code4sp
coding for social promotion

Co-funded by the
Erasmus+ Programme
of the European Union

3.1.: Code4SP Training Material Package

WP3: Code4SP Training Materials

Prepared by:

Project Information

Project Acronym: Code4SP

Project Title: Coding for Social Promotion

Project Reference: 621417-EPP-1-2020-1-PT-EPPKA3-IPI-SOC-IN

Project website: www.code4sp.eu

Authoring Partner: CEPROF

Document Version: 2

Date of Preparation: 22/03/2022

Document History			
Date	Version	Author	Description
11/02/2022	1	CEPROF	Draft
22/03/2022	2	CEPROF	Revision

Table of contents

Topic Information

Topic:

2. HTML

Prerequisites:

Basic computer literacy, basic software installed, and basic knowledge of working with files.

Workload:

10 hours.

Description:

In this topic, we cover the basics of HTML, to get learners up to speed in the world of programming, after having acquired some basic concepts. We define the elements, attributes and all the other important terms they may have heard and where these terms fit into the language. We also show how an HTML element is structured, how a typical HTML page is structured and explain other important basic features of the language.

Learning outcomes:

- Recognize the concept of HyperText Markup Language (HTML) in the family of document description languages.
- Distinguish between the structure, content and styles of a page.
- Use HTML in the construction of pages for the web.

Material required:

- Computer or laptop
- Internet connection
- Online website builder (https://sites.google.com/new)
- Online text editor (https://www.w3schools.com/html/default.asp)

Lesson Scenario:

The total time for this topic is 10 hours, and it will be up to the trainer/coach to decide how much time to dedicate to teaching each subtopic. In order to make the most of all the time available, we propose the use of the training materials produced by the project (PPT presentations), which were designed with an effective use of time in mind. These presentations are composed of the following elements:

- Development of the subtopic and main ideas to retain;
- Proposed Activities/Exercises.

That said, if the trainer/coach follows the logical sequence of the PPTs, he/she will certainly be able to complete the session within the stipulated time limit. These presentations can also be made available to learners for individual study.

Subtopics:

2.1. The basics of HTML

2.2. HTML advanced concepts

2.3. HTML5 features

2.4. HTML5 references

Additional resources:

- HTML reference guide
- W3Schools - Guide for every HTML element and CSS rule, and examples for each one of them
- Khan Academy: useful resources and videos on coding HTML & CSS, in many different languages

2.1. The basics of HTML

HTML (Hypertext Markup Language) **is not** a programming language. It is a markup language that communicates web browsers how to structure the web pages visited. It can be as complex or as simple as the web developer wants it to be. HTML comprises a series of elements, which you use to enclose, wrap, or rise different parts of content to make it appear or act in a certain way. The enclosing tags can make content into a hyperlink to connect to another page, italicize words, and so on. For example, considering the following line of text:

```
HTML is cool.
```

Figure 1 *- Line of text "HTML is cool" (**Source:** Author)*

If one wanted the text to stand by itself, one could specify that it is a paragraph by enclosing it in a paragraph (<p>) element:

```
<p> HTML is cool.</p>
```

Figure 2 *– Coding for the paragraph "HTML is cool" (**Source:** Author)*

Note: Tags in HTML are not case-sensitive. This means they can be written in uppercase or lowercase. For example, a <title> tag could be written as <title>, <TITLE>, <Title>, <TiTlE>, etc., and it will work. Nevertheless, it is best practice to write all tags in lowercase for consistency and readability.

Anatomy of an HTML element

Let's further explore our paragraph element from the previous section:

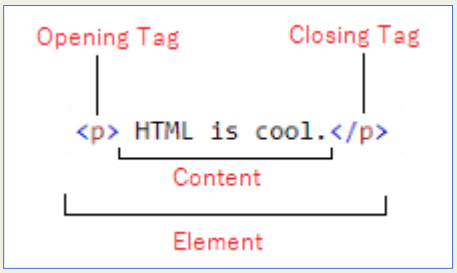


Figure 3 *– Anatomy of an HTML element (**Source:** Author)*

Therefore, the anatomy of the element is composed of:

The opening tag: the name of the element (in this example, *p* for paragraph), wrapped in opening and closing angle brackets. This opening tag marks where the element begins or starts to take effect. In this example, it comes first, at the start of the paragraph text.

The content: This is the content of the element. In this example, it is the paragraph text.

The closing tag: This is the same as the opening tag, except that it includes a forward slash before the element name. This marks where the element ends. Failing to include a closing tag is a common beginner error that can produce peculiar results.

The **element** is the opening tag, followed by content, followed by the closing tag.

Note: Some HTML elements have no content (as the <br> element). These elements are named “empty elements”. They do not have an end tag!

Web Browsers

The purpose of a web browser (Chrome, Edge, Firefox, Safari) is to read HTML documents and display them properly.

A browser does not display the HTML tags. It uses them to determine how to display the document:

Figure 4 *– An HTML document determined in a web browser (**Source:** https://www.w3schools.com/html/html_intro.asp)*

HTML Page Structure

An HTML page should be structured as follows:

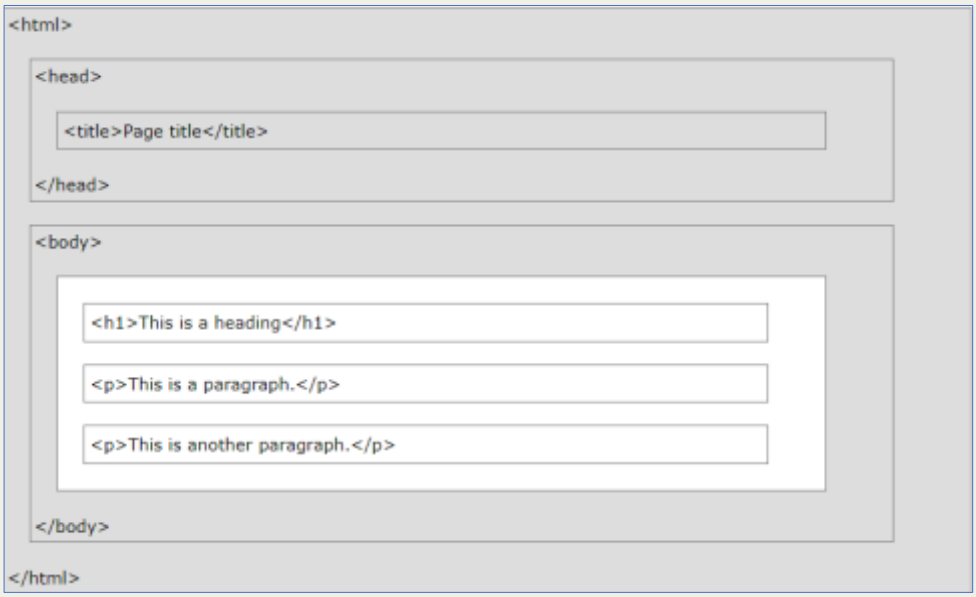


Figure 5 *– Structure of an HTML page (**Source:** https://www.w3schools.com/html/html_intro.asp)*

The content inside the <body> section (the white area above) will be displayed in a browser. The content inside the <title> element will be shown in the browser's title bar or in the page's tab.

HTML History

Since the early days of the World Wide Web, there have been many versions of HTML:

Year	Version
1989	Tim Berners-Lee invented www
1991	Tim Berners-Lee invented HTML
1993	Dave Raggett drafted HTML+
1995	HTML Working Group defined HTML 2.0
1997	W3C Recommendation: HTML 3.2
1999	W3C Recommendation: HTML 4.01
2000	W3C Recommendation: XHTML 1.0
2008	WHATWG HTML5 First Public Draft
2012	WHATWG HTML5 Living Standard
2014	W3C Recommendation: HTML5
2016	W3C Candidate Recommendation: HTML 5.1
2017	W3C Recommendation: HTML5.1 2nd Edition
2017 -	W3C Recommendation: HTML5.2

***Table 1** – HTML story (**Source:** https://www.w3schools.com/html/html_intro.asp)*

This manual follows the latest HTML5 standard.

HTML Editors

A simple text editor is all you need to learn HTML.

Learn HTML Using Notepad or TextEdit

Web pages can be created and modified by using professional HTML editors.
However, for learning HTML a simple text editor like Notepad (PC) or TextEdit (Mac) is recommended. Using a simple text editor can be a good way to learn HTML
The steps below should be followed to create learners' first web page with Notepad or TextEdit.

Step 1: Open Notepad (PC)

(Windows 8 or later: Open the Start Screen (the window symbol at the bottom left on your screen). Type Notepad.

Windows 7 or earlier: Open Start > Programs > Accessories > Notepad)

- Open TextEdit (Mac)
- Open Finder > Applications > TextEdit
- Get the application to save files correctly. In Preferences > Format > choose "Plain Text"
- Then under "Open and Save", check the box that says "Display HTML files as HTML code instead of formatted text".
- Then **open a new document** to place the code.

Step 2: Writing Some HTML

- Write or copy the following HTML code into Notepad:

```
<!DOCTYPE html>
```

```
<html>
<body>
<h1> My First Heading</h1>
<p>My first paragraph.</p>
</body>
</html>
```

Step 3: Save the HTML Page

- Save the file on your computer.
- Select File > Save as in the Notepad menu.
- Name the file "index.htm" and set the encoding to UTF-8 (which is the preferred encoding for HTML files).

Step 4: View the HTML Page in Your Browser

- Open the saved HTML file in your favorite browser (double click on the file, or right-click - and choose "Open with").

The result will be similar to the following:

***Figure 6** – An HTML document determined in a web browser (**Source:** https://www.w3schools.com/html/html_intro.asp)*

HTML Basic Examples

In this section, some basic HTML examples will be shared.

HTML Documents

All HTML documents must start with a document type declaration: <!DOCTYPE html>.

The <!DOCTYPE> declaration represents the document type and helps browsers to display web pages correctly. It must only appear once, at the top of the page (before any HTML tags). It is not case sensitive.

The <!DOCTYPE> declaration for HTML5 is: `<!DOCTYPE html>`

HTML Headings

HTML headings are defined with the <h1> to <h6> tags.
<h1> defines the most important heading. <h6> defines the least important heading:

```
<h1>This is heading 1</h1>
<h2>This is heading 2</h2>
<h3>This is heading 3</h3>
```

***Figure 7** – HTML headings (**Source:** https://www.w3schools.com/html)*

HTML Paragraphs

HTML paragraphs are defined with the <p> tag:

```
<p>This is a paragraph.</p>
<p>This is another paragraph.</p>
```

***Figure 8** – HTML paragraphs (**Source:** https://www.w3schools.com/html)*

HTML Links

HTML links are defined with the <a> tag:

```
<a href="https://www.w3schools.com">This is a link</a>
```

*Figure 9 – HTML link (**Source:** https://www.w3schools.com/html)*

The link's destination is specified in the href attribute.

Attributes are used to provide additional information about HTML elements.

More about attributes will be taught at a later stage.

HTML Images

HTML images are defined with the <img> tag.

The source file (src), alternative text (alt), width, and height are provided as attributes:

```
<img src="w3schools.jpg" alt="W3Schools.com" width="104" height="142">
```

*Figure 10 – HTML images (**Source:** https://www.w3schools.com/html)*

How to View HTML Source?

View HTML Source Code:

- Right-click in an HTML page and select "View Page Source" (in Chrome) or "View Source" (in Edge), or similar in other browsers. This will open a window containing the HTML source code of the page.

Inspect an HTML Element:

- Right-click on an element (or a blank area) and choose "Inspect" or "Inspect Element" to see what elements are made up of (you will see both the HTML and

the CSS). You can also edit the HTML or CSS on-the-fly in the Elements or Styles panel that opens.

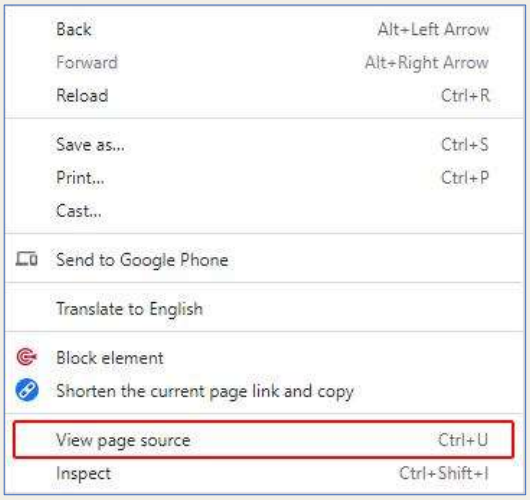


***Figure 11** – How to view page source (**Source:**Author)*

The structure of a HTML document

The HTML document itself begins with <html> and ends with </html>. The visible part of the HTML document is between <body> and </body>.

```
<!DOCTYPE html>
<html>
<body>

<h1>My First Heading</h1>
<p>My first paragraph.</p>

</body>
</html>
```

***Figure 12** – Document type declaration, HTML and visible part of the HTML document (**Source:** https://www.w3schools.com/html)*

HTML Attributes

HTML attributes provide additional information about HTML elements, and all HTML elements can have them. Attributes provide additional information about elements,

being always specified in the start tag. They usually come in name/value pairs like: name="value".

List of common attributes:

- **href** – the <a> tag defines a hyperlink. The href attribute specifies the URL of the page the link goes to, as follows:

```
<a href="https://www.w3schools.com">Visit W3Schools</a>
```

- **src** – the <img> tag is used to embed an image in an HTML page. The src attribute specifies the path to the image to be displayed, as seen below:

```
<img src="img_girl.jpg">
```

There are two ways to specify the URL in the src attribute:

1. **Absolute URL** - Links to an external image that is hosted on another website. E.g.: *src="https://www.w3schools.com/images/img_girl.jpg".*

Notes: External images might be under **copyright**. If one does not get permission to use it, he/she may be in violation of copyright laws. In addition, external images cannot be controlled; it can abruptly be removed or changed.

2. **Relative URL** - Links to an image that is hosted within the website. Here, the URL does not include the domain name. If the URL begins without a slash, it will be relative to the current page. E.g*: src="img_girl.jpg"*. If the URL begins with a slash, it will be relative to the domain. E.g.: *src="/images/img_girl.jpg".*
Hint: It is almost always best to use relative URLs. They will not break if the domain changes.

- **width and height attributes** – the <img> tag should also comprise the width and height attributes, which stipulates the width and height of the image (in pixels):

```
<img src="img_girl.jpg" width="500" height="600">
```

- **alt –** the required alt attribute for the <img> tag specifies an alternate text for an image, if the image cannot be displayed for some reason. This can be due to slow connection, or an error in the src attribute, or if the user uses a screen reader.

```
<img src="img_girl.jpg" alt="Girl with a jacket">
```

  If we try to display an image that does not exist, the value of the alt attribute will be displayed instead, as follows:

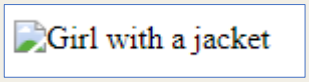
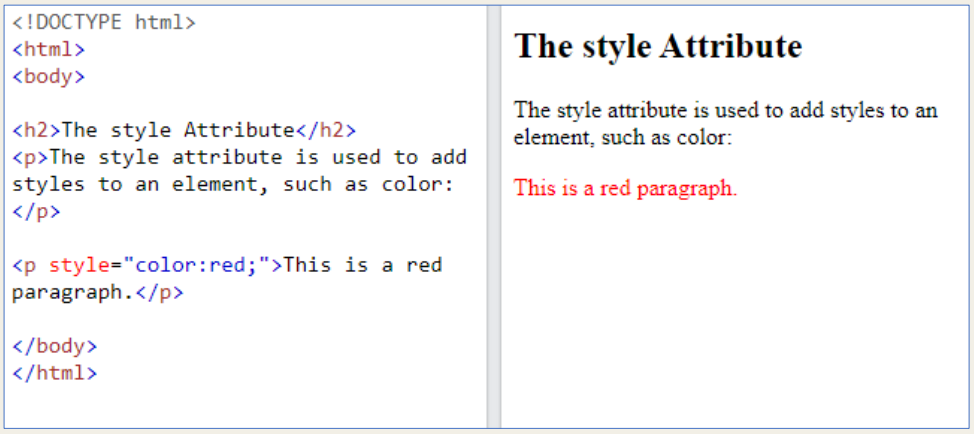


*Figure 13 – Alt value if image does not exist (**Source:** https://www.w3schools.com/html)*

- **style** – the style attribute is used to add styles to an element, such as color, font, size, and more.

```
<p style="color:red;">This is a red paragraph.</p>
```

Result:



```
<!DOCTYPE html>
<html>
<body>

<h2>The style Attribute</h2>
<p>The style attribute is used to add
styles to an element, such as color:
</p>

<p style="color:red;">This is a red
paragraph.</p>

</body>
</html>
```

The style Attribute

The style attribute is used to add styles to an element, such as color:

This is a red paragraph.

*Figure 14 – Code for coloring a paragraph (**Source:** https://www.w3schools.com/html)*

- **lang** – the lang attribute should always be included inside the <html> tag, to declare the language of the Web page. This is meant to support search engines and browsers. The example below stipulates English as the language in use:

```
<!DOCTYPE html>
<html lang="en">
<body>
…
</body>
</html>
```

Country codes can also be added to the language code in the lang attribute. So, the first two characters express the **language** of the HTML page, and the last two characters outline the **country**.

