LECTURE-3

- XML
- JSON
- Exceptions
- Debugging using Chrome Developer Tools
- JS Events

XML – EXTENDED MARKUP LANGUAGE

- XML is a markup language, like HTML
- Designed to carry data
  - Not to display data
- XML tags are NOT predefined.
  - Unlike HTML
  - You must define your own tags
- Self-descriptive
- Represented in plain text.

A SIMPLE EXAMPLE

```
<note>
    <to>Tove</to>
    <from>Jani</from>
    <heading>Reminder</heading>
    <body>Let's meet tomorrow!</body>
</note>
```

Summary:

1. User defined tags.
2. Self descriptive

JSON

- Text based
  - Very useful in transferring text data over the web
  - Language independent
    - Used in JS, Java, PHP, etc.
- Provides easy means to
  - Define JS objects
  - Can convert JS objects to strings and vice-versa
  - Different languages have functions for conversion.

JSON EXAMPLE

- var person =

```
{
"firstname":  "John",
"lastname": "Doe";
"age": 50,
"address":  {
        "street":"11 Broadway",
        "city":"New York City"
        }
};
```

- Can access data of individual fields
  - person.firstName (or) person[firstName]
  - person.lastName (or) person[lastName]
  - person.address.street(or) person.address[street]

JSON DATA TYPES

- A JSON object member can be of type
  - Number
  - String
  - Boolean
  - Null
  - Array
  - Another JSON object
    - Nested JSON objects

- Values of objects' members can be
  - Modified.
    - E.g., person.address[street] = "2 Columbia Way"
  - Deleted
    - E.g., delete person.age;

JSON DATA CONVERSION TO STRING

- JSON object to string conversion
  - var personString = JSON.stringify(person)
- JSON string to an JSON object
  - var person = JSON.parse(personString);
- Useful in sending JS objects over HTTP as strings.

JSON VS. XML

- Similarities
  - Self describing and text based.
  - Have user defined "tags" (unlike HTML)
  - Nested
  - Can be parsed in many languages
  - Can be fetched using XMLHTTPResponse (AJAX).

- Differences
  - JSON can be parsed by JS, XML can be parsed by XML parser
  - JSON does not have an end tag (e.g., NO </firstName>)
  - JSON can use arrays
  - JSON is less verbose

EXCEPTIONS

- Syntax and usage
  - Similar to Java/C++ exception handling

```
try
{
  // your code here
}
catch (exception)
{
  // handle error
  // optional throw
}
```

THROW

- Syntax
  - throw (exception)

Example

```
function( ) {
 // Some error condition
 throw (”<Some Exception String>”);
}
```

THROW EXAMPLE

```
<html>
 <body>
  <script>
    var x = prompt ("Enter a number between 0 and 10:","");
     try {
         if (x > 10)
            throw "Err1";
         else if (x < 0)
            throw "Err2";
      }
       catch (err) {
         if (err == "Err1")
            alert ("Error! The value is too high");
         if (err == "Err2")
            alert ("Error! The value is too low");
    }
  </script>
 </body>
</html>
```

TYPES OF EXCEPTIONS

- EvalError (old) — Error occurred in the eval function
- RangeError — Number out of range error
- ReferenceError — Illegal reference error
- SyntaxError — A syntax error
- TypeError — A type error
- URIError — encodeURI( ) function error

EVENTS

- Events in Javascript – "something" happening. Examples
  - Web page is loaded/unloaded
  - Mouse key clicked/double-clicked
  - Mouse hovering over/out-of a region
  - Any keyboard key is pressed/released
  - An error has occurred
  - A "submit" or "reset" button is pressed.
  - An element gets or loses focus.
- Complete list of events are given at

https://www.w3schools.com/js/js_events_examples.asp

EVENTS – ONLOAD( )

Called (if defined) when a web page is loaded.

Simple Example

```
<html>
  <head>
    <title> On Load event example </title>
    <script type="text/javascript">
    function onloadFn( )          // Function definition
     {
        alert ("Web page finished loading");
     }
    </script>
  </head>
  <body onload="onloadFn( )">// Call it when page is loaded
</body>
</html>
```

EVENTS – ONUNLOAD

- Opposite of onload
  - Called when
    - We go out of a web page or
    - A web page is re-loaded.
- Example
  - Same example as before
    - Except replace onload with onunload.

EVENTS – ONERROR

- Recall our example of error handling

Example:

```
onerror=handleErr // Call handleErr on errors

function handleErr(msg,url,l)
// msg – error msg, url – current URL, l – line #
{
   //Handle the error here
   return true or false
}
```

EVENTS – ONSUBMIT, ONRESET

- onsubmit – Event when a submit button is pressed.
  - E.g., When using forms.
- onreset – event when a reset button is pressed.
  - Typically, used to cancel/reset the values of all fields.

EVENTS – ONMOUSEUP, ONMOUSEDOWN, ONMOUSEOVER, ONMOUSEOUT

- onmousedown – event when a mouse button is pressed down.
- onmouseup – event when a mouse button is pressed up.
- onmouseover – event when a mouse hovers over (a specific region).
- onmouseout – event when a mouse comes out (of a specific region).

EVENTS – ONKEYPRESS, ONKEYDOWN, ONKEYUP

- onkeypress – Event when a key is pressed
- onkeydown – Event when a key is pressed or held down
  - Similar to onkeypress
- onkeyup – Event when a key is released (after being pressed)

EVENTS – ONCLICK, ONDBLCLICK, ONCHANGE

- onclick – event when a mouse button is clicked
- ondblclick – event when a button is double clicked
  - Try to avoid onclick when ondblclick is defined

EVENTS – ONFOCUS, ONBLUR, ONRESIZE

- onfocus – event when an element gets focus
- onblur – event when an element loses focus
  - Opposite of onfocus

EVENTS – ONRESIZE, ONCHANGE, ONABORT

- onresize – event when a browser window is resized (changed).
- onchange – event when the value of a field changes
- onabort – event when loading of an image is interrupted.

TIMING EVENTS

- setTimeout: execute something after a given time
  - Syntax: var t = setTimeout (code, time_in_msec);
  - Similar to sleep
  - Difference with sleep: Code "setTimeout" is executed immediately, no timeout there.
- clearTimeout: Cancel a timeout condition
  - Syntax: clearTimeout (t)
    - "t" was the variable returned by setTimeout