ΕΠΛ425

Τεχνολογίες Διαδικτύου

(Internet Technologies)

The Basics of HTML5

Διδάσκων

Δρ. Χριστόφορος Χριστοφόρου

christophoros@cs.ucy.ac.cy

Goals

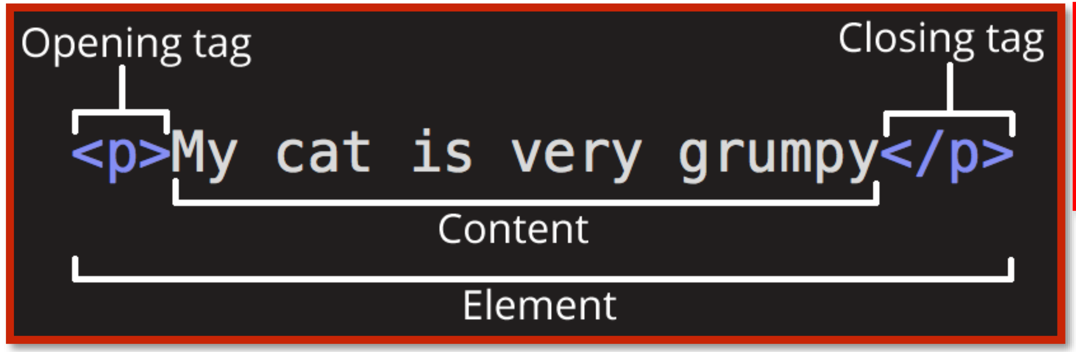
Introduction to Front-End Development:

- **HTML** to create the **document structure** and **content**
- **CSS** to control its visual/stylist aspect
- **Javascript** for interactivity

What is HTML

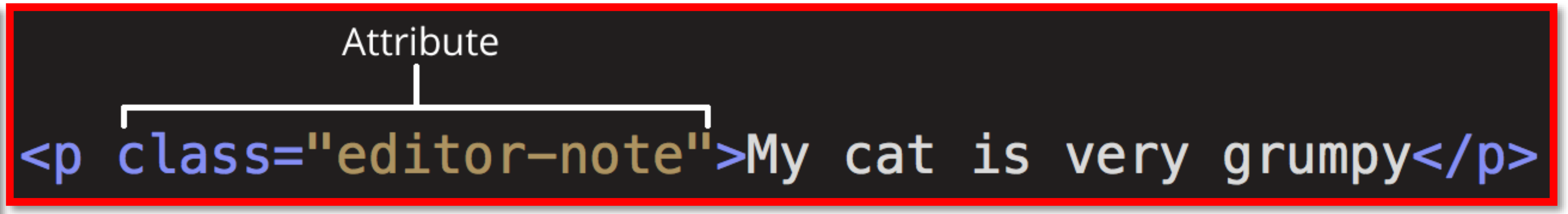
- HTML (**H**yper **T**ext **M**arkup **L**anguage) is a **Markup Language** for **describing web documents** (web pages) using a set of **markup tags** → Is the code that is used to **structure** a **web page** and its **content**:
- **HTML documents** are described by **HTML tags** → Each **HTML tag** describes **different document content** (i.e., a **paragraph**, a **list of bulleted points**, an **image**, a **data table**, etc.) **which is also referred as an HTML Element.**

Basic anatomy of an HTML element



```
<p>My cat is very grumpy</p>
```

HTML elements can also have attributes!!!



```
<p class="editor-note">My cat is very grumpy</p>
```

HTML Tags and Web Browsers

- **HTML tags** are **keywords** (**tag names**) surrounded by **angle brackets < >**:

<tagname> content </tagname>

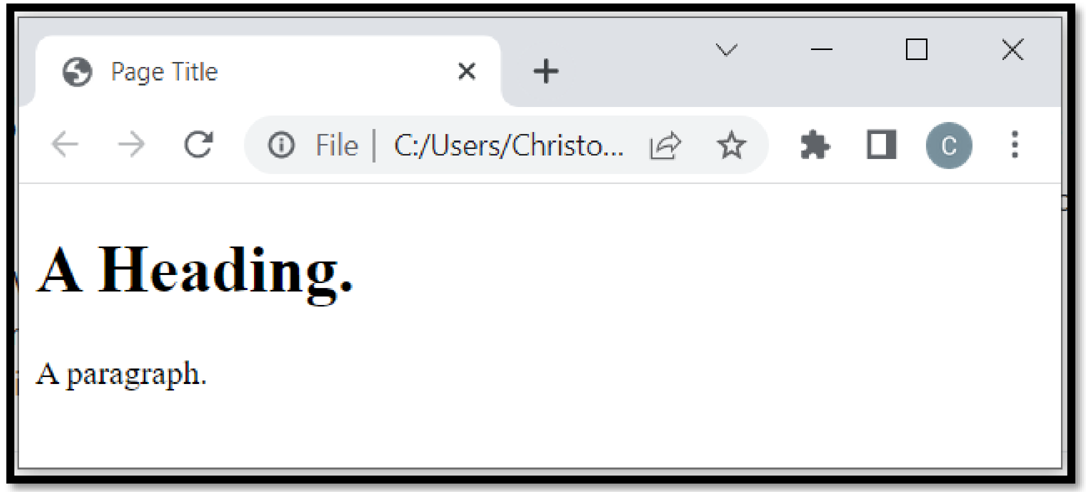
  - HTML tags normally **come in pairs** like: **<p> This is a paragraph </p>**
  - The **first tag** in a pair is the **start tag**, the **second tag** is the **end tag** written like the start tag, but **with a slash /** before the tag name.
  - Note that **some HTML tags do not have** an **end tag**, like for example the **<img >** tag.

Note: HTML tags are **NOT case sensitive**: **<P>** means the same as **<p>**. However, is **recommended** to **use lowercase tags.**

HTML Tags and Web Browsers

- The **purpose** of a **web browser** (Chrome, Internet Explorer, Firefox, Safari) is to **read HTML documents** and **display** them.
- The **browser does not display** the **HTML tags**, but **uses them** to **determine how to display the document.**

```
<!DOCTYPE html>
<html lang="en-US">

<head>
    <title>Page Title</title>
    <meta charset="utf-8">
</head>

<body>
    <h1>A Heading.</h1>
    <p>A paragraph.</p>
</body>

</html>
```



A Heading.

A paragraph.

Basic Anatomy of an HTML document

```
<!DOCTYPE html>
<html lang="en-US">

<head>
    <title>Page Title</title>
    <meta charset="utf-8">
</head>

<body>
    <h1>A Heading.</h1>
    <p>A paragraph.</p>
</body>

</html>
```

In a simple HTML document we have the following:

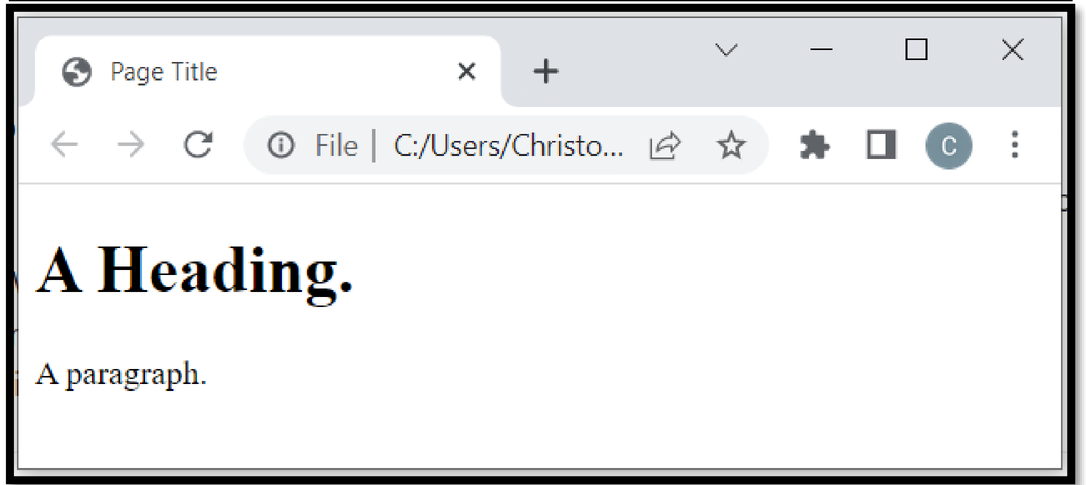
- **The** <!DOCTYPE html> declaration defines the **document type** to be HTML (and more specifically HTML5). It is a **required preamble**!
- The <html> </html> element **wraps all the content** on the entire page and is sometimes known as the **root element**. It describes an **HTML document.**
- The <head> </head> element acts as a **container** for **information** (e.g., title, style, script) **and meta data** for how the **document should** be **perceived**, and **rendered** by browsers, search engines, etc.

Basic Anatomy of an HTML document

- The <title> </title> element **sets the title of the page**, which is the title that **appears** in the **browser tab** the page is loaded in. It is also **used** to **describe** the **page** when you **bookmark/favorite it**.

- The <body> </body> element describes the **visible page content!** It contains **all the content** that you want **to show** to web users when they visit your page, whether that's **text**, **images**, **videos**, **games**, **playable audio tracks**, or whatever else.

```
<!DOCTYPE html>
<html lang="en-US">

<head>
    <title>Page Title</title>
    <meta charset="utf-8">
</head>

<body>
    <h1>A Heading.</h1>
    <p>A paragraph.</p>
</body>

</html>
```



A Heading.

A paragraph.

Basic Anatomy of an HTML document

<h1> defines the most important heading.
<h6> defines the least important heading.

```
<!DOCTYPE html>
<html lang="en-US">

<head>
    <title>Page Title</title>
    <meta charset="utf-8">
</head>

<body>
    <h1>A Heading.</h1>
    <p>A paragraph.</p>
</body>

</html>
```

- **HTML headings** are defined with the **<h1>** to **<h6>** tags:
  - <h1>This is a heading</h1>
  - <h2>This is another heading</h2>
  - <h3>This is another heading</h3>
  - ….
- **HTML paragraphs** are defined with the **<p>** tag
  - <p>This is a paragraph.</p>
  - <p>This is another paragraph.</p>

The `<!DOCTYPE html>` Declaration

Since the early days of the web, there have been **many versions** of HTML:

Version	Year
HTML	1991
HTML 2.0	1995
HTML 3.2	1997
HTML 4.01	1999
XHTML	2000
HTML5	2014

The `<!DOCTYPE html>` Declaration

- To display **a document correctly**, the browser **must know** both the **type** and **version** of the **HTML used for creating** the **web page**.
- The HTML <!DOCTYPE > declaration is not an HTML element or tag. It **is an instruction** that **tells the browser what type of document to expect**.
- Specifically, is a **declaration** that tells the browser **what version** of HTML the document is **written in**. This declaration appears as the very first line in an HTML file.

