

JavaScript

MPRI 2.26.2: Web Data Management

Antoine Amarilli

Friday, December 7th



Motivation

- **Dynamic changes** on the client without refreshing the page
- **Simple cases**: form validation, menus, etc.
 - going **beyond** what HTML and CSS allow
 - being **compatible** with old browsers
- **Complex cases**: writing entire in-browser applications in JS (aka **single page applications**, SPA)
- The language of the Web is **JavaScript** (ECMAScript standard)
 - Beware, it's unrelated to **Java**!

Table of Contents

Introduction

JavaScript

JavaScript and HTML

AJAX

Front-end Javascript frameworks

Old and new technologies

The JavaScript language

- **ECMAScript**: name of the standard (ECMA-262, 2011, 258 pages)
- JavaScript is a **full-featured** programming language!
- It is **interpreted** (traditionally) and high-level
- **Dynamic** typing
 - at runtime
 - typing of values, not of variables (like Python)
 - implicit **conversions** are very flexible (too flexible?)
- Dynamic code execution with **eval**

Evolution of JavaScript

- 1999: ECMAScript 3, addition of **regexps**, **try/catch**
- 2009: ECMAScript 5, addition of **strict mode**
- ES2015: addition of **classes**, **iterators**, **generator expressions**, **promises**, **arrow functions**
- ES2017: ECMAScript 8, adds **atomics**, **async/await**, etc.
- ES2018: asynchronous loops, promises extensions, regexp extensions (e.g., named capture groups)
- **Polyfills**: implement new features in old language versions for legacy browsers
- **Style guides** for JavaScript, e.g., Google¹, AirBNB²

¹<https://google.github.io/styleguide/javascriptguide.xml>

²<https://github.com/airbnb/javascript>

JavaScript language (cont'd)

- **Object-oriented** language, with **prototypes** (cloning existing objects) rather than classes
- **First-class** functions (anonymous functions, closures; functions are also objects)
- Support for **exceptions**
- **Object syntax:**
 - `foo.a` for member `a` of object `foo`
 - `foo.b(arg)` to call method `b` of `foo` with argument `arg`

JavaScript example

```
function factorial(n) {  
    if (n === 0) {  
        return 1;  
    }  
    return n * factorial(n - 1);  
}
```

JavaScript engines

IE Chakra, open-sourced in 2016, since IE9

Firefox SpiderMonkey, free and open-source, released in 1996

Chrome V8, released with Chrome in 2008

→ Also used by Opera and Node.js

Safari JavaScriptCore aka Nitro (2008)

→ Active research area to improve **performance!**

→ **Just-in-time compilation** ; not just interpreted

→ Some Javascript fragments are heavily optimized and used as compilation targets (`asm.js`, WebAssembly, ...)

Newer Javascript dialects

- **TypeScript**: Javascript with optional static typing
 - The main language of the **Angular** framework
- Also: **CoffeeScript**: adds syntactic sugar (matching, etc.)
- These languages can be **transpiled** to Javascript

Table of Contents

Introduction

JavaScript

JavaScript and HTML

AJAX

Front-end Javascript frameworks

Old and new technologies

Integrating Javascript in HTML

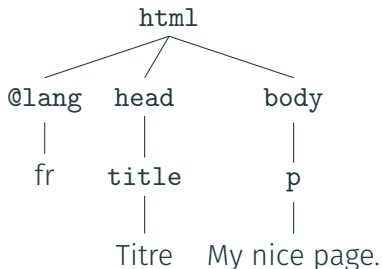
- A bit like **CSS**:

```
<head>  
  <script src="script.js" type="text/javascript">  
  </script>  
</head>
```

- **External dependency**, and **security risk** with third-party codes!
 - The code can also go **directly in** `<script>`
 - **Events** like `onClick` (explained later)
 - Also **links** starting with `javascript:`
- Beware, some browsers do not support Javascript! (e.g., robots)
It's better not to require it if possible

Document Object Model

```
<html lang="fr">
  <head>
    <title>Titre</title>
  </head>
  <body>
    <p>
      My nice page.
    </p>
  </body>
</html>
```



The document object

The `document` object represents the **current document**:

`document.documentElement` Get the **root** node

`document.getElementById("foo")` Get the node with attribute `id="foo"` (or `null` if none exists)

`document.getElementsByTagName("p")` Return a table of all `<p>` nodes

Node objects: basics

The `Node` objects represent **nodes** in the DOM tree:

