JavaScript in Browsers

These slides have been prepared by Cyril Concolato and Jean-Claude Dufourd.

pdf

JavaScript vs. ECMAScript

- What is ECMAScript ?
  - Programming/Scripting Language
  - Interpreted code (not compiled into machine code), Portable code
  - Standard syntax
  - Invented by Brendan Eich at Netscape (and Microsoft JScript)
- Versions
  - JavaScript 1.5-2.0
  - ECMA-262 3rd, (4th), 5th, 6th (2015), 7th edition (draft)
- In the Web Browser : JavaScript
  - Executed by the JavaScript engine of the browser according to a model
  - Used with specific interfaces (DOM, . . .)
- More : Tutorial Videos by Douglas Crockford

JavaScript Basics

Reminder of pure JavaScript

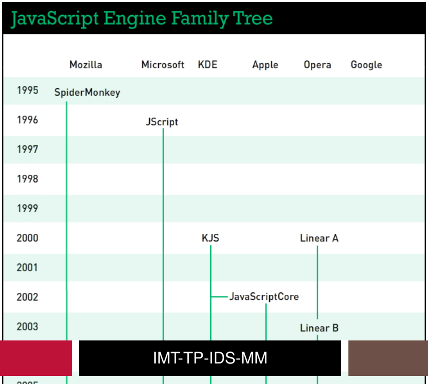
- How to declare/assign a variable ?
- How to define a function ?
- How to call a function ?
- Arrays
- Strings
- Objects
- Properties

Browsers and JavaScript

- The JavaScript Engine is a core component of browsers
  - Used for :
    - Interactivity, animations, media manipulations (Canvas, audio API, . . .)
  - Potential problems
    - Security
    - Performance

JavaScript Engines race

- IE9 Chakra, Opera Carakan, Safari Nitro, Firefox JägerMonkey, Chrome V8
- Benchmarks :
  - SunSpider
  - V8 Benchmark
  - Octane
- Optimizations : JIT, dead-branch ...



Web Applications=

HTML +

- Document structure
- Textual content and media resources (images, . . .)

CSS +

- Presentation information

JavaScript (=ECMAScript + Web APIs)

- Browser-interpreted code to provide the intelligence, behavior of the application

“There is a Web API for everything”

- Basic APIs
  - Document Object Model (DOM) : Core, Events, Window, . . .
- Specific APIs
  - Communication APIs
    - XHR, Push, WebSockets, . . .
  - Drawing APIs
    - Canvas, WebGL, . . .
  - Storage APIs
    - Files, Cookies, Database, . . .
  - Multimedia APIs
    - Audio, video, streaming, . . .
  - Device APIs
    - Battery, AdressBook, WebCam . . .
  - System APIs

Document Object Model (DOM) Interfaces

Interfaces to the document tree

- For access and modifications of content, structure, and style of documents

Specifications

- Level 1 (one single specification)
- Level 2 (6 specs) : Core, Style, (Views), . . .
- Level 3 (3 specs) : Core, . . .
- Level 4

DOM Interfaces Hierarchy

../tp ../logo-IPP-s

DOM Interfaces : methods and properties

- The Node interface

```
nodeType
parentNode
firstChild
nextChild
firstSibling
hasChildNodes()
hasAttributes()
appendChild()
removeChild()
```

- The Document interface

```
documentElement
getElementById()
getElementsByTagName()
querySelector()
```

../tp

../logo-IPP-s

The Window Object

API corresponding to the browser window or tab

Convenient API for various usages

- Timing (animations)
- General events (load, . . .)
- Navigation (history)
- Embedding (openURL)

JavaScript `global` object in browser

The Window Object

Examples of DOM manipulations in JS : Add an element

The page before

```
<html>
  <body>
  </body>
</html>
```

The JS code

```
var obj = document.createElement("p");
obj.textContent="some new text";
var body = document.getElementsByTagName("body")[0];
body.appendChild(obj);
```

The page after

```
<html>
  <body>
    <p>some new text</p>
  </body>
</html>
```

HTML editing

The page before

```
<html>
  <body>
    <p id="someid">some new text</p>
  </body>
</html>
```

The JS code

```
var obj = document.getElementById("someid");
obj.innerHTML = "some <span style='color: red;'>other</span> tex
```

The page after

```
<html>
  <body>
   <p id="someid">some <span style="color: red;">other</span> te
```

Working on attributes

The page before

```
<html>
  <body>
    <p id="someid">some new text</p>
  </body>
</html>
```

The JS code

```
var body = document.getElementsByTagName("body")[0];
body.onload="myfunction()";
var obj = document.getElementById("someid");
obj.setAttribute("align", "center");
```

The page after

```
<html>
```

Remove elements

The page before

```
<html>
  <body>
    <p id="someid">some new text</p>
  </body>
</html>
```

