COMP7306: Web technologies

Client-side Web design

30 January 2013

Pierre Senellart

Outline

Outline

Purpose

- CSS: Cascading StyleSheets
- W3C recommendation
- Several levels: CSS1 (1996), CSS2 (1998), CSS2.1 (2005), CSS3 (work in progress)
- Very disparate support by browsers, never complete (in particular, IE6 and IE7 have several important limitations) $\Longrightarrow$ focus on what works in all common browsers

Principle

- HTML describes structure and content
- CSS describes:
  - text formatting
  - layout of blocks with respect to one another

Inline styles

- Simplest way to use CSS
- Add a `style` attribute on HTML tags
- `<span>` or `<div>` can be used as non-semantic text-level and block-level tags
- Clutters HTML code with styling indications: not what we are looking for!

Examples

-
  ```
  This word in <em style="color: red;">emphasis</em>
  is also in red.
  ```

  ```
  <span style="text-decoration: underline">This multiword
  phrase</span> is underlined.
  ```

Styles in the head

- Integrating style rules to the head of the page with the `<style>` tag
- Using selectors to define which elements these rules apply to
- Disadvantage: mixes HTML and CSS in the same document, impossible to reuse CSS rules across documents

Example

```
<!DOCTYPE ... >
<html>
  <head> ...
    <style type="text/css">
      em { color: red; }
    </style>
  </head>
  <body> ... </body>
</html>
```

Linked stylesheet

- Putting the CSS stylesheet in a separate file (generally, one uses the *.css* file extension)
- Allows reuse of the same spreasheet in several Web pages
- Adding a `<link>` tag in the head whose `rel` attribute is set to `"stylesheet"`
- Possible to add `media="screen"` or `media="print"`, etc., to choose different stylesheets depending on the display mode

Example

```
<!DOCTYPE ... >
<html>
  <head> ...
    <link rel="stylesheet" href="style.css" type="text/css">
  </head>
  <body> ... </body>
</html>
```

Classes

- One often wants to add more structure and semantics to an HTML document
- One can use the `class` attribute on any HTML tag for that purpose
- CSS can be use to apply some style to everything that is in a given `class`

Example (Put person names in blue italic)

- HTML

```
<p>I would like to acknowledge
<span class="person">Mrs Foo</span>
and <span class="person">Mr Bar</span>.</p>
```

- CSS

```
.person { color: blue; font-style: italic; }
```

CSS syntax

- Set of rules of the form:

```
selector {
  property: value;
}
```

- `selector` : indicates which part of the document the rule applies to
- `property:` : specific display property to modify
- `value` : meaning depends on the property
- Stylesheets must be validated with an adequate validator, cf. `http://jigsaw.w3.org/css-validator/`
- Comments between `/*` and `*/`

Cascade

- Several stylesheets can apply to the same document:
  - Several `<link rel="stylesheet">` tags
  - `@import` directive in a stylesheet

    ```
    @import url(style_aux.css);
    ```
  - User stylesheet (configurable in some browser)
- Inside a given stylesheet, several rules can conflict
- Cascade: mechanisms to deal with these conflicts

Cascade

- If !important is specified after the value, the rule has priority
- Otherwise, the most specific the rule, the highest the priority
- Otherwise, the last rule declared applies

Outline

Simple, multiple, universal selectors

Simple selector: tag name

Multiple selector: several selectors joined by commas

Universal selector: ‘∗’, selects everything

Examples

- `ul { color: blue; }` displays all unordered list content in blue
- `h1,h2,h3,h4,h5,h6 { color: red; }` puts section titles in red
- `* { color: black; }` displays everything in black. In this particular case, one would preferably use `body { color: black; }`.

