INTERFACES WEB

MASTER INFORMATIQUE

SPÉCIALITÉ ANDROIDE

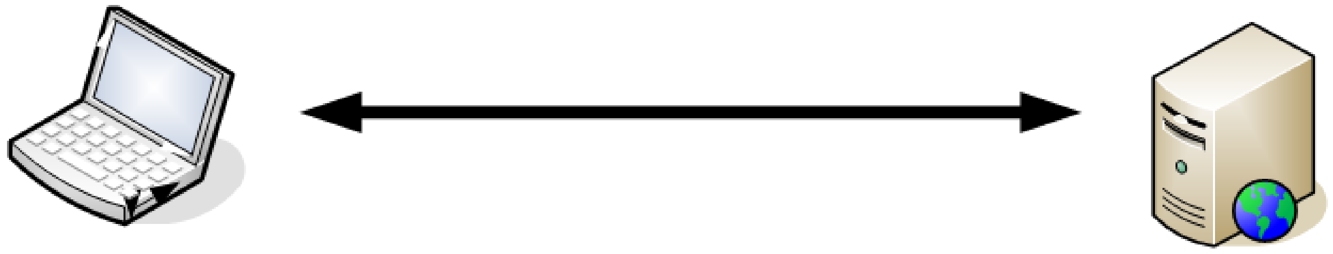
UE IHM

UPMC/TELECOM PARISTECH

2017

THE WEB

A CLIENT/SERVER ARCHITECTURE



- A variety of clients are used:
  - graphical browsers: destkop, smartphone, embedded, ...
  - textual browsers: w3m, lynx ...
    - used by visually-impaired people when sites are accessible
  - browsers with speech-synthesis engines
  - crawlers, spiders, robots ...
- Servers deliver web content to the clients:
  - static content (pages, images, ...)
  - dynamically generated content (php, js, asp, ...)
- Architectural choice: light-client/heavy-client

WHAT IS WEB CONTENT?

- Textual, visual or aural content experienced when using a browser
  - «Web Page»
  - «Web Site»
  - «Web Application»
- A mix of multiple languages and file formats
  - Used by the client (don't mix with server-side languages)
  - Each with its own usefulness (HTML, CSS, JS ...)
  - Hierarchically nested: e.g. CSS content in HTML content
  - Referencing each other: hyperlinks, e.g. JS content referenced from HTML content
- Statistics

LANGUAGES OF THE WEB

- HTML / XHTML
  - Content structuration
  - Basic rendering
- CSS
  - Presentation instructions to render the HTML content
  - Layout, animations, ...
- SVG
  - Presentation instructions to render rich graphical content
- JS
  - Programmatic behavior to be added to HTML or SVG content
- XML
  - Data exchange, validation, ...
- JSON
  - Data exchange

EXAMPLE OF MIXED LANGUAGES

```
<!DOCTYPE html>
<html>
 <head>
  <title>Hello</title>
  <script>
   window.onload=function(e) { alert("Page loaded!"); };
  </script>
  <style type="text/css">
   body { width: 30%; margin: auto; }
   p {
    font-size: 30px;
    font-family: sans-serif;
   }
  </style>
 </head>
 <body>
  <p>
   <img src="../pesto/image01.png" style="float: left; margin-right: 5px" onclick="alert('He
   This is a simple image, but next is a vector graphics image
   <svg style="float: right; width: 100px; height: 100px">
    <rect rx="5" width="50" height="50" fill="lightblue" onclick="alert('Rect click');"/>
   </svg>
  </p>
 </body>
</html>
```

TOOLS TO CREATE WEB CONTENT

- General purpose tools
  - Text Editor (Atom, Sublime …)
  - Integrated Development Environment (Visual Studio, Eclipse …)
- Specific Tools
  - DreamWeaver
  - Brackets
  - Aptana
  - …
- Code playgrounds
  - JS Fiddle
  - Code Pen
  - CSS Deck
  - JS Bin

TOOLS TO DEBUG

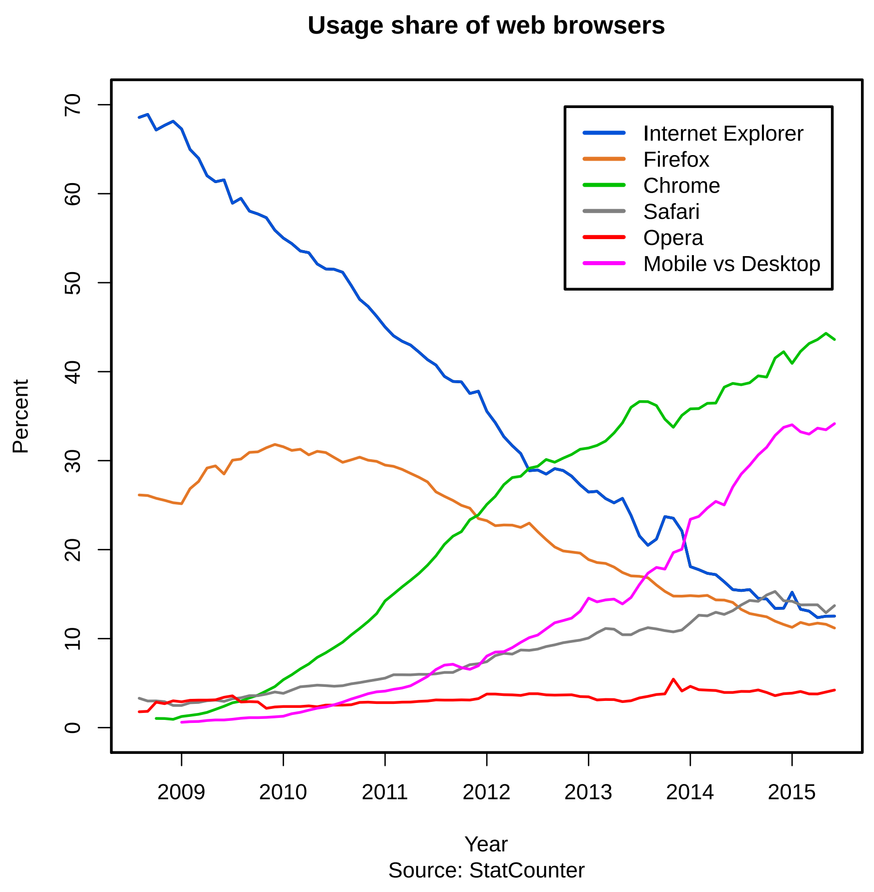
- What kind of debugging ?
  - Source Code Inspector: HTML, CSS, ...
  - Advanced Inspectors: DOM Tree, CSS Rules
  - JavaScript debug: step-by-step, breakpoints, ...
  - Network monitoring: requests, timing, ...
  - Performances: frame rate, CPU, smoothness, memory ...
- Browser-integrated debugger (F12 / Cmd + Opt + I on Mac)
  - Chrome Dev Tools
  - Mozilla Firebug
  - Microsoft Developer Tools
  - Safari Developer tools

WHAT IS A BROWSER?

- Processing of Web Resources
  - **Downloading** of HTML/JS/CSS/Images/Videos ... using Internet Protocols
    - Sequential/Syncronous vs. Parallel/Asynchronous
  - **Rendering** (aural and visual)
- Handling **dynamicity**
  - Reacting to **user interactions**
    - Navigation, Click, ...
  - Reacting to **network conditions**
    - TCP Congestion, Streaming, ...
  - Processing **animations**

