON and XML is basically just plain text.
  + It is not platform specific and therefore any system that receives this format can work with it
  + The strength is that the implementation on the server side is irrelevant when what is being communicated from client-server is JSON/XML. The webservice can have been made in Java, C#, C++, Python and so on and still function as long as we use JSON or XML when parsing data from client-server.
* **Explain the topic AJAX and how it has changed the way modern web-applications are created**

[**https://www.w3schools.com/js/js\_ajax\_intro.asp**](https://www.w3schools.com/js/js_ajax_intro.asp)

* + Read data from a web server - after the page has loaded
  + Update a web page without reloading the page
  + Send data to a web server - in the background
  + AJAX = Asynchronous JavaScript And XML.
  + AJAX is not a programming language.
  + AJAX just uses a combination of:
  + A browser built-in XMLHttpRequest object (to request data from a web server)
  + JavaScript and HTML DOM (to display or use the data)
  + AJAX is a misleading name. AJAX applications might use XML to transport data, but it is equally common to transport data as plain text or JSON text.
  + AJAX allows web pages to be updated asynchronously by exchanging data with a web server behind the scenes. This means that it is possible to update parts of a web page, without reloading the whole page



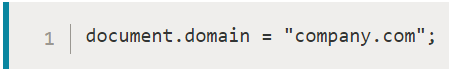
* **Explain the Same Origin Policy (for AJAX), and different ways to work around it:**

<https://developer.mozilla.org/en-US/docs/Web/Security/Same-origin_policy>

* + The same-origin policy restricts how a document or script loaded from one origin can interact with a resource from another origin. It is a critical security mechanism for isolating potentially malicious documents
  + Definition of an origin
  + Two pages have the same origin if the protocol, port (if one is specified), and host are the same for both pages. You'll see this referred to as the "scheme/host/port tuple" at times (where a "tuple" is a set of three components that together comprise a whole).

**Change an origin:**

* + A page may change its own origin with some limitations. A script can set the value of document.domain to its current domain or a superdomain of its current domain. If it sets it to a superdomain of its current domain, the shorter domain is used for subsequent origin checks. For example, assume a script in the document at http://store.company.com/dir/other.html executes the following statement:



* + After that statement executes, the page can pass the origin check with http://company.com/dir/page.html (assuming http://company.com/dir/page.html sets its document.domain to "company.com" to indicate that it wishes to allow that - see document.domain for more). However, company.com could not set document.domain to othercompany.com since that is not a superdomain of company.com.
  + The port number is kept separately by the browser. Any call to the setter, including document.domain = document.domain causes the port number to be overwritten with null. Therefore one cannot make company.com:8080 talk to company.com by only setting document.domain = "company.com" in the first. It has to be set in both so that port numbers are both null.

**Definition:**

* + The following table gives examples of origin comparisons to the URL http://store.company.com/dir/page.html:
  + **URL Outcome Reason**

<http://store.company.com/dir2/other.html> Success

http://store.company.com/dir/inner/another.html Success

<https://store.company.com/secure.html> Failure Different protocol

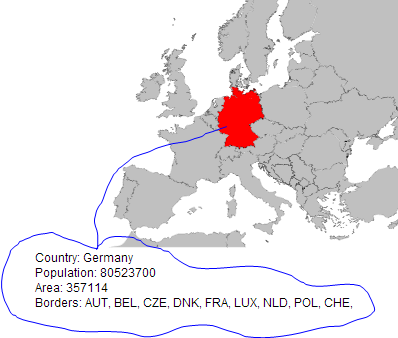
<http://store.company.com:81/dir/etc.html> Failure Different port

<http://news.company.com/dir/other.html> Failure Different host

**CA or Semester Project**

For a real exam exercise, this will be a small part where you are expected to talk, in about 5 minutes, about the semester project or one of the semester CA’s (related to the topic for this question).

**Practical part**

In this exercise, we will combine SVG with several of the topics we have been around this semester such as AJAX and REST-endpoints to obtain data, and Javascript for DOM-manipulation.