Class selectors

Class selector: class name, prefixed with ‘.’, as it appears in a `class` attribute of an HTML tag

Examples

- `.person { font-weight: bold; }` displays all tags with class `person` in bold
- `p.comment { font-style: italic; }` displays all `<p>` tags with class `comment` in italic

Identifier selector

Identifier: as defined by the `id` attribute of an HTML tag. Similar to classes, but only one tag with a given `id` in the whole HTML document

Identifier selector: identifier name, prefixed with ‘#’, as it appears in the `id` attribute of an HTML tag

Examples

- `#introduction { font-size: 120%; }` displays the tag with identifier `introduction` in larger font
- `p#introduction { font-size: 120%; }` displays the `<p>` tag with identifier `introduction` in larger font

Contextual selectors

Contextual selector: 2 selectors or more separated by spaces. *A B* selects *B*'s only if they are contained in *A*'s

Examples

- `h1 em { color: blue; }` puts words that are in emphasis also in blue
- `ul ol, ol ul, ul ul, ol ol { font-size: 80%; }` decreases the font size of nested lists

Pseudo-elements

Pseudo-element: allows to put some styling on something that is not defined by an HTML tag

Examples

- `p:first-line { font-weight: bold; }` puts the first line of every paragraph in bold
- `p:first-letter { font-weight: bold; }` puts the first letter of every paragraph in bold

Pseudo-classes

- Pseudo-classes: allows selecting of a given element in certain contexts
- `a:link` : `<a>` 's that are hyperlinks
- `a:visited` : `<a>` 's that are links to already visited pages
- `a:hover` : `<a>` 's that are links that one is pointing to (e.g., the mouse is hovering on them)
- `a:active` : `<a>` 's that are links that are being activated (e.g., being clicked on)

Remark

Caution! Always define these pseudo-classes in this order, because of the cascade rules. (Las Vegas (Forest) Has Animals)

Outline

Length properties

Properties expecting lengths allow different measuring units:

- as a percentage (with respect to the current value, or that of the containing element)
- relative values:

`em`	`font-size` value of the current font (height of the natural height of the font)
`ex`	height of an `x` character in the current font

- values relatives to the screen (to be avoided so as to have a design independent of screen resolution!

`px`	number of pixels

- absolute values (Nonsensical for Web pages to be displayed on a computer screen!): mm, cm, in, pt, pc

Fonts

- Several ways to control the font used in the document
- Main properties (`font-family`, `font-size`, `font-weight` and `line-height`) control appearance of text

Example

```
p {
  font-family: "Times New Roman";
  font-size: 130%;
  font-weight: bold;
  line-height: 150%;
}
```

`font-family` specifies the font family or category (`serif`, `sans-serif`, `cursive`, `fantasy`, `monospace`) to use for the browser. It is always recommended to use a generic family after specific font families (separated with commas) to substitute in case these font families are not available on a specific device

`font-size` defines the font size (most of the time expressed as a percentage of the current value)

`font-style:` `italic`, `oblique` ou `normal`.

`font-variant:` `normal` or `small-caps`

`font-weight:` `normal` or `bold`

`line-height` is the interline space. A good rule of thumb is that it should be 1.2 times the height of the current font (`1.2em` or `120%`).

`text-decoration` defines text ornaments: `none`, `underline`, `overline`, `blink` or `line-through`

Alignment and indenting

`text-align:` `left`, `right`, `center`, or `justify`

`vertical-align:` vertical alignment with respect to the base text line: `super` (superscript), `sub` (subscript), `baseline` (normal), etc.

`text-indent:` shifts the beginning of the first line with respect to subsequent lines

Properties of lists

`list-style-image` defines the image used as a list item marker. When the image is unavailable, replaced with that specified by `list-style-type`

`list-style-type:` `disc`, `circle`, `square`, `decimal`, `upper-alpha`, `lower-alpha`, `upper-roman`, `lower-roman`, or `none`

Example

- ● disc
- ○ circle
- □ square
- none

Colors

- Several ways to express colors:
  - predefined color names among `aqua`, `black`, `blue`, `fuchsia`, `gray`, `green`, `lime`, `maroon`, `navy`, `olive`, `purple`, `red`, `silver`, `teal`, `white` and `yellow`
  - RGB (Red, Green, Blue) code with values from 0 to 255
  - hexadecimal RGB code