The following example specifies Portuguese as the language and Portugal as the country:

```
<!DOCTYPE html>
<html lang="pt-PT">
<body>
...
</body>
</html>
```

HTML Language Code Reference includes all the language codes.

- **title –** this attribute describes some additional information about an element.

Its value will be displayed as a tooltip when the mouse pointer goes over the element, as follows:

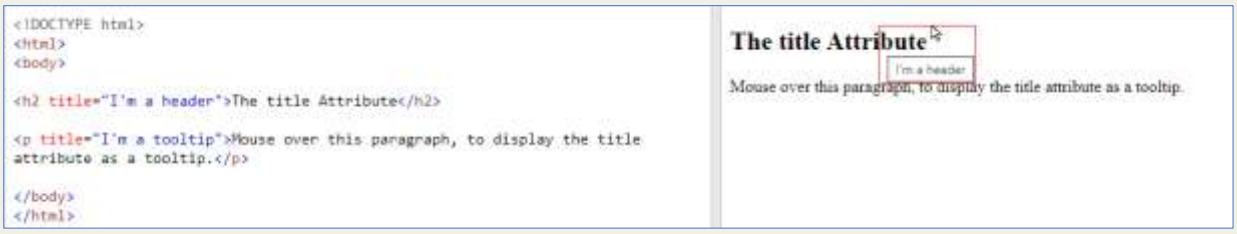


***Figure 15** – Code for displaying a 'title' tooltip (**Source:** Author)*

Recommendation: It is highly recommended to always use lowercase attributes and to always quote attribute values for stricter document types like XHTML.

HTML Headings

HTML headings are titles or subtitles that one wants to display on a webpage.

```
<!DOCTYPE html>
<html>
<body>

<h1>Code4SP 1</h1>
<h2>Code4SP 2</h2>
<h3>Code4SP 3</h3>
<h4>Code4SP 4</h4>
<h5>Code4SP 5</h5>
<h6>Code4SP 6</h6>

</body>
</html>
```

Code4SP 1

Code4SP 2

Code4SP 3

Code4SP 4

Code4SP 5

Code4SP 6

Figure 16 *– Code for adding headings (**Source:** Author)*

HTML headings are delineated with the <h1> to <h6> tags. <h1> identifies the most important heading. <h6> outlines the least important heading. <h1> headings should be used for main headings, followed by <h2> headings, then the less important <h3>, and so on.

Headings are of upmost importance since search engines use them to index the structure and content of webpages. Users often browse a page by its headings. It is important to use headings to exhibit the document structure.

It is important to state that, by definition, browsers automatically add a margin before and after a heading.

Recommendation: It is highly recommended to use HTML headings for headings only, not to make text big or bold.

Moreover, in the CSS topic, it will be taught that headings' size can be specified using the **style** attribute, using the CSS font-size property, as follows:

```
<h1 style="font-size:60px;">Heading 1</h1>
```

HTML Paragraphs

A paragraph constantly starts on a new line and is typically a block of text. It is defined by the HTML <p> element and, like headings, browsers automatically add some margin before and after a paragraph.

```
<!DOCTYPE html>
<html>
<body>

<p>Code4SP helps me to code.</p>
<p>I love Code4SP.</p>
<p>Coding is so great!</p>

</body>
</html>
```

Code4SP helps me to code.

I love Code4SP.

Coding is so great!

Figure 17 *– Code for adding paragraphs (**Source:** Author)*

HTML Display

One cannot be sure how HTML will be presented, as it can vary from screen to screen. With HTML, the display cannot be changed by adding extra spaces or extra lines in the HTML code.

The browser will automatically remove any extra spaces and lines when the page is displayed, as seen in the following example:

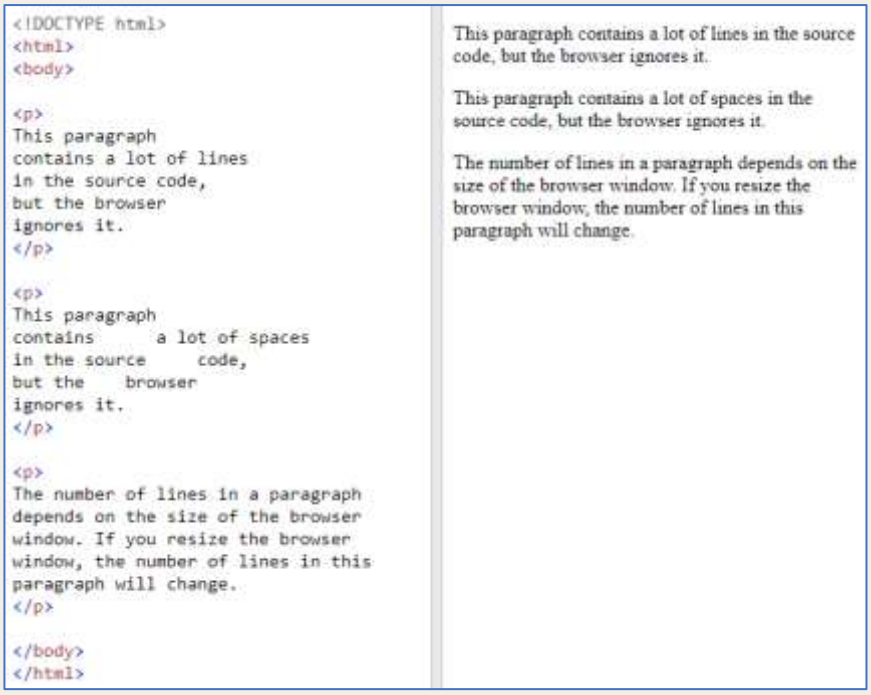


Figure 18 *– Example of how browsers tend to ignore spaces (**Source:** https://www.w3schools.com/html)*

HTML Line Breaks

The HTML <br> element defines a line break. It is an empty tag, which means that it has no end tag.

 should be used if one wants a new line without starting a new paragraph:

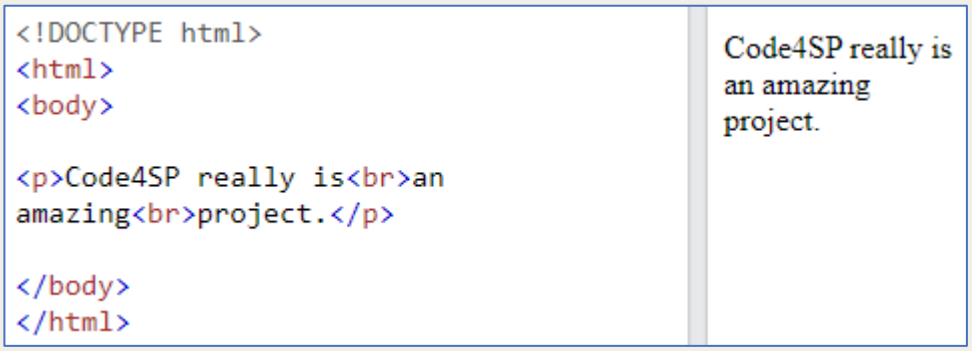


Figure 19 *– The <br> element (**Source:** Author)*

HTML Styles

The HTML style attribute is used to add styles to an element, such as color, font, size, etc. To set the style of an HTML element, the style attribute must be used. It has the following syntax (it should be noted that *property* and *value* are CSS features, to be learned later).

```
<tagname style="property:value;">
```

- **Background Colour**

The CSS *background-color* property specifies the background colour for an HTML element.

```
<!DOCTYPE html>
<html>
<body style="background-color:green;">

<h1>Code4SP</h1>
<p>Coding for Social Promotion.</p>

</body>
</html>
```

Figure 20 *– Setting a background colour (**Source:** Author)*

- **Text Colour**

The CSS *color* property outlines the text colour for an HTML element:

```
<!DOCTYPE html>
<html>
<body>

<h1 style="color:blue;">Code4SP</h1>
<p style="color:red;">Coding for Social
Promotion.</p>

</body>
</html>
```

Figure 21 *– Setting a background colour (**Source:** Author)*

- **Fonts**

The CSS *font-family* property defines the font to be used for an HTML element:

```
<!DOCTYPE html>
<html>
<body>

<h1 style="font-
family:verdana;">Code4SP</h1>
<p style="font-family:courier;">Coding
for Social Promotion.</p>

</body>
</html>
```

Code4SP

Coding for Social Promotion.

Figure 22 *– Setting the font to be used for an HTML element (**Source:** Author)*

- **Text Size**

The CSS *font-size* property identifies the text size for an HTML element:

```
<!DOCTYPE html>
<html>
<body>

<h1 style="font-
size:300%;">CODE4SP</h1>
<p style="font-size:160%;">Coding for
Social Promotion.</p>

</body>
</html>
```

CODE4SP

Coding for Social Promotion.

Figure 23 *– Setting the text size for an HTML element (**Source:** Author)*

- **Text Alignment**

The CSS *text-align* feature defines the horizontal text alignment for an HTML element:

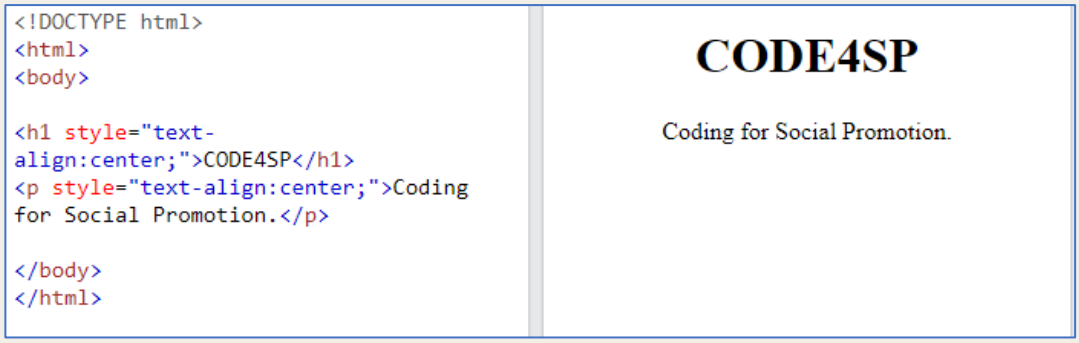


***Figure 24** – Text Alignment feature (**Source:** Author)*

HTML Text Formatting

HTML comprises various elements for defining text with a special implication (bold, italic, subscript, superscript, etc.).

The following are the **HTML formatting elements**:

Property	Outcome	Definition	Example
<b>	Bold text	The HTML <b> element specifies bold text, without any extra importance.	**Bold text**
	Important text	The HTML element describes text with strong importance. The content inside is usually displayed in bold.	**Important text**
<i>	Italic text	The HTML <i> element defines a part of text in an alternate voice or mood. The content inside is typically displayed in italic.	*Italic text*

	Emphasized text	The HTML element defines emphasized text. The content inside is typically displayed in italic.	*Emphasized text*
<mark>	Marked text	The HTML <mark> element defines text that should be marked or highlighted.	Marked text
<small>	Smaller text	The HTML <small> element defines smaller text.	Smaller text
<del>	Deleted text	The HTML <del> element defines text that has been deleted from a document. Browsers will usually strike a line through deleted text.	Deleted ~~text~~
<ins>	Inserted text	The HTML <ins> element defines a text that has been inserted into a document. Browsers will usually underline inserted text:	~~Inserted~~ text.
<sub>	Subscript text	The HTML <sub> element expresses subscript text. Subscript text appears half a character below the normal line and is sometimes rendered in a smaller font. Subscript text can be used for Chemistry, like H_2O.	Su$_b$script text
<sup>	Superscript text	The HTML <sup> element specifies superscript text.	Su^{p}erscript text

		Superscript text appears half a character above the normal line and is sometimes rendered in a smaller font. Superscript text can be used for footnotes, as WWW [1]:	

HTML Text Formatting

HTML <blockquote> for Quotations

The HTML <blockquote> element identifies a section that is quoted from another source. Browsers typically indent <blockquote> elements, as can be seen below:

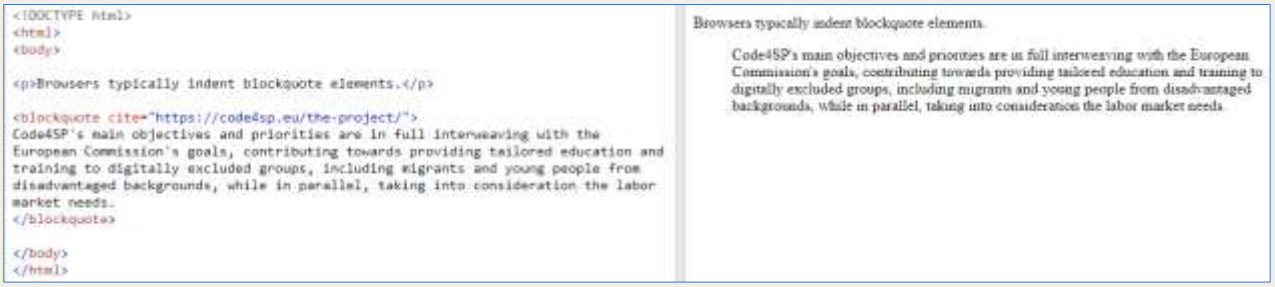


***Figure 25** – The blockquote element (**Source:** Author)*

HTML <q> for Short Quotations

The HTML <q> tag specifies a short quotation. Browsers generally insert quotation marks around the quotation, as follows:

```
<!DOCTYPE html>
<html>
<body>

<p>Browsers usually insert quotation
marks around the q element.</p>

<p>WWF's goal is to: <q>Build a future
where people live in harmony with
nature.</q></p>

</body>
</html>
```

Browsers usually insert quotation marks around the q element.

WWF's goal is to: "Build a future where people live in harmony with nature."

Figure 26 *– The short quotation's element (**Source:** https://www.w3schools.com/html/html_quotation_elements.asp)*

HTML <abbr> for Abbreviations

The HTML <abbr> tag specifies an abbreviation or an acronym, like "HTML", "CSS", "Mr.", "Dr.", "ASAP", etc. Marking abbreviations can give valuable information to browsers, translation systems and search engines, as seen previously. In case one does not know the meaning of any given abbreviation, he/she could use the global title attribute to show the description for the abbreviation/acronym when mousing over the element, as seen below:

```
<!DOCTYPE html>
<html>
<body>

<p>The <abbr title="World Health
Organization">WHO</abbr> was founded in
1948.</p>

<p>Marking up abbreviations can give
useful information to browsers,
translation systems and search-engines.
</p>

</body>
</html>
```

The WHO was founded in 1948.

Marking u World Health Organization useful information to browsers, translation systems and search-engines.

Figure 27 *– The <abbr> element and the mouse over function (**Source:** Author)*

HTML <address> for Contact Information

The HTML <address> tag defines the contact information for the author/owner of a document or an article. It can be an email address, URL, physical address, phone number, etc. The text comprised in the <address> element is typically presented in italic, and browsers will always add a line break before and after it, as follows:

```
<!DOCTYPE html>
<html>
<body>

<p>Please contact us:.</p>

<address>
https://code4sp.eu/<br>
https://www.facebook.com/Code4SP<br>
</address>

</body>
</html>
```

Please contact us:.

https://code4sp.eu/
https://www.facebook.com/Code4SP

***Figure 28** – The <address> element (**Source:** Author)*

HTML <cite> for Work Title

The HTML <cite> tag defines the title of a book, a poem, a song, a movie, a painting, and all creative works. It should be stated that the author's name is not the title of a work.

As in the aforementioned tags, the text in the <cite> element normally renders in italic:

```
<!DOCTYPE html>
<html>
<body>

<img src="img_the_scream.jpg"
width="220" height="277" alt="The
Scream">
<p><cite>The Scream</cite> by Edvard
Munch, 1893.</p>

</body>
</html>
```

The Scream by Edvard Munch, 1893.

Figure 29 *– The <cite> element (**Source:** https://www.w3schools.com/html/html_quotation_elements.asp)*

HTML <bdo> for Bi-Directional Override

HTML <bdo> tag stands for "bidirectional override" which is used to override the current/default text direction. This tag sets the direction of content within it to execute on browser from left to right or right to left (*rtl* – right to left; *ltf* – left to right).

```
<!DOCTYPE html>
<html>
<body>

<p>If your browser supports bi-
directional override (bdo), the next
line will be written from right to left
(rtl):</p>

<bdo dir="rtl">Code4SP</bdo>

</body>
</html>
```

If your browser supports bi-directional override (bdo), the next line will be written from right to left (rtl):

PS4edoC

Figure 30 *– The <bdo> element (**Source:** https://www.w3schools.com/html/html_quotation_elements.asp)*

HTML Comments

Add Comments

This element is used to add a comment to an HTML document. An HTML comment begins with <!— and closes with —>. HTML comments are visible to anyone that views the page source code but are not rendered when the HTML document is rendered by a browser. It should be noted that there is an exclamation point in the start tag, but not in the end tag. This feature is especially useful for placing notifications and reminders in the HTML code:

```
<!DOCTYPE html>
<html>
<body>

<!-- This is a comment -->
<p>Code4SP project.</p>
<!-- Comments are not displayed in the
browser -->

</body>
</html>
```

Code4SP project.

