Web Browsers

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

Erik Wilde, UC Berkeley School of Information

2009-02-09

Contents

Abstract (2)

This lecture looks at *Web browsers* and how they work. It introduces the basic functionalities of a browser; retrieval and rendering of Web pages. Any modern browser needs to support more than just HTTP and HTML; it must support CSS for stylesheets, JavaScript for scripted Web pages, various image formats, and popular applications such as Flash. In addition, browsers can support additional functionality such as off-line operation, or in general more application-oriented features such as *AIR* or *Silverlight*.

Browser Basics

What is a Web Browser? (4)

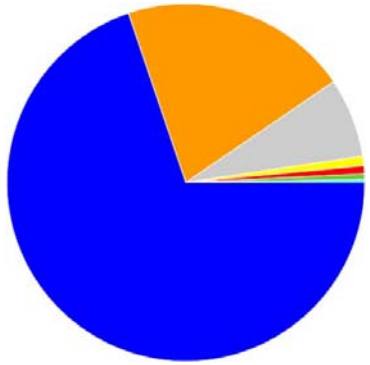
- Network access (HTTP, HTTPS, FTP, file system, …)
- Rendering HTML layout (a subset of CSS layout)
  - CSS specifies many more features
- Handling special HTML in the required way
  - images (in various formats) must be downloaded and embedded
  - forms must be rendered and form data must be submitted
- Running scripts and providing them access to the page
  - re-rendering when scripts change the page (DHTML)
  - providing scripts with network access (Ajax)
- Utility functions to make the browser more usable
  - tabs and bookmarks for more organized browsing
  - security policies for safer browsing
  - additional content types may be supported (by external software)
  - the browser may be extended (add-ons)

One Minute in the Life of a Browser (5)

1. Analyze URI and connect to server to retrieve resource
   - recursively repeat until all required resources are retrieved
2. Analyze HTML, correct errors, and compute a *DOM tree*
   - DOM is a memory representation of the HTML markup
3. Apply CSS and compute the layout of the styled DOM tree
   - compute CSS decorated DOM and apply formatting algorithm to it
4. Start executing Scripting [Scripting] code and change the DOM as required
   - scripting may have initial phase and user interaction phase
5. Continue executing scripting code in response to user interactions
   - for many dynamic Web pages, this is a continuous activity
6. If the user clicks on a link, start all over again

Browser Usage (6)

- Internet Explorer (69.80%)
- Mozilla Firefox (20.66%)
- Safari (7.18%)
- Chrome (0.87%)
- Opera (0.72%)
- Netscape (0.52%)
- Other (0.25%)

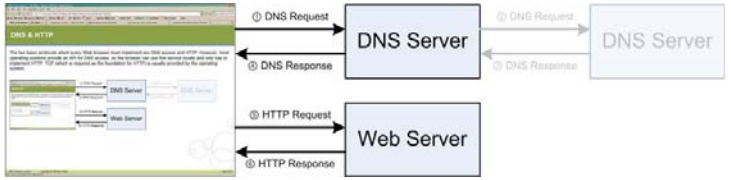


Browsers and CSS (7)

- Browsers have their own *built-in CSS code*
  - HTML pages with no CSS are still formatted in some way
  - HTML pages can provide their own CSS to change defaults
  - users can change the browser's default to their own preferences
- CSS has a well-defined way of how stylesheets are combined [http://www.w3.org/TR/CSS21/cascade.html#cascading-order]
  1. browser defaults
  2. user declarations
  3. page declarations
  4. page important [http://www.w3.org/TR/CSS21/cascade.html#important-rules] declarations
  5. user important [http://www.w3.org/TR/CSS21/cascade.html#important-rules] declarations
- Rendering of HTML/CSS depend on a variety of factors
  - default settings of the browser
  - preferences set by the user
  - CSS code provided by the page author
  - HTML/CSS capabilities of the browser

Browsers and the Internet (8)

Before retrieving the Web page http://www.berkeley.edu/ [http://www.berkeley.edu/], the browser first has to find out the IP [Internet Architecture; Internet Protocol (IP) (1)]address of the www.berkeley.edu server. Using this address, it can then open an HTTP [Web Foundations (URI and HTTP); Hypertext Transfer Protocol (HTTP) (1)] connection. The lookup service used by the browser is the Domain Name System (DNS) [Internet Architecture; Domain Name System (DNS) (1)].