Example (Expressing blue)

- `blue` keyword
- `rgb` keyword with absolute `rgb(0,0,255)` or proportional `rgb(0%,0%,100%)` values
- hexadecimal code `#0000FF`

Colors and backgrounds

- `color` and `background-color` properties for front and background colors
- `background-image` for using an image as background. Strongly recommended to also have a matching background color, that can be used as a replacement if the image cannot be shown

Example

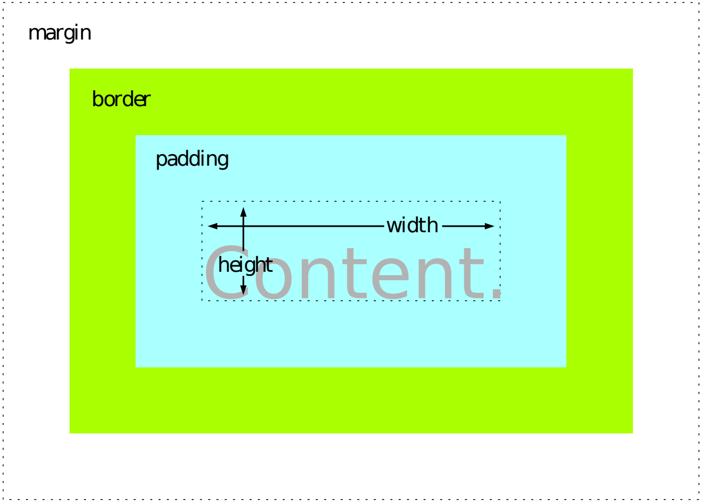
```
background-image: url("images/myImage.jpg");
background-color: white;
```

Outline

Block-level and inline elements

- Two kind of HTML elements:
  - Block-level: `<p>`, `<h1>`, `<ul>`...
    `<div>` is a generic block-level element
  - Inline elements that must be put inside blocks: `<a>`, `<img>`, `<em>`...
    `<span>` is a generic inline element
- How block-level elements are laid out with respect to one another?

CSS box model



Floating and positioning

- Floating and positioning are tools for complex layouts: column layout, elements on top of each other, etc.
- Positioning consists in specifying where elements must appear relatively to their normal position or to another element, or to the browser window
- Floating elements float around elements that follow (think of pictures or sidebars inside newspaper articles)

Floating

- Floating defined by the `float` position that can take the following values:
  - `left`: the element floats to the left of following elements
  - `right`: the element floats to the right of following elements
  - `none`: default value, mostly serves to overwrite existing style
- Every block (image, text, etc.) can be defined as floating
- A dozen of rules defining the behavior of floating elements

Positioning

- Positioning specified with the `position` property
- Four modes: `relative`, `absolute`, `fixed` and `static`.
- `static` positioning is the default behavior
- `relative` shifts an element by a given distance
- `absolute` removes an element from the normal flow and positions it with respect to its nearest container with relative positioning
- `fixed` removes an element from the normal flow and positions it with respect to the browser window. Caution: does not work with IE6

Remark

For absolute positioning with respect to the whole page (most frequent), `body { position: relative; }`.

Visibility and display

- Visibility of an element can be controlled using the `visibility` property:
  - `visible` sets an element as visible
  - `hidden` sets an element as invisible, but the general layout does not change
- One can also use the `display` property:
  - `none` removes the element from the flow of the page;
  - `block` displays the element as a block (default for `<div>`, `<p>`, etc.)
  - `inline` displays the element inline (default for `<span>`, `<a>`, etc.)
  - and other values not so well supported

Outline

In CSS2.1

- `p>em` : `<em>` directly inside a `<p>` (not IE6)
- `li+li` selects `<li>`'s that directly follow another (not IE6)
- `*:lang(fr)` selects elements whose declared language is French (declared with a `lang=` attribute, not IE6 and IE7)
- `em:after {content: "!"}` adds an “!” after all `<em>`'s (`*:before` also exists, not IE6 and IE7)
- The `inherit` value indicates that the value is inherited from hierarchy elements (not IE)
- `max-width` / `min-height` /etc. set minimum or maximum width/heights for blocks (not IE6)
- `display: table-cell;` (and `table`, `table-row`...) displays any set of elements as a table (not IE6 and IE7)
- Automatic counters (not IE6 and IE7), etc.