The <!DOCTYPE html> Declaration

- All HTML documents need to start with a <!DOCTYPE > declaration. The **declaration varies** depending on what **version of HTML** the document is written in.
- In **HTML5**, the doctype declaration is **<!DOCTYPE html>.** This is **easy to write** and **remember**, particularly when compared to the complicated doctype declarations of **previous versions** of HTML.
- The <!DOCTYPE > declaration is **not case sensitive**. All cases are acceptable!

```
<!DOCTYPE html>
<!DOCTYPE HTML>
<!doctype html>
<!Doctype Html>
```

✓

The <!DOCTYPE html> Declaration

- In the examples below with HTML 4 and HTML 1.0, the Document Type Definition (DTD) is **declared** in **external files** by the **World Wide Web Consortium** (W3C), which are referenced in quotation marks.

```
HTML 4 Example:
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
HTML 1.0 Example:
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

- It is **NO LONGER NECESSARY** to refer to a DTD when writing with the latest version of HTML, which is HTML5.

The <!DOCTYPE html> Declaration

- The <!DOCTYPE html> declaration tells the browser that this document type is HTML5 so it knows **which elements** are **valid**. For example:
  - The anchor element, <a>, is valid in the HTML5, HTML4, and XHTML doctypes.
  - The acronym element, <acronym>, is valid for HTML4 and XHTML but not for HTML5.
  - The dialog element, <dialog>, is only valid for HTML5 — not HTML4 or XHTML.

What is new in HTML5

- The **DOCTYPE declaration** for HTML5 is now very simple:

```
<!DOCTYPE html>
```

- Also the **default character encoding** in HTML5 is UTF-8. **Thus, even if you omit this** <meta charset="UTF-8">, **UTF-8** will be used.

```
<!DOCTYPE html>
<html>

<head>
    <meta charset="UTF-8">
    <title>Title of webpage</title>
</head>

<body>
    <!-- Content of the webpage... -->
</body>

</html>
```

Example of HTML5 Structure

What is new in HTML5

- The <meta charset="UTF-8"> element **sets** the **character set** your document should use to **UTF-8,** which **includes most characters from the vast majority of written languages**.
- Essentially, it can now **handle any textual content** you might put on it.

```
<!DOCTYPE html>
<html>

<head>
    <meta charset="UTF-8">
    <title>Title of webpage</title>
</head>

<body>
    <!--Content of the webpage...-->
</body>

</html>
```

Example of HTML5 Structure

What is new in HTML5

- **Also HTML5 comes with new API's (Application Programming Interfaces).** The **most interesting** new **API's** are:
  - HTML Geolocation
  - HTML Drag and Drop
  - HTML Local Storage (*This is a **powerful replacement** for cookies*)
  - HTML Application Cache
  - HTML Web Workers
  - HTML SSE (Server-Sent Events)

HTML Elements and Attributes

```
<!DOCTYPE html>
<html>

<head>
    <title>Title</title>
</head>

<body>
    <h1>A Heading.</h1>
    <br>
    <hr>
    <p>A paragraph.</p>
</body>

</html>
```

- **HTML elements** are written with a **start tag**, with an **end tag** and **(if applicable)**, with the **content in between**! For example:

```
<h1>A Heading.</h1>  <!-- This is an Element-->
<p>A paragraph.</p>  <!-- This is an Element-->
<title>Title</title>      <!-- This is an Element-->
```

- HTML elements with **no content** are called **empty elements**, like for example **<br> and <hr>**
  - **<br>** is **without** a **closing tag** which defines a **line break** in our HTML document.
  - **<hr>** is **without** a **closing tag** which **creates** a **horizontal line** in an HTML page (can be used to separate content)

HTML Elements and Attributes

- **HTML elements** can **have attributes** which are **used to provide additional information** about **HTML elements.**

These attributes are <u>**always specified** in the **start tag**</u> and come in **name="value"** pairs, like:

id="greeting"

src="images/ucyLogo.png"

```
<!DOCTYPE html>
<html lang="en-US">

<head>
   <title>Title</title>
</head>

<body>
  <p id="greeting">Welcome!</p>
  <a href="http://www.ucy.ac.cy"> This is a link to UCY</a>
  <img src="images/ucyLogo.png" alt="UCY Logo" width="104">
</body>

</html>
```

HTML Elements and Attributes

- **Also the document language** is declared **in the <html> tag** using the **lang attribute.**

Declaring a language is **important** for **accessibility applications** (e.g., **screen readers** for blind people) and **search engines.**

```
<!DOCTYPE html>
<html lang="en-US">

<head>
   <title>Title</title>
</head>

<body>
  <p id="greeting">Welcome!</p>
  <a href="http://www.ucy.ac.cy"> This is a link to UCY</a>
  <img src="images/ucyLogo.png" alt="UCY Logo" width="104">
</body>

</html>
```

The first two letters specify the language (en). If there is a dialect, use two more letters (US).

HTML Elements and Attributes

- **HTML links** are defined with the **<a>** tag. The **link's destination** is specified in the **href attribute**.

```
<!DOCTYPE html>
<html lang="en-US">

<head>
    <title>Page Title</title>
</head>

<body>
    <!-- This is a comment -->
    <a href="http://www.ucy.ac.cy">This is a link to UCY</a>
</body>

</html>
```

To **add comments** to your HTML source use the following syntax:

<!-- *A comment here* **-->**

HTML Elements and Attributes

- **HTML images** are defined with the **<img>** tag.
- In the **<img>** tag, the **source file** (**src**) and the **alternative text** (**alt**) can be provided (among others) as **attributes.**
- The **alt** attribute specifies an **alternative text** to be used, when **an HTML element cannot be displayed** (i.e. in case the **src** is not reachable).

```
<!DOCTYPE html>
<html lang="en-US">

<head>
    <title>Page Title</title>
</head>

<body>
    <img src="images/ucyLogo.jpg" alt="UCY Logo">
</body>

</html>
```

Also, the value of the **alt attribute can be read by "screen readers".** This way, someone "listening" to the webpage, i.e., a **blind person**, can "**hear**" the element.

HTML Elements and Attributes

Note: <img **src** >, <a **href** >, and <link **href** > can all take either **relative** or **absolute** paths to the resource.

```
<a href="about.html">About</a>
<img src="http://i.imgur.com/WJToVGv.jpg">
<link rel="stylesheet" href="css/style.css">
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5
.2.3/dist/css/bootstrap.min.css" rel="stylesheet">
```

CDN (Content Delivery Network)

HTML Elements and Attributes

```
<!DOCTYPE html>
<html lang="en-US">

<head>
</head>

<body>
    <p id="greeting" title="About UCY">
The University of Cyprus is a public
research university established in Cyprus
in 1989. It admitted its first students in
1992 and has approximately 7000 students
    </p>
</body>

</html>
```

- With a <p> tag the **title attribute** (**not the** <title> element) can also be used.
- In this case **when you move the mouse over** the **element** in the browser, the **title will be displayed** as a **tooltip**.

HTML Elements and Attributes

```
<!DOCTYPE HTML>
<html>

<head>
  <title>An Example using HTML</title>
</head>

<body>
  <pre>
    This paragraph            contains
    a lot of spaces       in the source
    code,   but the       browser will
    preserve them.
  </pre>

</body>

</html>
```

- With HTML, the browser will **remove extra spaces** and **extra lines** when the **page is displayed**. Any number of spaces, and any number of new lines, counts as **only one space**. For example the following will be displayed as a **single line.**

```
<p>
This paragraph          contains
a lot of spaces   in the source
code, but the      browser ignores it.
</p>
```

- **The HTML <pre> element** defines **preformatted text**. The text inside a **<pre>** element is **displayed** in a **fixed-width font** (usually Courier), and it **preserves both spaces and line breaks.**

HTML Elements and Attributes – Styling

- **Styling** can be added to **HTML elements** in **3 ways**:
  - **Inline** - using a **style attribute** **in HTML elements** **start tag**
  - **Internal** - using a **<style>** **element** in the HTML **<head>** **section**
  - **External** - using one or more **external CSS files**

The most common way to add styling, is to keep the styles in separate CSS files.

We will see more about external CSS files in a later lecture!!!

HTML Elements and Attributes – Inline styling

- **Inline styling** is used to apply a **unique style** to a **single HTML element.**
- We can define the **style** of an element using the **style attribute** and the **following syntax** :

Syntax: style="property1:value1; property2:value2;"

The property is a **CSS property** and the value is a **CSS value**.

HTML Elements and Attributes – Inline styling

- **Examples:**
  - Sets the background for a page to lightgrey

    ```
    <body style="background-color:lightgrey;">
    ```

  - Sets the text color to blue and align to center

    ```
    <h1 style="color:blue; text-align:center">This is a header</h1>
    ```

  - Sets the text color to green and used courier fonts

    ```
    <p style="color:green; font-family:courier;">This is a paragraph.</p>
    ```

HTML Elements and Attributes – Internal styling

- **Inline styling** is used to **define a style** for **one HTML element.**

- **Internal styling** is defined in the **<head>** section of an HTML page, within a **<style> element** (CSS code is used here). It **defines a style** for **one HTML page.**

```
<!DOCTYPE html>
<html>

<head>
    <title>Page Title</title>
    <style>
        body { background-color: lightgrey; }
        h1 { color: blue; font-family: verdana; }
        p { color: green; font-size: 160%; }
    </style>

</head>

<body>
    <h1>This is a heading</h1>
    <p>This is a paragraph.</p>
</body>

</html>
```

HTML Elements and Attributes – Using External CSS file

- An external style sheet (CSS) is used to **define** the **style** **for many pages.**
- With an **external style sheet, you can change the look of an entire web** site **by changing just one file!**
- **To use an external CSS file** in the **<head>** section of an HTML page in a **<link> element, add a link href to its source! Also the rel attribute set to "stylesheet" is mandatory!**

```
<!DOCTYPE html>
<html>

<head>
    <title>Page Title</title>
    <link rel="stylesheet" href="style.css">
</head>

