Scripting

Web Architecture and Information Management [./] Spring 2009 — INFO 190-02 (CCN 42509)

Erik Wilde, UC Berkeley School of Information

2009-03-30

Scripting ; Erik Wilde ; UC Berkeley School of Information http://localhost/drectures/web-spring09/scripting#(23)

Contents

Abstract (2)

Scripting is used on the majority of today's modern Web sites. Scripting can be used to improve the usability and accessibility of a Web site (for example for *validating form data on the client side*), it can vastly improve the user experience with new interface design (the *smooth scrolling of Google Maps* vs. older "click to scroll" map services), or it can be used to implement behavior that would be impossible without scripting (for example the *online applications of Google Docs*). This introductory lecture looks into scripting fundamentals such as JavaScript itself, the *Document Object Model (DOM)* for accessing the browser window's content, and `XMLHttpRequest` for script-server communications.

Scripting ; Erik Wilde ; UC Berkeley School of Information http://localhost/drectures/web-spring09/scripting#(23)

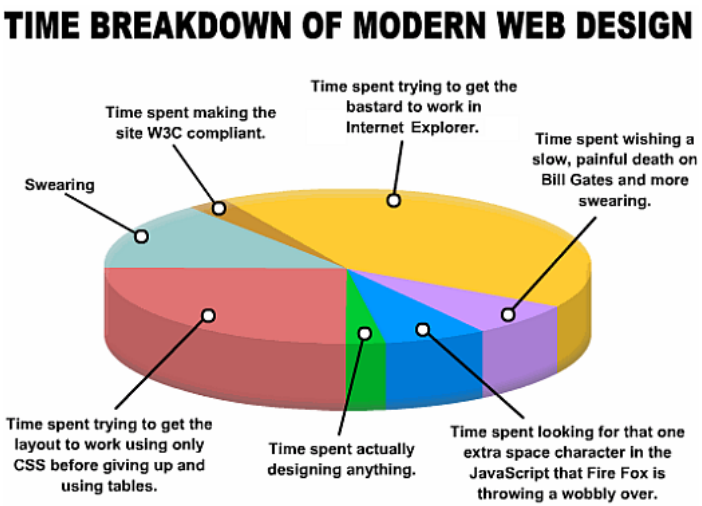
E. Wilde: Scripting

Scripting on the Web (3)

- Web pages were static HTML
  - HTML Forms [HTML Forms] were the only interactive part of Web pages
  - interaction was only possible by clicking links and visiting new pages
  - CSS introduced limited dynamic behavior (such as `mouseOver` events)
- Netscape invented the Document Object Model (DOM) [Document Object Model (DOM) (1)] and *LiveScript*
  - Java [http://en.wikipedia.org/wiki/Java_(programming_language)] was new and hip, so the language was renamed to JavaScript [JavaScript (1)]
  - pages with scripting (a.k.a. *Dynamic HTML* or *DHTML*) allowed richer user interfaces
  - other browsers invented their own "versions" of DOM/JavaScript
- Scripting was and is often used to implement "missing functionality"
  - good scripting supports graceful degradation (leaving the page functional)
  - bad scripting compromises accessibility when the scripting code does not work
- Any non-trivial scripting has to deal with browser differences
  - JavaScript Frameworks [JavaScript Frameworks (1)] help by providing a foundation to build on

Web Architecture and Information Management [./]
Spring 2009 — INFO 190-02 (CCN 42509)
2009-03-30

The Joys of Web Design (4)



Scripting ; Erik Wilde ; UC Berkeley School of Information http://localhost/drectures/web-spring09/scripting#(23)

E. Wilde: Scripting

Basic Scripting (DHTML) (5)

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
  <title>Well-Designed JavaScript</title>
  <script type="text/javascript" src="nicetitle.js"></script>
  <link rel="stylesheet" href="nicetitle.css" />
 </head>
 <body>
  <h1>Well-Designed JavaScript</h1>
  <p><a href="http://en.wikipedia.org/wiki/JavaScript"
title="Wikipedia: JavaScript is a scripting language used to enable
programmatic access to objects within other applications. It is
primarily used in the form of client-side JavaScript for the
development of dynamic websites.">JavaScript</a> should not make any
assumptions about browser support. Ideally, pages using scripting
should also be usable when scripting is turned off, so that more
basic browsers (for example, mobile phones or Kindles) can also be
used for using the page.</p>
 </body>
</html>
```

Web Architecture and Information Management [./]
Spring 2009 — INFO 190-02 (CCN 42509)
2009-03-30

Basic Scripting (JavaScript) (6)