BROWSERS CATEGORIES

BROWSER WARS

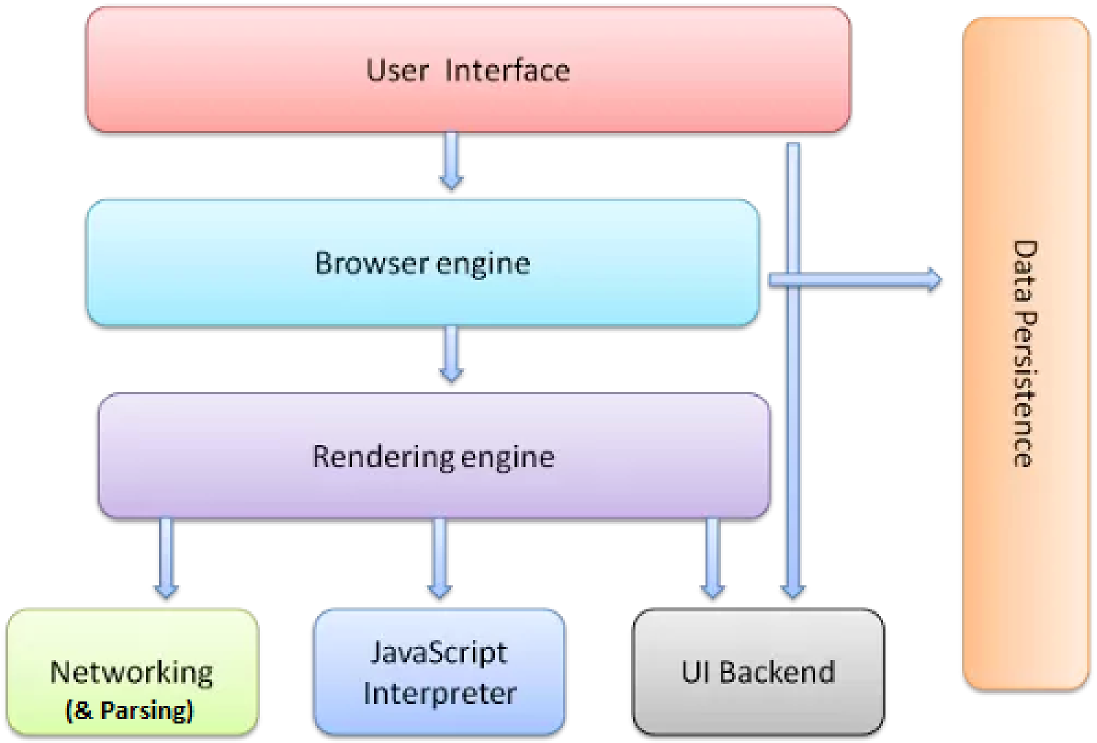


BROWSER HISTORY

- Long history of browsers
- Rapid evolution recently
    - Next versions of major browsers very often
        - Ex: Chrome release a new version every 6 weeks
        - Ex: Firefox 5 (June 2011), Firefox 25 (Oct. 2013)

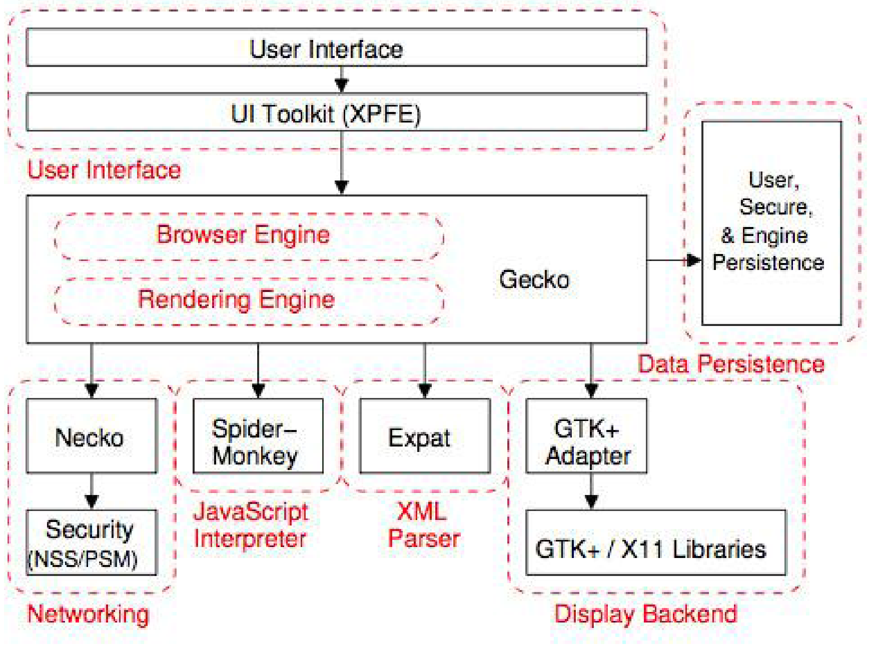
- Browsers are converging in standards support
    - But still have differences (see sites like CanIUse.com or QuirksMode)

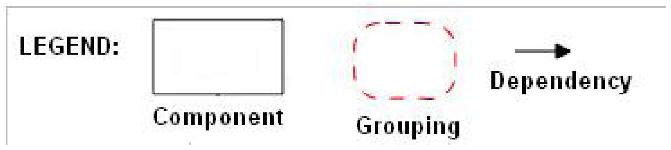
BROWSERS SIMPLIFIED ARCHITECTURE



HTML5 Rocks

FIREFOX ARCHITECTURE



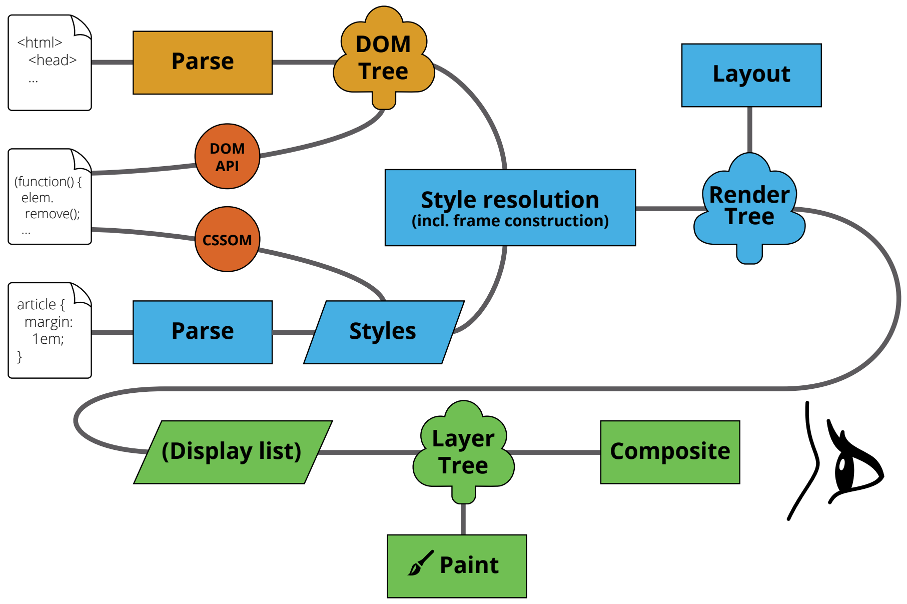


BROWSERS COMPONENTS

Browser	Rendering Engine	Scripting Engine
Edge	EdgeHTML	Chakra
Internet Explorer	Trident	Chakra
Firefox and alike (IceWeasel, Seamonkey...)	Gecko	(Spider)Monkey
Safari	WebKit	JavaScript Core
Chrome	Blink (previously WebKit)	V8
Opera	Blink (previously Presto)	V8 (previously Carakan)

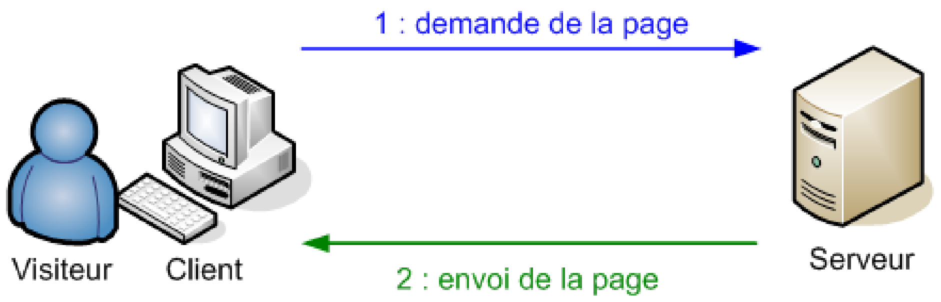
Browser Rendering Engine Scripting Engine

BROWSER PROCESSING



See Mozilla's presentation

HTTP



- Hyper Text Transfer Protocol, standardized by IETF
- Application protocol at the basis of the World Wide Web
- history & versions:
  - HTTP (1991, proposed by Tim Berners-Lee),
  - HTTP/1.0 (1996, initial version, RFC 1945),
  - HTTP/1.1 (1997, current deployments, RFC 2068 and 2616),
  - HTTP/2.0 (2015, latest version, in deployment, RFC 7540)
- Client/server protocol
  - The client is a "User-Agent" (Firefox, wget, curl ...)
  - HTTP servers: Apache, Microsoft IIS, node.js, ...
- Protocol used to download resources
  - identified by a URL