Figure 31 *– The Comments feature (**Source:** Author)*

Hide Content

Comments can also be used to hide content, and that can be helpful if one hides it for the moment. You can also hide more than one line, everything between the <!-- and the --> will be hidden from the display. Comments are also great for debugging HTML, because one can comment out HTML lines of code, one at a time, to search for errors.

```
<!DOCTYPE html>
<html>
<body>

<p>Code4SP project.</p>

<!-- <p>This content is hidden. </p> --
>

<p>But this will appear.</p>

</body>
</html>
```

Code4SP project.

But this will appear.

Figure 32 *– The hide content feature (**Source:** Author)*

HTML Colours

HTML colours are stipulated with predefined colour names, or with RGB, HEX, HSL, RGBA, or HSLA values.

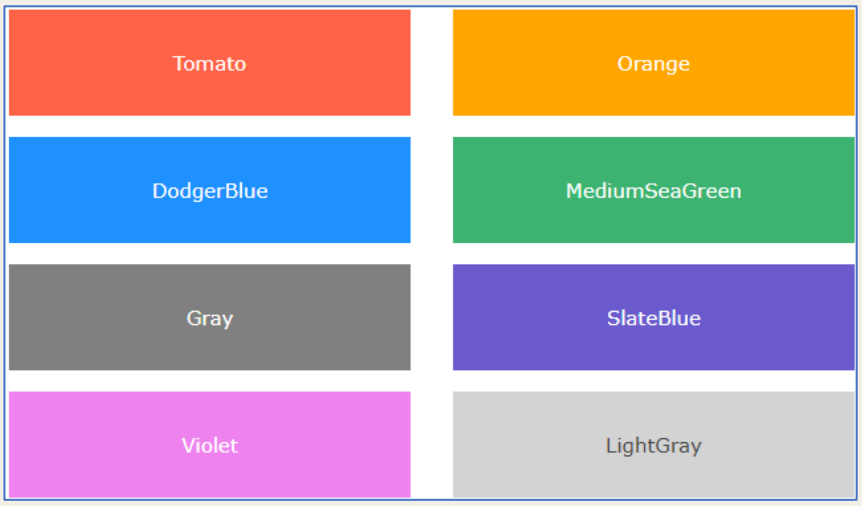


Figure 33 *– Some colour names one can define (**Source:** https://www.w3schools.com/html/html_colors.asp)*

Colours can also be set for the **page background**:

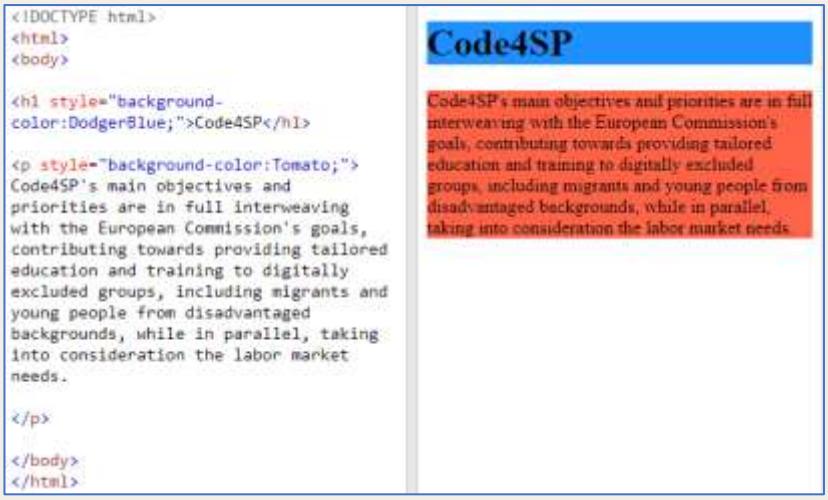


Figure 34 *– Defining a webpage's background colour (**Source:**Author)*

The same principle can be applied to the **text colour**:

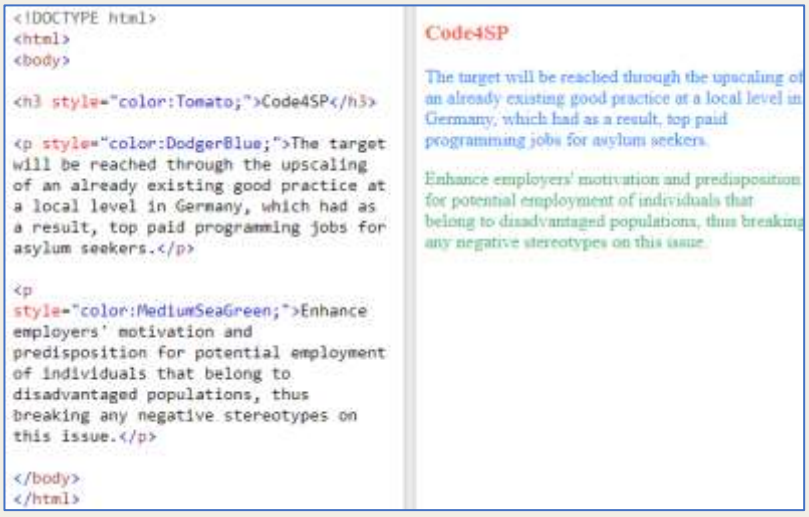


Figure 35 *– Defining the text colour (**Source:**Author)*

Also, for the colour of **borders**:

```
<!DOCTYPE html>
<html>
<body>

<h1 style="border: 2px solid
Tomato;">Code4SP</h1>

<h1 style="border: 2px solid
DodgerBlue;">Code4SP</h1>

<h1 style="border: 2px solid
Violet;">Code4SP</h1>

</body>
</html>
```

Code4SP

Code4SP

Code4SP

Figure 36 *– Adding borders and defining their colour (**Source:**Author)*

Colour Values

As stated above, HTML colours are stipulated with predefined colour names, or with RGB, HEX, HSL, RGBA, or HSLA values. The following three elements have their background color set with RGB, HEX, and HSL values:

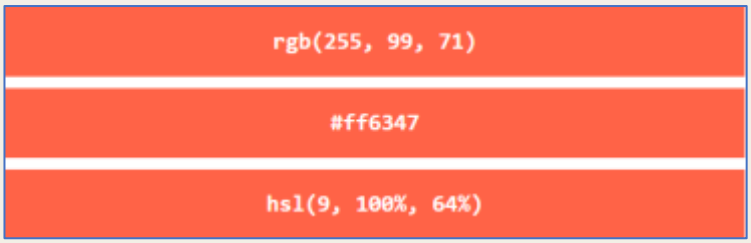


Figure 37 *– RGB, HEX, and HSL values for the Tomato colour (**Source:** https://www.w3schools.com/html/html_colors.asp)*

Transparency is also a feature that could be added when defining a colour, by adding an Alpha channel to it. The following example as 50% transparency:

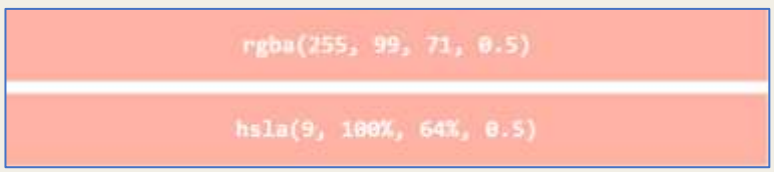


Figure 38 *– Setting transparency values for the Tomato colour (**Source:** https://www.w3schools.com/html/html_colors.asp)*

The code for setting up both features will ease learners' understanding. As seen, it comprises CSS features to be explored later:

```
<!DOCTYPE html>
<html>
<body>

<p>Same as color name "Tomato":</p>

<h1 style="background-color:rgb(255,
99, 71);">rgb(255, 99, 71)</h1>
<h1 style="background-
color:#ff6347;">#ff6347</h1>
<h1 style="background-color:hsl(9,
100%, 64%);">hsl(9, 100%, 64%)</h1>

<p>Same as color name "Tomato", but 50%
transparent:</p>
<h1 style="background-color:rgba(255,
99, 71, 0.5);">rgba(255, 99, 71, 0.5)
</h1>
<h1 style="background-color:hsla(9,
100%, 64%, 0.5);">hsla(9, 100%, 64%,
0.5)</h1>

<p>In addition to the predefined color
names, colors can be specified using
RGB, HEX, HSL, or even transparent
colors using RGBA or HSLA color values.
</p>

</body>
</html>
```

Same as color name "Tomato":

rgb(255, 99, 71)

#ff6347

hsl(9, 100%, 64%)

Same as color name "Tomato", but 50% transparent:

rgba(255, 99, 71, 0.5)

hsla(9, 100%, 64%, 0.5)

In addition to the predefined color names, colors can be specified using RGB, HEX, HSL, or even transparent colors using RGBA or HSLA color values.

***Figure 39** – Setting colour and transparency of the Tomato colour (**Source:** https://www.w3schools.com/html/html_colors.asp)*

More information concerning RGB, HEX, HSL, RGBA, or HSLA values can be found at https://www.w3schools.com/html/html_colors.asp

HTML Links

Links can be found in nearly all webpages. They allow internet users to navigate from page to page. It does not have to be text, as it can be an image or any other HTML element.

Hyperlinks

HTML links are hyperlinks, which can be clicked, jumping to another document.
When the mouse is moved over a link, the mouse arrow will turn into a little hand.

Syntax

The HTML <a> tag defines a hyperlink. It has the following syntax:

```
<a href="url">link text</a>
```

The most significant attribute of the <a> element is the href attribute, which indicates the link's destination. The link text is the part that will be visible to the user. By clicking on the link text, the reader will be redirected to the required URL address.

By default, links will be seen in all browsers as follows:

- An unvisited link is underlined and blue
- A visited link is underlined and purple
- An active link is underlined and red

Note: Link colours can be changed by using CSS features.

The target attribute

By default, the linked page will be shown in the current browser window. To modify this, learners must indicate another target for the link.

The target attribute indicates where to open the linked document. It can have one of the following values:

- _self - Default. Opens the document in the same window/tab as it was clicked
- _blank - Opens the document in a new window or tab
- _parent - Opens the document in the parent frame
- _top - Opens the document in the full body of the window

Using an image as a Link

To use an image as a link, just put the <img> tag inside the <a> tag, as it can be checked over the following tutorial: https://www.w3schools.com/html/tryit.asp?filename=tryhtml_links_image

Linking an email address

To create a link that opens the user's email software (letting them send a new email), mailto: should be added inside the href attribute, following the next example:

.

```
<a href="mailto:someone@example.com">Send email</a>
```

Create a Bookmark in HTML

HTML links can be applied to make bookmarks, so that readers can jump to particular parts of a web page, and it can be truly useful if the web page is very long. This process is composed of two very simple steps:

- To create a bookmark - first the bookmark should be created, and then a link should be added to it. To create, the id attribute should be used (e.g.: `<h2 id="C1">Chapter 1</h2>`), then, a link to the bookmark, from within the same page, should be added (e.g.: `<a href="#C4">Jump to Chapter 4</a>`)

- When the link is clicked, the page will scroll to the location with the bookmark.

HTML Images

Inserting images into webpages

Images improve visual look of the web pages by making them more appealing and colourful. The <img> tag is used to add images in the HTML pages. It is an empty element and contains attributes only.

Each image must have at least two attributes: the src and alt attributes. The src attribute informs the browser where to find the image, being its value the URL of the image file. The alt attribute provides an alternative text for the image if it is inaccessible or cannot be displayed for some reason (slow connection, image is not available at the specified URL, or if the user uses a screen reader or non-graphical browser). Its value should be a meaningful substitute for the image, preferably a suggestive text.

Images improve visual look of the web pages by making them more appealing and colourful. The <img> tag is used to add images in the HTML pages. It is an empty element and contains attributes only.

Each image must have at least two attributes: the src and alt attributes. The src attribute informs the browser where to find the image, being its value the URL of the image file. The alt attribute provides an alternative text for the image if it is inaccessible or cannot be displayed for some reason (slow connection, image is not available at the specified URL, or if the user uses a screen reader or non-graphical browser). Its value should be a meaningful substitute for the image, preferably a suggestive text.

Figure 40 *– Adding an image to a HTML document, using the src and alt attributes (**Source:**Author)*

Setting the Width and Height of an Image

The width and height attributes are used to indicate the width and height of an image. The values of these attributes are interpreted in pixels by default. It's a good practice to specify both the width and height attributes, so that browser can assign that much of space for the image before it is transferred.

Figure 41 *– Setting the height and width of an image (**Source:***
https://www.tutorialrepublic.com/codelab.php?topic=html&file=specify-dimensions-for-images)

The style attribute can also be used to indicate width and height. It prevents style sheets from changing the image size by accident, because inline style has the highest priority.

Using the HTML5 Picture Element

Now and then, scaling an image up or down to fit different devices (or screen sizes) does not work as expected. In addition, reducing the image dimension using the width and height attribute does not decrease the initial file size. In order to solve these problems, HTML5 has introduced the <picture> tag that allows to define multiple versions of an image to target different types of devices.

The <picture> element contains zero or more <source> elements, each referring to different image source, and one <img> element at the end. Likewise, each <source> element has the media attribute which specifies a media condition that is used by the browser to determine when a particular source should be used.

Image Maps

An image map allows one to define hotspots on an image that acts out just like a hyperlink. The key idea behind creating an image map is to give an simple way of linking various parts of an image without dividing it into separate image files. For example, a map of a country may have each city hyperlinked to more information about that city.

The following example is pretty accurate about these features:
https://www.tutorialrepublic.com/codelab.php?topic=html&file=image-maps

The name attribute of the <map> tag is used to reference the map from the <img> tag using its usemap attribute. The <area> tag is used inside the <map> element to define the clickable areas on an image. Any number of clickable areas can be defined within an image.

HTML Favicon

A favicon is a small image displayed to the left of the page title in the browser tab:

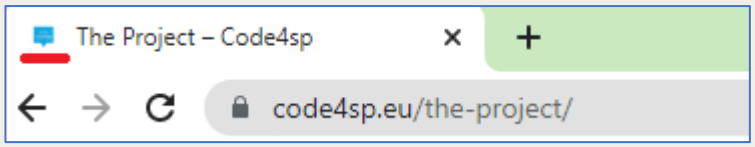


***Figure 42** – Code4SP's favoticon (**Source:** Author)*

To add a favicon to a website, a favicon image should be saved to the root directory of webserver. Other way is by creating a folder in the root directory called images, then saving the favicon image in this folder. A common name for a favicon image is "favicon.ico".

Next, a <link> element should be added to the "index.html" file, after the <title> element, as follows:

```
<!DOCTYPE html>
<html>
<head>
  <title>Code4SP</title>
  <link rel="icon" type="image/x-icon" href="/images/favicon.ico">
</head>
<body>
<h1>Our project</h1>
<p>Co-funded by the E+ fund of the EC.</p>
</body>
</html>
```

HTML Tables

Creating Tables in HTML

HTML tables allow to arrange data into rows and columns. They are generally used to display tabular data like product listings, customer's details, financial reports, etc.

A table can be created using the <table> element. Inside the <table> element, the <tr> elements can be utilized to create rows, and to create columns inside a row the <td> elements can be used. A cell can be defined as a header for a group of table cells using the <th> element.

Tables do not have any borders by default. The CSS border property can be used to add borders to the tables. Furthermore, table cells are sized just large enough to fit the contents by default. To add more space around the content in the table cells, the CSS padding property can be used.