In CSS3

- More color names (almost works in IE!)
- Different stylesheets depending on screen resolution or characteristics etc. (light support)
- Selectors depending on the value of an attribute (e.g., `a[href$=".pdf"]`, not IE6)
- Selecting *n*th, 2*n*th, etc., elements inside another (light support)
- Automatically downloaded fonts `@font-face` (all browsers, but font formats differ from one browser to another). cf. `http://craigmod.com/journal/font-face/`
- and many other things, still in development

Outline

Outline

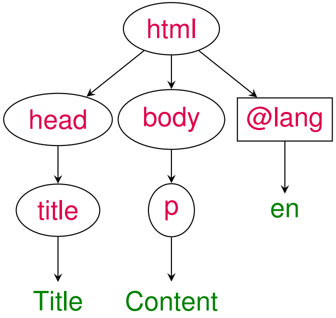
Purpose

- PHP, CGI. . . : allow dynamic behavior on the server side. Require a communication between the browser and the server (form submission, clicking on a link, timeout, etc.) for every needed dynamic behavior
- JavaScript: dynamic behavior on the client side: dealing with windows, dynamic loading of HTML/CSS, fine interaction with forms. . .
- Manipulates the DOM (Document Object Model), the tree representation of the HTML document, tags being nodes of the tree
- “ Dynamic HTML” (DHTML): JavaScript + DOM + CSS
- Alternatives: VBScript (Internet Explorer only), Java/Flash/Silverlight (more isolated from the rest of the Web page)
- Nothing to do with Java!

Document Object Model

Example

```
<html lang="en" xmlns="...">
  <head><title>Title</title></head>
  <body><p>Content</p></body>
</html>
```



The JavaScript language

- Programming language
- JavaScript script (or program): text file, instructions completed with semicolons
- Normalized under the name of EcmaScript (JavaScript is historically the name of the Netscape implementation, JScript is the Microsoft equivalent)
- Practically speaking, currently, only graphical browsers support JavaScript. Consequence: except for complex applications, a site must be usable without JavaScript.

Link with HTML

- `toto.js` contains JavaScript instructions and functions (function)
- in the HTML document:

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

- Event manager as HTML tag attributes (cf. later)
- One can also have JavaScript code inside `<script>` tags but:
  - Mixing languages inside a given document adds complexity
  - Full document must remains valid HTML/XHTML!
  - Impossible to share functions across Web pages without copying them.
- Last possibility: use the `javascript:` pseudo-protocol in a link

Outline

Object model

- JavaScript based on object-oriented programming paradigm
- Variables: complex objects, with properties (members) and features (methods).
- In JavaScript, one accesses the blah member of the toto object with toto.blah and the bouh method of the toto object with toto.bouh(arguments).
- In practice, JavaScript objects that are used represent the HTML document, nodes of the document, the window…

Outline

The Node class of objects

- The Node object is the central object of the DOM (Document Object Model).
- Each element, each attribute, each character data are represented as distinct nodes forming a tree.
- The Node has properties and methods to access different nodes, no matter according to where they are in the tree
- The nodeType property gives the type of node (element, attribute, texte, document, comment. . .)

Example

```
if(node.nodeType==Node.ELEMENT_NODE) {
  ...
}
```

Document nodes

- document is a predefined object representing the current document
- `node = document.documentElement` loads in the `node` variable the l'root element (i.e., the node corresponding to the `<html>` element)
- `node = document.getElementById("titi")` accesses an element by its identifier
- `value = node.nodeValue` accesses the value of a node:
  - for text node, its character data
  - for attribute nodes, the attribute value
- There is also a `document.getElementsByTagName` function that return an array of elements

CSS manipulation

- `node.className="new_class"` to change the CSS class of a node
- `node.style.borderStyle="value"` to change the border style of a node
- `node.style.visibility="value"` to change the visibility of a node
- `node.style.display="value"` to change the CSS display property of a node

General rule

One can change this way any CSS property. The JavaScript name is identical to the CSS name, except that hyphens are replaced with a following uppercase. JavaScript property values are identical to CSS values (but must be put into quotes).