<body>
    <h1>This is a heading</h1>
    <p>This is a paragraph.</p>
</body>

</html>
```

HTML Elements and Attributes – Some Tips

- **Always use lowercase attributes…**even if HTML5 is **not case sensitive** and thus **does not require lower case attribute names**.
- **Always use quotes in attribute values…**even if the HTML5 standard **does not require quotes** around attribute values.
- Note that sometimes it is **necessary** to use quotes.

  `<p title="About UCY">` ✓

  `<p title=About UCY>` ✗ **This will not display correctly, because it contains a space!**
- **Double quotes "** are the most common in HTML, but **single quote '** can also be used.

HTML Elements and Attributes – Some Tips

- A description of **all HTML tags** is provided here.
- A list of **all HTML attributes** and by **what HTML elements can be used** within is provided here.

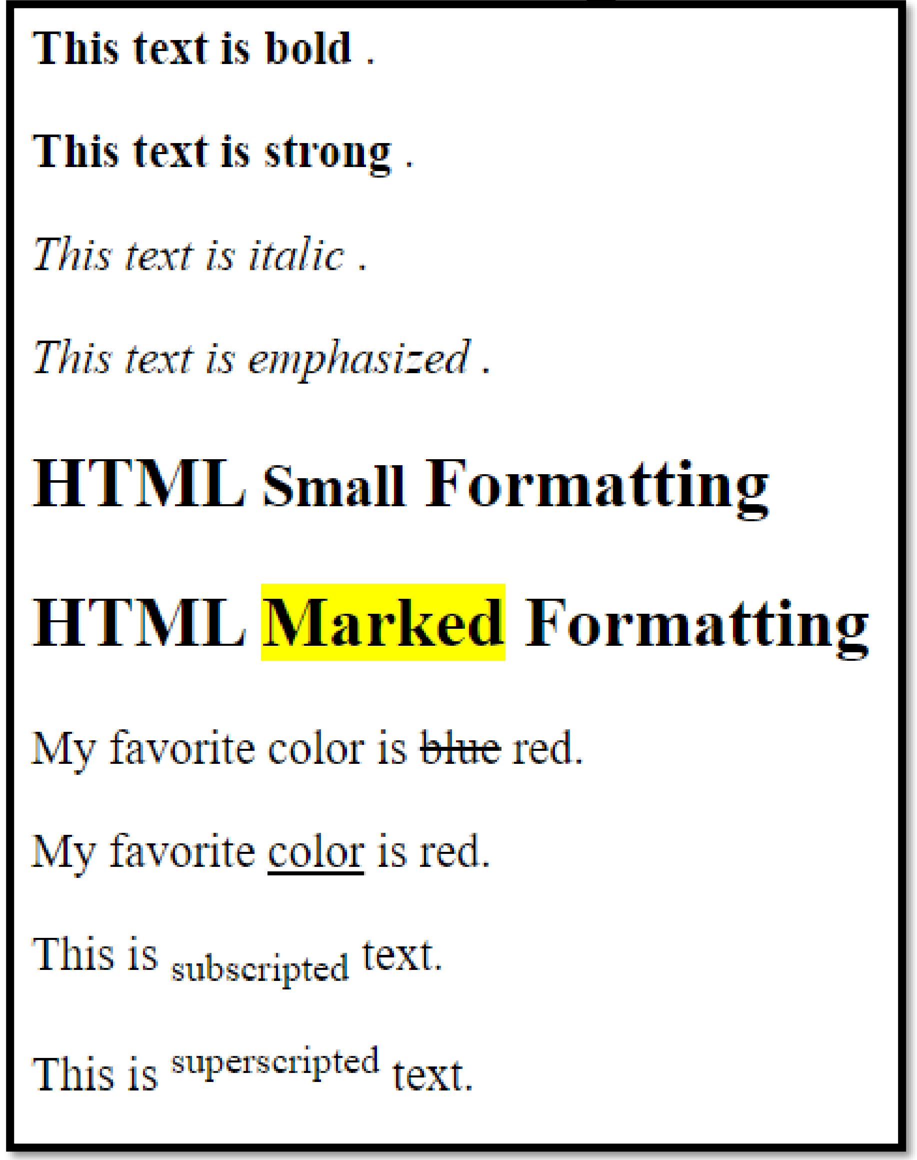
Text Formatting Elements

- **Text Formatting** elements were designed to display special **types of text**:

Bold text	<p>**<b>** This text is bold **</b>**.</p>
Important text	<p>**<strong>** This text is strong **</strong>**.</p>
Italic text	<p>**<i>** This text is italic **</i>**.</p>
Emphasized text	<p>**<em>** This text is emphasized **</em>**.</p>
Small text	<h2> HTML **<small>** Small **</small>** Formatting </h2>
Marked text	<h2> HTML **<mark>**Marked**</mark>** Formatting </h2>
Deleted text	<p> My favorite color is **<del>**blue**</del>** red. </p>
Inserted text	<p> My favorite **<ins>**color**</ins>** is red. </p>
Subscripts	<p> This is **<sub>**subscripted**</sub>** text. </p>
Superscripts	<p> This is **<sup>**superscripted**</sup>** text. </p>

Text Formatting Elements

- **Text Formatting** elements were designed to display special **types of text**:
  - Bold text
  - Important text
  - Italic text
  - Emphasized text
  - Small text
  - Marked text
  - Deleted text
  - Inserted text
  - Subscripts
  - Superscripts



This text is bold .

This text is strong .

This text is italic .

This text is emphasized .

HTML Small Formatting

HTML Marked Formatting

My favorite color is ~~blue~~ red.

My favorite color is red.

This is $_{\text{subscripted}}$ text.

This is $^{\text{superscripted}}$ text.

HTML Links – Hyperlinks and Local Links

- A **hyperlink** is a **text or an image** you can **click on**, and **jump to another document.**
- In HTML, links are defined with the **<a>** tag:
  - **Syntax:** <a href="*url*"> *link text or link image* </a>

```
<a href="http://www.ucy.ac.cy/">Visit our University</a>
<a href="http://www.ucy.ac.cy/"><img src="images/Earth_globe.png"></a>
```

The **href** attribute specifies the **destination address**: **http://www.ucy.ac.cy**

The ***link text*** (Visit our University) ***and the img*** is the **visible part**.

Clicking on it will **send you** to the **destination address**.

HTML Links – Hyperlinks and Local Links

- A **local link** (link to the **same web site**) is specified with a **relative URL** (without http://www....).

Example: `<a href="press_releases.html"> Announcements </a>`

HTML Links – Hyperlinks and Local Links

- Using the **target attribute** you can specify **where to open the linked document**. The following example will open the linked document in a **new browser window** or in a **new tab.**

```
<a href="http://www.ucy.ac.cy/" target="_blank"> Visit our University</a>
```

TARGET VALUE	DESCRIPTION
_blank	Opens the linked document in a new window or tab
_self	Opens the linked document in the same frame as it was clicked (this is the **default way**).
_parent	Opens the linked document in the parent frame
_top	Opens the linked document in the full body of the window
framename	Opens the linked document in a named frame

HTML Links – Create a Bookmark

- **HTML bookmarks** are used to allow readers to **jump to specific parts of a web page**.
- Bookmarks are practical **if your website has long pages**.
- To **make a bookmark**, you must first **create the bookmark**, and then **add a link to it**.
- When the link is **clicked**, the page will **scroll to the location** with the bookmark.

HTML Links – Create a Bookmark: An Example

- First, **create the bookmark** (e.g., using an <h2> header element) in your web page, **associating it** with an **id** attribute!

  <h2 id="tips"> Useful Tips for Registering to Classes **</h2>**

- Then, **add a link** to the **bookmark**, from **within the same page**.

  <a href="#tips"> Useful tips for registration! **</a>**

- If you want to **add a link** to the bookmark, **from another page**.

  <a href="registration.html#tips"> Useful tips for registration! **</a>**

HTML Links – Style and Colors

- When you **move the mouse over a link**, **two things** will normally **happen**:
  - The **mouse arrow** will **turn into** a **little hand.**
  - The **color of the link** element will **change.**
- **By default**, a link will appear like this (in all browsers):
  - An **unvisited link** is **underlined and blue**
  - A **visited link** is **underlined and purple**
  - An **active link** is **underlined and red**

HTML Links – Style and Colors

- You can **change the default style and color**, by using styles:

```
<style>
  a:link {color:green; background-color:transparent; text-decoration:none}
  a:visited {color:pink; background-color:transparent; text-decoration:none}
  a:hover  {color:red; background-color:transparent; text-decoration:underline}
  a:active {color:yellow; background-color:transparent; text-decoration:underline}
</style>
```

HTML Images – JPG, GIF, PNG….

* A **screen reader** is a software program that can read what is displayed on a screen. Screen readers are useful to people who are blind, visually impaired, or learning disabled.

- In HTML, images are defined with the **<img>** tag.
- The <img> tag can **contain attributes only**, and **does not have** a **closing tag**.
- The **src attribute** specifies the **location** that the **image can be found**:
- The **alt attribute** specifies an **alternate text** for an image, if the **image for some reason cannot be displayed** (because of slow connection, an error in the **src** attribute, or if the user uses a **screen reader***).

Syntax: <img src="*url*" alt="*some_text*">

Example: <img src="UCY_logo.jpg" alt="UCY Logo">

HTML Images – JPG, GIF, PNG….

- Some web sites **store** their **images** on **image servers**.
- You can **access images** from any web address in the world by declaring in the **src attribute** the **absolute URL** of the image.

Example:

```
<img src="https://www.helpguide.org/wp-content/uploads/king-
charles-spaniel-resting-head-768.jpg" alt="Spaniel Dog">
```

HTML Images – JPG, GIF, PNG….

- To specify the **size of the images** you can use the **style attribute**. The values are **specified in pixels** (use px after the value).

```
<img src="UCY_logo.jpg" alt="UCY Logo" style="width:128px; height:128px;">
```

- **Alternatively**, you can use **width** and **height attributes**. Here, the values are specified in **pixels by default**.

```
<img src="UCY_logo.jpg" alt="UCY Logo" width="128" height="128">
```

Note1: Better use the **style attribute**. It prevents CSS styles sheets from changing the original you defined for the size of images.

Note2: Even better…use an external CSS file

HTML Images – JPG, GIF, PNG….

- As shown earlier, you can also **use** an **image as link**. For example:

```
<a href="http://www.ucy.ac.cy/" target="_blank">
  <img src="UCY_logo.jpg" alt="UCY Logo" style="width:42px; height:42px; border:0";>
</a>
```

In this case the **link text** is **replaced** by an **image element**

Tip: "border:0;" is included to prevent Internet Explorer 9 (and earlier) from **displaying a border around the image**.

Image Maps - For more details on Image Maps check this link

- Use the **<map>** tag to define an **image-map.** An **image-map** is an image with **clickable areas**. An example is provided below.