By default, borders around the table and their cells are separated from each other. But they can be collapsed into one by using the border-collapse property on the <table> element. Additionally, text inside the <th> elements is displayed in bold font, aligned horizontally center in the cell by default. To change the default alignment, the CSS text-align property can be used. To do so, it is suggested that learners reach the CSS topic first. Most of the <table> element's attribute such as border, cellpadding, cellspacing, width, align, etc. for styling table appearances in earlier versions has been dropped in HTML5, so they should be avoided. **CSS should be privileged to style HTML tables.**

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Creating Tables in HTML</title>
</head>
<body>
    <h2>Spotify Top Songs of 2021 (USA)</h2>
    <table>
        <tr>
            <th>No.</th>
            <th>Song - Band </th>
            <th>Lenght</th>
        </tr>
        <tr>
            <td>1</td>
            <td>drivers licence - Olivia Rodrigo</td>
            <td>4:02</td>
        </tr>
        <tr>
            <td>2</td>
            <td>MONTERO (Call me by your name) - Lil Nas X</td>
            <td>2:17</td>
        </tr>
        <tr>
            <td>3</td>
            <td>STAY - The Kid LAROI ft. Justin Bieber</td>
            <td>2:21</td>
        </tr>
    </table>
</body>
</html>
```

Spotify Top Songs of 2021 (USA)

No.	Song - Band	Lenght
1	drivers licence - Olivia Rodrigo	4:02
2	MONTERO (Call me by your name) - Lil Nas X	2:17
3	STAY - The Kid LAROI ft. Justin Bieber	2:21

Figure 43 *– A basic table (**Source:** Author)*

Spanning Multiple Rows and Columns

Spanning allows to extend table rows and columns across multiple other rows and columns. Usually, a table cell cannot pass over into the space below or above another table cell. However, the rowspan or colspan attributes can be used to span multiple rows or columns in a table.

Likewise, the rowspan attribute can be used to create a cell that spans more than one row, as follows:

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Span Multiple Rows in an HTML Table</title>
    <style>
        table {
            width: 300px;
            border-collapse: collapse;
        }
        table, th, td {
            border: 1px solid black;
        }
        th, td {
            padding: 10px;
        }
    </style>
</head>
<body>
    <h2>Spanning Rows</h2>
    <table>
        <tr>
            <th>Name:</th>
            <td>John Carter</td>
        </tr>
        <tr>
            <th rowspan="2">Phone:</th>
            <td>55577854</td>
        </tr>
        <tr>
            <td>55577855</td>
        </tr>
    </table>
</body>
</html
```

Spanning Rows

Name:	John Carter
Phone:	55577854
	55577855

Figure 44 *– The rowspan attribute (**Source:** https://www.tutorialrepublic.com/html-tutorial/html-tables.php)*

Table captions

A caption (or title) for tables can be created by using the <caption> element. This element should be placed directly after the (opening) <table> tag. By default, caption appears at the top of the table, but this can be changed by using the CSS caption-side property.

```html
<!DOCTYPE html>
<html>
   <body>
      <table border=1>
       <caption> WIKITECHY WEBSITE </caption>
       <tr>
           <th>Firstname</th>
           <th>Lastname</th>
       </tr>
       <tr>
           <td>Wiki</td>
           <td>techy</td>
       </tr>
      </table>
   </body>
</html>
```

WIKITECHY WEBSITE

Firstname	Lastname
Wiki	techy

Figure 45 *– Adding table captions (**Source:** https://www.wikitechy.com)*

Defining a Table Header, Body, and Footer

HTML provides a series of tags <thead>, <tbody>, and <tfoot> that aid learners to create more coordinated tables, by defining header, body and footer regions, in that order.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>HTML Table with a Header, Footer and Body</title>
    <style>
        table {
            width: 300px;
            border-collapse: collapse;
        }
        table, th, td {
            border: 1px solid black;
        }
        th, td {
            padding: 10px;
            text-align: left;
        }
    </style>
</head>
<body>
    <table>
        <thead>
            <tr>
                <th>Items</th>
                <th>Expenditure</th>
            </tr>
        </thead>
        <tbody>
            <tr>
                <td>Stationary</td>
                <td>2,000</td>
            </tr>
            <tr>
                <td>Furniture</td>
                <td>10,000</td>
            </tr>
        </tbody>
        <tfoot>
            <tr>
                <th>Total</th>
                <td>12,000</td>
            </tr>
        </tfoot>
    </table>
</body>
</html
```

Items	Expenditure
Stationary	2,000
Furniture	10,000
Total	12,000

Figure 45 *– Adding table captions (**Source:** https://www.tutorialrepublic.com/html-tutorial/html-tables.php)*

HTML Lists

HTML lists are applied to present information in a well-formed and semantic way. Inside of them, one can add text, images, links, line breaks, etc. There are three different types of lists in HTML and each one has a specific purpose and meaning:

- **A) Unordered lists** — Used to create a list of related items, in no particular order.
- **B) Ordered lists** — Used to create a list of related items, in a certain order.
- **C) Description lists** — Used to create a list of terms and their descriptions.

A) Unordered lists

An unordered list created using the <ul> element, and each list item begins with the <li> element. It is marked with bullets, as follows:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Reasons why you should travel by train</title>
</head>
<body>
    <h2>Reasons why you should travel by train</h2>
    <ul>
        <li>It is less expensive</li>
        <li>It is eco-friendly</li>
        <li>You can enjoy the view</li>
    </ul>
</body>
</html>
```

Reasons why you should travel by train

- It is less expensive
- It is eco-friendly
- You can enjoy the view

Figure 46 *– Unordered Lists (**Source:** Author)*

B) Ordered lists

An ordered list is created by using the <ol> element, and each list item starts with the <li> element. Ordered lists are used when the order of the list's items is critical. It is marked with numbers, as follows:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>HTML Ordered List</title>
</head>
<body>
    <h2>How to cook your own veggie burger</h2>
    <ol>
        <li>Dump your ground meat into a bowl. (We go for ground meat with
around 20% fat.) Season it with salt, pepper, and whatever else you want; you can
add spices, perhaps, or Worcestershire sauce, or shallots, or chiles.</li>
        <li>Shape your burgers into patties, using your thumb to make an
indentation in the center; this will keep the burgers from puffing up. Keep in
mind that the burgers will shrink up a bit once you cook them, so make your
patties a bit bigger than you want them later.</li>
        <li>Oil your grill or a cast-iron pan, and grill or sear those patties.
(How many times to flip them is up for debate -- but when I'm grilling, I flip
once so I can get get those nice grill marks.) Cook them until your desired
doneness (around 125-130°F for medium rare, around 1 minute per side for each
inch of thickness). But before you take them off the grill...</li>
        <li>...add your cheese and toast your buns. Let the cheese melt while the
burgers are still on the grill; to speed things up, you can close the cover.
</li>
<li>Once your burgers are finished cooking, and your cheese is melty and your
buns are nicely charred, throw some condiments and toppings on those burgers.
Anything goes. (Really, anything goes.) Bite into it and let those juices run
down your chin, and rejoice that it's summer. And then make another round,
because now you know how.</li>
    </ol>
</body>
</html>
```

How to cook your own veggie burger

1. Dump your ground meat into a bowl. (We go for ground meat with around 20% fat.) Season it with salt, pepper, and whatever else you want; you can add spices, perhaps, or Worcestershire sauce, or shallots, or chiles.
2. Shape your burgers into patties, using your thumb to make an indentation in the center; this will keep the burgers from puffing up. Keep in mind that the burgers will shrink up a bit once you cook them, so make your patties a bit bigger than you want them later.
3. Oil your grill or a cast-iron pan, and grill or sear those patties. (How many times to flip them is up for debate -- but when I'm grilling, I flip once so I can get get those nice grill marks.) Cook them until your desired doneness (around 125-130°F for medium rare, around 1 minute per side for each inch of thickness). But before you take them off the grill...
4. ...add your cheese and toast your buns. Let the cheese melt while the burgers are still on the grill; to speed things up, you can close the cover.
5. Once your burgers are finished cooking, and your cheese is melty and your buns are nicely charred, throw some condiments and toppings on those burgers. Anything goes. (Really, anything goes.) Bite into it and let those juices run down your chin, and rejoice that it's summer. And then make another round, because now you know how.

Figure 47 *– Ordered Lists (**Source:** Author)*

C) Description Lists

A description list is a list of items with a description or definition of each item.

The description list is created using <dl> element. The <dl> element is used in conjunction with the <dt> element which specify a term, and the <dd> element which specify the term's definition.

Browsers usually render the definition lists by placing the terms and definitions in separate lines, where the term's definitions are slightly indented.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>HTML Description or Definition List</title>
</head>
<body>
    <h2>What are bread and coffee?</h2>
    <dl>
        <dt>Bread</dt>
        <dd>A baked food made of flour.</dd>
        <dt>Coffee</dt>
        <dd>A drink made from roasted coffee beans.</dd>
    </dl>
</body>
</html>
```

What are bread and coffee?

Bread
: A baked food made of flour.

Coffee
: A drink made from roasted coffee beans.

Figure 48 *– Description Lists (**Source:** Author)*

HTML Forms

HTML Forms are expected to collect different types of user inputs, such as contact details (name, email, phone number, bank account, etc.). Forms include special elements known as controls like input box, checkboxes, radio-buttons, submit buttons, etc. To this end, users fill in a form with this data, either via text or box selection, submitting it later to a webserver for processing this data.

The <form> tag is used to generate an HTML form. A simple example of a login form would be as follows:

Figure 49 *– Example of a login form (**Source:** https://www.tutorialrepublic.com/html-tutorial/html-forms.php)*

There are different types of controls to be applied in a HTML form, being **input elements** the most frequently used ones. They identify various types of user input fields, depending on the type attribute. Input elements can be **text fields, password fields, checkboxes, submit buttons, reset buttons, file select boxes**, as well as several new input types introduced in HTML5 (they can be checked out here).

Text fields are areas that let users add text. These are created by using an <input> element, whose type attribute has a value of text. It should be noted that the <label> tag is used to identify the labels for <input> elements. If the webmaster wants his/her users to enter several lines, <textarea> should be added instead.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>HTML Text Input Field</title>
</head>
<body>
    <form>
        <label for="user-name">Login:</label>
        <input type="text" name="username" id="user-name">
    </form>
</body>
</html>
```

Login:

Figure 50 *– Text field example (**Source:** Author)*

Password fields are like text fields, being the only difference characters in a password field are hidden, so they are displayed as asterisks or dots. Likewise, this procedure prevents someone else from reading the password on the screen. This is as well a single-line text input control generated using an <input> element whose type attribute has a value of password.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>HTML Password Input Field</title>
</head>
<body>
    <form>
        <label for="user-pwd">Password:</label>
        <input type="password" name="user-password" id="user-pwd">
    </form>
</body>
</html>
```

Password:

Figure 51 *– Password field example (**Source:** https://www.tutorialrepublic.com/html-tutorial/html-forms.php)*

Radio buttons are applied to allow the user select exactly one option from a pre-specified set of options. It is generated using an <input> element whose type attribute has a value of radio.

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <title>HTML Radio Buttons</title>
</head>
<body>
    <form>
        <input type="radio" name="Civil Status" value="single" id="single">
        <label for="single">Single</label>
        <input type="radio" name="Civil Status" value="married" id="married">
        <label for="married">Married</label>
      <input type="radio" name="Civil Status" value="other" id="other">
        <label for="other">Other</label>
    </form>
</body>
</html>
```

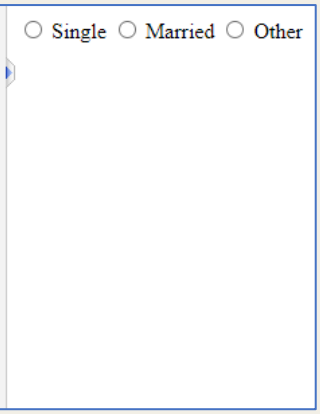


Figure 52 *– Code for setting up radio buttons (**Source:** Author)*

Checkboxes make available to the user one or more choices from a pre-defined set of options. It is generated using an <input> element whose type attribute has a value of checkbox. If one prefers to generate a checkbox (or radio button) selected by default, one simply must add the attribute checked to the input element (`<input type="checkbox" checked>`).

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <title>HTML Checkboxes</title>
</head>
<body>
    <form>
        <input type="checkbox" name="sports" value="football" id="football">
        <label for="football">Football</label>
        <input type="checkbox" name="sports" value="handball" id="handball">
        <label for="handball">Handball</label>
        <input type="checkbox" name="sports" value="basketball" id="basketball">
        <label for="basketball">Basketball</label>
    </form>
</body>
</html>
```

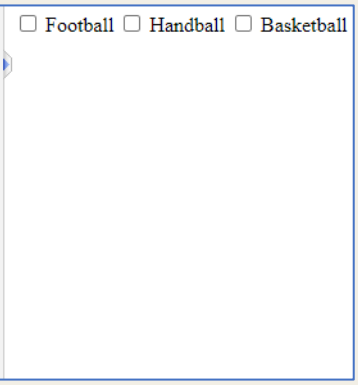


Figure 53 *– Code for setting up checkboxes (**Source:** Author)*

File select boxes allow a user to browse for a local file and send it as an attachment with the form data. E.g., Google Chrome provides a file select input field with a 'Browse' button that permits the user to select a file from his/her hard drive.

File select boxes are also created using an <input> element, whose type attribute value is set to file.

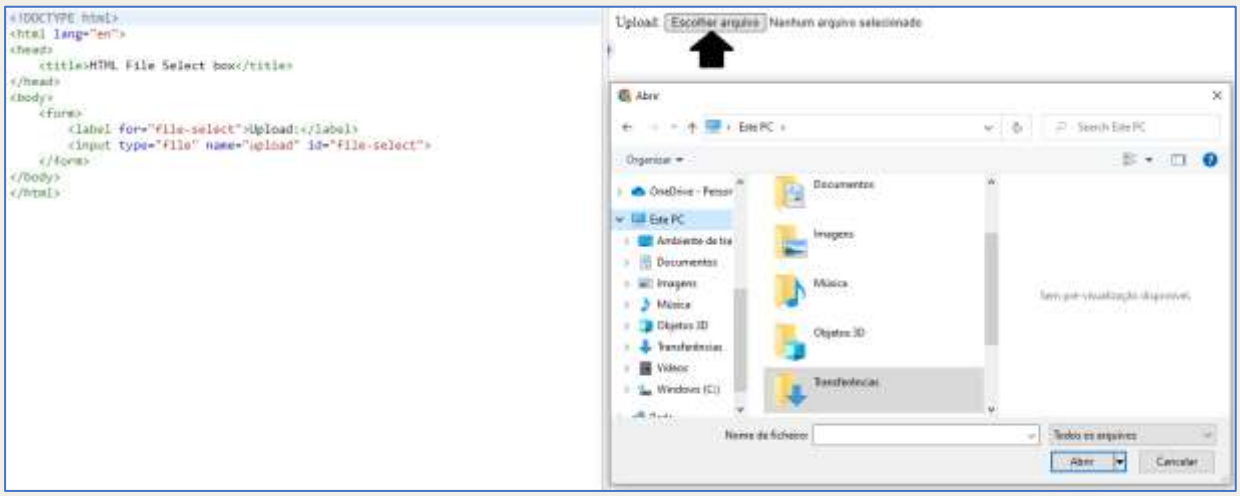


Figure 54 *– Code for setting up file select boxes (**Source:** Author)*

Textarea can be defined as a multiple-line text input control that allows to enter more than one line of text. These controls are created using an <textarea> element, as follows:

Figure 55 *– Code for setting up a multiple-line text input control (**Source:** Author)*

Select boxes are dropdown lists of options in which a user can select one or more options from a pull-down menu. They are created by using the <select> element and <option> element. The <option> elements within the <select> element define each list item.

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <title>HTML Select Box</title>
</head>
<body>
    <form>
        <label for="country">Country:</label>
        <select name="country" id="country">
            <option value="Portugal">Portugal</option>
            <option value="Cyprus">Cyprus</option>
            <option value="Greece">Greece</option>
        </select>
    </form>
</body>
</html>
```

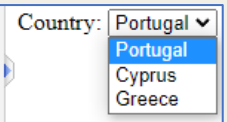


Figure 56 *– Code for setting up select boxes (**Source:** Author)*

Submit and reset buttons are very common in most of the websites. Submit buttons are used to send (form) data to a web server, while reset buttons are created to reset the form to default values. When the user clicks the submit button, the form data is sent to the file specified in the form's action attribute to process the submitted data. Buttons can also be created using the <button> element. They have the same purpose as buttons created with the input element. However, they offer more rendering possibilities, as they allow the embedding of other HTML elements.

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <title>HTML Submit and Reset Buttons</title>
</head>
<body>
    <form action="/examples/html/action.php" method="post">
        <label for="first-name">First Name:</label>
        <input type="text" name="first-name" id="first-name">
        <input type="submit" value="Submit">
        <input type="reset" value="Reset">
    </form>
</body>
</html>
```

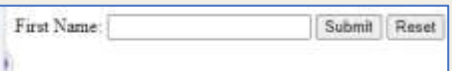


Figure 57 *– Code for setting up submit and reset buttons (**Source:** Author)*

Grouping form controls are a great tool for users to locate a control, making the form more accessible. The <legend> element is key for creating logically related controls, as seen in the picture below:

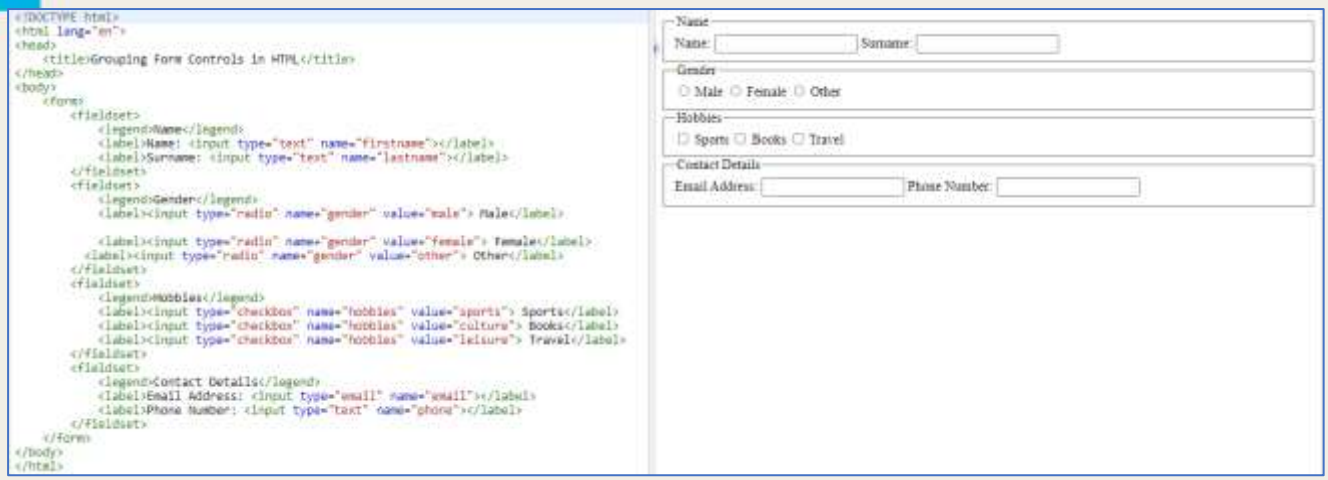


Figure 58 *– Code for setting up grouping form controls (**Source:** Author, check this for better definition)*

Below there is a list of frequently used form element's attributes:

Attribute	Description
`name`	Specifies the name of the form.
`action`	Specifies the URL of the program or script on the web server that will be used for processing the information submitted via form.
`method`	Specifies the HTTP method used for sending the data to the web server by the browser. The value can be either get (the default) and post.
`target`	Specifies where to display the response that is received after submitting the form. Possible values are _blank, _self, _parent and _top.
`enctype`	Specifies how the form data should be encoded when submitting the form to the server. Applicable only when the value of the method attribute is post.