Example

JavaScript:

```
function Test() {
  document.getElementById("paragraph").style.color = "blue";
}
```

HTML:

```
<p id="paragraph" style="color: red;">some text</p>
<a href="" onclick="Test()">Test</a>
```

Explanation: The example has a paragaph of identifier *paragraph* and a link that, when clicked, calls the *Test()* function. This function changes the CSS `color` property of the paragraph, such that this paragraph loses its red color and becomes blue.

Tree browsing

node.parentNode

node.childNodes

node.nodeName

node.getAttribute("name")

Tree modification

node.appendChild(child) adds a previously created node to the existing structure, as last child of a given node

node.removeChild(child)

node.cloneNode() constructs an identical copy of a node, with or without its tree substructure

node.setAttribute("name","value") change node attribute values

Window manipulation

alert("m")	creates a dialog box in which the message *m* is displayed
back()	allows returning to the last visited page
close("window_name")	destroys a client window
confirm("m")	creates a dialog box to confirm an action: OK or Cancel
open("URL","window_name","window_options")	opens a new browser window

These functions are actually methods of the window object, a predefined object corresponding to the browser window.

example

Example

JavaScript:

```
function OpenWindow() {
  Info = open("file.htm", "secondwindow");
}
```

HTML:

```
<body onload="OpenWindow()">
  <p><a href="" onclick="Info.close()">Close the window</a></p>
</body>
```

Explanation: Opens at load time a new popup window. If the user clicks on the link, this window is closed down.

Event handlers

- Event handlers are user actions, that will lead to some interactivity
- The event corresponds to a mouse click, the selection of a field, a key press, etc.
- JavaScript code can be associated with events
- In the event handler, this represents the current node
- Syntax is as follows:

```
onevent="JavaScript action or function"
```

Common events

onblur
onchange
onclick
onfocus
onload
onreset
onsubmit

Example

```
<form name="Test" action="">
  <input type="text"
    onfocus="this.value='Enter your name here'">
  <input type="text"
    onfocus="this.value='Enter your address here'">
  <input type="text"
    onfocus="this.value='Enter your age here'">
</form>
```

Outline

AJAX

- Asynchronous Javascript and XML
- not a technology, but a set of technologies, similar to DHTML

DHTML	HTML+CSS+JavaScript
AJAX	HTML+CSS+JavaScript(+XML)+XMLHttpRequest

- Important part: JavaScript XMLHttpRequest class
- Allows exchanges between client and server code inside a Web page, without reloading
- Asynchronous: data processed when they arrive, in a non-blocking manner
- XML: data historically usually returned in XML format, not always the case anymore
- Used for autocompletion, complex interfaces (e.g., webmails), refreshing information in a part of a page, etc.

Support in modern browsers

- Introduced by Microsoft in IE5, in a slightly unstable form
- Very good idea! Other ways to do the same things (e.g., `<iframe>`), but much less convenient
- Taken over by all other browsers since then, and standardized by the W3C in a simplified form

Principle

- Create an XMLHttpRequest object by giving the URL of a script (possibly with parameters) to contact
- Associate this object with a function that will handle retrieved data. Data can be retrieved as XML and processed in JavaScript using DOM browsing functions, as one would with HTML
- Send the request
- Once the response is returned, the processing function is called (in an asynchronous manner)

Example of use of XMLHttpRequest

Example

```
request = new XMLHttpRequest();
request.open("GET", url);

request.onreadystatechange = function() {
  if (request.readyState == 4 && request.status == 200) {
    result=request.responseXML;
    /* do something with responseXML */
  }
};
request.send(null);
```