`node.nodeName` Name of the element (BODY...) or attribute

`node.nodeType` Type, e.g., `Node.ELEMENT_NODE`, or `TEXT`, `ATTRIBUTE`, `COMMENT...`

`node.nodeValue` Value of text and attribute nodes (also:
`node.setAttribute("name", "value")` and
`node.getAttribute("name")`)

`node.className` Node class

`node.style` CSS style with the various properties (with dashes replaced by `camelCase`). For instance:
`node.style.borderStyle = "solid"`

Node **objects**: traversal

`node.parentNode` The parent node

`node.childNodes` Array of child nodes (or `null` if none)

`node.appendChild(child)` Add a child (at the end)

`node.removeChild(child)` Remove a child

`node.cloneNode(true)` Clone the node and its descendants

- The new clone is not yet added to the document
- Beware of duplicate `ids`!

`node.cloneNode(false)` Clone the node (but not its descendants)

`node.innerHTML` Get/set an HTML representation of the node contents

The `window` object

By default, the **methods** of `window` are in global scope:

`alert("x")` Message dialog showing "x"

`back()` Go to the previous page

`a = open("URL", "name")` Open a window (or tab) on "URL" with name "name"

`t = setTimeout("f()", 42000)` Call `f()` in 42 seconds, e.g.,

- for **animations**
- for **polling**

Events

- **Attribute** `onfoobar="return f(this)"` on `x`:
 - When `foobar` happens on element `x`, call `f(x)`
- **Many events**. The most useful:
 - `onclick` When a click happens; return `False` to cancel
 - `onchange` When the contents of a field change
 - `onfocus` When an element gets focus
 - `onsubmit` When a `<form>` is submitted; cancellable
 - `onload` On `<body>`: when the page is done loading
 - `onmouseover` When the element is hovered
 - `onresize` When the window is resized

Example: form validation

```
function vrf() {
  v1 = document.getElementById("pw1").value;
  v2 = document.getElementById("pw2").value;
  if (v1 != v2) {
    window.alert("The passwords don't match!");
    return false;
  }
}

<form action="test.php" method="post" onsubmit="return vrf()">
  <input type="text" name="name" placeholder="Name" required><br>
  <input type="password" name="pw1" id="pw1"
    placeholder="Password" required><br>
  <input type="password" name="pw2" id="pw2"
    placeholder="Retype password" required><br>
  <input type="submit">
</form>
```

Table of Contents

Introduction

JavaScript

JavaScript and HTML

AJAX

Front-end Javascript frameworks

Old and new technologies

Asynchronous queries

- JavaScript can also do **HTTP queries**
- **AJAX**: Asynchronous JavaScript And XML
- Exchange data with the **server** without reloading
- **Asynchronous** means that queries are not blocking
- Introduced in **Internet Explorer 5** in 1999, standardised by W3C (Working Draft, 2012)
- **Security** implications:
 - **Same-origin policy**: you cannot query a different domain (otherwise, danger!)
 - Special mechanisms for exceptions: `postMessage` between clients, Cross-Origin Resource Sharing

Example XMLHttpRequest

```
// Caution, not compatible with old IE versions
var req = new XMLHttpRequest();

req.onreadystatechange = function() {
  if (req.readyState === 4) { // is it ready?
    if (req.status === 200) { // did it go well?
      window.alert("Received: " + req.responseText);
    } else {
      window.alert("Problem with the query");
    }
  }
}

// true to be asynchronous, t to avoid caching
var t = new Date().getTime();
req.open("GET", "data.xml?t=" + t, true);
// can provide POST content (which you must encode)
req.send(null);
```

- Generally, AJAX is not used to retrieve a **Web page** but to retrieve **XML** or **JSON**
- The Javascript code will **parse** it and update the page accordingly
- See **next week** for details about XML and JSON

Pushing data from the server

- AJAX allows us to query the server, but doesn't allow the server to **push** data
- Legacy solutions:
 - **Polling**: query periodically (latency, wasteful)
 - **Comet**: make a request and wait for an answer, the server waits until something new has arrived
- Current solution: **WebSocket** IETF, RFC 6455, 2011
 - **Upgrade** the HTTP connection to be bidirectional
 - Backwards-compatibility is **tricky** (proxies, caches...)
 - Javascript library: **socket.io**
- Alternative solution: **Server-Sent Events** (not in IE)