Table 2 *– List of frequently used form element's attributes (**Source:** https://www.tutorialrepublic.com/html-tutorial/html-forms.php)*

More information regarding other attributes can be found here.

HTML iFrame

An iframe (*or inline frame*) is used to exhibit external objects within a web page, including other web pages. An iframe pretty much performs like a mini web browser within a web browser. Likewise, the content inside an iframe occurs separate from the adjacent elements.

The basic syntax for adding this feature to a web page is as follows:

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

The URL specified in the src attribute indicates the location of an external object or a web page.

Figure 59 *– Code for setting up an inline frame (**Source:** Author)*

The **height** and **width** of the iframe can be defined by applying the code disposed on *Figure 60* below. The width and height attribute values are stipulated in pixels by default, but users can also set these values in percentage, such as 50%, 100%, etc.. The default width of an iframe is 300 pixels, while the default height is 150 pixels.

Figure 60 *– Code for setting up height and width of an inline frame (**Source:** Author)*

As it could be verified, iframe has a border around it set by default. To modify or remove it, the best way is to use the CSS border feature (to be taught in the CSS topic).

An iframe can also be used as a target for the hyperlinks. It can be named using the name attribute. This means that when a link with a target attribute (with that name set as value) is clicked, the linked source will open in the same inline frame, as follows:

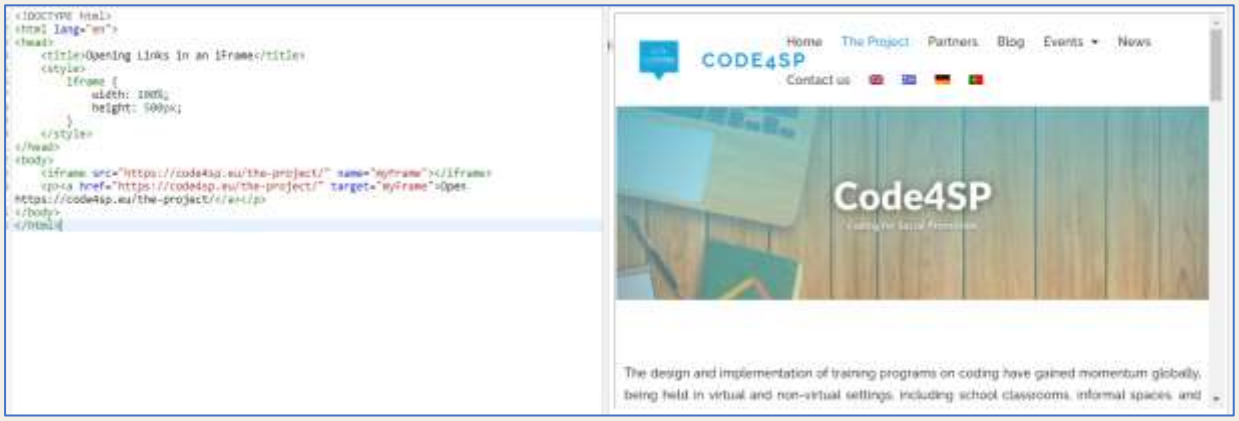


Figure 61 *– Using iframe as a target for a hyperlink (**Source:** Author)*

2.2. HTML advanced concepts

HTML Doctypes

A Document Type Declaration (DOCTYPE) is an instruction to the web browser concerning the version of markup language in which a web page is created. It appears at the top of a web page before all other elements. According to the HTML specification, every HTML document requires a valid document type declaration to ensure that web pages are displayed the way they are meant to be. The doctype declaration is frequently the first thing defined in an HTML document (even before the opening <html> tag); nevertheless, the doctype declaration itself is not an HTML tag.

The DOCTYPE for HTML5 is very short, concise, and case-insensitive: `<!DOCTYPE html>.`

The following markup can be used as a template to create a new HTML5 document:

```
<!DOCTYPE html> <html lang="en"> <head> <meta charset="utf-8">
<title><!-- Insert your title here --></title> </head> <body> <!--
Insert your content here --> </body> </html>
```

HTML Layouts

There are different methods of creating a web page layout, positioning the various elements that compose a web page in a well-structured manner and giving an engaging appearance to the website. Most websites usually display their content in multiple rows and columns, configured like a magazine to provide the users with a better reading and writing atmosphere. This can be easily attained by utilizing the HTML tags, such as <table>, , <header>, <footer>, <section>, etc. and combining some CSS styles to them.

The simplest way for creating layouts in HTML is indeed obtained by designing tables. As seen in the previous sections, this usually involves the process of putting contents (text, images, etc.) into rows and columns.

The layout below contains an HTML table with three rows and two columns. It should be noted that the first and last row spans both uses the colspan attribute. It should be stated that the method used for creating layout in this example, although it is not wrong, it is not recommended. Tables and inline styles for creating layouts should be avoided. Layouts created using tables often rendered very slowly. **Tables should only be used to display tabular data.** For creating such layouts, **CSS float techniques** are advised. Users shall learn about this tool moreover.

***Figure 62** – HTML basic layout (**Source:** Author, adapted from Tutorial Republic. Click here for better resolution)*

HTML5 has established new structural elements, for instance <header>, <footer>, <nav>, <section>, etc. to identify the different parts of a web page in a more semantic way. This example applies the new HTML5 structural elements to create the same layout created in *Figure 62*.

To know more about newly introduced tags, learners should visit this source.

HTML Head

The head element is the receptacle for all head elements, which provide extra information about the document or reference to other resources, required for the document to perform properly. It illustrates the properties of the document such as title, deliver meta information like character set, tell the browser where to find the style sheets or scripts that allows to expand the HTML document interactively.

The HTML elements that can be used inside the <head> element are: <title>, <base>, <link>, <style>, <meta>, <script> and <noscript>.

The HTML title Element

The <title> element identifies the title of the document and only one is required in all HTML/XHTML documents to produce a valid document. It must be placed within the <head> element. The title element comprises plain text and entities and it may not include other markup tags. It may be used for different functions:

- To show a title in the browser title bar and in the task bar;
- To deliver a title for the page when it is added to favourites or bookmarked;
- To present a title for the page in search-engine results (e.g., Google search).

It should be **short** and **specific** to the content of the document, as search engines would pay particular attention to the words present in the title – it should have ideally 65 characters.

A title in an HTML document should be displayed as follows:

```
<!DOCTYPE html> <html lang="en"> <head> <title>A simple HTML
document</title> </head> <body> <p>Hello World! </p> </body> </html>
```

The HTML base Element

The HTML <base> element is used to identify a base URL for all relative links contained in the document. For example, one can set the base URL once at the top of the page, and then all subsequent relative links will use that URL as a starting point, as follows:

```
<!DOCTYPE html> <html lang="en"> <head> <title>Defining a base URL</title> <base href=" https://code4sp.eu/the-project/ "> </head> <body> <p><a href=" https://code4sp.eu/the-project/ ">HTML Head</a>. </p> </body> </html>
```

Figure 63 *– HTML base element (**Source:** Author)*

The HTML <base> element must be found before any element that belongs to an external resource. HTML permits only one base element for each document.

The HTML link Element

The <link> element describes the correlation between the current document and an external document or resource. A common use of link element is to connect to external style sheets.

Figure 63 *– HTML base element (**Source:** https://www.tutorialrepublic.com/codelab.php?topic=html&file=linking-style-sheet)*

It should be stated that an HTML document's <head> element may include any number of <link> elements. The <link> element has attributes, but no contents.

The HTML style Element

The <style> element is applied to describe embedded style information for an HTML document. The style rules inside the <style> element indicate how HTML elements should render in a browser. An embedded style sheet should be used when a single document has a unique style. If the same style sheet is used in various documents, then an external style sheet would be more suitable.

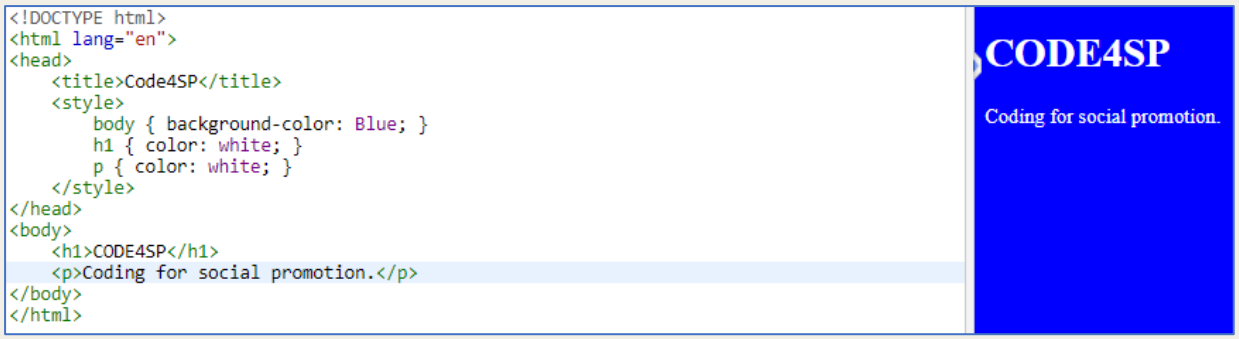


Figure 64 *– Coding various style elements (**Source:** Author)*

The HTML meta Element

The <meta> element provides metadata about the HTML document, which is a set of data that describes and gives information about other data. <meta> tags always appear inside the <head> element, and are typically used to specify character set, page description, keywords, author of the document, and viewport settings.

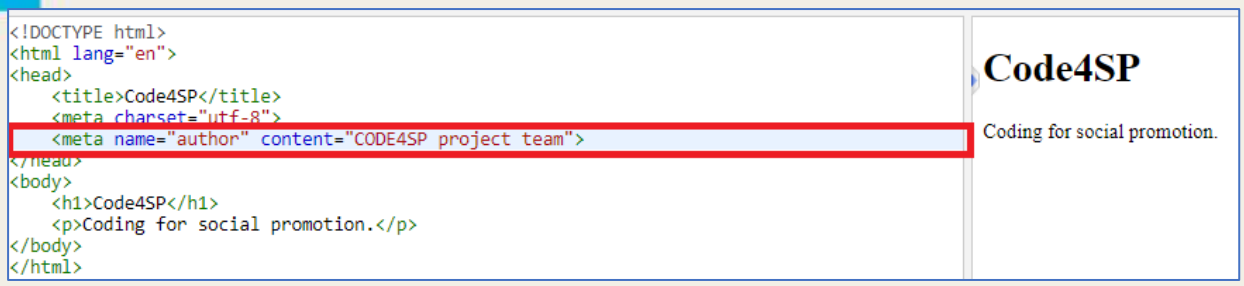


Figure 64 *– The meta element (**Source:** Author)*

As seen above, meta tags include information about a web page. It is not visible in the browser (it is machine parsable though). Metadata is utilised by browsers (how to display content or reload page), search engines (keywords), and other web services. Moreover, there is a technique to let web designers rule over the **viewport**, through the <meta> tag. The viewport is the user's visible area of a web page. It differs from device to device (it will be bigger on a computer screen than on a mobile phone).

The following <meta> element should be included in all web pages, as it will give the browser guidelines on how to assume the page dimensions and scaling: `<meta name="viewport" content="width=device-width, initial-scale=1.0">`
The `width=device-width` part sets the width of the page to follow the screen-width of the device (which will vary depending on the screen). The `initial-scale=1.0` part sets the initial zoom level when the page is initially loaded by the browser.

The difference between webpages with or without viewport meta tag is quite visible in the following example:

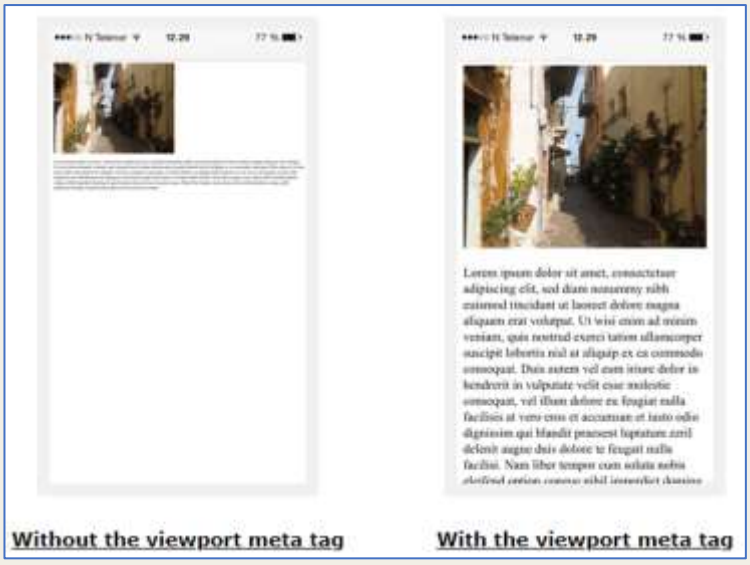


Figure 65 *– With or without viewport meta tags examples (**Source:** https://www.w3schools.com/tags/tag_meta.asp)*

The HTML script Element

The <script> element is used to define client-side script, such as JavaScript in HTML documents.

Figure 65 *– The script element (**Source:** Author)*

Working with Client-side Script

Client-side scripting is linked to the type of computer programs that are performed by the user's web browser. **JavaScript (JS)** is the most widespread client-side scripting language on the web. The <script> element is used to embed or reference JavaScript within an HTML document to add interactivity to web pages and to create a more user-friendly experience. Some of the most common uses of JavaScript are form validation,

generating alert messages, creating image gallery, show hide content, DOM manipulation, etc.

JavaScript can be **embedded** following two ways:

1. Directly inside the HTML page; or
2. Placed in an external script file and referenced inside the HTML page.

Note: both techniques use the <script> element.