Compatibility with Internet Explorer 5 and 6

```
request=null;
if(typeof XMLHttpRequest != "undefined")
  request = new XMLHttpRequest();
else {
  try {
    request = new ActiveXObject("Msxml2.XMLHTTP.6.0");
  } catch(e) {
    try {
      request = new ActiveXObject("Msxml2.XMLHTTP.3.0");
    } catch(e) {
      try {
        request = new ActiveXObject("Msxml2.XMLHTTP");
      } catch(e) {
        request = new ActiveXObject("Microsoft.XMLHTTP");
      }
    }
  }
}
```

Outline

JavaScript support

- Very large disparity in JavaScript support from one browser to the next
- EcmaScript and DOM standards arrived quite late
- IE is the most behind the standards, but even other browsers have numerous sources of varying behavior
- Implementation in progress of many features

JavaScript frameworks

- JavaScript function libraries that allow to work at a higher level and abstract out difference across browsers
- Features:
  - Browser detection
  - Nicer DOM manipulation
  - Animation effects
  - Complex interactions (e.g., click/drag)
  - Better event handling
  - Simplified AJAX processing
  - Ready-to-use interface elements such as calendars, dynamic tables, etc.
- Important to check which browser versions are supported!

Some frameworks

- Yahoo! User Interface Library
- Prototype/Scriptaculous/Rico
- jQuery/jQuery UI
- Qooxdoo
- Dojo
- see also some Web application frameworks (notably Google Web Toolkit)

Outline

Web applications

- Web is not only used to present textual information, but hosts real software applications

Examples

  - Messaging applications
  - Word processing and other office application
  - Mapping
  - Games
  - Multimedia content

- Generally speaking, applications designed to work only with graphical browsers
- HTML, CSS, and even JavaScript were not at all designed for that purpose: other technologies are (often) necessary

Possible dangers

- Limited accessibility. . .
- . . . justifiable for complex applications, not for text content!
- Loss of usual browser features (bookmarks, history, password management, etc.)
- What about other Web agents, notably search engine robots?
- Complexifying Web development (technology stacking), leading to possible bugs, design and maintenance issues
- Plug-ins to add to a browser: adding to potential security issues

Outline

The Java language

- Java: oriented-objet programming language, use for heavy applications
- Syntax close to JavaScript, but much stricter (strongly typed, all variables must be declared, etc.)
- Source compiled to virtual machine code
- Requires a development environment (or JDK) to transform source code into virtual machine code, and execution environment (or JRE) to run virtual machine code
- Reference implementation: Sun/Oracle, open source
- JRE available for many different platforms, including basic mobile smartphones
- Quite heavy!

Java applets

- Java program inside the browser window in a delimited area
- Integration via a browser plugin to install separately (available for all modern graphical browsers)
- Integration within HTML:

```html
<applet code="toto.class" width="300" height="400">
  <param name="param" value="value">
  <!-- Alternative content -->
</applet>
```

- `<applet>` is normally deprecated in favor of `<object>`, but no simple way to make Java applets work with `<object>` in all graphical browsers
- Access to the whole Java standard library (huge) and possible to use the numerous available third-party libraries

Trust system

Two trust level in an applet:

- By default, an applet (similarly as JavaScript, Flash...) is confined in a sandbox, does not have access to outside the browser (OS, hard drive content, etc.)
- A trusted applet (confirmation request made by the browser) will have on the other hand the same rights than regular software executed on the local machine

Pros/cons

- Quite old, initially designed as the way to have a complex application inside the browser
- Good support for various browsers and platformes, if the plugin is installed
- Quite heavyweight, applets take time to get initialized
- Trusted applets: good way to deploy a heavy application on a variety of platforms
- The language include graphics features, but not so easy to use in a Web context