Table of Contents

Introduction

JavaScript

JavaScript and HTML

AJAX

Front-end Javascript frameworks

Old and new technologies

- Released in **2006**
- About **87 kB**
- Free and open-source (MIT)
- Used by **97%** of the most visited websites.³
- Makes it **simpler** to write Javascript
- **Abstract away** some browser quirks

³https://w3techs.com/technologies/overview/javascript_library/all

Example jQuery

```
<script type="text/javascript" src="jquery-3.3.1.min.js">
</script>
<script type="text/javascript">
  $.ajax({
    url: "data.json",
    cache: false,
    success: function (data) {
      $( "#load" ).html( data.load );
      $( "#speed" ).html( data.speed );
    }
  });
</script>
```

General utility libraries

- **Backbone.js**, lightweight frontend JS framework
- **Underscore.js**, utility library for **Backbone.js**
- **Lodash**, successor of Underscore.js
- etc.

- **AngularJS** (version 1): a Javascript library
- **Angular** (version 2): a platform based on **TypeScript**
- **Basic idea:** update the view in a **declarative** way by specifying the link between the model and the view:
 - **one direction:** when the value changes, update the view
 - **other direction:** when the view is changed by the user, update the value

Templates

- **Idea:** instantiate HTML snippets with variable values provided by Javascript
- Several languages, e.g., **Mustache**, **Handlebars**,

```
<script id="entry-template" type="text/x-handlebars-template">
  <div class="entry">
    <h1>{{title}}</h1>
    <div class="body">
      {{body}}
    </div>
  </div>
</script>
```

React.js (Facebook)

- Allows you to create HTML **directly in Javascript** with JSX

```
function getGreeting(user) {  
  if (user) {  
    return <h1>Hello, {formatName(user)}!</h1>;  
  }  
  return <h1>Hello, Stranger.</h1>;  
}
```

- **Virtual DOM:** don't manipulate the DOM (slow) but a copy of it, and do a **tree diff** to update the real DOM
 - **Data flow** with **Redux**
- More alternatives: **Vue.js**, **Ember.js**

Widget libraries

- Collections of widgets: **jQuery UI**, **jQuery Widgets**, **Polymer**, **Angular Material**, etc.
- Rich text editing: **TinyMCE**
- Data visualization and plotting: **D3.js**
 - Bind data to visualizations
 - Define rules to update the view when elements are updated/added/deleted
- Also: **Web Components** (in-browser)
 - Support for **templates** and **includes**
 - **Shadow DOM** to limit the scope of embedded components

Development tools

- **Developer tools** in Web browsers
- **JSFiddle**: share and test Javascript snippets
- **Github gists**

Table of Contents

Introduction

JavaScript

JavaScript and HTML

AJAX

Front-end Javascript frameworks

Old and new technologies

Old technologies (not supported on mobile)

- **Flash:** released in **1996** by Macromedia (bought by Adobe), used for videos, video games, etc.
 - Still used on **7%** of the most popular websites⁴
 - Supported by **95 à 99%** of **desktop** clients in 2012 ⁵
 - No support on **mobile**, deprecated in **2017**
- **Java applets:** released in **1995**, rich applications but no interaction with the rest of the page: only on **< 0.1%** of the most popular websites⁶
- Old **Microsoft technologies:** ActiveX, Silverlight, etc.

⁴https://w3techs.com/technologies/overview/client_side_language/all

⁵Source: *The Tangled Web*, chapter 8, 2012

⁶<https://w3techs.com/technologies/details/cp-javaruntime/all/all>

New technologies

`<canvas>` Page region with efficient **bitmap drawing** in Javascript, useful for **video games**

SVG+JS Javascript manipulation of **vector graphics**

WebGL Javascript API for **accelerated 3D** using a GPU

WebVR Describe **scenes** for virtual reality headsets with `a-scene`

WebRTC Voice and video communication across browsers

Offline Web applications that can be used offline

Web development outside the Web

- Creating **Web applications** and porting them to other **platforms**
 - **Apache Cordova** for mobile: embed the app in a **Web View**
 - Also: **React Native**
 - **Firefox OS** (abandoned)
 - **Electron**: bundle the app with Node.js and Chromium
- Advantage: portability (don't need to develop multiple apps)
- Drawback: performance

- Matériel de cours inspiré de notes par Pierre Senellart
- Merci à Pablo Rauzy et à Pierre Senellart pour leur relecture