

# HTML

## MPRI 2.26.2: Web Data Management

---

Antoine Amarilli



# General presentation

- **HyperText Markup Language**
  - Describes a **Web page** (not the only kind of Web content...)
  - Normalized by the **W3C** (industry+academia) and **WHATWG**
  - Language with **tags**, giving the **structure** and **content** of the document
  - For **presentation**, we will see CSS
  - For **dynamic behavior**, there is JavaScript
  - **Main version:** HTML5 (also older versions, XHTML, etc.)
  - Official W3C standard: HTML5 specification (548 pages, 2014)
  - Now also a **living standard** (WHATWG)
- Lots of **slack** in how documents respect the standard or not!

# Table of Contents

General notions

Structure

Text, lists, tables

Multimedia

Forms

# Markup principles

- **Opening** (`<em>`) and **closing** (`</em>`) tags:

Here is an `<em>example</em>`

- **Consecutive spaces** are ignored:

Doesn't  
matter!

- **Self-closing** tags, e.g., `<br>` (line break).  
(Or `<br/>` in XHTML.)

- **Attributes** :

```
<p>A <a href="https://example.com/">link</a>.</p>
```

- **Comments** :

```
<!-- not shown -->
```

- **HTML entities**: special characters and escaping

```
<p>&lt;p&gt;Club m&eacute;t&nbsp;!&lt;/p&gt;</p>
```

# Nesting

- Tags should be opened and closed in the **correct order**  
<a><b></b></a> and not <a><b></a></b>
- **Rules** about which tags can go where (in theory at least)
- Obvious **tree interpretation**

```
<table>
```

```
<tr>
```

```
<th>A1</th>
```

```
<th>B1</th>
```

```
</tr>
```

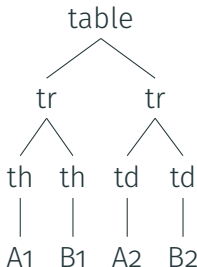
```
<tr>
```

```
<td>A2</td>
```

```
<td>B2</td>
```

```
</tr>
```

```
</table>
```



# General structure

- DOCTYPE to indicate which HTML **version** is used (here, HTML5)
- Optional lang attribute (language)

```
<!DOCTYPE html>  
<html lang="fr">  
  <head>  
    <!-- meta-information -->  
  </head>  
  <body>  
    <!-- main document contents -->  
  </body>  
</html>
```

# Header

- The `<head>` tag contains **metadata**:

```
<head>  
  <meta charset="utf-8">  
  <title>Title of the page</title>  
  <meta name="description" content="blah">  
  <!-- ... -->  
</head>
```

- Indicate the **character encoding** (e.g., UTF-8)
- Indicate the **title**
- Other information for search engines, caching
- Also **scripts** and **CSS** (see later)

# Tag soup

- HTML was not historically written by **programmers** so documents in the wild are often **broken**
- Web browsers are **very resilient**
- **Validators** allow you to check if your markup is correct (e.g., `validator.w3.org`)



# Table of Contents

General notions

Structure

Text, lists, tables

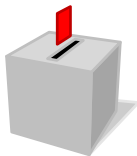
Multimedia

Forms

# POLL: tags

Which is the modern tag that will (by default) format the text in bold?

- **A:** This one: `<b>`
- **B:** This one: `<em>`
- **C:** This one: `<strong>`
- **D:** This one: `<bold>`



# POLL: tags

Which is the modern tag that will (by default) format the text in bold?

- **A:** This one: `<b>`
- **B:** This one: `<em>`
- **C: This one:** `<strong>`
- **D:** This one: `<bold>`



- Text should go in **paragraphs** delimited by `<p> ... </p>`
- **Line break** with `<br>`
- To **stress** some content, use:
  - `<em> ... </em>`
  - `<strong> ... </strong>`
  - `<mark> ... </mark>`

# Titles and structure

- **Titles:** from `<h1>` (main title) to `<h6>` (lowest title)
- Other tags to indicate the **page structure** (semantically):
  - `<header>`
  - `<footer>`
  - `<nav>`: navigation element
  - `<article>`: e.g., on a blog
  - `<main>`: main page content
  - `<dialog>`: dialog box