Outline

Flash

- Adobe Flash (previously, Macromedia Flash)
- Made originally for vector animations: vector images + ActionScript (based on EcmaScript)
- Like Java applets, executes in a sandbox
- Flash programs compiled in a binary format (SWF)
- Correct but not great support:
  - no support for iOS and some other smartphones
  - not support for exotic Unix: AIX, HP-UX, OpenBSD, etc.
- SWF format description is public, but until May 2008, Adobe forbid to use it to create Flash players: no free software credible alternative to the official Adobe Flash player
- Development environment: pay-for and not cheap
- Flash development given up on for smartphones

Adobe Flex

- Programming language to create Flash applets by programming (and not graphically as with the IDE)
- Nothing changes for the user
- Free software compiler (but not IDE) offered by Adobe

Integration with HTML

With the `<object>` tag:

```
<object data="toto.swf"
     type="application/x-shockwave-flash"
     width="300" height="400" >
  <param name="param" value="value">
  Alternative content
</object>
```

Standard method, may requires painful workaround to support old IE versions (using `<embed>`)

Outline

Silverlight

- Microsoft technology, relatively recent (first version December 2006)
- Alternative to Flash, works on the same principle and using the same model
- Support very light (Windows and MacOS X mainly, not all browsers)
- Uses Microsoft's .NET architecture

ActiveX

- Microsoft technology
- Browser applets allowing a strong interaction with the OS
- No sandbox system, but plugin execution must be confirmed by the user
- Only on Windows, mainly on Internet Explorer
- Out of fashion nowadays, like Java applets that ActiveX was supposed to compete with

Canvas+JavaScript

- `<canvas>` tag defining a zone for bitmap drawing
- JavaScript to make this drawing dynamic
- Part of HTML5
- Supported in Safari, Opera, Firefox, Chrome, IE9
- Examples:
  - http://www.abrahamjoffe.com.au/ben/canvascape/
  - http://radikalfx.com/files/collage/demo.html
  - http://www.canvasdemos.com/

SVG/VML+JavaScript

- Vector image formats, processable in JavaScript
- Scalable Vector Graphics
- Vector Markup Language
- SVG: XML dialect, supported (to various degrees) in Firefox, Opera, Safari, Chrome, IE9; Adobe SVG Plugin for Internet Explorer ≤ 8
- VML: Internet Explorer only
- Still a bit experimental
- SVG+JavaScript: alternative to Flash, using open and standard formats
- For example used in Google Maps to display routes (VML in IE, SVG elsewhere)
- Example: `http://isthis4real.com/orbit.xml`

But also. . .

Shockwave	initially a counterpart to Flash in Macromedia's offer, now sometimes used for interactive games and 3D environments
VRML, WebGL	3D images and models
. . .	

Outline

HTML and multimedia content

- HTML only represents (structured) text and images (`<img>`)
- Nothing for audio and video! (except a non-standard `<bgsound>`, better to be forgotten)
- Standard solution: use the general mechanism of `<object>`
- HTML5 adds `<audio>` and `<video>` tags

Containers and codecs

Audio codecs	MPEG-1/I, MPEG-1/II, MPEG-1/III (MP3), AAC, AC3, RealAudio… Way audio data are compressed (with or without loss) by blocks
Video codecs	MPEG-2, H.264, Theora, RealVideo, DivX, VP8… Way video sequences are compressed (with or without loss) by blocks
Containers	WAV, ASF… (soybd), AVI, MPEG-2, MPEG-4, MOV, OGG, RealMedia, DivX, WebM… (sound+video) Way audio and video compressed by codecs are arranged in a file, with additional features to interlace audio and video, to allow random access to the video, etc.

Quite a mess! Hard to find software that supports all of this

Software patents

- Numerous software patents protecting the encoding or decoding according to given codecs (license granted against royalties)
- Not clear which patents are valid! Numerous companies claim a patent on MP3 compression or decompression MP3.
- Note: software patents only exist in Europe if they describe a industrial process. Unclear whether this applies to encoding/decoding audio and video
- Consequence: free software to play video, such as vlc, may be illegal! Small chance to see video and audio support for proprietary formats (H.264/MPEG-4) in free browsers
- Exempt from these issues:
  - Old formats (RIFF, MPEG-1...)
  - Formats designed to be free (Ogg, Vorbis, Theora, VP8, WebM); but some companies are afraid they might still violate submarine patents, not yet revealed!