To embed JS in an HTML file, the user must add the code as the content of the <script> element, following the example of *Figure 66*. Preferably, script elements should be placed at the end of the page, before the closing body tag (</body>), because when browser encounters a script, it pauses rendering the rest of the page until it deconstructs the script that may drastically impact the website performance.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>Embedding JavaScript</title>
</head>
<body>
    <div id="greet"></div>
    <script>
        document.getElementById("greet").innerHTML = "Code4SP";
    </script>
</body>
</html>
```

Figure 66 *– Embedding JS in an HTML doc (**Source:** Author)*

JavaScript codes can also be placed into a separate file, calling up a .js extension, while corresponding it in the HTML document by using the src attribute of the <script> tag. This can be especially useful if the web designer wants to make the same script available for multiple documents, so he/she does not need to perform the same tasks repeatedly. When the src attribute is indicated, the <script> element should be empty,

so the web designer cannot use the same <script> element to both embed the JavaScript and to link to an external JavaScript file in an HTML document.

If a browser, for some reason, does not support client-side scripting, or users have disabled JS in their browser, the <no script> element can be used to provide an alternative content. This element can include any HTML elements, except <script>, as it can be included in the <body> element of a HTML page.

HTML Entities

Most learners will certainly be curious about how to display special characters and symbols in their programming processes. Hence, this subchapter is intended to explain how they can be successful in doing such thing.

First, it is important to understand what a HTML entity is. As perceived in the previous chapters, some characters are quite reserved in HTML. For instance, one cannot use signs such as '<' or '>', as the browser could mistake them for a markup. Moreover, some characters are just not available in the keyboard (e.g., the copyright symbol).

These special characters can be displayed by simply being replaced with the character entities (or just entities), then solving the aforementioned troubles. Below is a list of the most frequently used entities in HTML:

Result	Description	Entity Name	Numerical reference
	non-breaking space		
<	less than	&lt;	&#60;
>	greater than	&gt;	&#62;

&	ampersand	&amp;	&#38;
"	quotation mark	&quot;	&#34;
'	apostrophe	&apos;	&#39;
¢	cent	&cent;	&#162;
£	pound	&pound;	&#163;
¥	yen	&yen;	&#165;
€	euro	&euro;	&#8364;
©	copyright	&copy;	&#169;
®	registered trademark	&reg;	&#174;
™	trademark	&trade;	&#8482;

***Table 3** – The most frequent entities in HTML (**Source:** https://www.tutorialrepublic.com/html-tutorial/html-entities.php)*

Numeric character references can also be used as an alternative for entity names, especially because they have stronger browser support, and can be used to specify any Unicode character, however entities are limited to a subgroup of this.

HTML URL

URL: how many times has this abbreviation appeared in virtual environments? It stands for Uniform Resource Locator, and it works as the global address of documents and other resources on the World Wide Web. Its goal is to identify the location of a document and other resources available online, while specifying the mechanism for accessing it using a web browser. For instance, https://code4sp.eu/ is an URL.

The general syntax of URLs can be described as follows: `scheme://host:port/path?query-string#fragment-id`

URLs have a linear structure and are composed of the following components:

- **Scheme name** — The scheme recognizes the protocol to be used to access a resource on the Internet. The scheme names are followed by the three characters :// (a colon and two slashes). The most used protocols are http://, https://, ftp://, and mailto://.
- **Host name** — identifies the host where the resource is situated. A hostname is a domain name given to a host computer. This is usually a combination of the host's local name with its parent domain's name. For example, www.code4sp.eu/ consists of host's machine name www and the domain name code4sp.eu.
- **Port Number** — Servers often deliver more than one type of service, so they must be told what service is being requested. These requests are made by port number. Well-known port numbers for a service are normally omitted from the URL. For example, web service HTTP runs by default over port 80, HTTPS runs by default over port 443.
- **Path** — identifies the specific resource within the host that the user wants to access. For example, /html/html-url.php, /news/technology/, etc.
- **Query String** — contains data to be passed to server-side scripts, running on the web server. For instance, parameters for a search. The query string preceded by a question mark (?), is usually a string of name and value pairs separated by ampersand (&), for example, ?first_name=John&last_name=Corner, q=mobile+phone, and so on.
- **Fragment identifier** —specifies a location within the page. Browser may scroll to display that part of the page. The fragment identifier introduced by a hash character (#) is the optional last part of a URL for a document.

HTML URL Encoding

It is well-known that sometimes data is not safely transmitted over the internet. This happens mostly because URLs are not fully or accurately encoded and causes some misunderstandings among internet users.

According to RFC 3986, the characters in a URL only restricted to a defined set of reserved and unreserved US-ASCII characters. Any other characters are not allowed in a URL (that is why some Latin and Cyrillic characters are not seen in URL). But URL often comprises characters outside the US-ASCII character set, so they must be converted to a valid US-ASCII format for worldwide interoperability. URL-encoding is a process of converting URL information so that it can be safely transmitted over the internet.

To map the large range of characters used globally, a two-step method is followed:

- Firstly, the data is encoded in relation to the UTF-8 character encoding.
- Then only those bytes that do not correspond to characters in the unreserved set should be percent-encoded like %HH, where HH is the hexadecimal value of the byte.

For instance, look at this Portuguese popular saying:

“Quem vê caras, não vê corações.” [“Faces we see, hearts we do not know.”]

This sentence would be encoded as:

Quem%20v%C3%AA%20caras%2C%20n%C3%A3o%20v%C3%AA%20cora%C3%A7%C3%B5es.

Ç, ç (c-cedilla) is a Latin script letter, as well as well as the accents used (~ and ^).

Certain characters are restricted from use in a URL, as they may (or may not) be defined as delimiters by the generic syntax in a particular URL scheme (for instance, forward slash / characters are used to separate different parts of a URL).

Reserved characters in a URL are as follows:

!	#	$	&	'	(	)	*	+	,	/	:	;	=	?	@	[	]
%21	%23	%24	%26	%27	%28	%29	%2A	%2B	%2C	%2F	%3A	%3B	%3D	%3F	%40	%5B	%5D

***Figure 67** – Reserved characters in a URL (**Source:** https://www.tutorialrepublic.com/html-tutorial/html-url-encode.php)*

However, there are also characters that, despite of being allowed in a URL, do not have a reserved purpose – that is why they are called “unreserved characters”.
These include uppercase and lowercase letters, decimal digits, hyphen, period, underscore, and tilde. The following table lists all the unreserved characters in a URL:

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
0	1	2	3	4	5	6	7	8	9	-	_	.	~												

***Figure 68** – Unreserved characters in a URL (**Source:** https://www.tutorialrepublic.com/html-tutorial/html-url-encode.php)*

For encoding/decoding characters, users may use this converter.

HTML Validation

To make sure a HTML code follows the current web standards, free from errors, it is of upmost importance to figure out how to validate a HTML code. Beginners often will make mistakes when writing HTML codes, and incorrect or non-standard codes will certainly cause unexpected results in how a web page is displayed in a browser.

In order to prevent this to happen, users can test their codes within the formal guidelines and standards set by the Wide Web Consortium (W3C) for HTML/XHTML web pages. There is an online tool that automatically checks HTML codes and points out any problems/errors, like missing closing tags or missing quotes around attributes.

The process of validating a web page is ensuring the respect for the norms/standards defined by the W3C, so it is very important. Some reasons for validating a web page are:

- It helps to create web pages that are cross-browser, cross-platform compatible. Most likely they will be compatible with the future version of web browsers and web standards.
- Standards compliant web pages increase the search engine spiders and crawlers visibility, as a result your web pages will more likely be appear in search results.
- It will reduce unexpected errors and make your web pages more accessible to the visitor.

HTML5 Features

In this subchapter, the features of the fifth (and last) major HTML version recommended by the W3C will be explained.

HTML5 New Input Types

HTML5 introduces several new <input> types like email, date, time, color, range, and so on, with the goal of improving the user experience and to make the forms more interactive. Nevertheless, if a browser failed to recognize these new input types, it will consider them a normal text box.

The following are some new input types:

- color
- date
- datetime-local
- email
- month
- number
- range
- search
- tel
- time
- url
- week

Regarding the color input, it allows to select a colour from a colour picker and gives information about the colour value in hexadecimal format (e.g., #000000, which is black, the default colour if the user does not specify a value), as verified in *Figure 69* below.

It should be noted that the color input is supported in all major modern web browsers (Firefox, Chrome, Opera, Safari (12.1+), Edge (14+)), but it is not supported by the Microsoft Internet Explorer and older versions of Apple Safari browsers.

Figure 69 *– Picking a colour using the color input (**Source:** https://www.tutorialrepublic.com/codelab.php?topic=html5&file=color-input-type)*

Concerning the date input type, it allows the user to select a date from a drop-down calendar, in which he/she may choose the year, month and day (but not time). This feature is also supported by most browsers, except for Internet Explorer and Safari.

Figure 70 *– The date input type (**Source:** https://www.tutorialrepublic.com/codelab.php?topic=html5&file=color-input-type)*

The datetime-local input type makes available to the user to select both local date and time, including the year, month, and day including the time in hours and minutes. This input is supported by Safari, Firefox and IE, but not by Chrome, Edge and Opera.

Figure 71 *– The datetime-local input type (**Source:** https://www.tutorialrepublic.com/codelab.php?topic=html5&file=color-input-type)*

To allow the user to enter his/her e-mail address, email is the preferred input type. It is quite like a standard text input type, but if it is applied in combination with the required attribute, browsers may search for the patterns to guarantee a properly formatted e-mail address should be entered. The email input field can be styled for different validation states, when a value is entered using the :valid, :invalid or :required pseudo-classes. This input type is supported by all major browsers.

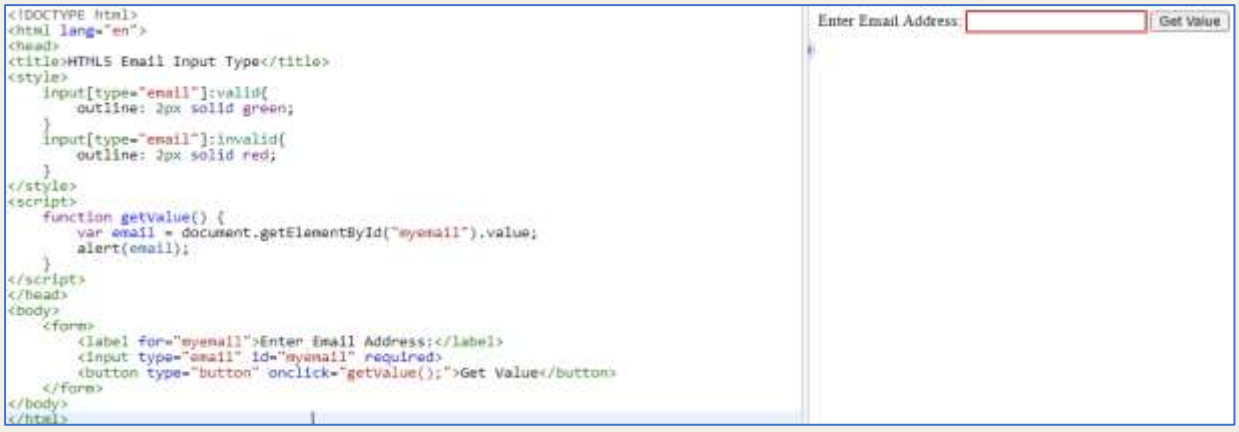


Figure 72 *– The email input type (**Source:** https://www.tutorialrepublic.com/html-tutorial/html5-new-input-types.php)*

In what regards month Input type, this is a very similar feature to the previous time ones since it allows to select a month and year from a drop-down calendar (being 'YYYY' the year and ''MM'' the month. It should be noted that this is not supported by Firefox, Safari and Internet Explorer. Only Chrome, Edge and Opera browsers support it.

Figure 73 *– The month input type (**Source:** https://www.tutorialrepublic.com/html-tutorial/html5-new-input-types.php)*

Concerning the number input type, it is used for inserting a numerical value. The web designer can also limit the user to enter only acceptable values. For this to happen, the additional attributes min, max, and step should be used. This feature is supported by all major web browsers.

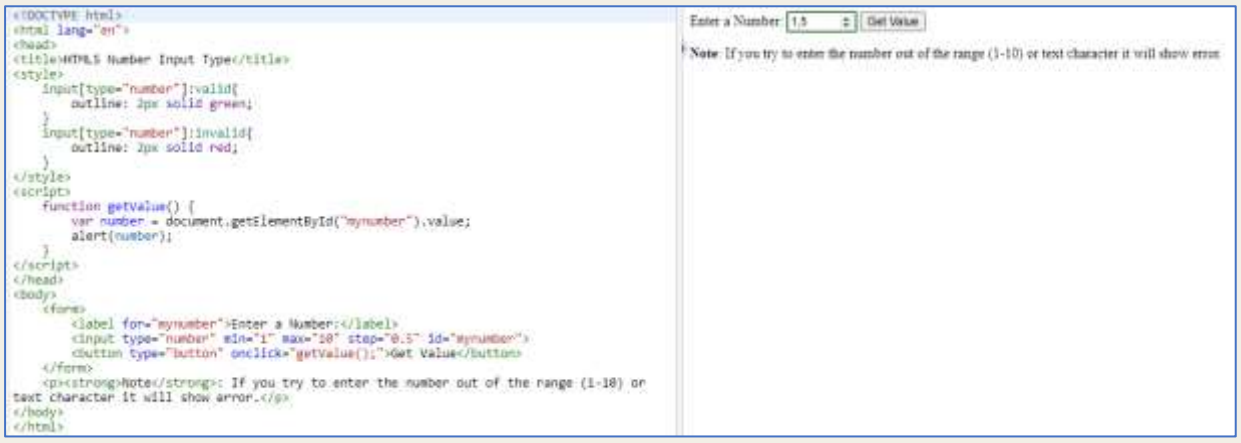
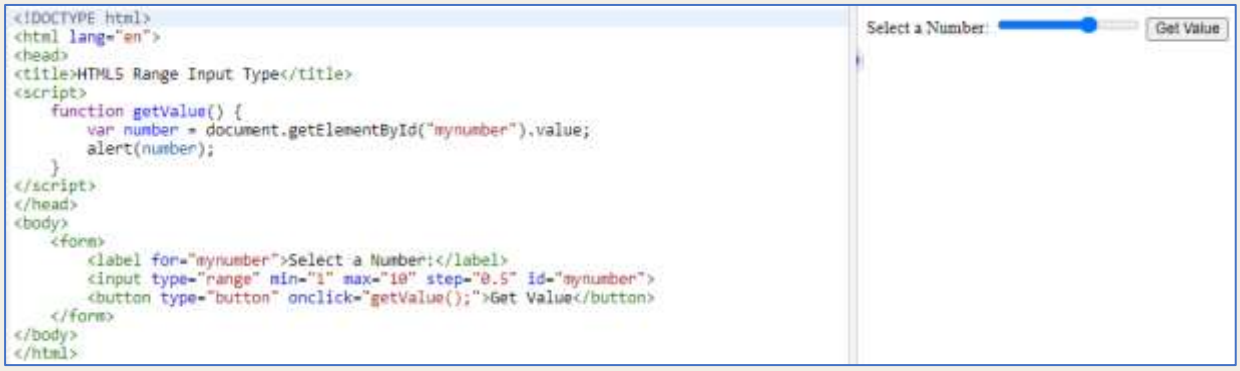


Figure 74 *– The number input type (**Source:** https://www.tutorialrepublic.com/html-tutorial/html5-new-input-types.php)*

The range input type can be applied for registering a numerical value within a specific range. It works very similar to number input above, although it presents an easier way for inserting a number. This input type is supported by all major web browsers.




```
<!DOCTYPE html>
<html lang="en">
<head>
<title>HTML5 Range Input Type</title>
<script>
    function getValue() {
        var number = document.getElementById("mynumber").value;
        alert(number);
    }
</script>
</head>
<body>
    <form>
        <label for="mynumber">Select a Number:</label>
        <input type="range" min="1" max="10" step="0.5" id="mynumber">
        <button type="button" onclick="getValue();">Get Value</button>
    </form>
</body>
</html>
```

Figure 75 *– The range input type (**Source:** https://www.tutorialrepublic.com/html-tutorial/html5-new-input-types.php)*

The search input type is suitable for generating search input fields. It must be stated that, in some browsers (i.e., Chrome and Safari), as soon as the user begins to type in the search box, a tiny cross emerges on the right side of the field, which can be useful for clearing the entire search field. It is supported by all major browsers.