```
if( !document.links )
{
    document.links = document.getElementsByTagName("a");
}
for (var ti=0;ti<document.links.length;ti++) {
    var lnk = document.links[ti];
    if ( lnk.title ) {
        lnk.setAttribute("nicetitle",lnk.title);
        lnk.removeAttribute("title");
        addEvent(lnk,"mouseover",showNiceTitle);
        addEvent(lnk,"mouseout",hideNiceTitle);
        addEvent(lnk,"focus",showNiceTitle);
        addEvent(lnk,"blur",hideNiceTitle);
    }
}

var d = document.createElementNS(XHTMLNS,"div");
d.className = "nicetitle";
tnt = document.createTextNode(nicetitle);
pat = document.createElementNS(XHTMLNS,"p");
pat.className = "titletext";
pat.appendChild(tnt);
d.appendChild(pat);
```


Scripting ; Erik Wilde ; UC Berkeley School of Information http://localhost/drectures/web-spring09/scripting#(23)

E. Wilde: Scripting

Basic Scripting (CSS) (7)

```
div.nicetitle {
    position: absolute;
    padding: 4px;
    top: 0px;
    left: 0px;
    color: white;
    width: 25em;
    background: url(nicetitle-bg.png);

    /* Mozilla proprietary */
    -moz-border-radius: 12px;
}
div.nicetitle p {
    margin: 0; padding: 0 3px;
}
```

Web Architecture and Information Management [./]
Spring 2009 — INFO 190-02 (CCN 42509)
2009-03-30

JavaScript

Browsers are Platforms (9)

- *Runtime environments* are critical for running applications
  - popular computer environments are Windows, MacOS, Linux, and Java
  - popular mobile environments are iPhone, Android, Blackberry, S60, and JavaME/JavaFX
  - popular Web-based environments are JavaScript, Flash, and Java applets
  - popular Web-oriented environments are Silverlight and AIR
- JavaScript is a scripting language supported by most browsers
  - access to the current document's DOM [Document Object Model (DOM) (1)] is the most important part of DHTML
  - JavaScript has code, functions, and interacts with the user through the browser

```
<p>It is <script type="text/javascript">
var currentTime = new Date() ;
document.write(currentTime.getHours() + ":" +
currentTime.getMinutes()) ;
</script> hours</p>