Supported URI Schemes (9)

- Most Web pages are available over HTTP [Web Foundations (URI and HTTP); Hypertext Transfer Protocol (HTTP) (1)]
  - one popular exception are pages available over HTTPS [Security & Privacy; HTTP over SSL (HTTPS) (1)]
- Most browsers support more than just the HTTP and HTTP URI Schemes [Web Foundations (URI and HTTP); URI Schemes (1)]
  - http: and https: are necessary (these are the Web protocols)
  - file: [http://en.wikipedia.org/wiki/File_URI_scheme] allows the browser to load local files
  - ftp: is useful because many documents are available on FTP servers
  - mailto: usually is not built into the browser (the mail tool is started)
  - tel: is a useful scheme for devices with telephone functionality
- Firefox 3 allows to register protocol handlers [https://developer.mozilla.org/en/Web-based_protocol_handlers]

Caching (10)

- Browsers retrieve resources for rendering Web pages
- In a typical user session, many resources are used repeatedly
  - using the browser's "back" button
  - accessing pages reusing the same CSS or images
- *Caching* is a frequently used optimization in computer systems
  1. store retrieved data locally
  2. reuse that data when it is used again instead of fetching it again
  3. the hard (and important) part is *cache invalidation*
- *Prefetching* [http://en.wikipedia.org/wiki/Link_prefetching] allows browsers to load pages in advance
  - predicting user behavior usually is hard or impossible
  - unnecessary prefetching generates unnecessary load on servers and the network
  - faster networks make load delays less painful

```
<link rel="prefetch" href="http://www.example.com/">
```

- Google Web Accelerator [http://en.wikipedia.org/wiki/Google_Web_Accelerator] was a Google-specific approach
  - using prefetching and compression (connecting to a Google cache)
  - serious privacy implications (*all* traffic is routed through Google)

Security and Privacy (11)

- Browsers store a lot of security-sensitive data
  - data entered in forms is stored for future visits
  - authentication credentials (cookies [State Management (Cookies)]) are stored on behalf of servers
  - the browsing history of visited pages is stored
- Connecting to HTTPS Web sites requires a certificate validity check
  - browsers come with a large set of pre-installed *certification authorities*
  - users implicitly trust this list of pre-installed authorities
- Browsers provide control over these features in complicated settings
- Browsers start providing more user-friendly "private modes"
  - Safari calls the feature *private browsing*
  - IE8 has an InPrivate [http://www.microsoft.com/windows/internet-explorer/beta/features/browse-privately.aspx] mode
  - Firefox 3.1 includes such a feature as *Private Browsing*
- Security/Privacy (as always) is a trade-off with convenience

Browsers and Scripting (12)

- Scripting [Scripting] is essential for most modern Web pages
  - well-designed Web pages also work when scripting is turned off
  - many Web pages are not designed all that well
  - when scripting is turned on, behavior should be predictable and consistent
- Scripting problems plagued Web developers for a long time
  - major parts of Web development go into ensuring compatibility
  - ill-behaving browsers (such as IE) make it impossible to develop simple code
  - JavaScript Frameworks [Scripting; JavaScript Frameworks (1)] provide "compatibility layers" on top of browsers
- Browsers can morph into "runtime environments"
  - using Google Docs [http://docs.google.com/] has little to do with Web browsing
  - some essential features are missing (offline capabilities, local storage)
  - Chrome [Chrome (1)] is Google's attempt to morph the Web into an application platform

Content Type Handling

Content Types (14)

- Browsers retrieve resources and render them
  1. URI identifies a resource to be retrieved
  2. HTTP request is sent to the server requesting the resource
  3. HTTP response is received containing and describing the resource
  4. the media type [Media Types] determines how the browser handles the resource
- Browsers can handle resource in four different ways
  1. Built-In Support [Built-In Support (1)] allows the browser to handle the resource by itself
  2. Add-Ons [Add-Ons (1)] extend the browser with additional capabilities
  3. Plug-Ins [Plug-Ins (1)] are platform-specific extensions
  4. External Viewers [External Viewers (1)] are programs to which the browser passes the resource

Built-In Support

Built into the Browser (16)

- The Web is built on few universal media types
- Variety on the Web is achieved through two major factors:
  1. the established media types are not application-specific
  2. advanced content can be based on *browser-based runtime environments*
- Examples of built-in media types:
  - HTML+CSS for rendering formatted Web pages
  - popular image formats (GIF, JPEG, PNG, ICO [Multimedia Content; Icons (1)])

Advantages/Disadvantages (17)

- Advantages of built-in media types:
  - fast and seamless rendering
  - should be supported by any browser (safe choice for developers)
- Disadvantages of built-in media types:
  - cannot be added by users (not extensible)
  - browser must be upgraded to add new types

