File and Network

Laurent Jouanneau
Course on Mozilla Education and Technologies @ Evry
December, 2011

XmlHttpRequest in documents

- Used to retrieve textual or XML content over http (or others protocols)
- Initialization:
  - Create a new xmlHttpRequest object
  - Set a callback to do things when the request is finished, on the onreadystatechange property
- Start the request
  - Open the connection with open(), by giving the url
  - Send the content with send() (POST parameters for example)
- Retrieve the content sent by the server: responseText or responseXml properties

XmlHttpRequest in documents

- ```
  var req = new XMLHttpRequest();
  req.onreadystatechange = function () {
      if (req.readyState == 4) {
          if(req.status == 200)
            alert(req.responseText);
          else
            alert("Error loading content\n");
      }
  };
  ```

- GET method

```
req.open('GET', 'http://xulfr.org/', true);
req.send(null);
```

- POST method

```
req.open('POST', 'http://monsite.tld/mon_script.php', true);
req.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
req.send("param1=val1&param2=val2");
```
```

XmlHttpRequest: progression

- Instead of onreadystatechanged, use event listeners
- For Firefox 3.0-

```
req.onprogress = function(evt) {
    var percentComplete = (evt.position / evt.totalSize)*100;
}
req.onload = function(evt) { alert("The transfer is complete.");}
req.onerror = function(evt) { alert("An error occurred.");}
```

- For Firefox 3.5+

```
function onProgress(evt) {
    if (evt.lengthComputable) {
        var percentComplete = evt.loaded / evt.total;
        // ...
    }
}
function onLoad(evt)  { alert("The transfer is complete.");}
function onError(evt) { alert("An error occurred while transferring the file.");}
function onAbort(evt) { alert("The transfer has been canceled by the user.");}

req.addEventListener("progress", onProgress, false);
req.addEventListener("load", onLoad, false);
req.addEventListener("error", onError, false);
req.addEventListener("abort", onAbort, false);
```

XmlHttpRequest in XPCom

- XmlHttpRequest is available as an XPCOM component

```
var req = Components.classes["@mozilla.org/xmlextras/xmlhttprequest;1"]
                    .createInstance(Components.interfaces.nsIXMLHttpRequest);
```

Files

- A nsIFile object should be used to handle a descriptor, the path, of the file
- It allows to abstract differences between plateforms
- It is used in many components, and can be retrieve from several other components.

```
var file = Components.classes["@mozilla.org/file/local;1"]
                .createInstance(Components.interfaces.nsILocalFile);
file.initWithPath(aPath); // warning, system dependant

file.append("a");
```

The directory service

- It allows to retrieve the path (or a file descriptor) of a specific directory
- The « alias » of the directory could be:
  - ProfD = the user's profile directory
  - CurProcD = application directory
  - Home = the user's home
  - TmpD = system tmp directory

```
var file = Components.classes["@mozilla.org/file/directory_service;1"]
                     .getService(Components.interfaces.nsIProperties)
                     .get("ProfD", Components.interfaces.nsIFile);
// file is a nsIFile object
var thePath = file.path;

//or

Components.utils.import("resource://gre/modules/Services.jsm");
var file = Services.dirsvc.get("ProfD", Components.interfaces.nsIFile);
```

FileUtils.jsm

- Since Firefox 3.6 (gecko 1.9.2), provides helpers
- ```
  Components.utils.import("resource://gre/modules/FileUtils.jsm");

  var file = new FileUtils.File(aPath);  // warning, system dependant

  file.append("a");
  ```
- Getting a file in a known directory

```
var file = FileUtils.getFile("ProfD", ["sub","path","myfile"],true) ;
// file is a nsIFile object
var thePath = file.path;
```
```

Reading an XML file

- Steps:
  - Create a file descriptor
  - Open an input stream
  - Use a DOM parser to create the DOM from the input stream

```
var file = Components.classes["@mozilla.org/file/local;1"]
                     .createInstance(Components.interfaces.nsILocalFile);
file.initWithPath(aPath);

var stream = Components.classes["@mozilla.org/network/file-input-stream;1"]
                       .createInstance(Components.interfaces.nsIFileInputStream);
stream.init(file, -1, -1, Components.interfaces.nsIFileInputStream.CLOSE_ON_EOF);

var parser = Components.classes["@mozilla.org/xmlextras/domparser;1"]
                       .createInstance(Components.interfaces.nsIDOMParser);

var doc = parser.parseFromStream(stream, null, file.fileSize, "text/xml");
```

Writing an XML file

- Steps:
  - Create a file descriptor
  - Open an Output stream
  - Serialize the XML content to the output stream

```
var file = Components.classes["@mozilla.org/file/local;1"]
            .createInstance(Components.interfaces.nsILocalFile);
file.initWithPath(aPath);

var stream = Components.classes["@mozilla.org/network/file-output-stream;1"]
            .createInstance(Components.interfaces.nsIFileOutputStream);

stream.init(file, -1, -1, 0);

var encoder = Components.classes["@mozilla.org/layout/documentEncoder;1?type=text/xml"]
               .createInstance(nsIDocumentEncoder);
encoder.init(aDomDoc, "text/xml", 0);
encoder.encodeToStream(stream);
```

IO Service

- The IO Service allows to:
  - Create a nsIURI object from a nsIFile or an Url string
  - Create channels to read or write over a network protocol

```
var ioService = Components.classes["@mozilla.org/network/io-service;1"]
                          .getService(Components.interfaces.nsIIOService);
// or
Components.utils.import("resource://gre/modules/Services.jsm");
var ioService = Services.io;


var uri = ioService.newURI("http://xulfr.org/fichier.txt", null,null);
var uri2 = ioService.newFileURI(myNsIFileObject);

var channel = ioService.newChannelFromURI(uri);
```

Simple Http request using streams

```
var ioService = Components.classes["@mozilla.org/network/io-service;1"]
                        .getService(Components.interfaces.nsIIOService);

var uri = ioService.newURI("http://xulfr.org/fichier.txt", null,null);

var channel = ioService.newChannelFromURI(uri);

var listener = {
    data : "",

    onStartRequest : function(aRequest, aContext) {

    },
    onStopRequest : function(aRequest, aContext, aStatusCode) {

    },
    onDataAvailable : function(aRequest, aContext, aInputStream, aOffset, aCount) {

    }
}

var context = null;
channel.asyncOpen(listener, context);
```