URL

UNIFORM RESOURCE LOCATOR

- Initially standardized by IETF (new versions in development by IETF/W3C/WHATWG)

`https`	`://`	`www.example.com`	`:443`	`/path/to/doc`	`? name=foo&town=bar`	`#para`
scheme		hostname	port	path	query	fragment

- scheme:
  - way the resource can be accessed; generally http or https, but also ftp, data, ws, wss, ...
- hostname:
  - domain name of a host (cf. DNS); hostname of a website may start with www., but not a rule.
- port:
  - TCP port; defaults: 80 for http and 443 for https
- path:
  - logical path of the document for the server, with/without extensions or with extensions corresponding to content generated content (e.g. php)
- query string:
  - optional additional parameters (dynamic documents)
- fragment:
  - optional subpart of the document

RELATIVE URLS

- with respect to a context (e.g., the URL of the parent document, the **base URL**):
- If context is : `https://www.example.com/toto/index.html`

relative URL	**Absolute URL**
/titi	https://www.example.com/titi
tata	https://www.example.com/titi/tata
#tutu	https://www.example.com/index.html#tutu

IDENTIFYING WEB RESOURCES

- File/URL extension
  - Resources may not have one, or it may be wrong
    - Ex: http://www.example.org/
    - Ex: http://www.example.org/generate.cgi?user=12
  - Not reliable!
- Sniffed type
  - E.g. use of 'magic number' (registered in MIME type)
    - Ex: "47 49 46 38 37 61" GIF89a
  - E.g Detection of file header (XML)
  - May be abused
- MIME type or Internet Media Type
  - Used in HTTP Content-Type header
  - '/' (';' parameters)*
  - 5 major types: audio, video, image, text, application
  - Subtypes specific to a payload ('x-…' are proprietary)
  - Should be trusted

HTTP MESSAGES

- Message = Header + Body
  - Textual header (not necessarily for the resources)
- Message type = Requests or responses
  - Request=Method+URL+ProtocolVersion+Header(+data)
  - Method
    - GET
    - POST
    - HEAD
    - OPTIONS
    - PUT
    - DELETE
    - TRACE
    - CONNECT
    - PATCH
  - Response=ProtocolVersion+Response Code+Header+Resource

GET

- Simplest type of request.
- Possible parameter are sent at the end of a URL, after a '?'
  - Not applicable when there are too many parameters, or when their values are too long (total length < 2000 chars).

- Example:
  - URL in the browser

```
http://www.google.com/search?q=hello
```

  - Corresponding HTTP Request

```
GET /search?q=hello HTTP/1.1
Host: www.google.com
```

POST

- Method only used for submitting forms.
- Example:

```
POST /php/test.php HTTP/1.1
Host: www.w3.org
Content-Type: application/x-www-form-urlencoded
Content-Length: 100
type=search&title=The+Dictator&format=long&country=US
```

- By default, parameters are sent using: `name1=value1&name2=value2`
  - special characters (accented characters, spaces...) are replaced by codes such as +, %20
  - This way of sending parameters is called `application/x-www-form-urlencoded`.
  - Also used with GET requests in the URL: `http://www.example.org?name1=value1&name2=value2`

- For the POST method, another heavier encoding can be used (several lines per parameter)
  - similar to the way emails are built: mostly useful for sending large quantity of information.
  - Encoding named `multipart/form-data`.

RESPONSE CODES

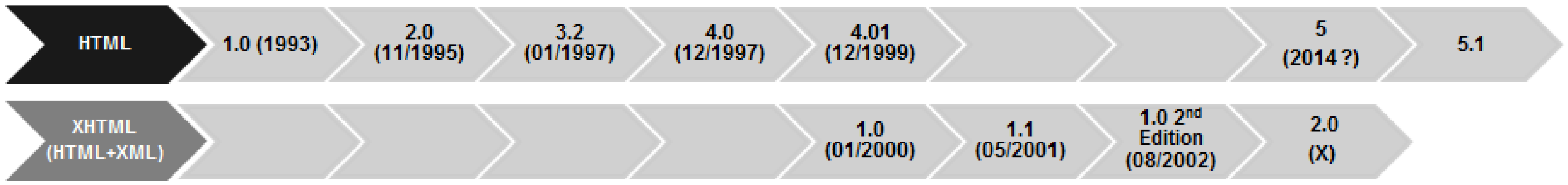
- Success (2xx)
  - OK (200)
  - …
- Redirections (3xx)
  - Permanent redirection (301)
  - Temporary redirection (302)
  - No modification (304)
  - …
- Request Errors (4xx)
  - Bad request (400)
  - …
  - Forbidden(403)
  - Not found (404)
- Server Errors (5xx)
  - Internal Error (500)
  - …

HYPERTEXT MARKUP LANGUAGE

HTML

HTML

A BIT OF HISTORY



- Initial version created by Tim Berners-Lee in 1989
  - as an open language, royaltee-free
  - Then developped by the World Wide Web Consortium (W3C)
  - Now developped by W3C and WHATWG
- Several versions
  - HTML 4 Strict, Transitional, Frameset
  - HTML vs. XHTML
  - HTML 5 (HTML.next, HTML 5.1)

HTML 5

THE LANGUAGE

- 1 language, 2 syntaxes
  - **HTML**, identified by documents of type `text/html`
  - **XHTML** (XML), identified by `application/xhtml+xml`
  - Similar syntaxes but different processing (e.g. +/- strict)
- Text-based (not binary, as opposed to Flash)
  - mix of **tags** (markup) and text
  - no compilation step
  - can easily view the source code
- Presentation agnostic
  - Might be rendered by different renderers (screen, printer, text-only, speech, ...)
  - Rendering can be configured via **CSS**
- Basic Interactivity (navigation, forms)
  - Advanced Interactivity to be provided by **JS**
- Associated with a tree representation and JS APIs: **DOM**

HTML 5

TAGS

- start (opening) tag :

```
<mytag>
```

- end (closing) tag:

```
</mytag>
```

- Tags should be closed
  - in XML-compatible syntax: always
    - in particular with self-closing tags:

```
<mytag/>
```

  - in non-XML syntax: most of the time
    - except for some tags (historical reasons): `img`, `br`, `input`, ...

```
<div><br></div>
```

  - must be closed in the right order

```
<a><b></a></b> // wrong
<a><b></b></a> // correct
```

- Tags structure the content of an HTML document into a tree: the DOM Tree

HTML 5

ATTRIBUTE

- An attribute indicates a property of a DOM element
  - specified on the corresponding start tag or self-closing tag

```
<mytag property-name='property-value'></mytag>
<mytag property-name='property-value'/>
```

  - Using quotes " or single-quotes '

```
<mytag name="value"></mytag>
<mytag name='value'/>
```

  - Possibly with nested quotes

```
<mytag name="value with 'inside'">
<mytag name='value with "inside"'>
```

  - Aternate HTML 5 syntaxes (not XML-compatible)

```
<mytag name=value> <!-- no quote or single-quote needed when value has no space --
```

```
<mytag name> <!-- no =, no value when the attribute is boolean or can have only one
```

HTML 5

ATTRIBUTES

- Multiple attributes can be specified:
  - space separated

```
<mytag attr1="value1" attr2="value2">
```

  - order is not important

```
<mytag attr2="value2" attr1="value1"> <!-- equivalent -->
```

  - cannot duplicate the same attribute twice

```
<mytag attr1="value2" attr1="value1"> <!-- wrong!! -->
```

HTML 5

A LARGE STANDARD

- Defines many tags
    - Paragraphs, Tables, Forms
    - Multimedia: images, videos, audios
    - Graphical Primitives
    - ...
- Defines JavaScript APIs
    - Basic document manipulations
    - Element-specific APIs (e.g. video)
    - Advanced APIs (Offline Storage, Database, communications)
    - ...
- Defines how to integrate with other Web technologies
    - Mix of SVG and MathML within the HTML page
- ...

HTML 5

HELLO WORLD!

- As simple as that!

```
Hello World!
```

- Browser's parsing algorithms are very robust (tag soup)
  - Will create the page structure for you!
  - Will try to close tags for you!
  - ...

