metaprogramming for the Web?

(newbie intro to OSIS2016 **“state of the art web technologies”** workshop)

Basile STARYNKEVITCH
`gcc-melt.org` and **`starynkevitch.net/Basile/`**
`basile@starynkevitch.net` or `basile.starynkevitch@cea.fr`

CEA, LIST (Software Reliability Lab.), Palaiseau, France
[within Université Paris Saclay]

June 28th, 2016,
Open Source Innovation Spring, Jussieu, Paris, France

Overview

All **opinions are mine only**

Introduction (audience)

Expected **audience** (OSIS2016) :

- **developers** curious of Web technologies
- ***web developers*** curious on non-mainstream Web technologies
- **free-software friendly** and knowledgable

Why am *I* (Basile) interested by web technologies?

- I am a compiler & static source code analysis guy (**`gcc-melt.org`**), very far from the Web!
- Web technologies are about **half** of IT economy
- static source analysis tools **need powerful user interfaces** (because they compute a *lot* of things)
- GUI toolkits (e.g. Qt) are becoming *out of fashion*
- many companies dislike installing software on their own developer's laptops, prefering Web applications (running elsewhere)
- I feel the need for a persistent tool with a web interface ("MELT monitor", for a *small* team working on the same software)
- **I am still learning**, and a Web newbie!
- I am **old** enough (born in 1959, grandfather) to not have learned anything about the Web
- I have lots of questions! (perhaps better answered offline)

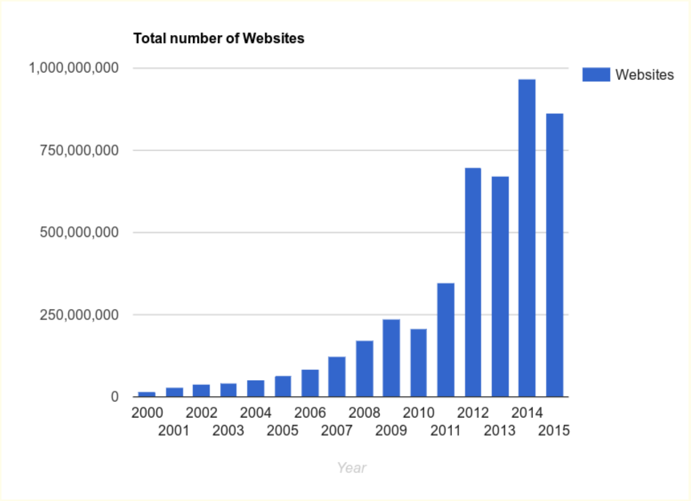
History and evolution of the *World Wide Web*

Cf `evolutionoftheweb.com` & wikipage **History of the World Wide Web**

- *Tim Berner Lee*'s pionnering work at CERN: 1980 - **1990**: web for helping scientists to work together (distributed hypertext)
- **HTTP** (HyperText Transfer Protocol), a textual client-server protocol (HTTP 0.9 in 1990, HTTP 1.1 in 1997-**1999** *RFC2616*, HTTP 2 in **2015** for 8% of websites today).
- **HTML** (HyperText Markup Language, inspired by SGML), evolved as HTML4 (1997) & HTML5 (2007) with Canvas, Video, etc...
- **URL**s (Uniform Resource Locator, RFC1738 in 1994)
- **CSS** (Cascading Style Sheet, 1996; CSS3 modules, 2011; CSS4 WIP)
- **JavaScript** (developed in *10 days !* in may 1995, by B.Eich at Netscape); now standardized as **ECMAScript 7** (June 2016) & **DOM** (document object model, DOM1 by W3C 1998, DOM4 by WHATWG, 2015)
- **AJAX** (asynchronous JavaScript & XML, Feb. 2005; precursor ActiveX at MicroSoft -1999- & `XMLHttpRequest` in Gecko -2000-)
- **WebAssembly** (2016, inspired by `PNaCl`, Google, 2011)

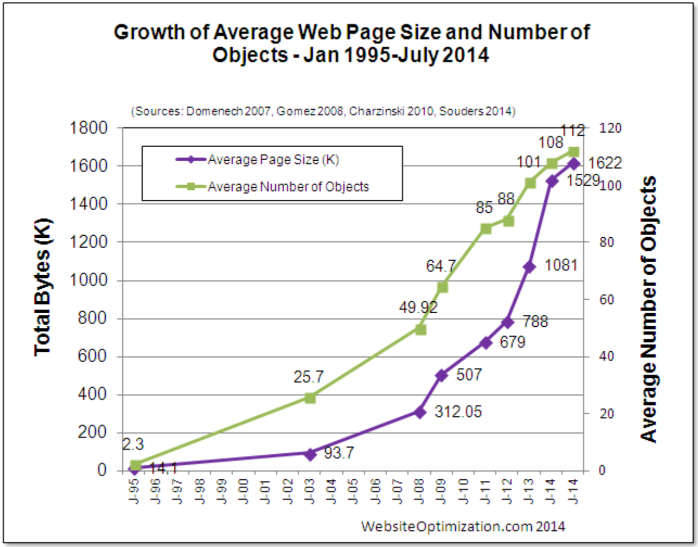
Web technologies evolved *organically*, no "grand design" (economical & social pressure), so **complex** & **layered**