```
<!DOCTYPE html>
<html lang="en">
<head>
<title>HTML5 Search Input Type</title>
<script>
    function getValue() {
        var term = document.getElementById("mysearch").value;
        alert(term);
    }
</script>
</head>
<body>
    <form>
        <label for="mysearch">Search Website:</label>
        <input type="search" id="mysearch" placeholder="Type something...">
        <button type="button" onclick="getValue();">Get Value</button>
    </form>
</body>
</html>
```

Figure 76 *– The search input type (**Source:** https://www.tutorialrepublic.com/html-tutorial/html5-new-input-types.php)*

Concerning the tel input type, it is especially useful for inserting a phone number. As browsers do not support tel input validation by default, the placeholder attribute can be

used to help users inserting the correct format for their phone number or indicate a regular expression to validate the user's input by applying the pattern attribute. This feature is not supported by any browser, as phone numbers vary a lot worldwide.

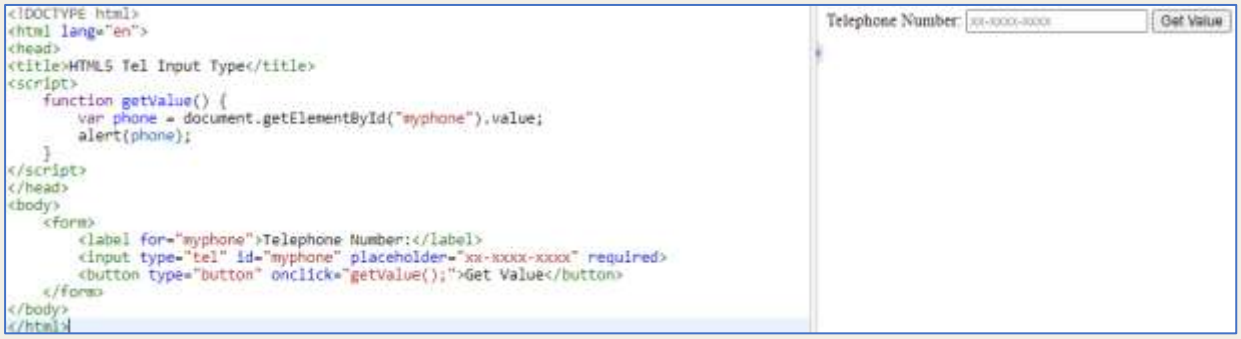


Figure 77 *– The tel input type (**Source:** https://www.tutorialrepublic.com/html-tutorial/html5-new-input-types.php)*

Regarding the time input type, it can be used for entering any given time (hours and minutes), and the browser can use both hour formats (12 or 24-hour) for inserting times, depending on the region. This input is not supported by IE and Safari browsers.

Figure 78 *– The time input type (**Source:** https://www.tutorialrepublic.com/html-tutorial/html5-new-input-types.php)*

Concerning the url input type, it can be used for inserting URL's or web addresses. The multiple attribute can be used for inserting more than one URL. Moreover, if required attribute is restricted, the browser will automatically perform validation to ensure that

only text that meets the standard format for URLs goes into the input box. All major browsers support this input type.

Figure 79 *– The url input type (**Source:** https://www.tutorialrepublic.com/html-tutorial/html5-new-input-types.php)*

Finally, the week input type enables the user to choose a week and year from a drop-down calendar. This feature is not supported by Firefox, Safari and IE, but it is currently supported by Chrome, Edge and Opera browsers.

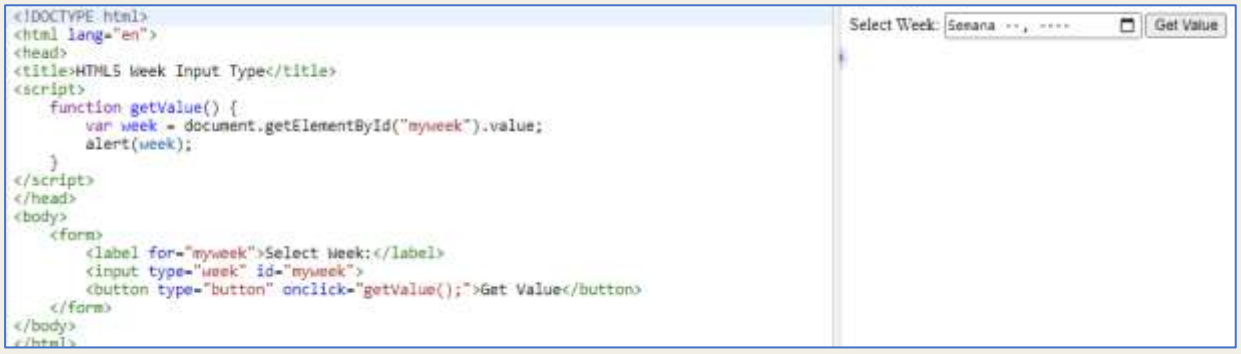


Figure 80 *– The week input type (**Source:** https://www.tutorialrepublic.com/html-tutorial/html5-new-input-types.php)*

HTML5 Canvas

This subchapter will be essentially useful for learning how to draw graphics using the HTML5 canvas element. It was originally created by Apple for the Mac OS dashboard widgets and to enable graphics in Safari. Moreover, it was adopted by other browsers, like Firefox, Google Chrome and Opera.

By default, the <canvas> element has 300px of width and 150px of height without any border and content. Nevertheless, custom width and height can be specified using the CSS height and width feature.
The canvas is a two-dimensional four-sided area. The coordinates of the top-left corner of the canvas are (0, 0), identified as origin, and the coordinates of the bottom-right corner are (*canvas width, canvas height*), as can be seen using the interactive tool available here.

To draw basic paths and shapes utilizing the recently introduced HTML5 canvas element and JavaScript, it will be useful to take a look at several templates.

Firstly, the base template for drawing paths and shapes onto the 2D HTML5 canvas:

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Embedding Canvas Into HTML Pages</title>
<style>
    canvas {
        border: 1px solid #000;
    }
</style>
<script>
    window.onload = function() {
        var canvas = document.getElementById("myCanvas");
        var context = canvas.getContext("2d");
        // draw stuff here
    };
</script>
</head>
<body>
    <canvas id="myCanvas" width="300" height="200"></canvas>
</body>
</html>
```

Figure 81 *– The base template for drawing paths and shapes onto the 2D HTML5 canvas (**Source:** https://www.tutorialrepublic.com/html-tutorial/html5-new-input-types.php)*

All the lines except those from 7 to 11 are quite straight forward and easy to interpret. The anonymous function affixed to the window.onload event will execute when the page loads. As soon as the page is loaded, one can access the canvas element with document.getElementById() method. Later, a 2D canvas context is defined by passing 2d into the getContext() method of the canvas object.

The initial step to draw on canvas is a **straight line**. The most important procedures used for this purpose are moveTo(), lineTo() and the stroke(). The moveTo() method identifies the position of drawing cursor onto the canvas, while the lineTo() method used to define the coordinates of the line's end point, and finally the stroke() method is used to make the line visible:

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Drawing a Line on the Canvas</title>
<style>
    canvas {
        border: 1px solid #000;
    }
</style>
<script>
    window.onload = function() {
        var canvas = document.getElementById("myCanvas");
        var context = canvas.getContext("2d");
        context.moveTo(50, 150);
        context.lineTo(250, 50);
        context.stroke();
    };
</script>
</head>
<body>
    <canvas id="myCanvas" width="300" height="200"></canvas>
</body>
</html>
```

Figure 82 *– The moveTo(), lineTo() and the stroke() procedures (**Source:** https://www.tutorialrepublic.com/html-tutorial/html5-new-input-types.php)*

What about **drawing an arc**? It can be obtained by simply using the arc() method, which syntax is:

```
context.arc(centerX, centerY, radius, startingAngle, endingAngle,
counterclockwise);
```

In the example above, an arc was drawn on canvas by inserting a JavaScript code:

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Drawing an Arc on the Canvas</title>
<style>
    canvas {
        border: 1px solid #000;
    }
</style>
<script>
    window.onload = function() {
        var canvas = document.getElementById("myCanvas");
        var context = canvas.getContext("2d");
        context.arc(150, 150, 80, 1.2 * Math.PI, 1.8 * Math.PI, false);
        context.stroke();
    };
</script>
</head>
<body>
    <canvas id="myCanvas" width="300" height="200"></canvas>
</body>
</html>
```

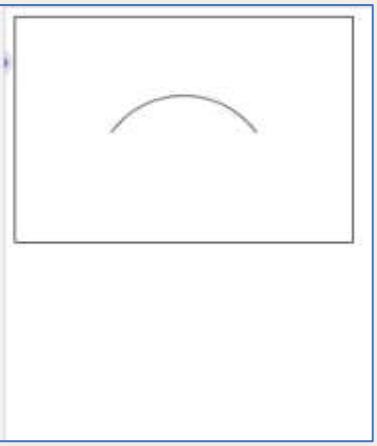


Figure 83 *– Drawing an arc using a JS code (**Source:** https://www.tutorialrepublic.com/html-tutorial/html5-new-input-types.php)*

To draw a **rectangle and square shapes**, the rect() method is the way to go. It entails four parameters: x, y positions of the rectangle and its width and height. The basic syntax of the rect() method is the following:

```
context.rect(x, y, width, height);
```

To draw it using a JS code:

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Drawing a Rectangle on the Canvas</title>
<style>
    canvas {
        border: 1px solid #000;
    }
</style>
<script>
    window.onload = function() {
        var canvas = document.getElementById("myCanvas");
        var context = canvas.getContext("2d");
        context.rect(50, 50, 200, 100);
        context.stroke();
    };
</script>
</head>
<body>
    <canvas id="myCanvas" width="300" height="200"></canvas>
</body>
</html>
```

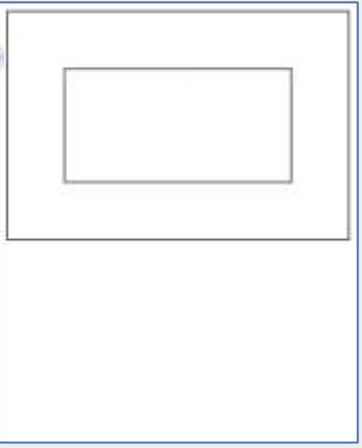


Figure 84 *– Drawing a rectangle using a JS code (**Source:** https://www.tutorialrepublic.com/html-tutorial/html5-new-input-types.php)*

In opposite to the rect() method, there is no specific procedure for drawing a circle. However, the result can be obtained by creating a fully enclosed arc, by using the arc() method. The synthax for drawing a complete circle using the arc() method is the following:

```
context.arc(centerX, centerY, radius, 0, 2 * Math.PI, false);
```

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Drawing a Circle on the Canvas</title>
<style>
    canvas {
        border: 1px solid #000;
    }
</style>
<script>
    window.onload = function() {
        var canvas = document.getElementById("myCanvas");
        var context = canvas.getContext("2d");
        context.arc(150, 100, 70, 0, 2 * Math.PI, false);
        context.stroke();
    };
</script>
</head>
<body>
    <canvas id="myCanvas" width="300" height="200"></canvas>
</body>
</html>
```

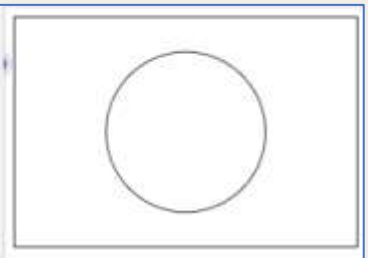


Figure 85 *– Drawing a circle on HTML5 canvas (**Source:** https://www.tutorialrepublic.com/html-tutorial/html5-new-input-types.php)*

In regard to **styles and colours on stroke**, the default colour of the stroke is black, being its thickness 1 pixel. However, these attributes can be changed using the strokeStyle and lineWidth properties, as follows on *Figure 86*.

In *Figure 87*, it is possible to set the cap style for the lines by using the lineCap property, with three styles available: butt, round, and square.

One can also fill colour inside the canvas shapes by using the fillStyle() approach. *Figure 88* shows how to fill up a solid colour within a rectangle shape. While designing the shapes on canvas, it is suggested to use the fill() method before the stroke() method to render the stroke appropriately.

```html
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Setting Stroke Color and Width on the Canvas</title>
<style>
    canvas {
        border: 1px solid #000;
    }
</style>
<script>
    window.onload = function() {
        var canvas = document.getElementById("myCanvas");
        var context = canvas.getContext("2d");
        context.lineWidth = 5;
        context.strokeStyle = "orange";
        context.moveTo(50, 150);
        context.lineTo(250, 50);
        context.stroke();
    };
</script>
</head>
<body>
    <canvas id="myCanvas" width="300" height="200"></canvas>
</body>
</html>
```

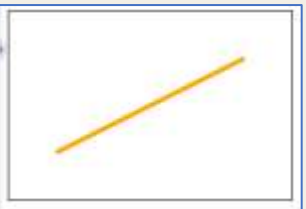


Figure 86 *– Setting the styles and colours on stroke using strokeStyle and lineWidth properties (**Source:** https://www.tutorialrepublic.com/html-tutorial/html5-new-input-types.php)*

```html
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Setting Stroke Cap Style on the Canvas</title>
<style>
    canvas {
        border: 1px solid #000;
    }
</style>
<script>
    window.onload = function() {
        var canvas = document.getElementById("myCanvas");
        var context = canvas.getContext("2d");
        context.lineWidth = 10;
        context.strokeStyle = "orange";
        context.lineCap = "round";
        context.arc(150, 150, 80, 1.2 * Math.PI, 1.8 * Math.PI, false);
        context.stroke();
    };
</script>
</head>
<body>
    <canvas id="myCanvas" width="300" height="200"></canvas>
</body>
</html>
```

Figure 87 *– Setting the cap style for the lines using the lineCap property (**Source:** https://www.tutorialrepublic.com/html-tutorial/html5-new-input-types.php)*

```html
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Filling Color inside a Rectangle on the Canvas</title>
<style>
    canvas {
        border: 1px solid #000;
    }
</style>
<script>
    window.onload = function() {
        var canvas = document.getElementById("myCanvas");
        var context = canvas.getContext("2d");
        context.rect(50, 50, 200, 100);
        context.fillStyle = "#f08080";
        context.fill();
        context.lineWidth = 5;
        context.strokeStyle = "black";
        context.stroke();
    };
</script>
</head>
<body>
    <canvas id="myCanvas" width="300" height="200"></canvas>
</body>
</html>
```

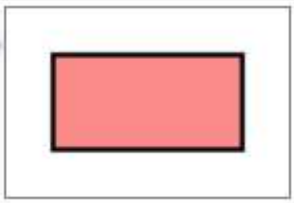


Figure 88 *– Filingl colour inside the canvas shapes by using the fillStyle() approach (**Source:** https://www.tutorialrepublic.com/html-tutorial/html5-new-input-types.php)*

It is also possible to fill gradient colour (a smooth visual transition from one colour to another) within the canvas shapes. There are two types of gradients here: linear and radial.

The basic syntax for creating a **linear gradient** is:

```
var grd = context.createLinearGradient(startX, startY, endX, endY);
```

The basic syntax for creating a **radial gradient** is:

```
var grd = context.createRadialGradient(startX, startY, startRadius,
endX, endY, endRadius);
```

Figure 89 shows how to fill a **linear gradient** colour inside a rectangle using the createLinearGradient() method:

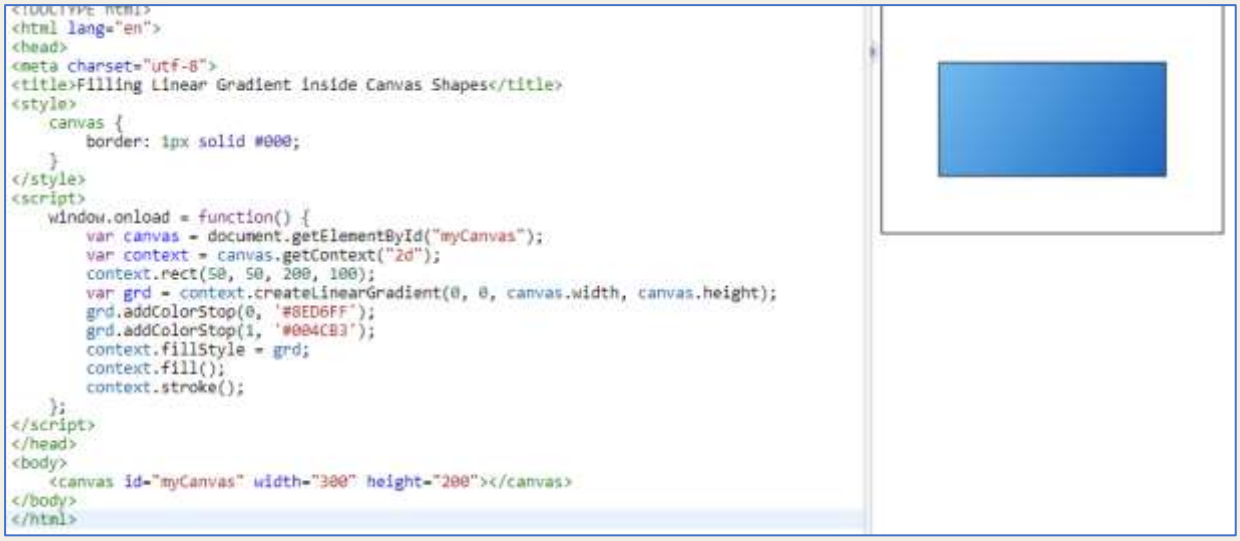
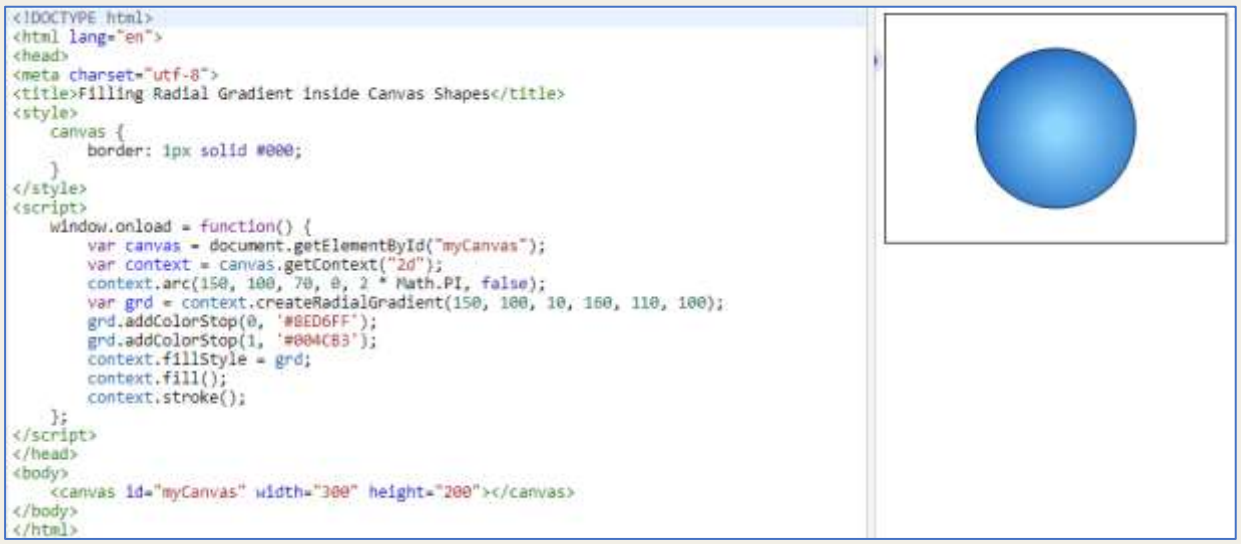