Alternatives for audio and video on the Web (1/2)

- `<embed>` tag: non-standard, old, to be forgotten
- Using a plugin specific to a given media type (does not always work with IE, requires having this plugin!)

```
<object type="video/quicktime" data="test.mov"
       width="320" height="240">
  <a href="test.mov">test.mov</a>
</object>
```

- Using a specific ActiveX plugin (IE only)

```
<object classid="clsid:02BF25D5-8C17-4B23-BC80-D3488ABDDC6B"
       width="320" height="240">
  <param name="src" value="test.mov" >
  <a href="test.mov">test.mov</a>
</object>
```

Alternatives for audio and video on the Web (2/2)

- Simple link to the content (works if any software handling this content is installed, but no integration with the browser)

```
<a href="test.mov">test.mov</a>
```

- `<audio>` and `<video>` tags (HTML5, implemented in recent versions of Opera/Chrome/Firefox/Safari + IE9, promising, but uncertainty on formats; possible to reference several formats

```
<video src="test.mov" width="320" height="240">
  <a href="test.mov">test.mov</a>
</video>
```

  Demo: `http://htmlfive.appspot.com/static/video.html`
- Integrated into a Flash application, works… if Flash works

Best solution

- `<video>` tag for modern browsers with multiple formats (H.264 for Safari/Android, Theora for old versions of Chrome/Firefox/Opera, WebM for recent versions of Chrome/Firefox/Opera and IE9 when a plugin is installed)
- Flash if `<video>` is not supported
- A link if neither Flash nor `<video>` is supported

Putting it all together

```
<video id="movie" width="320" height="240" preload controls>
 <source src="pr6.webm" type='video/webm; codecs="vp8, vorbis"' />
 <source src="pr6.ogv" type='video/ogg; codecs="theora, vorbis"' />
 <source src="pr6.mp4" />
 <object width="320" height="240"
  type="application/x-shockwave-flash" data="flowplayer-3.2.1.swf">
  <param name="movie" value="flowplayer-3.2.1.swf" />
  <param name="allowfullscreen" value="true" />
  <param name="flashvars" value='config={"clip": {"url":
"http://wearehugh.com/dih5/pr6.mp4", "autoPlay":false,
"autoBuffering":true}}' />
  <p>Download video as <a href="pr6.mp4">MP4</a>, <a
href="pr6.webm">WebM</a>, or <a href="pr6.ogv">Ogg</a>.</p>
 </object>
</video>
```

From `http://diveintohtml5.info/video.html`

Outline

Conclusion

What you should remember

- CSS to display content
- JavaScript to animate it
- Better to separate structure/styling/animation as much as possible
- Many technologies for rich content
- Standard technologies, integrated to the browser, seem to have a more promising future than proprietary ones

References: Software

- Any text editor
- Commonly used versions of graphical browsers
- `http://browsershots.org/`
- No style function of browsers
- JavaScript console of browsers
- Firebug Firefox extension (console, DOM inspector, JavaScript debugger), or similar in other browsers
- CSS validator: `http://jigsaw.w3.org/css-validator/`

References: To go further

- CSS specifications:
  - `http://www.w3.org/TR/REC-CSS1`
  - `http://www.w3.org/TR/CSS21/`
  - `http://www.w3.org/Style/CSS/current-work`
- Acid CSS conformity tests:
  - `http://acid1.acidtests.org/`
  - `http://acid2.acidtests.org/`
  - `http://acid3.acidtests.org/`
- JavaScript standards (much less readable than HTML or CSS):
  - EcmaScript language, `http://www.ecma-international.org/publications/standards/Ecma-262.htm`
  - DOM specifications, `http://www.w3.org/DOM/DOMTR`
- *HTML5, Up and Running*, O'Reilly
- *HTML5, Canvas*, O'Reilly

Licence de droits d'usage

Contexte public } avec modifications