evolution of the number of websites



From `http://www.internetlivestats.com/total-number-of-websites/`

growth of web page size & number of objects



Web technologies “outside” of, or “close” to, the Web

- **web indexing**
- **hidden** web (see `robots.txt` “standard”)
- **embedded** web servers: user interfaces of printers, routers, . . . ; free software HTTP server libraries (**libonion**, **Wt**, **ocamlnet**, etc . . .)
- **mobile** web
- **web services** - using **SOAP** (Simple Object Access Protocol, W3C, 2000), often **REST**-ful (Representational state transfer, R.Fielding 2000)
- **JSON** (a notation, 2001, used for Ajax) & **JSON-RPC**
- embeddable HTTP clients (`libcurl`...) & browsers (`QtWeb`, `WebKit`...)
- `FastCGI` to connect a software to a running web server
- Proxies & **ICAP** (Internet Content Adaptation Protocol, 1999)

Generating HTML

- **PHP** (R.Lerdorf, 1994; PHP7: 2015, faster!): dynamically typed interpreter
- and many **other tools**, sometimes generating HTML in batch
  - **`HeVeA`**, a LaTeX → HTML5 converter
  - zillions of HTML **generators**
  - many HTML **editors**
- several libraries, e.g. **FreeMarker** (for Java) etc, & “template engines”
- some interesting but dead languages: **Kaya**, etc ...

But what about **JavaScript**?

Issues in JavaScript

Today (2016) JavaScript is [nearly] **the *only* way to code for the browser**.
(but JavaScript was designed & implemented in only *ten days*!)

JavaScript is:

- **hard** to learn (books of 800-1000 pages) with **weird syntax & *semantics*** [1]
- difficult prototype based object model ; dynamically typed
- **difficult** to implement efficiently
- have a big and **complex *specification*** (≈ 566 pages, EcmaScript$_{v7}$, 2016)
- here to stay, **won't disappear** soon!
- **many criticisms** available on the web (and security[2] issues)
- many **frameworks** and **toolkits** (JQuery, . . .)
- better **generated** than handwritten

[1] `https://wiki.haskell.org/The_JavaScript_Problem`
[2] `http://www.ssi.gouv.fr/uploads/IMG/pdf/NP_Securite_Web_NoteTech.pdf`, in French.

headache with browser $\leftrightarrow$ server executions

Some **dynamic behavior inside the web browser** is required, notably in **single-page web applications**

- server initialize the web page at first HTTP requests
- actual content is dynamic (JavaScript code in browser handling events and modifying the DOM)
- most of the page content gets filled after **asynchronous** AJAX requests
- WebSocket-s enable **asynchrony**: the server would send some messages (often JSON) to the browser

execution oscillates between web server and browser, giving motion sickness:

- C.Queinnec's talk **Continuations & the web**
- G.Potdevin's talk **Control inversion in web development**

$\Rightarrow$ better **generate code mix** in server and browser thru **meta-programs**

meta-programming the web

Many (non-mainstream) tools and languages are producing code both in browser and in server:

- Opa - a typesafe language for full-stack application (generates: server program for Node.js + browser program in JavaScript + database for MongoDb)
- Haxe - an open source toolkit and language
- M.Serrano's talk on **Mixing computation in web server and in browser** (HOP)
- V.Balat's talk on **Developing multi-platform web & mobile applications** (Ocsigen)

is `contenteditable` useful?

(in the context of some “syntax directed editor” or “AST wiki”)

See **why is `contenteditable` terrible?** (N.Santos, Medium, 2014)

- how to disambiguate cursor position
  ```
  |<em><a href='http://example.com/'>some link</a> here</em>
  ```
  vs
  ```
  <em>|<a href='http://example.com/'>some link</a> here</em>
  ```
- **exhaustive** list of user actions and events modifying the editable element? Relation to `execCommand`?
- W3C Editor’s draft `https://w3c.github.io/editing/contentEditable.html` - dream or `contenteditable='events'` soon implemented?
- `beforeinput` & `input` events?

dealing with the clipboard (copy & paste)

Traditional (e.g X11 **EWMH** - Extended Window Manager Hints) clipboard involves a *negotiation* on the exchange format or MIME type (plain text, rich text, image, HTML, . . .). So copying & pasting some web page fragment to LibreOffice preserves italics, but copying the same to a terminal only sends text.

exchanging structured data

Could two similar web applications exchange e.g. JSON data during copy & paste ? How?

Security issues?

need of better static analysis tools?

Is there a need (& a “market”) for open source static analysis tools for Web programming languages?

(perhaps security is not relevant enough on the web . . .)