HTML 5

BASIC PAGE STRUCTURE

```html
<!DOCTYPE html>
<html lang="en">
<head>
<title>This is the title</title>
</head>
<!-- this is a comment -->
<body>
<!-- visible content goes here -->
</body>
</html>
```

HTML 5

HEADER

- The header of a document is delimited by the head tags.

```
<head> ... </head>
```

- The header contains meta-informations about the document, such as its title, encoding, associated files, etc.
- Some common items are:
  - metadata
    - The character set of the page, usually at the very beginning of the header (not reliable)

```
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
```

  - The title of the page, displayed in the title bar of Web browsers.

```
<title>My great website</title>
```

  - Javascript & CSS links

```
<script src="...">
<link src="...">
<style >
```

HTML 5

BODY

- The content of the document is delimited by the body tags.

```
<body> ... </body>
```

- The body is structured into sections, paragraphs, lists, etc.

HTML 5

BODY CONTENT

- Typically uses tags describe sections, by decreasing order of importance:

```
<h1>Title of the page</h1>
```

```
<h2>Title of a main section</h2>
```

```
<h3>Title of a subsection</h3>
```

```
<h4>Title of a subsubsection</h4>
```

- Or paragraphs of text:

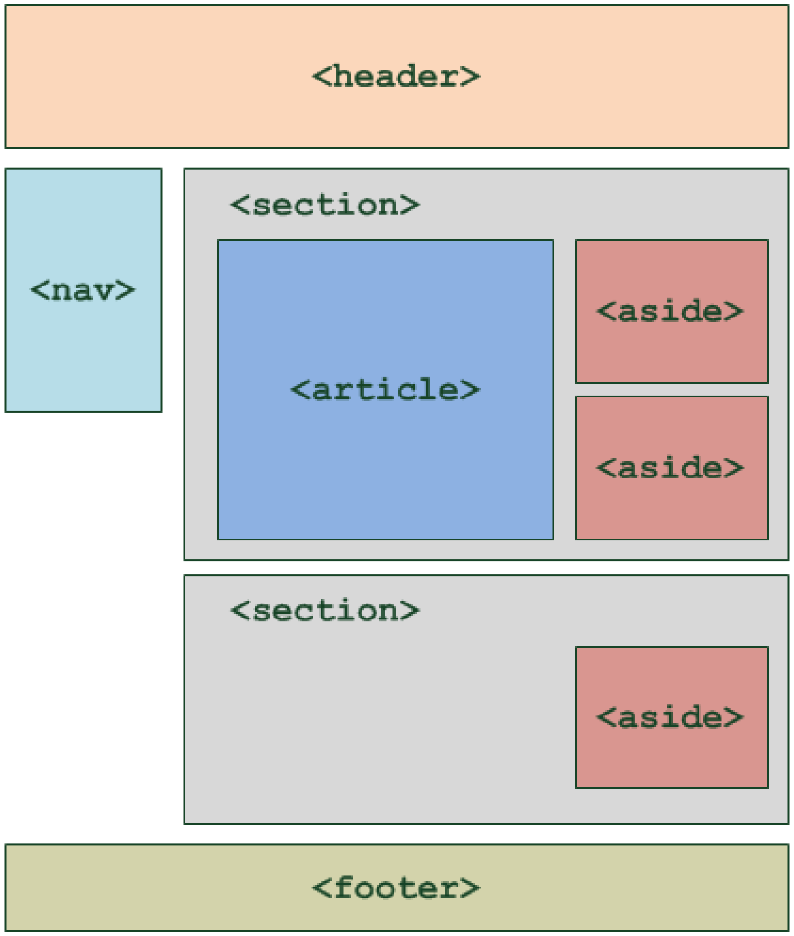
```
<p> ... </p>
```

- Or simple grouping elements without semantics:

```
<div> ... </div>
```

HTML 5

STRUCTURED BODY OF A PAGE



HTML 5

LINKS

- What differetiates Web pages (hypertext pages) from normal documents: links!
- Introduced with `<a> ... </a>`
- Navigating a link can bring to:
  - a resource on another server or another file of the same server

```
<a href="http://www.cnrs.fr/"> <!-- Absolute URL -->
  <img src="../pesto/images/cnrs.gif" alt="CNRS">
</a>
<a href="bio/indexbioinfo.html">Bioinformatics</a> <!-- Relative URL -->
```

  - another part of the same document with anchors

HTML 5

ANCHORS

- Anchors serve to reach a precise point in the document.
- They are defined, either on an existing tag by using the `id` attribute, or with an `<a id="...">`

```
<h3 id="tutorials">Tutorials</h3>
<!-- or -->
<a id="tutorials">
```

- Then, one can link to this anchor:

```
<a href="#tutorials">tutorials</a> <!-- relative URL linking to the same document -
<a href="http://www.w3.org/#tutorials">tutorials</a> <!-- absolute URL linking to a
```

HTML 5

LISTS

- Unordered lists

```
<ul>
<li>First bullet point</li>
<li>Second bullet point</li>
</ul>
```

• First bullet point
• Second bullet point

- Ordered lists

```
<ol>
<li>First ordered point</li>
<li>Second ordered point</li>
</ol>
```

1. First ordered point
2. Second ordered point

HTML 5

TABLES

```html
<table>
 <tr>
  <td>row 1 - column 1</td>
  <td>row 1 - column 2</td>
 </tr>
 <tr>
  <td>row 2 - column 1</td>
  <td>row 2 - column 2</td>
 </tr>
</table>
```

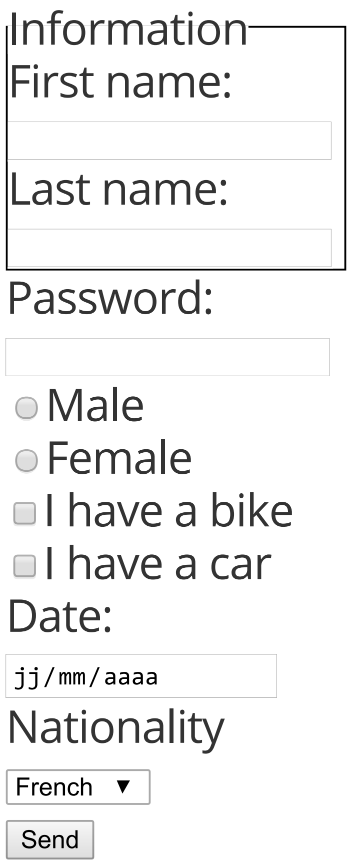
row 1 - column 1	row 1 - column 2
row 2 - column 1	row 2 - column 2

Other options: `th`, `caption`, `thead`, `tbody`, `tfoot`, `col`, `colgroup`

HTML 5

FORMS

```
<form  action="demo_form_action.asp" method="get">
 <fieldset>
  <legend>Information</legend>
  First name: <input type="text" name="firstname"><br>
  Last name: <input type="text" name="lastname"><br>
 </fieldset>
 Password: <input type="password" name="pwd"><br>
 <input type="radio" name="sex" value="male">Male<br>
 <input type="radio" name="sex" value="female">Female<br>
 <input type="checkbox" name="vehicle" value="Bike">I have a bi
 <input type="checkbox" name="vehicle" value="Car">I have a car
 Date: <input type="date" name="date"><br>
 Nationality: <select name="nationality">
  <option value="french">French</option>
  <option value="English">English</option>
 </select>
 <input type="submit" value="Send">
</form>
```



Information
First name:
Last name:
Password:
Male
Female
I have a bike
I have a car
Date:
jj/mm/aaaa
Nationality
French ▼
Send

Other options: colors, time, …

HTML 5

NESTED DOCUMENTS

- Render the content of another page in the current page
- Using `<iframe>` tags

```
<iframe width="400" height="215" frameborder="0
          scrolling="yes" marginheight="0" marginwidth="0"
          src="http://www.telecom-paristech.fr/...">
</iframe>
```

(formation-et-innovation-dans-le-numerique.html)

English (http://www.telecom-paristech.fr/eng)

(http://www.telecom-paristech.fr)

À l'affiche (actualites/affiche.html)

Best Paper Award pour des chercheurs de Comelec à la conférence Modelsward (nc/actualites/actualite.html?idactus=3035)

(nc/actualites/ac idactus=3035)

MOOC ABC du langage C (https://www.fun-mooc.fr/courses/MinesTelecom/04019S02/session02/about)

(https://www.fun-mooc.fr/courses

À vos agendas (nc/actualites/agenda.html)