Add-Ons

Browser-Specific Additions (19)

- Supported by the browser itself (built into the browser framework)
- Usually support *additional functionality* and not new media types
- Add-ons are productivity enhancements for users
- Examples of add-ons:
  - Minimap Sidebar [https://addons.mozilla.org/en-US/firefox/addon/5203] for better support for geolocation data
  - Operator [https://addons.mozilla.org/en-US/firefox/addon/4106] for visualizing a Web page's metadata

Advantages/Disadvantages (20)

- Advantages of add-ons:
  - only browser-dependent (i.e., not OS dependent)
  - can be installed individually and specific for users
  - allow browser specific behavior (e.g., Firebug [Hypertext Markup Language (HTML); Firebug (1)] for Firefox debugging)
- Disadvantages of add-ons:
  - cannot be used across browsers
  - conflicts between add-ons can lead to instabilities
  - Web applications cannot count on them being available (e.g., Google Gears [http://gears.google.com/] and Chrome [Chrome (1)])

Plug-Ins

Platform Code in the Browser (22)

- Platform-specific code running inside the browser
  - window created by an object with given window dimensions
  - the window displays whatever the plug-in code generates as display
- Examples of plug-ins:
  - Adobe Reader [http://get.adobe.com/reader/] for rendering PDF documents
  - Adobe Flash Player [http://get.adobe.com/flashplayer/] for running Flash applications
  - Java Runtime Environment (JRE) [http://java.com/en/download/help/5000011200.xml] for executing Java Applets

Advantages/Disadvantages (23)

- Advantages of plug-ins:
  - high performance (OS-specific code)
  - reasonably easy to implement if OS specific code already exists
- Disadvantages of plug-ins:
  - hard to implement for a new OS (needs OS-specific code)
  - no easy fallback if not supported by a browser
  - plug-in internals are "invisible" to the browser

External Viewers

Browser and Applications (25)

- Applications often are not integrated with a browser
  - designed as standalone applications
  - capable of handling certain media types
  - registered with the OS and/or with a browser
- External applications can do anything they like with a resource
  - the browser downloads the resource to a temporary file
  - the external viewer is started and handed the file's location
- Examples of external viewers:
  - Office Suites [http://en.wikipedia.org/wiki/Office_suite] for handling documents, spreadsheets, and presentations
  - Adobe Reader [http://get.adobe.com/reader/] for rendering PDF documents
  - Google Earth [http://earth.google.com/] for handling *KML* files

Advantages/Disadvantages (26)

- Advantages of external viewers:
  - almost no integration effort with a browser (just registering)
  - sophisticated and highly optimized applications
  - can be taken offline and used for non-Web activities
- Disadvantages of external viewers:
  - high dependency on platform and configuration of a user's browser
  - completely breaks the workflow of working with a browser

Others

Greasemonkey (28)

- Greasemonkey is an "add-on [Add-Ons (1)] for add-ons"
- Runtime environment for *user scripts* [http://userscripts.org/]
  - specifically addressing a Web page or a Web site
  - locally changing the Web page (in the browser)
  - support for changing a Web site's display according to my user needs
- User scripts analyze/extract/update a Web page
  - when Web pages are redesigned, user scripts often break
  - a fragile way of repurposing the information from a Web site

Extended Browsers

Chrome (30)

- Google's goal is to move more information online
- Today's browsers lack good offline support
  - Google Gears [http://gears.google.com/] Add-Ons [Add-Ons (1)] adds offline capabilities for Firefox
- Chrome [http://www.google.com/chrome/] is built around Google's strategic goals
  - robust and high quality rendering based on WebKit [http://webkit.org/]
  - efficient and fast execution of scripting code with V8 [http://code.google.com/p/v8/]
  - includes Google Gears for offline mode
  - minimal user interface to look more like an application

Accessibility Browsers (31)

- Browsers for blind users depend on well-designed Web pages
  - some browsers read aloud the textual information on a page
  - Braille readers are an alternative to read-aloud browsers
- Only well-designed Web content is accessible to these browsers
  - HTML and CSS content can be analyzed in the browser
  - Plug-Ins [Plug-Ins (1)] content (such as Flash) cannot be handled at all
  - External Viewers [External Viewers (1)] may have their own accessibility features
- Usability and Accessibility [Usability and Accessibility] should play an important role in Web design
  - Section 508 [http://www.section508.gov/] requires public information to be made accessible

Conclusions (32)

- Browsers are much more than just "HTML viewers"
- Users can customize their browsing experience
- Information providers should be aware of browser issues