```
<img src="images/planets.gif" alt="Planets" usemap="#planetmap" style="width:400px;height:auto;">

<map name="planetmap">
     <area shape="rect" coords="0,0,82,126" alt="Sun" href="sun.htm">
     <area shape="circle" coords="90,58,3" alt="Mercury" href="mercur.htm">
     <area shape="circle" coords="124,58,8" alt="Venus" href="venus.htm">
</map>
```

- The **name attribute** of the **<map> tag** is **associated** with **usemap attribute** of the **<img>** → **This creates a relationship between** the **image** and the **map**.
- The **<map> tag** contains a number of **<area> tags** that **defines the clickable areas** in the **image-map**.

HTML Tables

- **Tables** are **defined** with the **<table>** tag.
- Tables are **divided** into **table rows** with the **<tr> tag** and into **table data (i.e., table columns)** with the **<td>** tag.
- **Table data <tb>** are the **containers of the data**.
- They can **contain all sorts** of **HTML elements** like **text**, **images**, **lists**, **other tables**, **etc**.

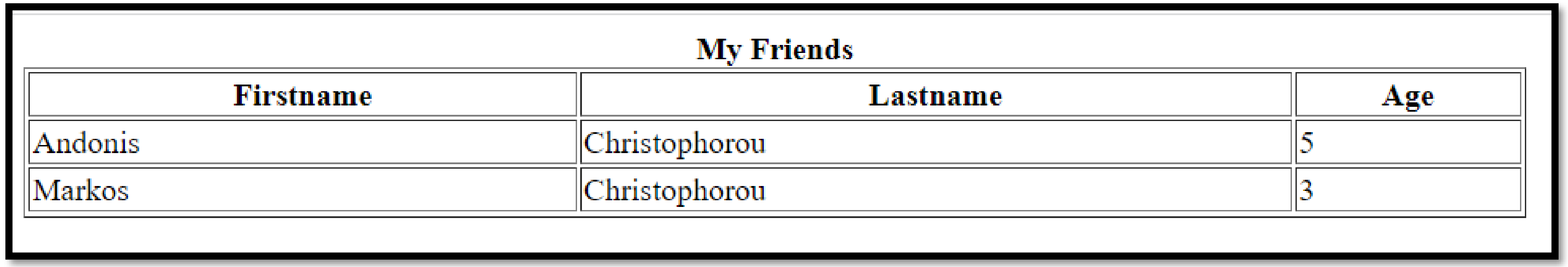
HTML Tables

- A **table row** can also be **divided** into **table headings** with the **<th>** tag. **By default**, all major browsers **display table headings** as **bold** and **centered.**
- To **add a caption** to a table, use the **<caption>** tag. The **<caption>** tag **must be inserted immediately after** the **<table> tag**.
- To **define** a **special style** (**CSS**) or to **perform a specific function** (**JavaScript**) for a special table, **add** an **id attribute to the table (or if needed an id attribute to every element of the Table)**.

HTML Tables – An example:

Further details on the CSS styling (i.e., colors, border spacing, text alignment, etc.) will be provided when we talk about CSS

```html
<table id="t01" border="1" style="width:50%">
    <caption><strong>My Friends</strong></caption>
    <tr>
        <th>Firstname</th>
        <th>Lastname</th>
        <th>Age</th>
    </tr>
    <tr>
        <td>Andonis</td>
        <td>Christophorou</td>
        <td>5</td>
    </tr>
    <tr>
        <td>Markos</td>
        <td>Christophorou</td>
        <td>3</td>
    </tr>
</table>
```



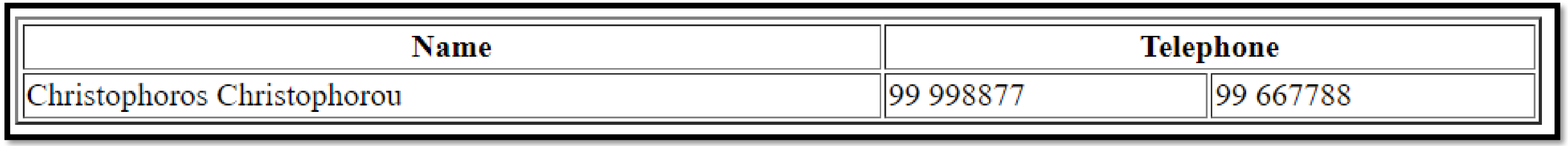
My Friends

Firstname	Lastname	Age
Andonis	Christophorou	5
Markos	Christophorou	3

HTML Tables

- To make a cell **span more than one column**, use the **colspan attribute**:

```
<table border="2" style="width:50%">
    <tr>
        <th>Name</th>
        <th colspan="2">Telephone</th>
    </tr>
    <tr>
        <td>Christophoros Christophorou</td>
        <td>99 998877</td>
        <td>99 667788</td>
    </tr>
</table>
```

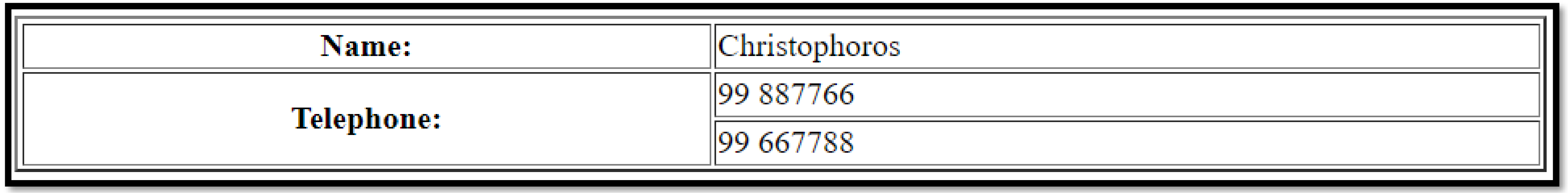


Name	Telephone	
Christophoros Christophorou	99 998877	99 667788

HTML Tables

- To make a cell **span more than one row**, use the **rowspan** attribute:

```
<table border="2" style="width:50%">
    <tr>
        <th>Name:</th>
        <td>Christophoros</td>
    </tr>
    <tr>
        <th rowspan="2">Telephone:</th>
        <td>99 887766</td>
    </tr>
    <tr>
        <td>99 667788</td>
    </tr>
</table>
```



Name:	Christophoros
Telephone:	99 887766
	99 667788

HTML Lists (Unordered and Ordered Lists)

- An **unordered list** starts with the <ul> tag. Each list item starts with the <li> tag. By default, the list items will be **marked with bullets** (small black circles). You can also define the style of the "bullets" in a **style** attribute.

```
<ul>
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ul>
```

```
style="list-style-type: square"
```

- An **ordered list** starts with the <ol> tag. Each list item starts with the <li> tag. **By default**, the list items will be **marked with numbers**.

```
<ol>
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ol>
```

HTML Lists (Unordered and Ordered Lists)

- A **type** attribute can be added to an ordered list, to **define the type of the marker**.

```
<ol type="A">
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ol>
```

TYPE	DESCRIPTION
type="1"	The list items will be numbered with numbers (default)
type="A"	The list items will be numbered with uppercase letters
type="a"	The list items will be numbered with lowercase letters
type="I"	The list items will be numbered with uppercase roman numbers
type="i"	The list items will be numbered with lowercase roman numbers

HTML Block and Inline Elements

- Every HTML element has a **default display value** depending on **what type of element it is**.
- The **default display** value for **most elements** is **block** or **inline**.
- **Block-level Elements:** A block-level element always **starts on a new line** and **takes up** the **full width available** (stretches out to the left and right as far as it can). **Examples** of **block-level** elements:
  - **<div>, <h1> - <h6>, <p>, <form> ....**
- **Inline Elements:** An inline element **does NOT start on a new line** and **only takes up as much width as necessary**. Examples of inline elements:
  - **<span>, <a>, <em>, <b>, <i> .....**

The <div> Element

- The **<div>** element is a **block-level** element that is often **used as a container for other HTML elements**.

- The **<div>** element has **no required attributes**, but **style** and **class** are **common (Note: style is better to be included using External CSS)**.

- When used together with CSS, the **<div>** element can be used to **style blocks of content** (i.e., the **contained elements** will **inherit the styling**).

```
<div class="cities" style="background-color:black; color:white; padding:20px;">
 <h1>London</h1>
 <p>London is the capital city of England with over 13 million inhabitants</p>
</div>
```

The <span> Element

- The **<span>** element is an **inline** element that is **often used** as a **container for some text**. The <span> element has no required attributes, but **style** and **class** are common.
- **When used together with CSS**, the **<span>** element **can be used to style parts of the text**.

```
<h1> My <span class="imp" style="color:red">Important</span> Heading </h1>
```

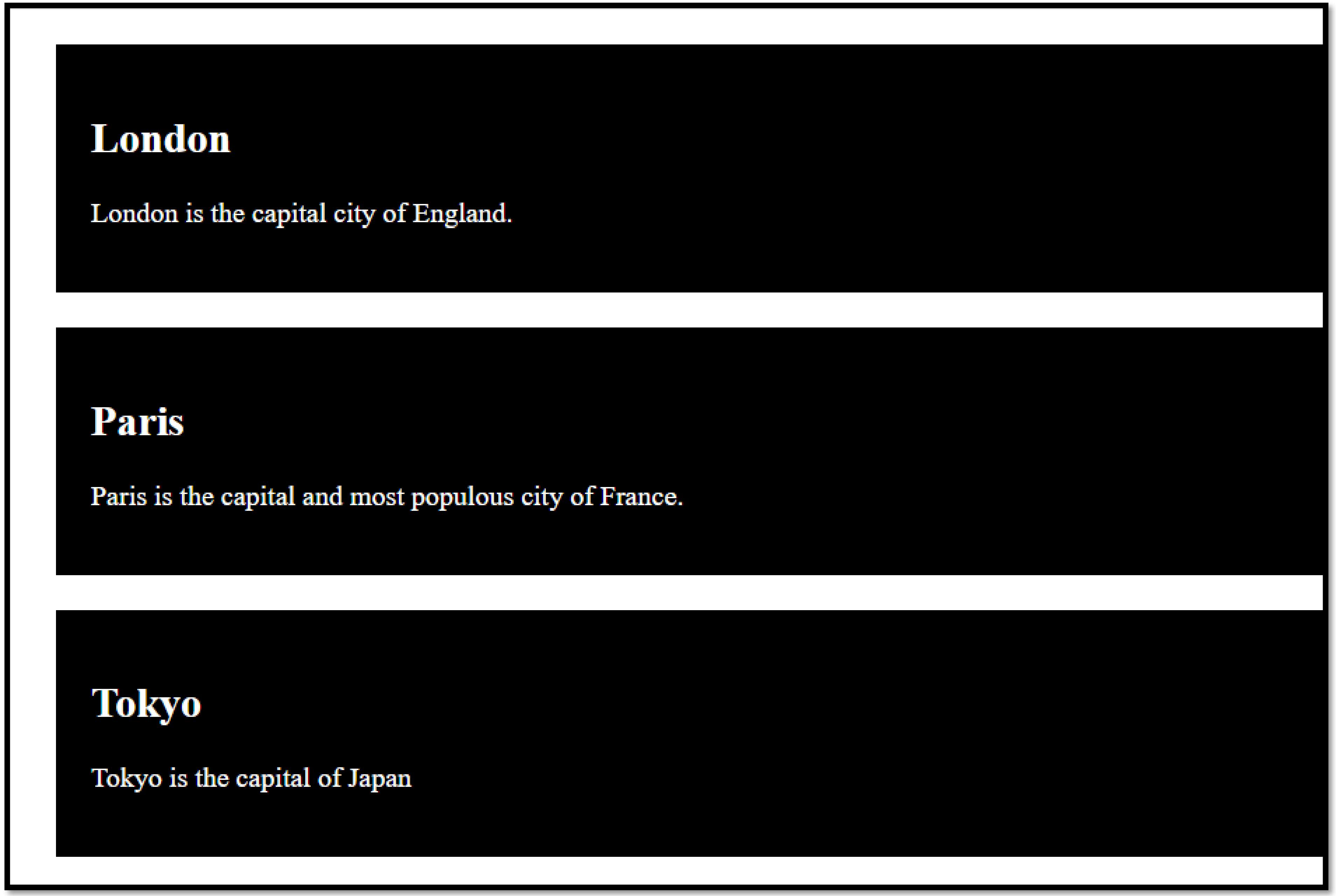
HMTL **class** Attribute

The HTML **class attribute makes it possible** to define, for example, **equal styles** for **"equal" <div> elements**