The JS code

```
var body = document.getElementsByTagName("body")[0];
var obj = document.getElementById("someid");
body.removeChild(obj);
```

The page after

```
<html>
  <body>
```

CSS and JavaScript

- The JavaScript style property
  - Used to set a new style on an element
  - Used to query the style on this element

```
var e = document.getElementById("SomeElementId");
e.style.top = 10px;
```

- The `getComputedStyle()` method
  - To ask for all styles (inherited, computed, . . .) of an element

```
var e = document.getElementById("SomeElementId");
var style = window.getComputedStyle(e);
var height = style.getPropertyValue("height");
```

../tp ../logo-IPP-s

Script processing in HTML

- Ways to use JS

```
<script>var x=0;</script> // inline code

<script src="file.js"></script> // external code

onload="doSomething();" // inline code
```

- One JavaScript global context per document (i.e. per HTML source)
  - Shared variables, shared functions
  - Ability to split the code into multiple files, to create modules

Script processing in HTML 2

- Code execution
  - Many operations can run in parallel in a browser (HTML parsing, CSS parsing, JS, rendering ...)
  - By default, JS processing is run in the main thread and synchronously :
    - download and execution blocks the rest of the parser
    - the code is interpreted as soon as it is read in `<script>` except if `async` or `defer` attributes are used

```
<script src="file3.js" async></script> // script will be ex
<script src="file1.js"></script>
<script src="file2.js"></script>
```

  - when events are triggered : “run-to-completion” approach (“script is taking too long” pop-up)
  - may be blocked when style sheets are being processed
- Where to put the `<script>` elements (head, bottom, middle) ?

JSON – JavaScript Object Notation

- Format for exchanging data
  - Text based
  - Structured
  - Easy serializing/parsing
- Based on JavaScript
- Literal notations
  - Object {}
  - Array []
  - String ""

JSON - Example

```
{
  "firstName": "John",
  "lastName": "Smith",
  "age": 25,
  "address": {
    "streetAddress": "21 2nd Street",
    "city": "New York",
    "state": "NY",
    "postalCode": 10021
  },
  "phoneNumbers": [
    { "type": "home", "number": "212 555-1234" },
    { "type": "fax", "number": "646 555-4567" }
  ]
}
```

JSON vs. XML

```
<person>
 <age>12</age>
 <name>Danielle</name>
</person>
```

```
{
  "age" : 12,
  "name" : "Danielle"
}
```

JavaScript libraries

- Principles
  - Simplify the JS code written by Web Developers
  - Provide a unique interface for all browsers (bugs)
- Many librairies
  - JQuery,
  - Angular,
  - Bootstrap . . .
- JavaScript “beautifier”/“minifier”

Scripted animations

- Use of timers and callback functions
  - Ex : using the `window` object
  - Ex : using an `SVGTimer` object
  - Ex : using `requestAnimationFrame`
- Management of the synchronization by the script

Animations with JS

```
<rect id='R' width="120" height="50" fill="blue">
<script>
function doAnimation(){
  var rect=document.getElementById('R');
  x=x+xincr;
  rect.setAttribute('x', x);
  window.setTimeout("doAnimation()", 10);
}
doAnimation();
</script>
```

```
function animloop() { // function to be called
  render();
  requestAnimFrame(animloop);
}
requestAnimFrame(animloop);
```

Interactivity & Scripting

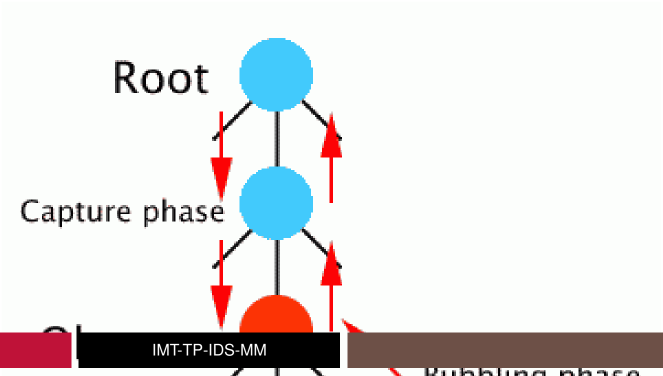
- Simple interactivity does not require scripting
  - Forms filing and submitting
  - Navigation
  - Triggering animations or transitions
  - . . .
- More complex interactions require Javascript with
  - DOM events
  - AJAX Pattern

DOM Events

API to indicate to the browser how to process events in JavaScript

Based on a specific Event Propagation model

- Capture phase, target phase, bubbling phase
- Cancellation of events,
- Default action