Lancement du cycle d'événements sur la confiance à l'ère numérique : la Confiance, une valeur en mutation dans la société au NUMA (https://www.fondation-telecom.org/news/lancement-du-cycle-devenements-sur-la-confiance-a-lere-numerique/)
mercredi 1 mars 2017

Merci d'accepter les cookies pour de meilleurs contenus et services (détails sur mentions légales)
Je comprends () Mentions légales (/c-mentions-legales.html)

(https://www.fon telecom.org/new

HTML5

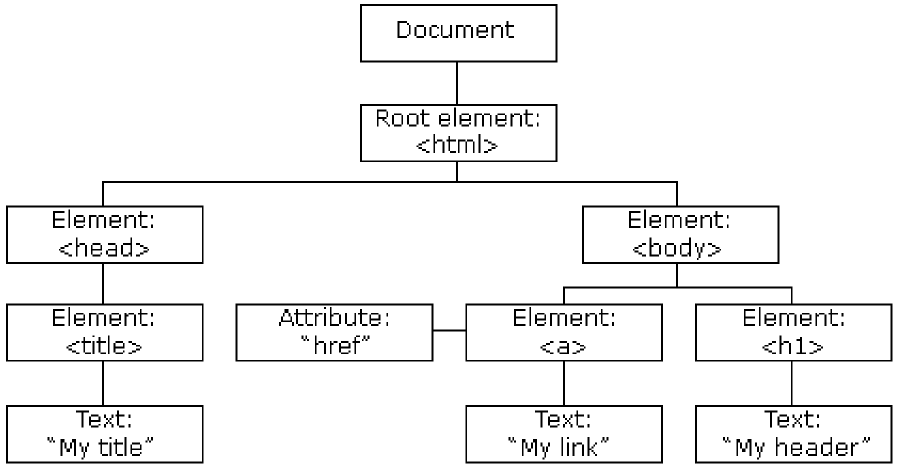
DOCUMENT OBJECT MODEL

- Tree-based representation of an HTML document
    - DOM Node=
        - DOM Text node
        - DOM Comment node
        - DOM Element
        - DOM Attribute

- DOM Nodes, DOM Elements … can be manipulated by script via specific interfaces

HTML 5

DOM TREE EXAMPLE



```
<html>
 <head>
  <title>My title</title>
 </head>
 <body>
  <a href="http://....">My link</a>
  <h1>my header</h1>
 </body>
</html>
```

Simplified tree: be careful of DOM Text nodes

CASCADING STYLE SHEETS

CONCEPT

- Language used to associate **styles** to documents
  - Companion specification to HTML
    - But can be applied to any document structured with a tree (e.g. HTML, XML, SVG)

- Separation CSS / HTML
  - To manage **presentation aspects** (CSS) separately from **structural aspects** (HTML)
  - To present the content differently to different users using different CSS
  - To present different HTML content with the same presentation aspects, same CSS

- Demonstration
  - Deactivate CSS

A BIT OF HISTORY

- CSS 1.0 (1996)
- CSS 1.0 (2nd ed., 1999)
- CSS 2.1 (2011):
  - Stable version, implemented interoperably by browsers
- CSS 3:
  - Modular specification of CSS 2.1
  - Many additions (50+ modules, see list of specifications)
  - Partly implemented by browsers

PRINCIPLES

- Language based on **rules** to be associated with document elements
- Each rule sets some **properties** on some elements
  - A rule is one or more **selectors** and a **declaration block** (block of properties)

- Types of properties (more than 400 defined)
  - Visual properties (`background-*`, `border-*`, ...)
  - Text properties (`text-*`, `font-*`, `color`, ...)
  - Box properties (`padding-*`, `margin-*`, ...)
  - other properties (`visibility`, `display`, `z-index`, ...)

- **Style Sheet**
  - A set of rules in a separate file is a style sheet
  - Multiple style sheets can be applied to a document
    - Author style sheets
    - User style sheets
    - Device Style sheets

DECLARATION OF PROPERTIES

- each property is declared using the syntax: property_name + ':' + value

```
font-weight: 600         /* property with a unitless number value */
```

```
font-size: 16px          /* property with a number value with units */
```

```
width: 99%               /* property with a percentage value */
```

```
background-color: red   /* property with a keyword value */
```

```
font-family: 'Arial'    /* property with a string value */
```

```
background-image: url('http://my.server.com/clear.png') /* property with a complex
```

- use of ; to group properties applying to the same element(s)

```
background-color: red; font-size: 16px;
color: red;
width: 50%;
```

CSS UNITS

- Size and position units
  - Absolute units
    - `px`
    - `pt`, `pc`, `cm`, `mm`, `in`
      - 1in = 2.54cm = 25.4mm = 72pt = 6pc
  - Relative units
    - percentage units (%)
    - Font-relative units
      - `em`,`ex`,`ch`,`rem`
    - Viewport relative units
      - `vw`,`vh`,`vmin`,`vmax`
- Other units
  - `deg`,`grad`,`rad`,`turn`
  - `s`,`ms`
  - `Hz`,`kHz`
  - `dpi`,`dpcm`,`dppx`

SELECTORS

- Select to which element(s) a block of properties apply (using `{ }`)
  - Selecting elements in the document tree by tag name

```
p { /* these properties apply to all p elements in the page */
  border-style:solid;
  border-width:5px;
}
```

  - Selecting using multiple tag names (separated by a comma)

```
h1, em { /* these properties apply to all h1 and em elements in the page */
  color: blue;
}
```

SELECTORS - MORE

- Addressing of 1 specific element in the document tree by `id` attribute using #

```
<!-- HTML -->
<p id="p1">text 1</p> <!-- each paragraph has a unique id attribute -->
<p id="p2">text 2</p>
```

```
/* CSS */
#p2 { /* this property applies to the element whose id is p2 */
    color: red;
}
#p1 { /* this property applies to the element whose id is p1 */
    color: blue;
}
```

- Addressing of several specific elements by `class` name using `.`

```
<!-- HTML -->
<!-- each paragraph has a class attribute with one or more class values -->
<p class="pType1">text 1</p>
<p class="pType1">text 2</p>
```

```
/* CSS */
.pType1 { /* this property applies to all elements whose class attribute contains p
    color: blue;
}
```

LINKING CSS CONTENT WITH HTML CONTENT

- Via the `style` attribute (**inline stylesheet**)
  - Styles attached to a given element (*syntax without selector*)

```
<p style="color:red;">text</p>
```

  - should be avoided

- Via the `style` element (**internal stylesheet**)
  - Styles attached to a given document

```
<head>
 <style>
 p { color: red; }
 </style>
</head>
```

  - should be avoided

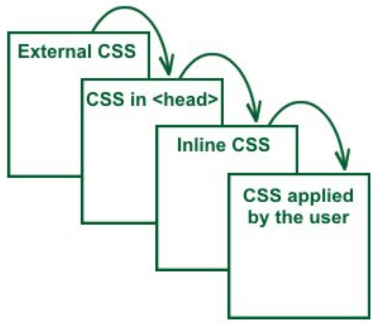
- Via an external stylesheet (separate file)
  - Styles can be attached to a document

```
<link href="file.css" type="text/css" rel="stylesheet"/>
```

- should be preferred

CSS CASCADE

- If different rules conflict (e.g. when multiple style sheets are used)
- The rule that has precedence is determined by:
  - media type of style sheet
  - origin of rule (user agent, user, author, !important author, !important user)
  - specificity of the selector
  - order in file



EXAMPLE OF A CSS PROPERTY DEFINITION

- The **border-top-width** property

Syntax:
<length> | thin | medium | thick
Definition:

Initial value	`medium`
Applies to	all elements. It also applies to `::first-letter`.
Inherited	no
Media	visual
Computed value	the absolute length or `0` if `border-top-style` is none or `hidden`
Animatable	yes, as a length

CSS INHERITANCE

- For a given element, if the value for a given property is **not specified**, the value is obtained as follows:
  - if the property is "inheritable" (i.e. "inherited: yes" in its definition),
    - if the element has a parent in the DOM tree, the **computed value** on that parent is use

      ```
      p { color: green }
      ```

      ```
      <p>The text and the span will be <span>green</span> because 'color' is inheritable.
      ```