```
<!DOCTYPE html>
<html>

<head>
    <style>
        div.cities {
            background-color: black; color: white;
            margin: 20px; padding: 20px;
        }
    </style>
</head>

<body>
    <div class="cities">
        <h2>London</h2>
        <p>London is the capital city of England.</p>
    </div>
    <div class="cities">
        <h2>Paris</h2>
        <p>Paris is the capital and most populous city of France.</p>
    </div>
    <div class="cities">
        <h2>Tokyo</h2>
        <p>Tokyo is the capital of Japan </p>
    </div>
</body>

</html>
```

HMTL class Attribute

The previous HTML code will display this in the browser.



London

London is the capital city of England.

Paris

Paris is the capital and most populous city of France.

Tokyo

Tokyo is the capital of Japan

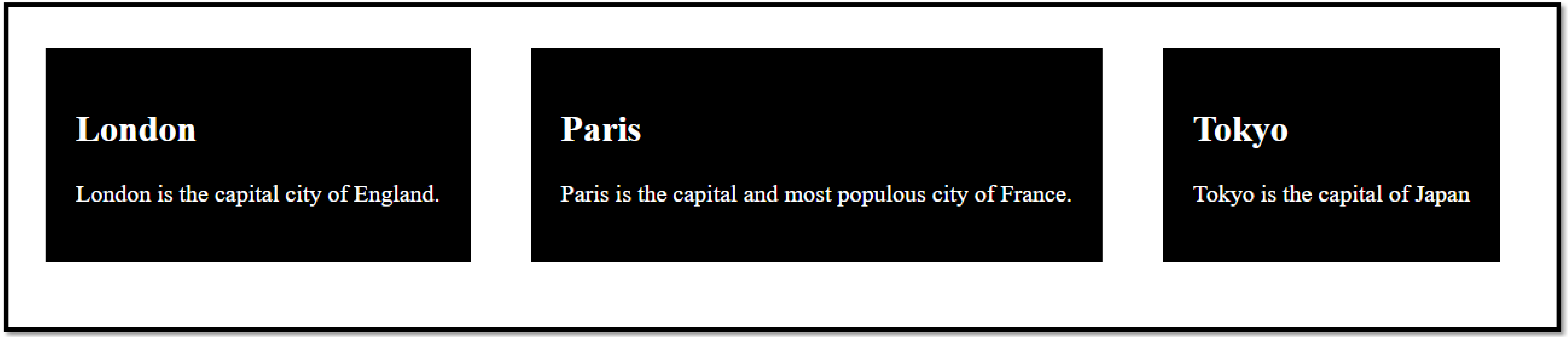
HMTL class Attribute with float property

- You can also use the **float** CSS style **property** to allow the elements of a **class float** to the **left** of its container (**in this case the container is the body**)…

```
<!DOCTYPE html>
<html>

<head>
    <style>
        div.cities {
            background-color: black; color: white;
            margin: 20px; padding: 20px; float: left;
        }
    </style>
</head>

<body>
    <div class="cities">
        <h2>London</h2>
        <p>London is the capital city of England.</p>
    </div>
    <div class="cities">
        <h2>Paris</h2>
        <p>Paris is the capital and most populous city of France.</p>
    </div>
    <div class="cities">
        <h2>Tokyo</h2>
        <p>Tokyo is the capital of Japan </p>
    </div>
</body>

</html>
```

HMTL class Attribute with float property

In this case the previous HTML code will display this in the browser



London

London is the capital city of England.

Paris

Paris is the capital and most populous city of France.

Tokyo

Tokyo is the capital of Japan

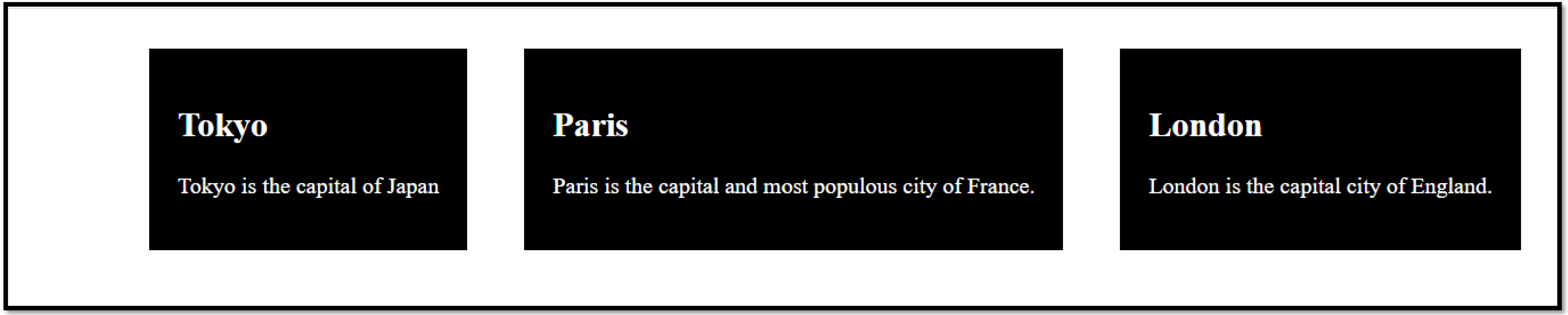
HMTL class Attribute with float property

- You can also use the **float** CSS/style **property** to allow the elements of a **class float** to the **right** of its container.

```
<!DOCTYPE html>
<html>

<head>
    <style>
        div.cities {
            background-color: black; color: white;
            margin: 20px; padding: 20px; float: right;
        }
    </style>
</head>

<body>
    <div class="cities">
        <h2>London</h2>
        <p>London is the capital city of England.</p>
    </div>
    <div class="cities">
        <h2>Paris</h2>
        <p>Paris is the capital and most populous city of France.</p>
    </div>
    <div class="cities">
        <h2>Tokyo</h2>
        <p>Tokyo is the capital of Japan </p>
    </div>
</body>

</html>
```

HMTL class Attribute with float property

In this case the previous HTML code will display this in the browser



Tokyo

Tokyo is the capital of Japan

Paris

Paris is the capital and most populous city of France.

London

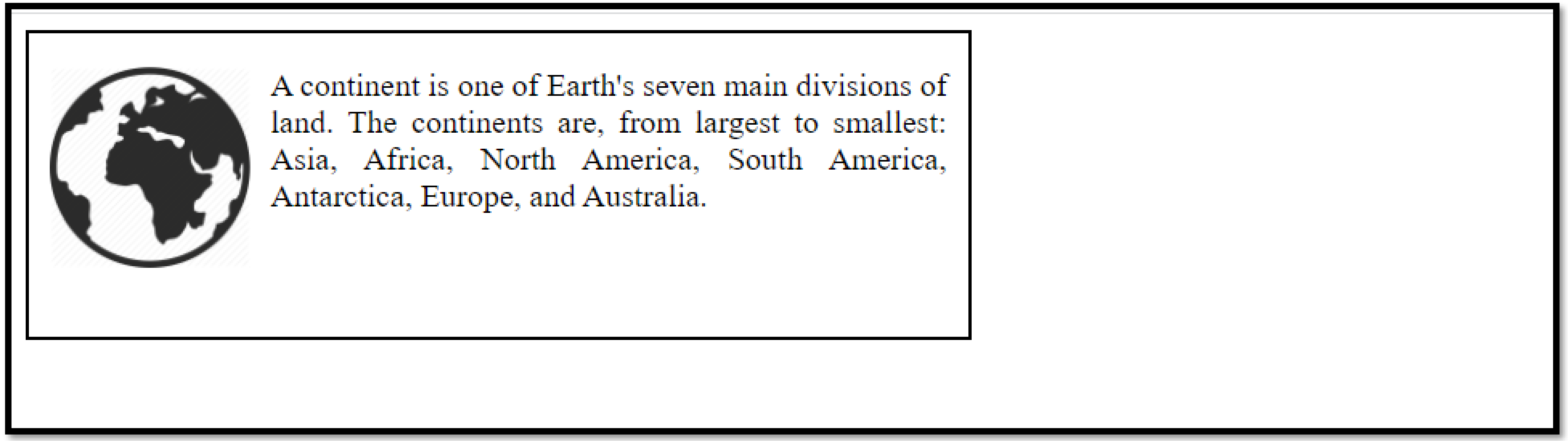
London is the capital city of England.

The CSS style float property with <img> elements

Example: In this case we want the image to **float** to the **left** of its container, which is the <div> element and write the text included in the <p> element next to it.

```
<!DOCTYPE html>
<html>

<head>
    <style>
        #myDIV {height: 150px;  width: 50%; background-color: white; border: solid 2px; padding: 20px;}
        img {float: left; height: 100px; width: auto; padding: 0px 10px;}
        p {text-align: justify; padding: 0px 10px;}
    </style>
</head>

<body>
    <div id="myDIV">
        <div>
            <img src="images/Earth_globe.png" alt="Earth Globe">
            <p>A continent is one of Earth's seven main divisions of land. The continents are, from
largest to smallest: Asia, Africa, North America, South America, Antarctica, Europe, and Australia.</p>
        </div>
    </div>
</body>

</html>
```

The CSS style float property with <img> elements



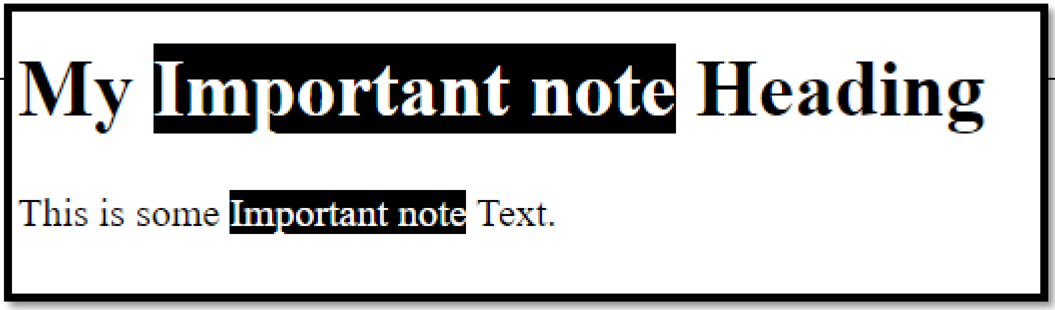
A continent is one of Earth's seven main divisions of land. The continents are, from largest to smallest: Asia, Africa, North America, South America, Antarctica, Europe, and Australia.

In this case the previous HTML code will display this in the browser

HMTL class Attribute with <span> elements

- The HTML **class attribute** also makes it possible to **define equal styles** for **"equal" <span>** elements:

```
<!DOCTYPE html>
<html>

<head>
    <style>
        span.notes {
            background-color: black; color: white;
        }
    </style>
</head>

<body>
    <h1>My <span class="notes">Important note</span> Heading</h1>
    <p>This is some <span class="notes">Important note</span> Text.</p>
</body>

</html>
```



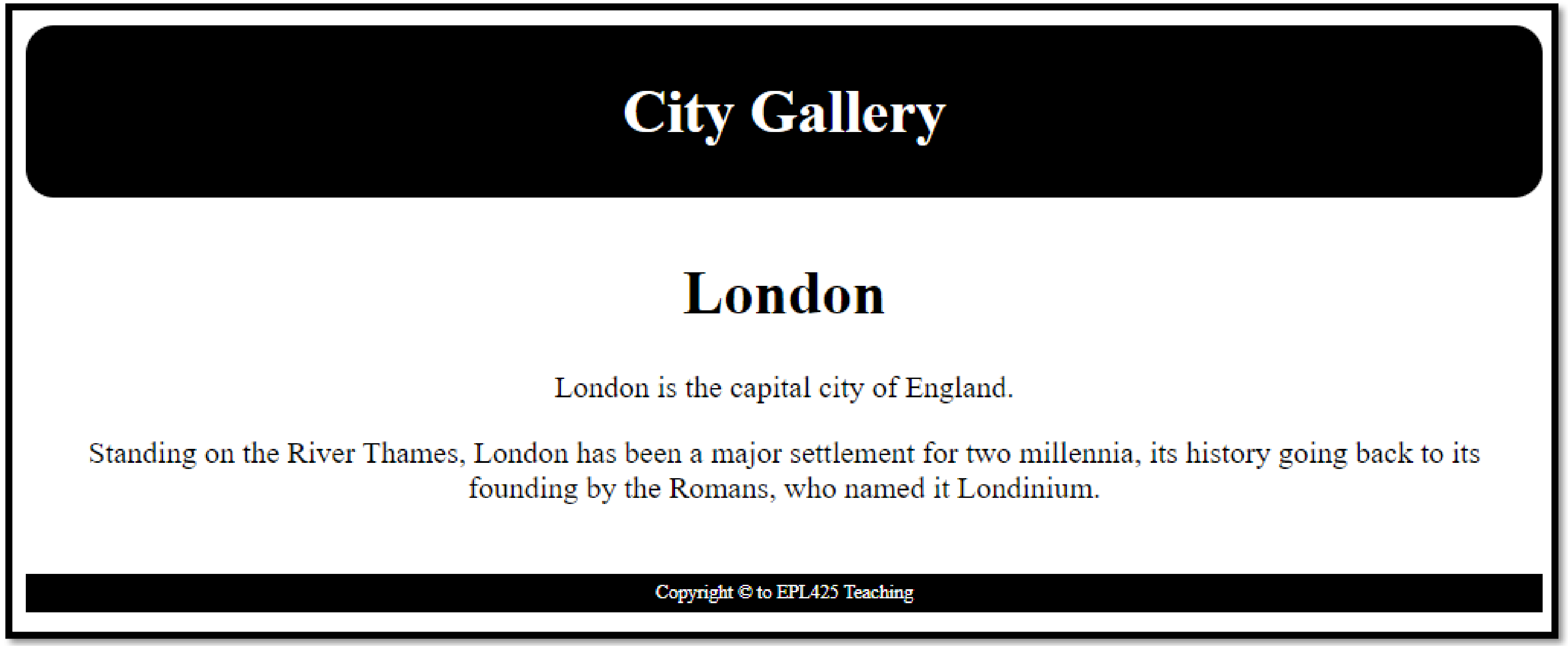
My Important note Heading

This is some Important note Text.

HMTL Layouts

- Websites often display content in **multiple rows** (like a magazine or newspaper).

- The **<div>** element is often **used** as a **layout tool**, because it can **easily** be **positioned with CSS**.



City Gallery

London

London is the capital city of England.

Standing on the River Thames, London has been a major settlement for two millennia, its history going back to its founding by the Romans, who named it Londinium.

Copyright © to EPL425 Teaching

The HTML Code for this layout is provided in the next slide

HMTL Layouts - Example

```
<!DOCTYPE html>
<html>

<head>
  <style>
    #header {background-color:black; border-radius: 15px; color:white; text-align:center; padding:5px;}
    #section {text-align:center; padding:10px;}
    #footer {background-color:black; color:white; text-align:center; padding:5px; font-size: 10px; line-height: 20px;}
  </style>
</head>

<body>
    <div id="header">
        <h1>City Gallery</h1>
    </div>

    <div id="section">
        <h1>London</h1>
        <p>London is the capital city of England.</p>
        <p>Standing on the River Thames, London has been a major settlement for two millennia, its history going back
to its founding by the Romans, who named it Londinium.</p>
    </div>

    <div id="footer">
        <p>Copyright © to EPL425 Teaching</p>
    </div>
</body>

</html>
```

This example uses three <div> elements used to create a multiple row layout in the body par.

HMTL Responsive Web Design

- Your **web page** should **look good**, and be **easy to use**, **regardless of the device** → **Responsive Web Design makes your web page** look good on **all devices** (desktops, tablets, and phones).
- Responsive Web Design is about using **CSS** and **HTML** to **resize**, **hide**, **shrink**, **enlarge**, or **move the content** to **make it look good on any screen.**

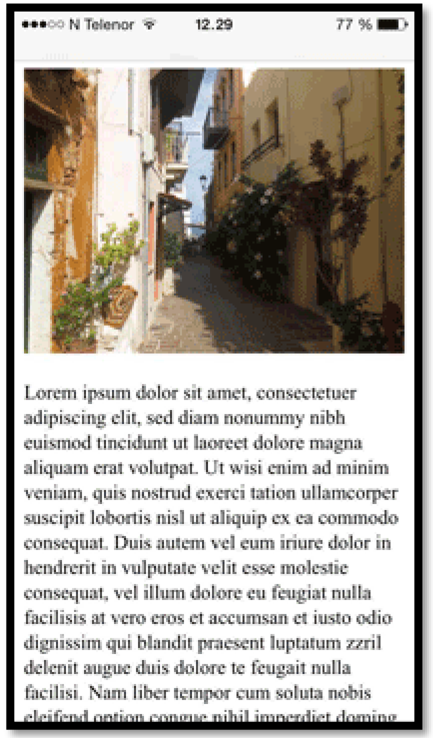
We will see more when we talk about CSS!!!

HMTL Responsive Web Design - Mobile vs Desktop browsers

- To **prevent phone browsers** from **rendering the page at desktop width** and **zooming out**, **use the meta viewport tag**:

```
<meta name="viewport" content="width=device-width, initial-scale=1">
```

- This belongs in the **<head>** section of your HTML, where the <title>, <link>, and **other metadata elements** are included.



Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat. Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et iusto odio dignissim qui blandit praesent luptatum zzril delenit augue duis dolore te feugait nulla facilisi. Nam liber tempor cum soluta nobis eleifend option congue nihil imperdiet doming

HMTL Responsive Web Design - Mobile vs Desktop browsers

```
<meta name="viewport" content="width=device-width, initial-scale=1">
```

- name="viewport" → Browser, I am going to tell you how I want the viewport to look."
- content="width=device-width, initial-scale=1":
  - width=device-width → The viewport's width should always start at the device's width.
  - initial-scale=1 → Start at zoom level of 100%.

You should pretty much always include this tag in your HTML!!

HMTL iFrames

- An **iframe** is used to **display** a **web page within a web page**. The syntax for adding an iframe is:

```
<iframe src="URL"> </iframe>
```

- The **src** attribute specifies the URL (web address) of the iframe page.
- For **more details** on iframes check this link

HMTL Scripts

- To **make HTML pages** more **dynamic** and **interactive** we can use **JavaScript**.
- The **<script>** tag is used to **define** and **embed** in our web page a **client-side script**, such as a **JavaScript**.
- The **<script> element** either **contains scripting statements** or it **points to an external script file** through the **src** attribute.
- Common uses for JavaScript are **image manipulation**, **form validation**, **dynamic changes** of HTML document’s **structure** and **content, etc.**

HMTL Scripts

Here are some examples of what JavaScript can do:

JavaScript can change HTML content:
document.getElementById("demo").**innerHTML** = "Hello JavaScript!";
JavaScript can change HTML styles:
document.getElementById("demo").**style.fontSize** = "25px";
JavaScript can change HTML attributes:
document.getElementById("image").**src** = "picture.gif";

We will learn more about JavaScript in a later lecture!!!

The HMTL <head> Element

- The **<head>** element is a **container for metadata** (data about data). HTML **metadata** is data **about** the **HTML document**.
- **Metadata** is **not displayed**. These data typically **define document title**, **styles**, **links**, **scripts**, and **other meta information** (e.g., author, keywords, page description, etc.).
- The following tags describe **metadata**: **<title>,** <style>, <meta>, <link>, <script>, and <base>.

The HMTL <head> Element

- In the HTML5 standard, the **<head>** tag **can be omitted**. **By default, browsers** will **add** all elements **before** <body>, to a **default** <head> element.
- In the **HTML5 standard**, the <html> tag and the <body> tag **can be omitted as well**
- However it is **recommended NOT to omit** these tags!
- The <title> element defines the title of the document and it is **required** in all HTML/XHTML documents.