# Lists

- A
- B
- C

```
<ul>
```

```
<li>A</li>
```

```
<li>B</li>
```

```
<li>C</li>
```

```
</ul>
```

1. A
2. B
3. C

```
<ol>
```

```
<li>A</li>
```

```
<li>B</li>
```

```
<li>C</li>
```

```
</ol>
```

- |          |   |
|----------|---|
| <b>X</b> | A |
| <b>Y</b> | B |
| <b>Z</b> | C |

```
<dl>
```

```
<dt>X</dt> <dd>A</dd>
```

```
<dt>Y</dt> <dd>B</dd>
```

```
<dt>Z</dt> <dd>C</dd>
```

```
</dl>
```

# Tables

```
<table>
  <tr>
    <th>X1</th>
    <th>X2</th>
  </tr>
  <tr>
    <td>A1</td>
    <td>A2</td>
  </tr>
  <tr>
    <td>B1</td>
    <td>B2</td>
  </tr>
</table>
```

<b>X1</b>	<b>X2</b>
A1	A2
B1	B2

What is the correct way to link to the website of Télécom Paris?

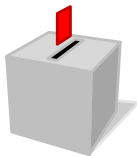
- **A:** This one: `<a href="www.telecom-paris.fr">text</a>`
- **B:** This one: `<a href="http://www.telecom-paris.fr">text</a>`
- **C:** This one: `<a href="https://www.telecom-paris.fr">text</a>`
- **D:** This one: `<a href=https://www.telecom-paris.fr>text</a>`





What is the correct way to link to the website of Télécom Paris?

- **A:** This one: `<a href="www.telecom-paris.fr">text</a>`
- **B:** This one: `<a href="http://www.telecom-paris.fr">text</a>`
- **C: This one:** `<a href="https://www.telecom-paris.fr">text</a>`
- **D:** This one: `<a href=https://www.telecom-paris.fr>text</a>`



# Hypertext links (hence HTTP, HTML)

```
<a href="https://www.telecom-paris.fr/">Telecom</a>
```

The URL can be **absolute** (as above) or **relative**

On the page `https://example.com/foo/bar.html`

---

Link	Resolution
/quux	https://example.com/quux
..	https://example.com/
bar2.html	https://example.com/foo/bar2.html
baz/toto.html	https://example.com/foo/baz/toto.html
#top	https://example.com/foo/bar.html#top

---

- The **fragment** #top will make the browser **scroll** to the element with attribute `id="top"`.

# POLL: request

When requesting

`https://www.telecom-paris.fr/foo#bar`, which part of the URL corresponds to the path in the HTTP query?

- **A:** This one: `www.telecom-paris.fr/foo#bar`
- **B:** This one: `www.telecom-paris.fr/foo`
- **C:** This one: `/foo/#bar`
- **D:** This one: `/foo`



## POLL: request

When requesting

`https://www.telecom-paris.fr/foo#bar`, which part of the URL corresponds to the path in the HTTP query?

- **A:** This one: `www.telecom-paris.fr/foo#bar`
- **B:** This one: `www.telecom-paris.fr/foo`
- **C:** This one: `/foo/#bar`
- **D: This one:** `/foo`



- Display **another page** in the current page:

```
<iframe src="https://en.wikipedia.org/">  
  <p>Sorry, your browser does not support iFrames.</p>  
</iframe>
```

- Not very **flexible** and **confusing** for the user, plus **security** issues

# Table of Contents

General notions

Structure

Text, lists, tables

Multimedia

Forms

# Images

```

```

- `src` gives the **URL** of the image (absolute or relative)
- `alt` indicates a text to replace the image (required)
- To get the image, the browser will make a **new query**, possibly to a completely different **server**
- Main **image formats**: JPEG, PNG, GIF, SVG, WebP

# Sound and video

```
<audio src="audio.ogg">  
  <p>No audio support, you can  
  <a href="audio.ogg">download audio.ogg</a>.</p>  
</audio>
```

- Video is similar, with `<video>`
- Main **sound formats**: MP3, Opus
- Main **video formats**: H.264, WebM
- Videos require large **bandwidth!**
  - Host them on a platform like **Youtube**
  - Use **BitTorrent** in **JavaScript**, e.g., **Peertube** using **WebTorrent**