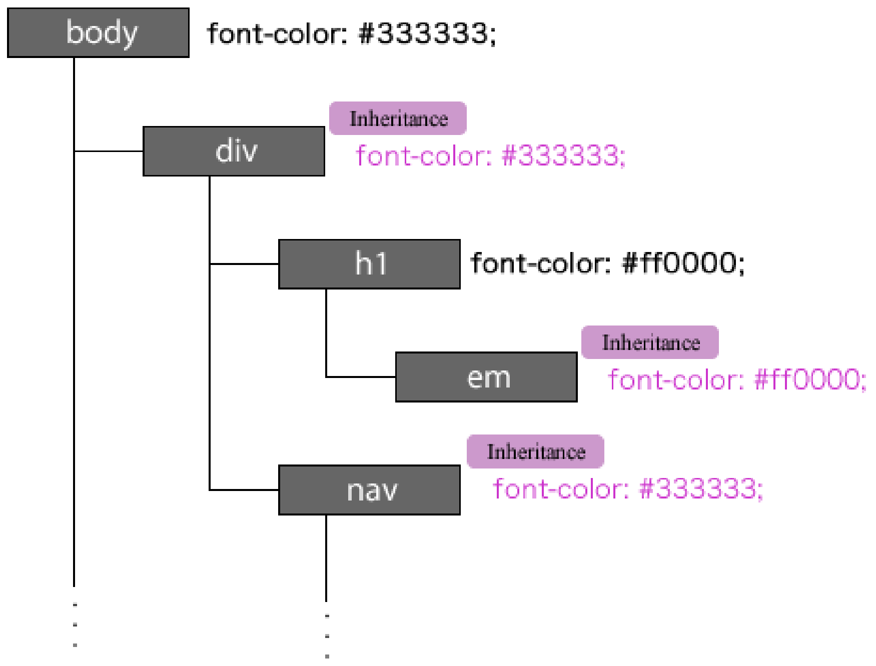
    - otherwise (for the root), the **initial value** is used.
  - if not (i.e. "inherited: no"), the **initial value** is used

    ```
    p { border-width: 1px }
    ```

    ```
    <p>Only the text will have <span>a border</span> because 'border-width' is not inhe
    ```

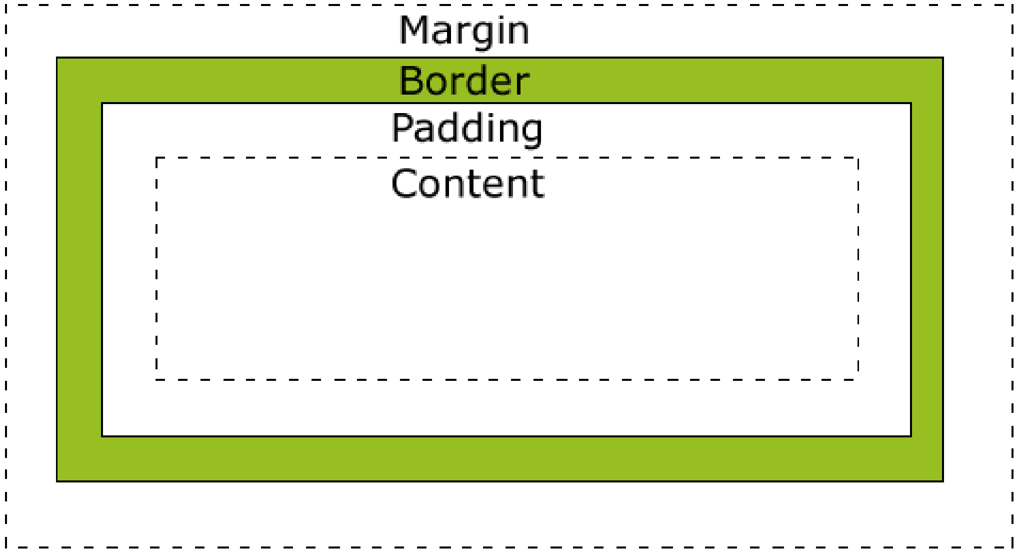
- The computed value is obtained:
  - by converting a relative value (when possible) to an absolute value
  - otherwise (% values when layout is involved), using the relative value

CSS INHERITANCE



THE CSS BOX MODEL

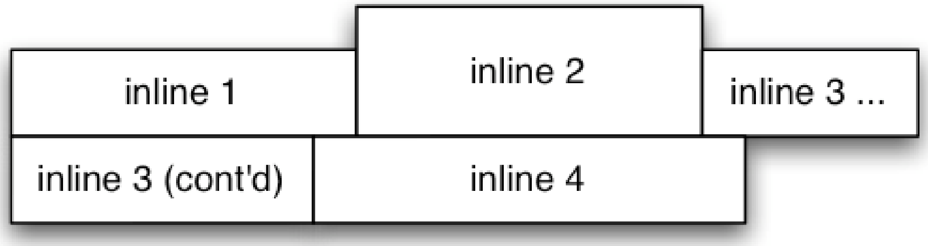
- Each element in the DOM produces zero, one or several boxes depending on the type of element
  - The page rendering consists in displaying those boxes
- Each box has generic properties that controls some generic aspects: margin, border,

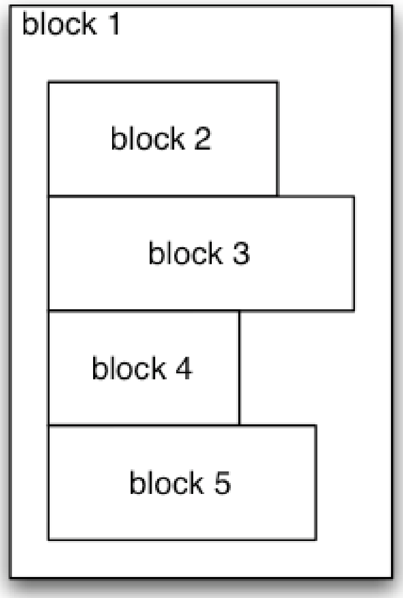


padding
- The layout (size and position) of a box depends on multiple factors:
  - The size of the box and of its content (e.g. images)
  - The type of box (block, inline, ...)
  - The positioning scheme: normal, absolute, float
  - The other elements and boxes around (siblings, parent, containers)
  - The viewport (e.g. the window size)

CSS BOX TYPES

- There are 2 main types of boxes:
  - **block** boxes: Boxes that don't display on the same line as the previous box and as the next box
    - Sizing properties such as `width` and `height` can be used.
  - **inline** boxes: Boxes that stay on the same line as the previous box and the next box (when possible)





- The type of box is defined by the standard:
  - block boxes: p, div, h1, h2, footer ...
  - inline boxes: a, img, span ...
- The default type can be overriden by the **display** property

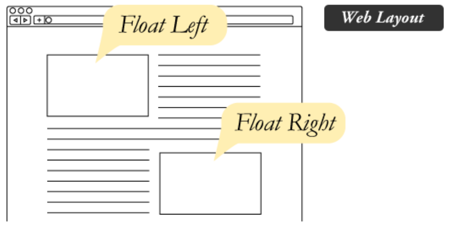
```
<p>A first par</p>
<p>A second par</p>
<a>A first link</a>
<a>A second link</a>
```

```
p { display: inline; }
a { display: block; }
```

CSS POSITIONING SCHEMES

- CSS defines the `position` property with the values
  - `static`: default value
  - `relative`: moved compared to its original position (initial place left empty)
  - `absolute`: positioned relative to the origin of the parent box
  - `fixed`: positioned relative to the window

- Floats



- z-index

RESPONSIVE DESIGN

- Principles
  - Design pages that adapt to the screen size using CSS Media Queries

CSS MEDIA QUERIES

- Adapt the CSS rules to apply based on client characteristics
  - Screen size, aspect-ratio, resolution or orientation
  - Type of device (pc, mobile, printer ...)
  - Number of colors

```
<link rel="stylesheet" media="screen and (max-width: 1280px)" href="file.css" />
```

or

```
<link rel="stylesheet" href="file-with-mediaqueries.css" />
```

```
@media screen and (max-width: 1280px)
{
    /* SomeCSS ruleshere */
}
```

AUTHORING CSS

- Many web sites offer free CSS templates
  - http://www.free-css.com/
  - http://templated.co/
  - …
- CSS tools
  - Pre-processors to generate CSS
    - SASS
    - LESS
  - WYSIWYG editors
    - BlueGriffon
    - SelfCSS
  - Responsive front-end frameworks
    - Bootstrap
    - Foundation