```
<!DOCTYPE html>
<html>
<title>Page Title</title>

<body>
    <h1>This is a heading</h1>
    <p>This is a paragraph.</p>
</body>

</html>
```

The **<title>** element: i) **defines** a title in the **browser tab**; ii) provides a title for the page when it is added to **favorites**; iii) displays a title for the page in **search engine results**!!!

The HMTL <meta> and <base> Elements

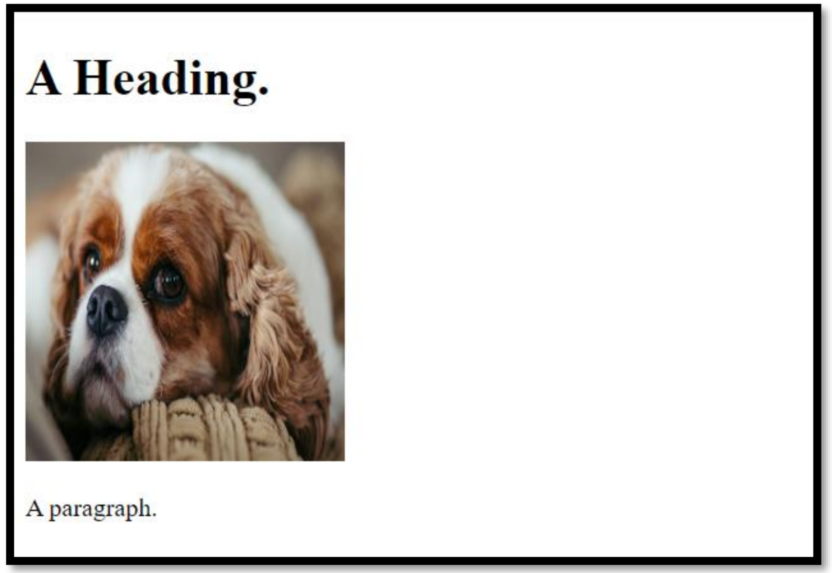
- The **<meta>** element is used to **specify** **page description**, **keywords**, **author**, and **other metadata**.
- **Metadata** is **used by** **browsers** (how to display content), by **search engines** (keywords), and other **web services**.
- At the right side are some examples of meta data

Define keywords for search engines:
`<meta name="keywords" content="EPL425, HTML, CSS,  JavaScript">`
Define a description of your web page:
`<meta name="description" content="Introduction to HTML, CSS, JS">`
Define the character set used:
`<meta charset="UTF-8">`
Define the author of a page:
`<meta name="author" content="Christophoros Christophorou">`
Refresh document every 30 seconds:
`<meta http-equiv="refresh" content="30">`
Specifies the base URL and base target for all relative URLs in a page
`<base href="http://www.mywebsite.com/images/" target="_blank">`

- The **<base>** element **specifies** the **base URL** and **base target** for **all relative URLs** in a page.

The HMTL <meta> and <base> Elements

```
<!DOCTYPE html>
<html>

<head>
    <title>Introduction to Web Development</title>
    <base href="https://www.helpguide.org/wp-content/uploads/" target="_blank">
    <meta name="keywords" content="EPL425, HTML, CSS, JavaScript">
    <meta name="description" content="Introduction to HTML, CSS, JS">
    <meta name="author" content="Christophoros Christophorou">
    <meta charset="UTF-8">
    <meta http-equiv="refresh" content="30">
</head>

<body>
    <h1>A Heading.</h1>
    <img src="king-charles-spaniel-resting-head-768.jpg" alt="Spaniel Dog" style="width: 200px; height: 200px;">
    <p>A paragraph.</p>

</body>

</html>
```


A Heading.

A paragraph.

HMTL Forms

- HTML forms are used to **collect user input**. The **<form> element defines** an **HTML form**.

<form>

Form elements - different types input elements, checkboxes, radio buttons, submit buttons, and more.

<form>

The <form> Element

- This **element** can **contain many other elements** as well including the below:
  - `<input>`
  - `<output>`
  - `<label>`
  - `<select>`
  - `<button>`
  - `<option>`
  - `<textarea>`
  - `<optgroup>`
  - `<fieldset>`

We will talk more about HTML forms, the elements of a form and AJAX, in a later lecture, however here we provide an Introduction!

The <input> Element

- The **<input>** element is the most important form element. It has **many variations**, **depending** on the **type** **attribute.** Below are some examples.

- For the full Input type attribute list check this link.

TYPE	DESCRIPTION
<input type="text"	Displays a single-line text input field
<input type="password"	Displays a password field (for showing asterisks in the text box)
<input type="radio">	Displays a radio button (for selecting one of many choices)
<input type="checkbox">	Displays a checkbox (for selecting zero or more of many choices).
<input type="submit">	Displays a submit button for submitting the form input to a form-handler
<input type="button">	Displays a clickable button (i.e., for calling a JavaScript function to handle the form input)

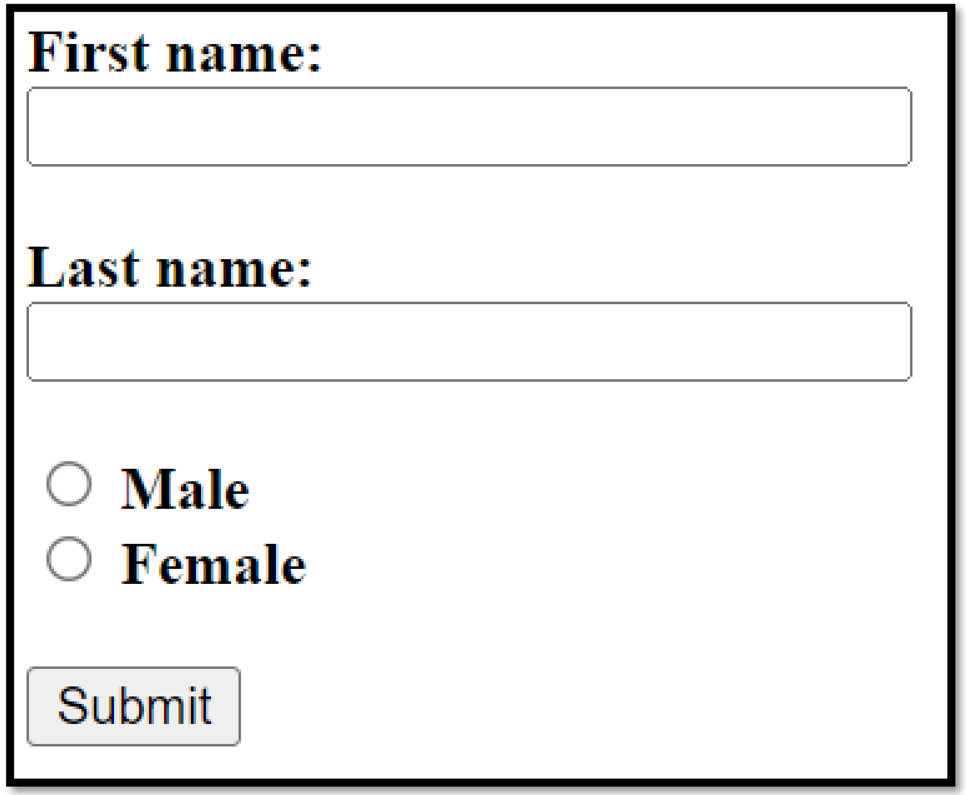
The <input> Element

The following HTML code…..

```
<!DOCTYPE html>
<html>
<head>
    <title>The first Input Form</title>
</head>

<body>
    <form action="action_page.php" method="post">
        <b>First name: </b> <br>
        <input type="text" name="firstname" size="30" maxlength="30"> <br><br>
        <b>Last name: </b> <br>
        <input type="text" name="lastname" size="30" maxlength="30"> <br><br>

        <input type="radio" name="sex1" value="male"> <b>Male</b> <br>
        <input type="radio" name="sex2" value="female"> <b>Female</b> <br><br>
        <input type="submit" value="Submit">
    </form>
</body>
</html>
```

The <input> Element

….will look like this in a browser!



Note that the **default size** of a **text field** is **20 characters**

The action and method attribute

- The **action attribute** defines the **action** to be **performed** when the **form** is **submitted**. The **method attribute** specifies the **HTTP method** (GET or POST) to be **used** when **submitting the forms (AJAX is used)**:

```
<form action="action_page.php" method="post">
```

- The **common way** to **submit** a form to a server, is by using a **submit button.**

```
<input type="submit" value="Submit">
```

- Normally, the **form is submitted** to a **web server** through the **form-handler**. The **form-handler** is typically a **server page** with a **script** for **processing input data** (i.e., **a php script file**). The form-handler is specified in the form's **action attribute.**

The name attribute

For **more details** on **form attributes** check this link.

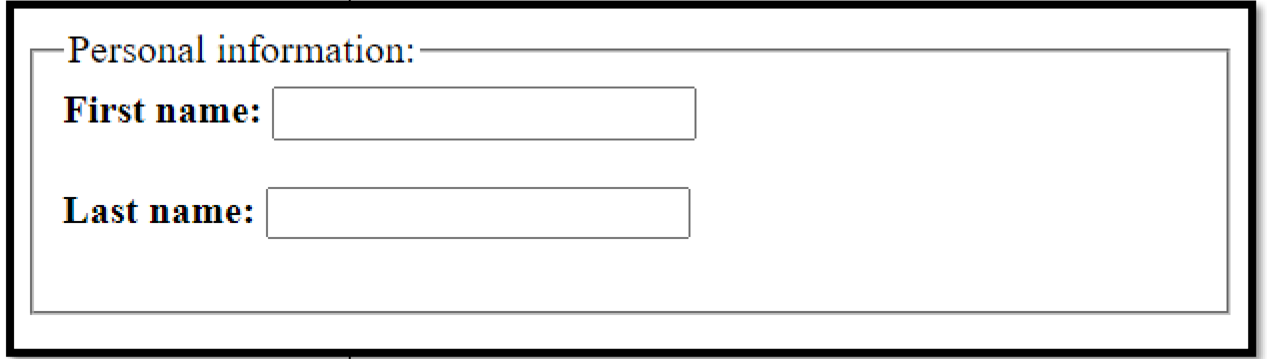
- The **name attribute** specifies the **name** of an <input> element. It is used:
  - To **reference elements** in a **JavaScript script (similarly to the id attribute)**, in case a JavaScript function is invoked using a **Button click**.
  - To **reference form data after a form is submitted (using the Submit button)**.

Note: Only form elements with a name attribute will have their values passed when submitting a form.

Grouping Form Data with <fieldset> and <legent> elements

- The **<fieldset>** element **groups related data** in a form while the **<legend>** element **defines** a **caption** for the **<fieldset>** element.

```
<!DOCTYPE html>
<html>
<title>The first Input Form</title>
<body>
    <form action="action_page.php">
        <fieldset>
            <legend>Personal information:</legend>
            <b>First name: </b>
            <input type="text" name="firstname"> <br><br>
            <b>Last name: </b>
            <input type="text" name="lastname"> <br><br>
        </fieldset>
    </form>
</body>
</html>
```


Personal information:
First name:
Last name:

In a browser it will look like this!

Note: If the **form input data** will be **handled by a JavaScript function** the **action attribute IS NOT NEEDED**. *Will see more details about this when we talk about HTML Forms and AJAX.*

What HTML elements can I use?

- **Q1: Instead of `<span class="highlight"></span>`, can I create a `<highlight>` element, even if this is not a valid HTML element?**

```
<p>EPL425 is <highlight>the best course ever!!!</highlight> </p>
```

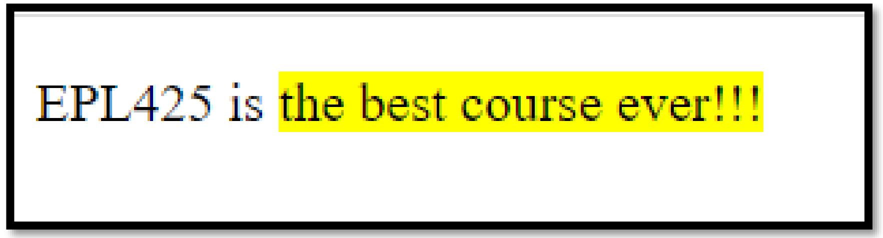
```
<style>
    highlight {
        background-color: yellow;
    }
</style>
```

Q2: Does this will work?

What HTML elements can I use?