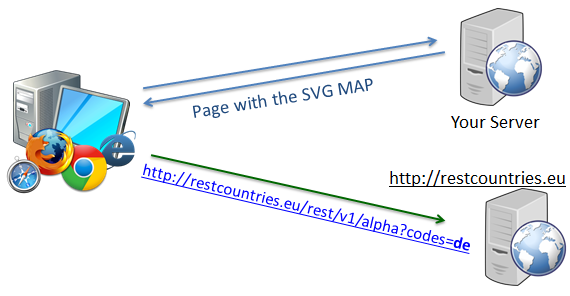
The task is to create a web-page with a map of Europe which, when a country is selected with a mouse click, should highlight the country and print details about the country as sketched below.

Get the map [Countries\_EuropAe.svg](https://github.com/Cphdat3sem2017f/StartcodeExercises/blob/master/JS/Countries_Europe.svg) and copy it into the clipboard. Create a new web project, include an html-file and paste the content into the body.

This is an SVG image where each country is given the ISO-country code as the id. This is very convenient, because using the public REST API given here: <http://restcountries.eu/rest/v1/alpha?codes=>[**de**](http://restcountries.eu/rest/v1/alpha?codes=de)

you will obtain a JSON encapsulated data package with all the information needed (+ a lot more) to fill out the details (just test the link above in a Browser).

1. So the exercise boils down to. Hook up an event handler on the map, get the id, perform an AJAX request to fetch the JSON-data from the link given above and update the GUI using the JSON returned as sketched above.



1. For the previous task it was possible to obtain data right from *restcountries.eu* via an AJAX call made from within your Browser (as sketched to the right). Use Chrome Developer tools to explain (with focus on the Same Origin Policy) why this is possible.
2. Let's assume restcountries.eu had not allowed Cross Origin Calls.

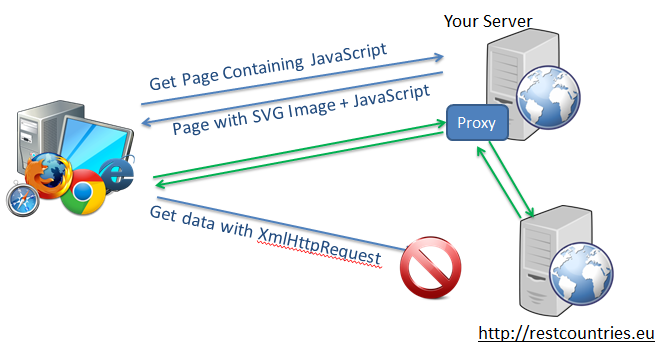
Design a Web Proxy Solution (using a plain Servlet or JAX-RS) where your browser will send the request to your proxy who should forward the request on to the remote server and send back the received response.

**Note: This is not a part of the "real" exercise. It's meant as FYI and this section will not be included with the real question.**

Hints:

1. Hook up a click-handler on the overall map (id =svg2), and in that, find the id for the actual element that was clicked (= the country code) via the target property of the event handler.
2. Change the colour of the selected country by changing its fill property

3)

***Same Origin Policy****: If your page, and the Service you try to call (via AJAX), are not located on the same Origin (see slides, and links on slides) the request is not permitted by the browser. The two solutions presented in the class (there are others, but focus on these) where either CORS, which is what made it possible for you to fetch data from within your browser, or to fetch the data from a Proxy on the* ***Origin*** *Server and provide it as a service from here .* 

*For part three you should use this second option as sketched in the figure above. Provide a Proxy-service on your own (origin) server, and make an http request from this to the remote server. Just return the result you get from this call in your own REST service.*

*The code below shows a simple way to perform a programmatically HTTP-GET request (requesting JSON) in Java.*

URL url = new URL("http://restcountries.eu/rest/v1/alpha...");

HttpURLConnection con = (HttpURLConnection)url.openConnection();

con.setRequestMethod("GET");

con.setRequestProperty("Accept", "application/json;charset=UTF-8");

Scanner scan = new Scanner(con.getInputStream());

String jsonStr=null;

if (scan.hasNext()) {

jsonStr = scan.nextLine();

}

scan.close();

System.out.println(jsonStr);