JAVASCRIPT

A.K.A. ECMASCRIPT

- What is ECMAScript?
  - Programming/Scripting Language
  - Interpreted code (not compiled into machine code), Portable code
  - Standard syntax
  - Invented by Brendan Eich at Netscape (and Microsoft JScript)
- Versions
  - JavaScript 1.5-2.0
  - ECMA-262 3rd, (4th), 5th, 6th (2015), 7th edition (draft)
- In the Web Browser: JavaScript
  - Executed by the JavaScript engine of the browser according to a model
  - Used with specific interfaces (DOM, ...)
- More: Tutorial Videos by Douglas Crockford

JAVASCRIPT BASICS

Reminders of simple JavaScript

- How to declare/assign a variable?
- How to define a function?
- How to call a function?
- Arrays
- Strings
- Objects
- Properties

BROWSERS AND JAVASCRIPT

- The JavaScript Engine is a core component of browsers
  - Used for:
    - Interactivity, animations, media manipulations (Canvas, audio API, ...)
  - Potential problems
    - Security
    - Performance

WEB APPLICATIONS=

- HTML +
  - Document structure
  - Textual content and media resources (images, ...)
- CSS +
  - Presentation information
- JavaScript (=ECMAScript + Web APIs)
  - Browser-interpreted code to provide the intelligence, behavior of the application

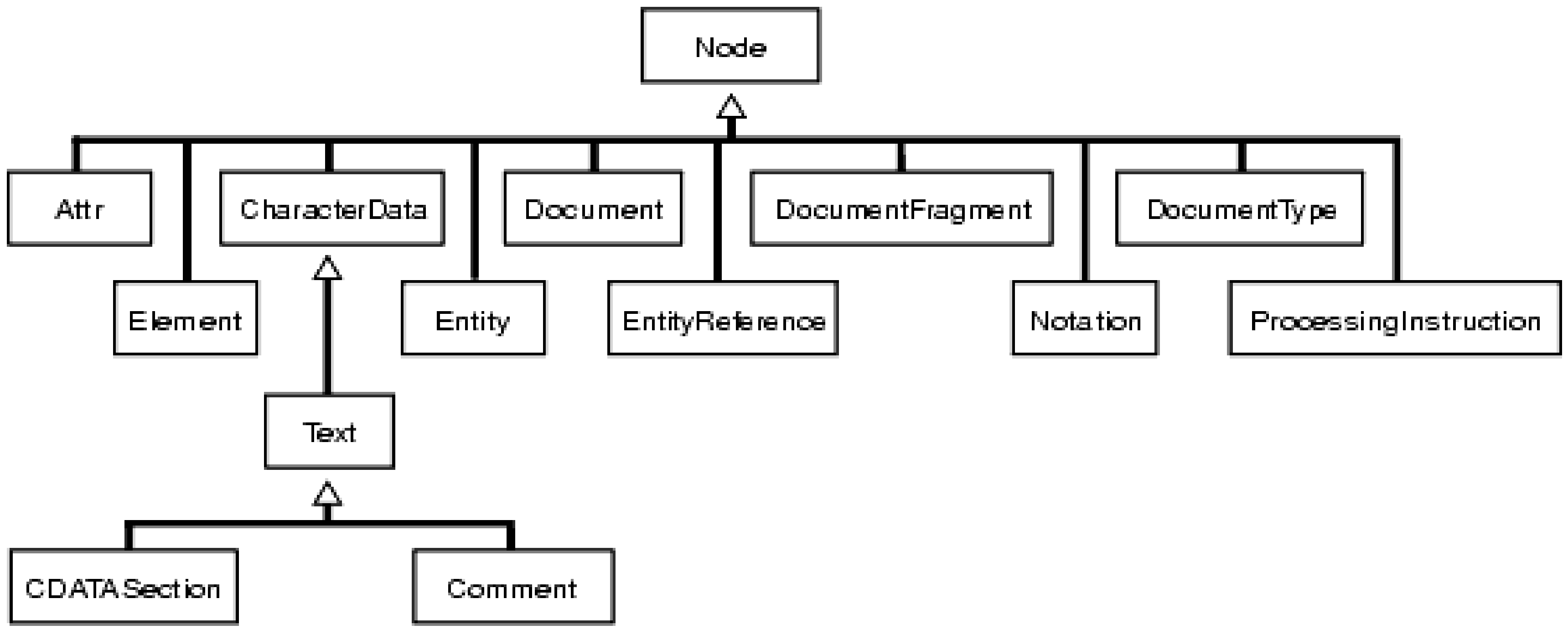
“THERE IS A WEB API FOR EVERYTHING”

- Basic APIs
  - Document Object Model (DOM): Core, Events, Window, ...
- Specific APIs
  - Communication APIs
    - XHR, Push, WebSockets, ...
  - Drawing APIs
    - Canvas, WebGL, ...
  - Storage APIs
    - Files, Cookies, Database, ...
  - Multimedia APIs
    - Audio, video, streaming, ...
  - Device APIs
    - Battery, AdressBook, WebCam ...
  - System APIs

DOCUMENT OBJECT MODEL (DOM) INTERFACES

- Interfaces to the document tree
  - For access and modifications of content, structure, and style of documents
- Specifications
  - Level 1 (one single specification)
  - Level 2 (6 specs): Core, Style, (Views), ...
  - Level 3 (3 specs): Core, ...
  - Level 4

DOM INTERFACES HIERARCHY



DOM INTERFACES: METHODS AND PROPERTIES

- The Node interface

```
nodeType
parentNode
firstChild
nextChild
firstSibling
hasChildNodes()
hasAttributes()
appendChild()
removeChild()
```

- The Document interface

```
documentElement
getElementById()
getElementsByTagName()
querySelector()
createElement()
```

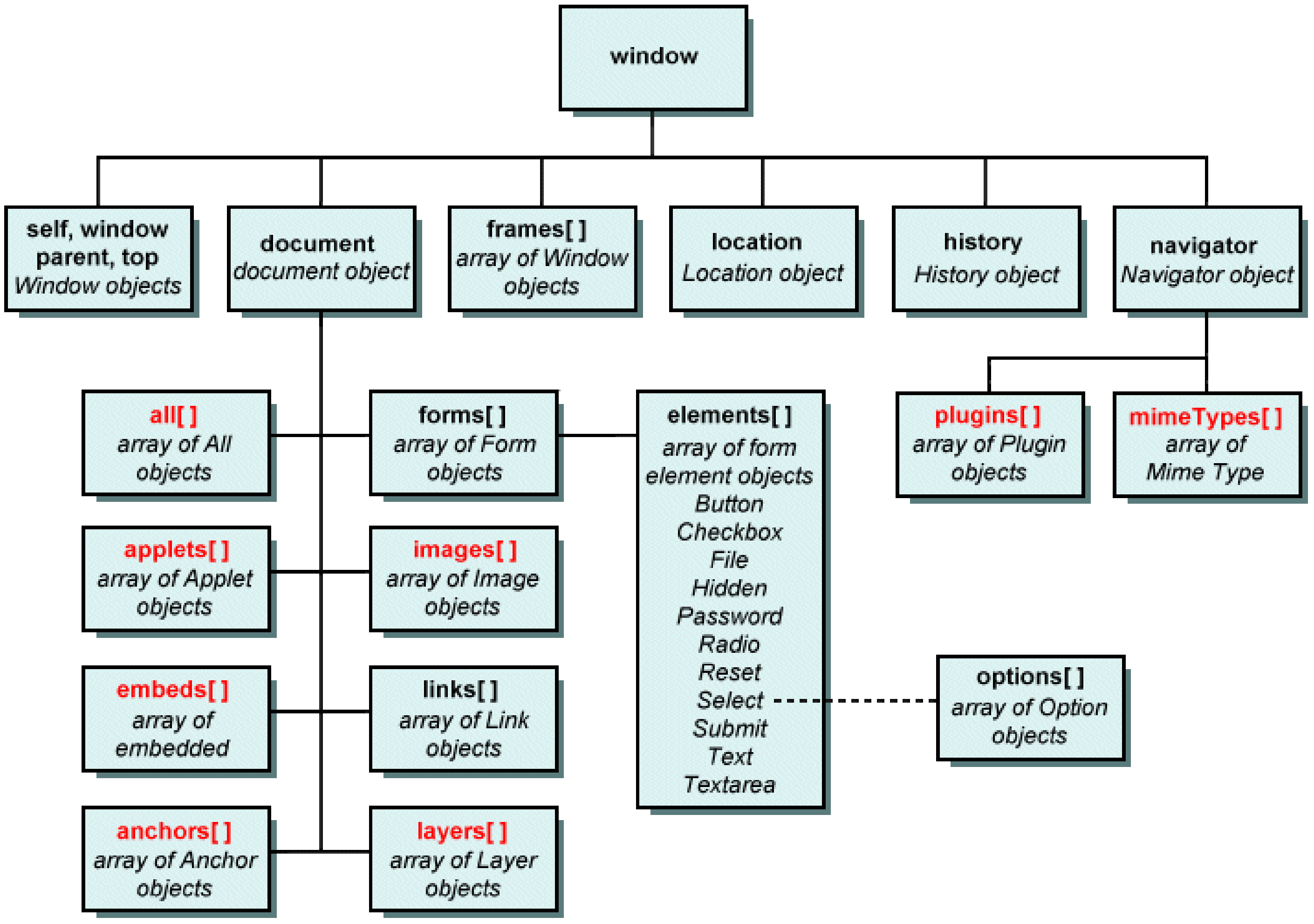
- The Element interface

```
innerHTML
getAttribute
setAttribute
removeAttribute
```