```
<!DOCTYPE html>
<html>

<style>
    highlight {
        background-color: yellow;
    }
</style>

<title>Custom made elements</title>

<body>
    <p>
        EPL425 is <highlight>the best course ever!!!</highlight>
    </p>
</body>

</html>
```

This renders correctly!!


EPL425 is the best course ever!!!

But you shouldn't do this!

It is NOT a standard behavior.

What HTML elements can I use?

- What is "**standard**" HTML?
- Why does **invalid HTML/CSS still work** sometimes?
  - If my **Java code** is **wrong**, I get a **compiler error**…
  - If my **HTML** or **CSS** is **wrong**, **why don't I get an error**?

Q: **Why does it matter** that I **follow** " **standard**" **HTML coding?**

A very brief history of HTML….

HTML History

- Since the early days of the web, there have been **many versions** of HTML:

Version	Year
Tim Berners-Lee invented www	1989
Tim Berners-Lee invented HTML	1991
Dave Raggett drafted HTML+	1993
HTML Working Group defined HTML 2.0	1995
W3C Recommended HTML 3.2	1997
W3C Recommended HTML 4.01	1999
W3C Recommended XHTML 1.0	2000
HTML5 WHATWG First Public Draft	2008
HTML5 WHATWG Living Standard	2012
HTML5 W3C Final Recommendation	2014

HTML History

- **Tim Berners-Lee invented** the "**World Wide Web"** in **1989**, and the Internet took off in the 1990s.
- **1994**: **World Wide Web Consortium** (**W3C**) is **created** with **main goal** to **maintain** and **develop** **standards** about **how the web should work.** Oversees several languages: HTML, CSS, DOM, XML, etc.
- **1999**: "HTML4" was published. The **first major stable version** of HTML.

HTML History

- From **1991** to **1998**, HTML developed from **version 1** to **version 4.**
- In **2000**, the **World Wide Web Consortium (W3C)** **recommended** **XHTML 1.0**.
- The **XHTML** **syntax** **WAS STRICT**, and the **developers** **were forced** **to write** **valid** **and "well-formed" code**.

HTML History

- **In 2004, WHATWG (Web Hypertext Application Technology Working Group) was formed.**

- **This was done** in **response to slow** World Wide Web Consortium (**W3C) development**, and the *"crazy"* **W3C's decision** to **close down** the **development of HTML, in favor of XHTML**.

HTML History

- **WHATWG wanted to develop HTML, consistent with how the web was used, while being backward compatible** with **older versions of HTML.**
- In the period **2004-2006**, the **WHATWG initiative, gained support** by the **major browser vendors**.
- **In 2006, W3C announced** that they **would support WHATWG**.
- **In 2008, the first HTML5 public draft was released.**

HTML History

- In **2012**, **WHATWG** and **W3C** **decided on a separation**:
  - **WHATWG will develop HTML as a "Living Standard"**. A living standard is **never fully complete**, but **always updated** and **improved**. **New features CAN BE ADDED**, but **old functionality** **CANNOT BE REMOVED**. The WHATWG Living Standard was **published** in **2012**, and is **continuously updated**.
  - **W3C will develop a definitive HTML5 and XHTML5 standard**, as a **"snapshot"** of WHATWG. The W3C HTML5 recommendation was released **28 October 2014**.

HTML History …. Lets See what Really Happened

- The **W3C HTML spec**, lists several design principles, and one is the **"degrading gracefully"** principle:

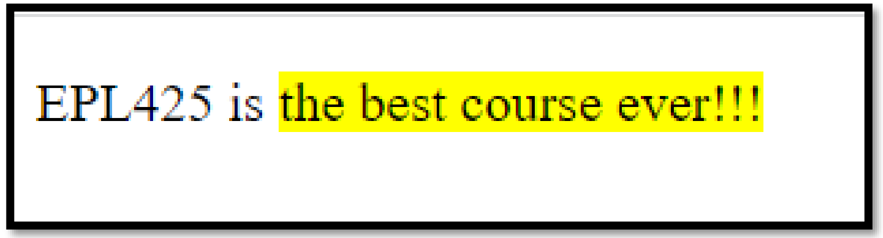
"An escalator **CAN NEVER BREAK!!!**
It can only BECOME STAIRS!!!"

This is why browsers do a **best-effort** to **render non-standard ("invalid") HTML** and **CSS**.

Best-effort rendering

It's also why <highlight> "works", **even though** it's **Invalid** HTML.

```
<!DOCTYPE html>
<html>

<style>
    highlight {
        background-color: yellow;
    }
</style>

<title>Custom made elements</title>

<body>
    <p>
        EPL425 is <highlight>the best course ever!!!</highlight>
    </p>
</body>

</html>
```


EPL425 is the best course ever!!!

Why not enforce strict HTML?

It's super weird that:

- **Browsers** **don't fail** when given **invalid** **HTML / CSS**
- Browsers **not only don't fail**, but they **render invalid HTML/CSS** **seemingly** **"correctly"!!!**

Q: Why the browser does not reject poorly written HTML/CSS?

Why not enforce strict HTML?

A: There was a (failed) attempt to enforce valid HTML/CSS code, but IT WAS TOO LATE!!!

The Internet GREW TOO BIG!!!!

Look what happened →

State of the world, 1997: In 1997, things were kind of a mess!

W3C®

Standards say one thing!!!

Main Page - Wikipedia, the free encyclopedia - Netscape Navigator

http://en.wikipedia.org/wiki/Main_Page

main page · discussion · view source · history

31,914 have donated.

You can help Wikipedia change the world! » Donate now!

From the fundraising blog » Put your money where your mind is, not where we tell you to

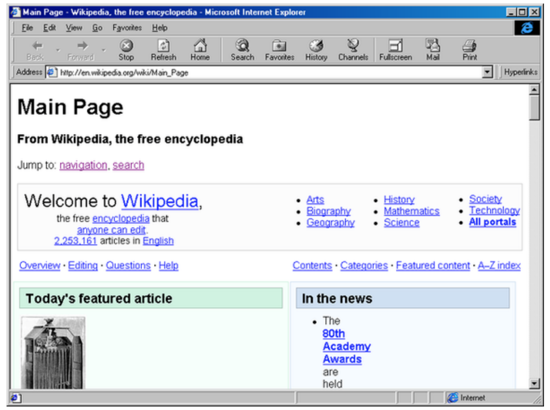
Welcome to Wikipedia, the free encyclopedia that anyone can edit

Today's featured article

Hoysala architecture is the distinctive building style developed under the rule of the Hoysala Empire in the region known today as Karnataka, India, between the 11th and the 14th centuries. Hoysala influence was at its peak in the 13th century, when it dominated the Southern Deccan Plateau region. Large and small temples built during this era remain as examples of the Hoysala architectural style, including the Chennakesava Temple at Belur, the Hoysaleswara Temple at Halebidu, and the Kesava Temple at Somanathapura. Other examples of fine Hoysala craftmanship are the temples at Belavadi, Amrithapura, Hosaholalu and Nuggehalli. Study of the Hoysala architectural style has

In the news

- A hostage crisis at Democratic presidential candidate Hillary Clinton's campaign office in Rochester, New Hampshire, USA ends peacefully.
- Protesters in Sudan demand the execution of a former Unity High School teacher for insulting Muhammad after the naming of the class teddy bear after the Islamic prophet.
- The wreckage of an Atlasjet plane carrying 56 passengers is found near Isparta in central Turkey, with no survivors.



Main Page - Wikipedia, the free encyclopedia - Microsoft Internet Explorer

Address http://en.wikipedia.org/wiki/Main_Page

Main Page

From Wikipedia, the free encyclopedia

Jump to: navigation, search

Welcome to Wikipedia, the free encyclopedia that anyone can edit. 2,253,161 articles in English

- Arts
- Biography
- Geography
- History
- Mathematics
- Science
- Society
- Technology
- All portals

Overview · Editing · Questions · Help

Contents · Categories · Featured content · A–Z index

Today's featured article

In the news

- The 80th Academy Awards are held

Browsers do another thing!!!

Developers write weird, non-standard code!!!

2000: W3C Recommended XHTML 1.0

W3C

Oops, that "degrading gracefully" thing was a mistake.

Browsers, stop rendering pages that have invalid HTML!!!

(This was the proposal of XHTML 1.1 in 2000)

2000: W3C Recommended XHTML 1.0

W3C

Oops, that "degradi graceful a

Browsers, p rendering pages that TML.

Sure, whatever

What?!?!

We can't do that!

That would break the internet!

2004: WHATWG formed

In 2004

Let's burn everything and start from scratch with XHTML 1.1

This broke approx. 64 million websites!!!

Let's work on HTML5

(an imperfect but realistic standard)

Fast forward 2017?!

- W3C **gave up** **XHTML 1.1** in 2007
- W3C and WHATWG are **mostly friends** (I think), though they are still separate entities

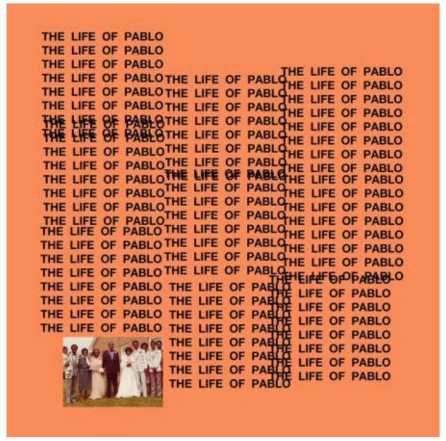
"HTML5" vs HTML

W3C now maintains HTML5:

- More **stable version** of WHATWG's HTML
- Usually **copies** what WHATWG does **after the dust settles**

WHATWG maintains HTML: The Living Standard

- No number, no versions
- **Updated frequently** and being updated today!
- **Most browsers implement WHATWG**
- This is why we don’t refer to it as "HTML5"



What you need to know

Q: What HTML elements can I choose from?

- Check MDN's list of HTML tags

Q: How do I know if an HTML tag (or CSS property, or JS feature) **is implemented on all browsers?**

- Check caniuse.com

What you need to know

Q: Why shouldn't I use non-standard HTML-CSS-JavaScript, even if it works in every browser?

- Because it **won't be guaranteed** to **work in the future**!!!
- Because it **won't be guaranteed** to work **on all "user agents"** (not just browsers)!!!

In computing, a **user agent** is any software, acting on behalf of a user, which "retrieves, renders and facilitates end-user interaction with Web content". Some prominent examples of user agents are web browsers and email readers.

Ερωτήσεις?