159352 Tutorial – Week 6

In this exercise we will play with XMLHTTPRequest (XHR) object, and explore XML document object model, eXtensible Stylesheet Transformations, and JavaScript Object Notation.

While you are free to install your own web server for this exercise, you will find that the versatility of the basic HTTP server (server.py from earlier tutorials) makes it is more than adequate for this tutorial.

Setup

1) Download the files (server.py, memberdata.xml, member.html, member.xsl, busstop.json, and busstop.html) into a single directory on your system
2) Execute the server program

```
# python server.py
```

3) Open your browser (preferably Chrome) and point it your web server address e.g. http://localhost:8080/member.html

Part A XML DOM

Inspect the memberdata.xml file which is a subset of the XML data about the membership of the House of Representatives in the US Congress obtained from http://clerk.house.gov/xml/lists/MemberData.xml

Examine the JavaScript code in member.html file, and see how the XMLHTTPRequest object is used to get the memberdata.xml file and render it using HTML DOM. You may find the XHR documentation handy.

With your web server running, point your browser to
http://localhost:8080/member.html

You will see a list of names of members of House of Representatives and their respective state. You will note that 56th element in the list is vacant. Your task is to fill the vacancy by modify the JavaScript in (member.html) as follows:

a) Clone the last member node (Walters, Maxine)
b) Change the name "Walters, Maxine" to "Smith, Jane" in all the relevant child elements of your cloned node.
c) Replace the vacant node with your cloned node.

Part B XSLT

The memberdata.xml carries a link to the member.xsl stylesheet document as shown below

```
<?xml-stylesheet type="text/xsl" href="member.xsl"?>
```

Point your browser to http://localhost:8080/memberdata.xml and you will see the XML data is rendered in a HTML table. Your task is modify the member.xsl file as follows:

a) Add a new column in the HTML table to display the phone numbers of members
b) Change the text colour of the members details to blue for democrats, and red for republicans.

Part C – XML vs JSON

Convert the following XML data into an equivalent JSON representation.

```
<member>

	<statedistrict>AK00</statedistrict>

	<member-info>
		<namelist>Young, Don</namelist>
		<bioguideID>Y000033</bioguideID>
		<lastname>Young</lastname>
		<firstname>Don</firstname>
		<middlename />
		<sort-name>YOUNG,DON</sort-name>
		<suffix />
		<courtesy>Mr.</courtesy>
		<prior-congress>114</prior-congress>
		<official-name>Don Young</official-name>
		<formal-name>Mr. Young of Alaska</formal-name>
		<party>R</party>
		<caucus>R</caucus>
		<state postal-code="AK">
			<state-fullname>Alaska</state-fullname>
		</state>
		<district>At Large</district>
		<townname>Fort Yukon</townname>
		<office-building>RHOB</office-building>
		<office-room>2314</office-room>
		<office-zip>20515</office-zip>
		<office-zip-suffix>0200</office-zip-suffix>
		<phone>(202) 225-5765</phone>
		<elected-date date="20161108">November  8, 2016</elected-date>
		<sworn-date date="20170103">January  3, 2017</sworn-date>
	</member-info>

	<committee-assignments>
		<committee comcode="II00" rank="2" />
		<committee comcode="PW00" rank="2" />
		<subcommittee subcomcode="II10" rank="2" />
		<subcommittee subcomcode="II24" rank="2" />
		<subcommittee subcomcode="PW05" rank="2" />
		<subcommittee subcomcode="PW07" rank="2" />
		<subcommittee subcomcode="PW12" rank="2" />
	</committee-assignments>

</member>
```

Part D – Parsing JSON

The JavaScript Object Notation (JSON) is a format for data exchange between client and server. JSON data is communicated over the network in text format. In JavaScript we use the JSON.stringify() function to convert a JavaScript object into text format for transmission. When JSON string text is received, we use JSON.parse() function to convert it to JavaScript object in order to be able to parse it.

Take a look at the busstop.json file, which shows the bus stop locations for Blenheim bus service (sourced from **https://beta.data.govt.nz/dataset/bus-stops2**)

Examine the JavaScript code in busstop.html, which uses to XHR object to get JSON data from the server. It is common for server side programs written in PHP to generate the JSON data on the fly and send it to the client. In this exercise the web server serves the JSON data from a static busstop.json file.

Point you're your browser to http://localhost:8080/busstop.html You will see a lonely page with text the "FeatureCollection" which is the value for the first (name, value) pair in the JSON response from the server. Your task is to modify the JavaScript in busstop.html file to display a pretty list of all BusStopLocations.

Submit the individual scripts for parts A-D on Stream.
Part A – member.html
Part B – member.xsl
Part C – member.json
Part D – busstop.html