<p>It is 0:11 hours</p>
```


Compiled vs. Interpreted Languages (10)

- Programming languages can be *compiled* or *interpreted*
- *Compilers* can do more than just translate
  - compiled languages are translated before they can be executed
  - check the code for errors that can be determined statically
  - augment the code for performance or analysis purposes
- *Interpreters* provide a less heavyweight environment
  - interpreted languages can be executed directly
  - any change in the code can be tested immediately
  - less tightly coupled software bundles than compiled packages
- *Scripting* has become much more popular in the past years
  - server-side languages such as PHP, Ruby/Rails, and Python

Scripting ; Erik Wilde ; UC Berkeley School of Information http://localhost/drectures/web-spring09/scripting#(23)

JavaScript and Browsers (11)

- Scripting used to be "too slow for serious applications"
  - processors have become much faster
  - language interpreters have become much smarter
- Implementations now deliver high-performance scripting
  - Safari uses the SquirrelFish JavaScript engine [http://trac.webkit.org/wiki/SquirrelFish]
  - Chrome uses the V8 JavaScript engine [http://code.google.com/p/v8/]
- Modern implementations allow sophisticated applications
  - Google Docs [http://docs.google.com/] is a set of browser-based "desktop applications"
  - Processing.js [http://ejohn.org/blog/processingjs/] is a JavaScript version of a popular visualization environment

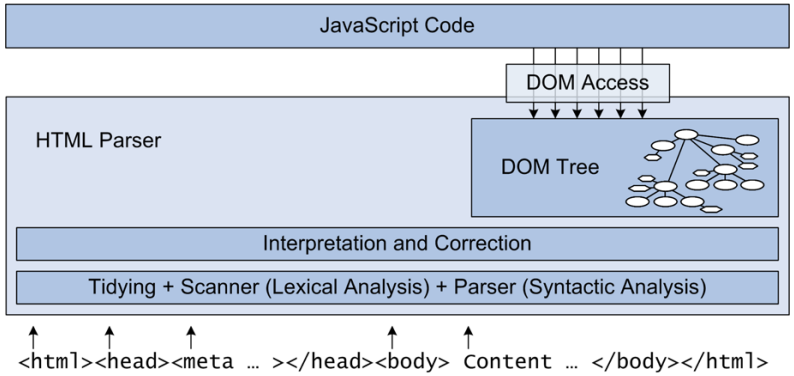
Document Object Model (DOM)

From HTML to DOM (13)

- HTML [Hypertext Markup Language (HTML)] is a representation for hypermedia documents
  - a representation is required to store and transmit the document
  - HTML uses markup [Setup and Environment; Elements (1)] for representing the document structure
- Browsers must render HTML documents (i.e., apply CSS and execute JavaScript)
  1. GET HTML from server and receive as text/html document
  2. parse document and deal with any errors by "fixing them"
  3. interpret document as if it had been error-free
  4. GET all additional resources (CSS, images, JavaScript, …)
  5. build internal model (DOM) based on error-free interpretation
  6. apply CSS rules to determine styling of document (e.g., margins and font sizes)
  7. render into visual structure
  8. start executing JavaScript code
  9. listen for events (keyboard, mouse, timer) and execute code
  10. discard everything and start over when user navigates to a different page

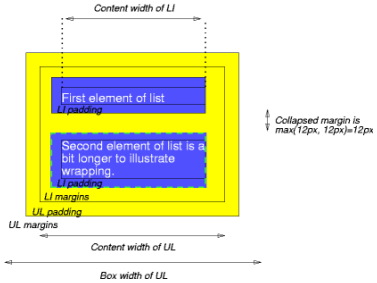
Web Architecture and Information Management [./] 2009-03-30
Spring 2009 — INFO 190-02 (CCN 42509)

Browser Handling of HTML (14)



Scripting ; Erik Wilde ; UC Berkeley School of Information http://localhost/drectures/web-spring09/scripting#(23)

Elements, Objects, and Boxes (15)



Document (16)

- The document (HTML) is the *interface language for Web applications*
- Most programming environments have visual interface models
  - almost everything has moved to *window-oriented interfaces*
  - Windows, MacOS, and Linux provide similar visual metaphors
- Web applications must use HTML as their model for the interface
  - HTML Forms [HTML Forms] are a simple way to build an interface
  - forms can be extended with client-side helpers (validation, repeating entries)

Scripting ; Erik Wilde ; UC Berkeley School of Information http://localhost/drectures/web-spring09/scripting#(23)

E. Wilde: Scripting Document Object Model (DOM)

Object (17)

- Documents are static, programming is dynamic
  - documents and code must be connected
  - *objects* are a common abstraction in programming languages
- Objects usually have a *type* and *methods*
  - *types* for HTML-based objects are based on HTML's elements
  - *methods* define the allowed to interact with objects
  - interactions can be read-only or they can change the document structure