THE WINDOW OBJECT

- API corresponding to the browser window or tab
- Convenient API for various usages
    - Timing (animations)
    - General events (load, ...)
    - Navigation (history)
    - Embedding (openURL)
- JavaScript `global` object in browser

THE WINDOW OBJECT



EXAMPLES OF DOM MANIPULATIONS IN JS: ADD AN ELEMENT

The page before

```
<html>
  <body>
  </body>
</html>
```

The JS code

```
var obj = document.createElement("p");
obj.textContent="some new text";
var body = document.getElementsByTagName("body")[0];
body.appendChild(obj);
```

The page after

```
<html>
  <body>
    <p>some new text</p>
  </body>
</html>
```

HTML EDITING

The page before

```html
<html>
  <body>
    <p id="someid">some new text</p>
  </body>
</html>
```

The JS code

```javascript
var obj = document.getElementById("someid");
obj.innerHTML = "some <span style='color: red;'>other</span> text";
```

The page after

```html
<html>
  <body>
    <p id="someid">some <span style="color: red;">other</span> text</p>
  </body>
</html>
```

WORKING ON ATTRIBUTES

The page before

```
<html>
  <body>
    <p id="someid">some new text</p>
  </body>
</html>
```

The JS code

```
var body = document.getElementsByTagName("body")[0];
body.onload="myfunction()";
var obj = document.getElementById("someid");
obj.setAttribute("align", "center");
```

The page after

```
<html>
  <body onload="myfunction()">
    <p align="center" id="someid">some new text</p>
  </body>
</html>
```

REMOVE ELEMENTS

The page before

```
<html>
  <body>
    <p id="someid">some new text</p>
  </body>
</html>
```

The JS code

```
var body = document.getElementsByTagName("body")[0];
var obj = document.getElementById("someid");
body.removeChild(obj);
```

The page after

```
<html>
  <body>
  </body>
</html>
```

CSS AND JAVASCRIPT

- The JavaScript style property
  - Used to set a new style on an element
  - Used to query the style on this element

```
var e = document.getElementById("SomeElementId");
e.style.top = 10px;
```

- The `getComputedStyle()` method
  - To ask for all styles (inherited, computed, ...) of an element

```
var e = document.getElementById("SomeElementId");
var style = window.getComputedStyle(e);
var height = style.getPropertyValue("height");
```

SCRIPT PROCESSING IN HTML

JAVASCRIPT LIBRAIRIES

- Principles
  - Simplify the JS code written by Web Developers
  - Provide a unique interface for all browsers (bugs)
- Many librairies
  - JQuery,
  - Angular,
  - Bootstrap …
- JavaScript “beautifier”/“minifier”

SCRIPTED ANIMATIONS

- Use of timers and callback functions
  - Ex: using the `window` object
  - Ex: using an `SVGTimer` object
  - Ex: using `requestAnimationFrame`
- Management of the synchronization by the script

ANIMATIONS WITH JS

```
<rect id='R' width="120" height="50" fill="blue">
<script>
function doAnimation(){
  var rect=document.getElementById('R');
  x=x+xincr;
  rect.setAttribute('x', x);
  window.setTimeout("doAnimation()", 10);
}
</script>
```

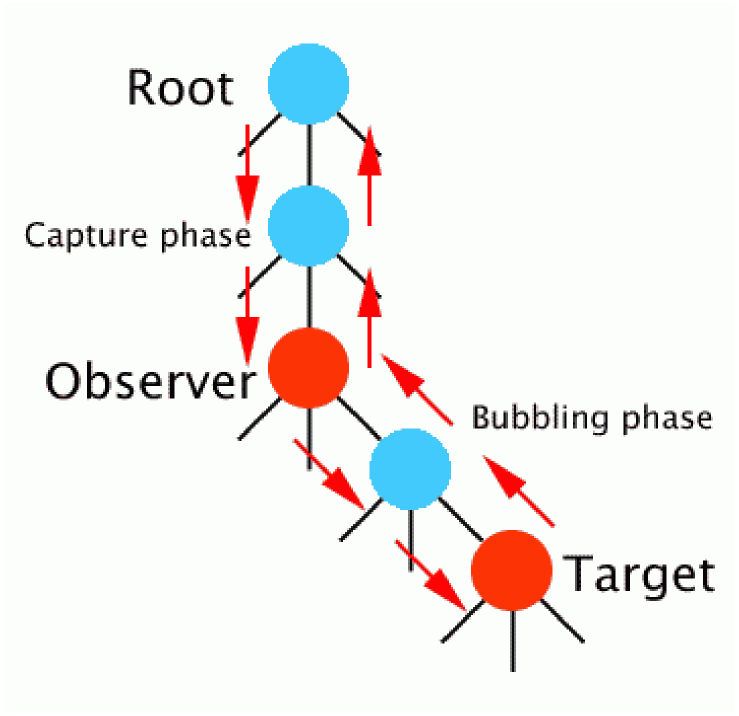
```
function animloop() {
  requestAnimFrame(animloop);
  render();
}
```

INTERACTIVITY & SCRIPTING

- Simple interactivity does not require scripting
  - Forms filing and submitting
  - Navigation
  - Triggering animations or transitions
  - …
- More complex interactions require Javascript with
  - DOM events
  - AJAX Pattern

DOM EVENTS

- API to indicate to the browser how to process events in JavaScript
- Based on a specific Event Propagation model
  - Capture phase, target phase, bubbling phase
  - Cancellation of events,
  - Default action



EXAMPLES OF DOM EVENTS CODE

```
<script type="application/ecmascript" >
  function doSomething(evt) { … }
</script>
<text onclick=“doSomething(evt)” >Hello World!</text>
```

```
<script type="application/ecmascript" >
 function doSomething(evt) { … }
 e=document.getElementById(‘T’);
 e.addEventListener(‘click’, doSomething, false);
</script>
<text id=“T” >Hello World!</text>
```

```
<script type="application/ecmascript" >
 function doSomething(evt) { … }
 e=document.getElementById(‘T’);
 e.onclick=doSomething;
</script>
<text id=“T” >Hello World!</text>
```

DOM EVENT TYPES

- Mouse Events
  - click, mousedown, mouseup, mouseover, mousemove, mouseout
- Key Events
  - keypress, keyrelease
- Touch events
  - touchstart, touchend, touchleave, touchmove, …
- Drag events
  - dragstart, dragend, …
- Network events
  - load, error, abort, progress
- Form events
  - submit, focus …
- Media events
  - play, pause …

AJAX "ASYNCHRONOUS JAVASCRIPT AND XML"

- Used to make asynchronous HTTP requests and retrieve data (e.g. text, XML, binary …)
- Combined usage of different technologies
  - HTML (or SVG, …)
  - ECMAScript
  - XML (or JSON, …)
  - HTTP Download

- Exemples
  - HTML/SVG + XML + DOM + XMLHttpRequest
  - Flash + ActionScript + LoadVars + XML

- Benefits
  - Requests are asynchronous to the rendering
    - Avoids waiting for the response to further interact

  - Enables client-side heavy interactivity
    - Data base requests and response handling

AJAX EXAMPLE

```
var xhr = new XMLHttpRequest();
xhr.open("GET", "test.txt");
xhr.onload = function() {
  alert(this.responseText);
}
xhr.send();
```