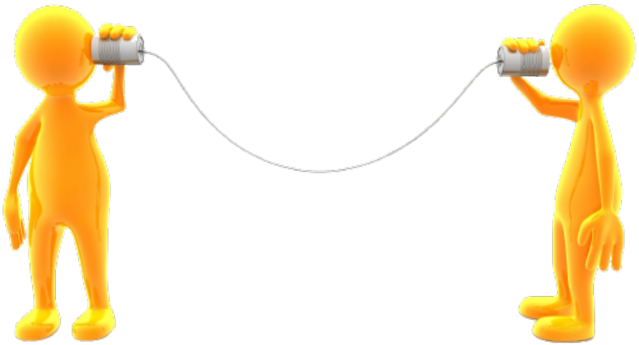
Examples of DOM Events code

```
<script type="application/ecmascript" >
  function doSomething(evt) { . . . }
</script>
<text onclick=“doSomething(evt)” >Hello World!</text>


<script type="application/ecmascript" >
 function doSomething(evt) { . . . }
 e=document.getElementById(‘T’);
 e.addEventListener(‘click’, doSomething, false);
</script>
<text id=“T” >Hello World!</text>


<script type="application/ecmascript" >
 function doSomething(evt) { . . . }
 e=document.getElementById(‘T’);
 e.onclick=doSomething;
</script>
```

DOM Event types

- Mouse Events :
  - click, mousedown, mouseup, mouseover, mousemove, mouseout
- Key Events :
  - keypress, keyrelease
- Touch events :
  - touchstart, touchend, touchleave, touchmove, . . .
- Drag events :
  - dragstart, dragend, . . .
- Network events :
  - load, error, abort, progress
- Form events :
  - submit, focus . . .
- Media events :
  - play, pause . . .

Web Pages can communicate



Web API for Communications

The `XMLHttpRequest` object

- HTTP communication with a server
- Core of the AJAX programming model

The `WebSocket` Interface

- Lightweight communication with servers
- Companion with IETF WebSocket protocol
- Upgrade from HTTP but different from HTTP

Server-Sent Events

- Used to deliver messages in push mode (notifications)

Web Messaging

- Messaging between Javascript contexts in the same browser (pages, workers)

WebRTC

AJAX “Asynchronous JavaScript and XML”

Used to make asynchronous HTTP requests and retrieve data (e.g. text, XML, binary . . .)

Combined usage of different technologies

- HTML (or SVG, . . .)
- ECMAScript
- XML (or JSON, . . .)
- HTTP Download

Example : HTML/SVG + JSON + DOM + XMLHttpRequest

Benefits

- Requests are asynchronous to the rendering
  - Avoids waiting for the response to further interact
- Enables client-side heavy interactivity
  - Data base requests and response handling

AJAX Example

```
var xhr = new XMLHttpRequest();
xhr.open("GET", "test.txt");
xhr.onload = function() {
  alert(this.responseText);
}
xhr.send();
```

WebSocket Example

```
const socket=new WebSocket('ws://example.com:12010/');

socket.onopen=function () {
 setInterval(function() {
     if (socket.bufferedAmount==0) {
         socket.send(getUpdateData());
     }
   },
   50);
};

socket.onmessage=function (evt) {
 const received_msg=evt.data;
 alert("Message is received...");
};

socket.onclose=function() {
 // WebSocket is closed.
```

Web Workers

- Equivalent to threads in JavaScript (without shared memory)
- Used for long-running scripts, in background, in parallel on multi-core CPU
- Thread and Messaging across scripts

```
var worker=new Worker('worker.js');
worker.onmessage = function (event) {
  document.getElementById('result').textContent=event.data;
};
var otherWorker = /* findotherworker */;
otherWorker.postMessage("A message");
```

File & Web Storage

Web Storage

Part of HTML5 Scope

Similar to HTTP Cookies mechanism with extensions

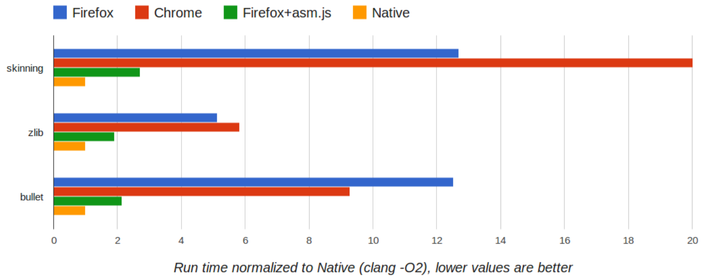
- Persistent Storage of structured data at the client side
- Cross window storage for the same site
- Larger storage capacity than cookies

File API

- Handling files, directories, file systems in browsers taking security issues into account
- API : FileReader, FileWriter

JavaScript Compilation

- Google Web Toolkit (GWT) : from Java to Javascript
- Emscripten : from C/C++ to Javascript
- Asm.js : restricted JS with improved execution speed



Summary of this lesson

- JS engine in the browser, JS APIs, Web Apps
- DOM, window, HTML editing, script tag, JSON
- JS libraries, animation, events, communication