# Table of Contents

General notions

Structure

Text, lists, tables

Multimedia

Forms

# Basics

- General structure of a form:

```
<form action="action.php" method="get">  
  <!-- form contents go here -->  
  <input type="submit">  
</form>
```

`<input>` Control (here, a button to submit the form)

`action` URL to which we should **submit**

`method` The **HTTP method** to use when submitting

`enctype` The **encoding** for the form data

- When submitting the form, the browser will query the `action` URL while providing the value indicated in the form fields
- The server should then **process** this data

## Back to HTTP

- **GET**: the data is given in the **path**:

```
GET action.php?first=Jean&last=Dupont HTTP/1.1
Host: example.com
```

- **POST** and `application/x-www-form-urlencoded` (default): ditto but the data is in the **request body**:

```
POST action.php HTTP/1.1
Host: example.com
Content-Type: application/x-www-form-urlencoded

first=Jean&last=Dupont
```

## Back to HTTP (cont'd)

- **POST** and multipart/form-data: less concise but more efficient when the data is large:

```
POST action.php HTTP/1.1
```

```
Host: example.com
```

```
Content-Type: multipart/form-data; boundary=--e06
```

```
----e06
```

```
Content-Disposition: form-data; name="first"
```

```
Jean
```

```
----e06
```

```
Content-Disposition: form-data; name="last"
```

```
Dupont
```

```
----e06--
```

## Text fields

```
<form action="action.php" method="get">
  <label for="firstname">First name</label>
  <input name="first" id="firstname" type="text"><br>
  <label for="lastname">Last name</label>
  <input name="last" id="lastname" type="text">
  <input type="submit">
</form>
```

- The `<label>` element **describes** the field, its `for` attribute points to the `id` attribute of the field
- The `name` attributes indicates the **name** in the HTTP query:

```
POST action.php HTTP/1.1
```

```
Host: example.com
```

```
Content-Type: application/x-www-form-urlencoded
```

```
first=Jean&last=Dupont
```

## Some attributes of a text field

placeholder **Help text** indicated when the field is empty

value Indicate the **default value**

required Must be filled in to submit the form

type Several kinds

→ **hidden** (not visible)

→ **password** (stars)

→ **email**

→ **range** (with min and max), **tel**, **number**, **color**...

pattern Check with a **regular expression** (also in JS)

→ In **all cases**, client constraints (pattern, required, hidden, etc.) **must** be revalidated on the server!

# POLL: Input validation

Which technique guarantees that the user will not change the value of a field when submitting a form

- **A:** The `disabled` attribute
- **B:** The `pattern` attribute
- **C:** Javascript validation
- **D:** Another technique



# POLL: Input validation

Which technique guarantees that the user will not change the value of a field when submitting a form

- **A:** The `disabled` attribute
- **B:** The `pattern` attribute
- **C:** Javascript validation
- **D: Another technique**





# Uploading files

- Attaching a file to the form:

```
<label for="thefile">File to submit</label>
```

```
<input type="file" name="thefile" id="thefile">
```

- **Must** use post and multipart/form-data!
- The **maximal size** should be imposed on the server
- The expected **MIME type** can be specified but should be revalidated

## Other controls

- `<input type="submit">`: Button to submit the form
- `<textarea>`: Multiline text area
- Radio buttons (`type="radio"`), checkboxes (`type="checkbox"`), dropdown lists `<select>...`

# Shorthand

Various **shorthands** to avoid writing HTML documents, e.g., **Markdown**

```
<h1>My title</h1>
```

```
# My title
```

```
<p>This is a <em>document</em>  
with <code>monospace</code>  
and here is a list:</p>
```

```
This is a document  
with monospace  
and here is a list:
```

```
<ul>  
<li>One</li>  
<li>Two</li>  
</ul>
```

- \* One
- \* Two

```
and here is a link:  
[A] (https://a3nm.net/)
```

```
<p>and here is a link: <a  
href="https://a3nm.net/">A</a>  
</p>
```

- Matériel de cours inspiré de notes par Pierre Senellart
- Merci à Marc Jeanmougin, Antonin Delpeuch et Pierre Senellart pour leur relecture