```
for (var ti=0;ti<document.links.length;ti++) {
    var lnk = document.links[ti];
    if ( lnk.title ) {
        lnk.setAttribute("nicetitle",lnk.title);
        lnk.removeAttribute("title");
        addEvent(lnk,"mouseover",showNiceTitle);
        addEvent(lnk,"mouseout",hideNiceTitle);
        addEvent(lnk,"focus",showNiceTitle);
        addEvent(lnk,"blur",hideNiceTitle);
    }
}
```

Model (18)

- Models are idealized/abstract representations of something
- Models allow to share that idealized/abstract view
- DOM introduced a *common way of how browsers deal with HTML*
  - without a DOM, there can be no interoperable scripting
- Abstractions are also limitations
  - some vendors introduced/supported "extensions" to the basic DOM model
  - any code based on these extensions is not interoperable
- DOM is under constant revision
  - DOM0 was invented by Netscape (backing the LiveScript/JavaScript)
  - DOM1 was the first DOM version produced by the W3C
  - DOM2 is the currently available stable version of DOM
  - DOM3 is highly modularized and still under development

Scripting ; Erik Wilde ; UC Berkeley School of Information http://localhost/drectures/web-spring09/scripting#(23)

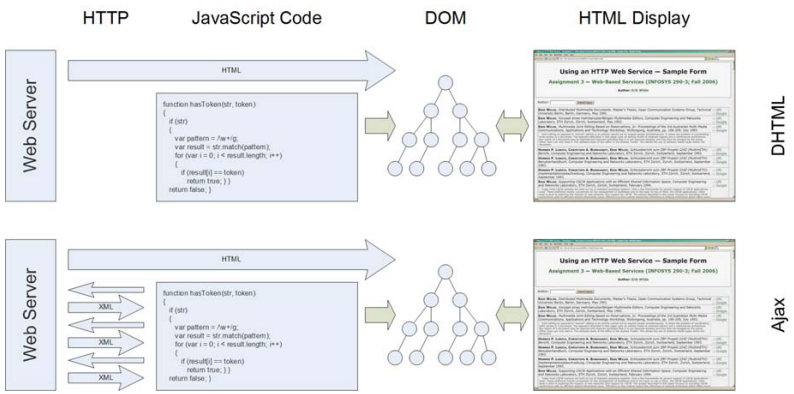
Ajax Basics

Ajax = DHTML + HTTP (20)

- Basic Scripting (DHTML) [Basic Scripting (DHTML) (1)] uses JavaScript "locally"
  - the scripting code reacts to user events and accesses the DOM structure
  - changes are either hardcoded or derived from user events
- Ajax adds an HTTP request [Web Foundations (URI and HTTP); HTTP Requests (1)] method to JavaScript
  - scripting code can now request additional data from an HTTP server
  - changes can thus be made based on any data received from the server
- Ajax dramatically reduces the number of page reloads
  - any change of the page can be done without a complete reload
  - based on user interactions, parts of the page can be reloaded
- Ajax has the same interoperability problems as DHTML

Scripting ; Erik Wilde ; UC Berkeley School of Information http://localhost/drectures/web-spring09/scripting#(23)

Ajax and DHTML (21)



JavaScript Frameworks

Abstraction and Reality (23)

- Browsers are not entirely standards-compliant
  - Acid Tests [http://www.acidtests.org/] are a way how to test browser compliance
  - compliance depends on what you test for (versions of the standards)
- Running Acid3 [http://acid3.acidtests.org/] for current browsers is disappointing
  - Chrome and Safari are equal (because they both use WebKit [http://webkit.org/]
  - Firefox and Opera are not that bad (but not perfect)
  - IE8 is a disaster
- In some cases, implementations have to make guesses
  - complex combinations of HTML, CSS, and JavaScript interactions
- JavaScript frameworks have two major functions
  1. hiding the fact that browsers need a lot of special case handling
  2. providing support for common Web design patterns

Web Architecture and Information Management [./]
Spring 2009 — INFO 190-02 (CCN 42509)
2009-03-30

Web Design Patterns (24)

- Many Web pages use similar ideas/visualizations
- Factoring them into *design patters* enables tool support
- Providing access to a tree-structured set of resources
  - folder views [http://developer.yahoo.com/yui/examples/treeview/folder_style.html] are a common design pattern
- Displaying image captions based on mouse events
  - dynamic image captions [http://masterfidgeter.com/projects/captify/] help merging images and their captions
- Tabs are well-known from desktop applications and popular in Web design
  - Ajax Tabs [http://extjs.com/deploy/dev/examples/tabs/tabs.html] can even load content dynamically
- Image-heavy Web sites might need image viewing support
  - image zooming [http://demos.dojotoolkit.org/demos/cropper/] can make it more convenient to zoom into images

Scripting ; Erik Wilde ; UC Berkeley School of Information http://localhost/drectures/web-spring09/scripting#(23)

E. Wilde: Scripting JavaScript Frameworks

Popular Frameworks (25)

- Different needs produce different frameworks
  - Dojo [http://www.dojotoolkit.org/]
  - Ext JS [http://extjs.com/]
  - jQuery [http://jquery.com/]
  - Yahoo! Interface Library [http://developer.yahoo.com/yui/]
- There is no such thing as the "best JavaScript framework"
  - for any given project, decide on the support you need
  - evaluate frameworks for the support they provide
  - evaluate for *functional requirements* ("is there a collapse/expand folder view?")
  - evaluate for *non-functional requirements* ("is the framework openly licensed")

Web Architecture and Information Management [./] 2009-03-30
Spring 2009 — INFO 190-02 (CCN 42509)

Important Framework Questions (26)

- How big is it?
- How is it licensed?
- How is it maintained?
- How well does it support graceful degradation?
- How well does it mix with other JavaScript code?
- For professional Web development, don't overuse effects

Scripting ; Erik Wilde ; UC Berkeley School of Information http://localhost/drectures/web-spring09/scripting#(23)

Conclusions (27)

- Scripting has become as essential part of Web-based applications
- DHTML is local scripting, Ajax is scripting + server access
- Deciding between client-side and server-side is hard
- JavaScript frameworks help developing script-based applications
- Graceful degradation becomes more important on the mobile Web