Figure 89 *– Filing a linear gradient colour inside a rectangle using the createLinearGradient() method (**Source:** https://www.tutorialrepublic.com/html-tutorial/html5-new-input-types.php)*

Figure 90 demonstrates how to fill a radial gradient colour inside a circle through the createRadialGradient() method:



```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Filling Radial Gradient inside Canvas Shapes</title>
<style>
    canvas {
        border: 1px solid #000;
    }
</style>
<script>
    window.onload = function() {
        var canvas = document.getElementById("myCanvas");
        var context = canvas.getContext("2d");
        context.arc(150, 100, 70, 0, 2 * Math.PI, false);
        var grd = context.createRadialGradient(150, 100, 10, 160, 110, 100);
        grd.addColorStop(0, '#8ED6FF');
        grd.addColorStop(1, '#004CB3');
        context.fillStyle = grd;
        context.fill();
        context.stroke();
    };
</script>
</head>
<body>
    <canvas id="myCanvas" width="300" height="200"></canvas>
</body>
</html>
```

Figure 90 *– Filling a radial gradient colour inside a circle through the createRadialGradient() method (**Source:** https://www.tutorialrepublic.com/html-tutorial/html5-new-input-types.php)*

It is also possible to **draw text** on canvas (containing Unicode characters only), as well as adding it **colour and alignment**, and even apply stroke on text using the strokeText() method, which will colour the perimeter of the text instead of filling it. Nonetheless, if the web designer intends to set both the fill and stroke on the text element, he/she can use the fillText() and the strokeText() methods together. it is suggested to use the fillText() method before the strokeText() method to render the stroke accurately.


```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Adding Stroke to Canvas Text</title>
<style>
    canvas {
        border: 1px solid #000;
    }
</style>
<script>
    window.onload = function() {
        var canvas = document.getElementById("myCanvas");
        var context = canvas.getContext("2d");
        context.font = "bold 32px Arial";
        context.textAlign = "center";
        context.textBaseline = "middle";
        context.strokeStyle = "blue";
        context.strokeText("Code4SP", 150, 100);
    };
</script>
</head>
<body>
    <canvas id="myCanvas" width="300" height="200"></canvas>
</body>
</html>
```

Figure 91 *– Drawing text on canvas (**Source:** https://www.tutorialrepublic.com/html-tutorial/html5-new-input-types.php)*

HTML5 SVG

This subchapter is expected to be clear on how to use HTML5 svg elements to draw vector graphics on a web page. To do so, firstly it is important to define **SVG**.

SVG stands for Scalable Vector Graphics, and it is an XML-based image format that is used to define two-dimensional vector-based graphics for the web. Distinct from raster image (e.g. .jpg, .gif, .png, and other two-dimensional formats), a vector image can be scaled up or down to any extent without losing the quality of the image. Vector images are comprised of a series of shapes defined by math, while raster images are composed of a fixed set of dots (pixels). Like other topics discussed along this chapter, SVG is also a W3C recommendation.

An SVG image is built by using a sequence of statements that go along with the XML schema, so, **SVG images** can be created and edited with a text editor like Notepad. There are numerous advantages of using SVG images rather than other image formats, as follows:

- They can be searched, indexed, scripted, and compressed.
- They can be created and modified using JavaScript in real time.

- They can be printed with high quality at any resolution.
- They can be animated using the built-in animation elements.
- They can contain hyperlinks to other documents.

SVG graphics can be embedded directly in a document by using the HTML5 <svg> element (see *Figure 92* below).

All the major modern web browsers (Chrome, Firefox, Safari, and Opera), as well as Internet Explorer 9 and above are compatible with inline SVG rendering.

Figure 92 *– Embedding SVG graphics directly by using the svg element (**Source:** https://www.tutorialrepublic.com/html-tutorial/html5-svg.php)*

Basic vector-based paths and shapes on the webpages can be drawn by utilizing the HTML5 <svg> element.

The most basic path to work with SVG is to **draw a straight line**. For that to happen, the SVG <line> element should be used. As can be seen in *Figure 93*, the attributes x1, x2, y1 and y2 of the <line> element draw a line from (x1,y1) to (x2,y2).

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Create a Line with HTML5 SVG</title>
<style>
    svg {
        border: 1px solid black;
    }
</style>
</head>
<body>
    <svg width="300" height="200">
        <line x1="50" y1="50" x2="250" y2="150" style="stroke:red; stroke-width:3;" />
    </svg>
</body>
</html>
```

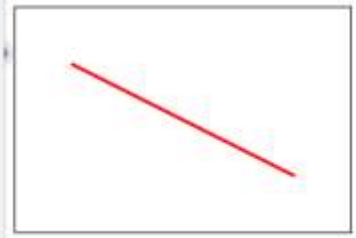


Figure 93 *– Drawing a straight line with SVG <line> element (**Source:** https://www.tutorialrepublic.com/html-tutorial/html5-svg.php)*

For drawing a **rectangle** and squares, the <rect> SVG element is the most appropriate way. The attributes x and y of <rect> element specify the co-ordinates of the top-left corner of the rectangle. The attributes width and height specify the width and height of the shape.

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Create a Rectangle with HTML5 SVG</title>
<style>
    svg {
        border: 1px solid black;
    }
</style>
</head>
<body>
    <svg width="300" height="200">
        <rect x="50" y="50" width="200" height="100" style="fill:orange; stroke:black;
stroke-width:3;" />
    </svg>
</body>
</html>
```

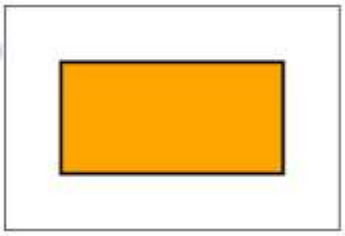


Figure 94 *– Drawing a rectangle with SVG <rect> element (**Source:** https://www.tutorialrepublic.com/html-tutorial/html5-svg.php)*

If a **circle** is the chosen shape, the SVG element <circle> is the most suitable. The attributes cx and cy of the <circle> element specifies the co-ordinates of the center of the circle and the attribute r identifies the radius of the circle. Still, if the attributes cx and cy are absent or not specified, the center of the circle is set to (0,0).

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Create a Circle with HTML5 SVG</title>
<style>
    svg {
        border: 1px solid black;
    }
</style>
</head>
<body>
    <svg width="300" height="200">
        <circle cx="150" cy="100" r="70" style="fill:red; stroke:black; stroke-width:3;"
/>
    </svg>
</body>
</html>
```

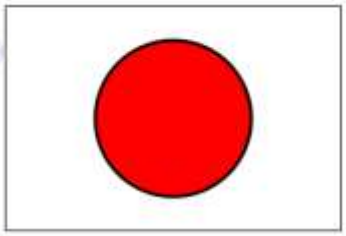


Figure 94 *– Drawing a circle with SVG <circle> element (**Source:** https://www.tutorialrepublic.com/html-tutorial/html5-svg.php)*

It is also possible to **draw text** with SVG. The text in SVG is rendered as a graphic so the web designer can use all the graphic transformation to it, but it still performs as text, so it can be selected and copied as text by the user. The attributes x and y of the <text> element identify the location of the top-left corner in absolute terms although the attributes dx and dy indicates the relative location.

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Render Text with HTML5 SVG</title>
<style>
    svg {
        border: 1px solid black;
    }
</style>
</head>
<body>
    <svg width="300" height="200">
        <text x="20" y="30" style="fill:purple; font-size:22px;">
            Welcome to Our Website!
        </text>
        <text x="20" y="30" dx="0" dy="20" style="fill:navy; font-size:14px;">
            Here you will find lots of useful information.
        </text>
    </svg>
</body>
</html>
```

Figure 95 *– Drawing text with SVG <text> element (**Source:** https://www.tutorialrepublic.com/html-tutorial/html5-svg.php)*

In alternative, web designers may use the <tspan> element to resize or relocate the span of text included within a <text> element. The text is included in separate tspans, but inside the same text element can all be selected at the same moment — when clicking and dragging to select the text. Yet, the text in separate text elements cannot be picked at the same time.

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Rotate and Render Text with HTML5 SVG</title>
<style>
    svg {
        border: 1px solid black;
    }
</style>
</head>
<body>
    <svg width="300" height="200">
        <text x="30" y="15" style="fill:purple; font-size:22px; transform:rotate(30deg);">
            <tspan style="fill:purple; font-size:22px;">
                Welcome to Our Website!
            </tspan>
            <tspan dx="-230" dy="20" style="fill:navy; font-size:14px;">
                Here you will find lots of useful information.
            </tspan>
        </text>
    </svg>
</body>
</html>
```

Figure 96 *– Drawing text with SVG <tspan> element (**Source:** https://www.tutorialrepublic.com/html-tutorial/html5-svg.php)*

Even though the new graphical elements <canvas> and <svg> have been introduced by HTML 5 in order to create high-quality graphics on the web, they differ quite a lot.

In the table below, the differences between both are summarized, and this will help learners on how to use them in an appropriate, effective way:

SVG	Canvas
Vector based (composed of shapes)	Raster based (composed of pixel)
Multiple graphical elements, which become the part of the page's DOM tree	Single element similar to <img> in behavior. Canvas diagram can be saved to PNG or JPG format
Modified through script and CSS	Modified through script only
Good text rendering capabilities	Poor text rendering capabilities
Give better performance with smaller number of objects or larger surface, or both	Give better performance with larger number of objects or smaller surface, or both

SVG	Canvas
Better scalability. Can be printed with high quality at any resolution. Pixelation does not occur	Poor scalability. Not suitable for printing on higher resolution. Pixelation may occur

Figure 96 *– The differences between SVG and canvas (**Source:** https://www.tutorialrepublic.com/html-tutorial/html5-svg.php)*

HTML 5 Audio

This subchapter is intended to explain how to embed audio in an HTML document.

As web browsers did not have a uniform standard for embedding media files as audio, it was not an easy task to perform in the past. However, there are many ways to embed sound in a webpage, from simply use a simple link to using the HTML5 <audio> element. This element provides a standard way to insert audio in web pages. Since the audio element is somewhat new, it runs in most of the modern web browsers.

There are many ways of inserting an audio into a HTML5 document. One of them is by using the browser's default group of controls, with one source defined by the src attribute, as it can be verified in this code, in which it is possible to hear a group of birds singing.

Another way can be achieved by using the <object> element, which is used for embedding different types of media files. This example embeds an audio file into a web page following the aforementioned method. It should be stated that the <object> element is not supported broadly and it depends on the type of the object that is being implanted. Other methods like HTML5 <audio> element or third-party HTML5 audio players could be a better option in a lot of cases.

Finally, the <embed> element can also be another way of inserting media in an HTML document, following this example. Even though the <embed> element is superbly supported in nowadays browsers and defined as standard in HTML5, the inserted audio might not be played due to absence of browser support for that file format or inaccessibility of plugins.

HTML5 Video

It is now time to learn how to embed video content into an HTML document.

Pretty much like sound, video contents were also difficult to insert into a web page, and for the same reason (web browsers did not have a uniform standard for defining embedded media files like video). In the next paragraphs, many ways of inserting these contents will be explained.

The newly introduced <video> element works in most of the modern browsers. This example explains how to simply insert a video into the HTML document, using the browser default set of controls, with one source defined by the src attribute.

The <object> element is also used to embed different types of media files. Following this example, one can understand how to embed a Flash video into a webpage (only browsers/applications supporting Flash will be able to play it). It should be noted that the <object> element is not supported extensively and depends a lot on the type of the object embedded. Other methods could be a better choice in many cases as, for instance, iPad and iPhone device cannot display Flash videos.

And what about **embedding YouTube videos**? That is actually the most simple and common way of embedding video files in webpages nowadays. The web designer must simply upload the video on YouTube and insert HTML code to display the video on his/her webpage. Here is a step-by-step mini guide:

Step 1 – Upload a video on YouTube.

Step 2 – After uploading a video on YouTube, the web designer should look for the 'Share' button, which is located below a video running on the platform's video player, just like as follows:

***Figure 96** – 'Share' button on YouTube (**Source:** Author)*

When clicking the 'Share' button, a share panel will open exhibiting some more options. Now, the '**Embed**' button should be clicked, as it will generate the HTML code to directly embed the video into the web page. For that to happen, the web designer should copy and paste that code into the HTML document.

***Figure 97** – 'Embed' option on YouTube (**Source:** Author)*

This code can be furtherly customized, and for that to happen the web designer just needs to select the customization option given just below the embed-code input box.

The insertion of a YouTube video on a webpage is explained in [this example].

HTML5 Web Storage

Ever wondered how to use HTML5 web storage feature to store data on user's browser? The following paragraphs will be useful for fully understanding this.

Firstly, it is important to understand the implications of 'web storage'.

With web storage, web applications can store data locally within the user's browser. Prior to HTML5, application data had to be stored in cookies, incorporated in every server request. Web storage is more secure, and substantial amounts of data can be stored locally, without impacting website performance. The information kept in the web storage is not sent to the web server as opposite to the cookies where data is delivered to the server with every request. Moreover, cookies only allow to store a small amount of data (nearly 4KB), while the web storage permits to store up to 5MB.

There are two types of web storage:

- **Local storage** — makes use of the localStorage object to store data for the entire website on a *permanent basis*. That being said, the stored local data will be available on the next day, the next week, or the next year unless it is removed.
- **Session storage** — it uses the sessionStorage object to store data on a *temporary basis*, for a single browser window or tab. The data vanishes when session ends, for instance when the user shuts that browser window or tab.

As regards the **local storage**, each piece of data is collected in a key/value pair. The key identifies the name of the information (i.e. 'first_name'), and the value is the value related to the same key (i.e. 'Peter'). The following JS code expresses the following:

- localStorage.setItem(key, value) stores the value associated with a key.
- localStorage.getItem(key) saves the value associated with the key.

It is also possible to remove a particular item from the storage, by passing the key name to the removeItem() method, i.e. localStorage.removeItem("first_name").

However, if the web designer intends to remove the complete storage, he/she should use the clear() method, i.e. localStorage.clear(). The clear() method simply clears all key/value pairs from localStorage at once, **so it must be used cautiously**. The web storage data will not be accessible between different browsers.

Finally, the sessionStorage object works similarly to localStorage, except that it stores the data only for one session. The following example is quite explanatory on how this works.

HTML5 Application Cache

During the subchapter, learners can get closer on how to create offline applications using **HTML5 caching feature**.

It is well-known that most web-based application will not work if the web designer is offline. However, HTML5 brings in an application cache mechanism that allows the browser to automatically save the HTML file and all the other resources that requires to display it properly on the local machine, that way the browser can still access the web page and its resources without establishing a connection to the internet. This is supported in all major modern web browsers (Firefox, Chrome, Opera, Safari, and Internet Explorer 10 and above.

There are several advantages in using this feature:

- **Offline browsing** — Visitors can use the application even when they are not online or there are unexpected interruptions in the network connection.

- **Improve performance** — Cached resources load directly from the user's machine instead of the remote server so web pages load quicker and perform better.
- **Reduce HTTP request and server load** — The browser will only have to download the updated/changed resources from the remote server that reduce the HTTP requests and saves valuable bandwidth as well as decrease the load on the web server.

There are a few steps to go through in order to cache the files for offline use:

STEP 1 – Create a Cache Manifest file. This is a special text file that informs browsers what files should they store (and not), and what files to replace. It always starts with the words CACHE MANIFEST (always in uppercase).

Figure 98 below is an example of a manifest file:

Example ⬇ Download

```
CACHE MANIFEST
# v1.0 : 10-08-2014

CACHE:
# pages
index.html

# styles & scripts
css/theme.css
js/jquery.min.js
js/default.js

# images
/favicon.ico
images/logo.png

NETWORK:
login.php

FALLBACK:
/ /offline.html
```

Figure 98 *– Example of a manifest file (**Source:** https://www.tutorialrepublic.com/html-tutorial/html5-application-cache.php)*

It is now time to explain the coding of *Figure 98*.

- Firstly, it is important to understand that manifest files can have three different sections: **CACHE, NETWORK**, and **FALLBACK**.
- The files listed under the CACHE: section header (or right after the CACHE MANIFEST line) are clearly cached after they are downloaded for the first time;
- Files under the NETWORK: section header are white-listed resources that are never cached and are only available online. It means users cannot never access login.php page when they're offline;
- The FALLBACK: section specifies alternative pages the browser should use in case the connection to the server cannot be done. Each entry in this section lists two URIs — first is the primary resource, the second is the fallback. For instance, in *Figure 98* case, offline.html page will be displayed if the user is offline. Also, both URIs must be from the same origin as the manifest file.
- It should be noted that lines starting with '#' (hash symbol) are comment lines.

Therefore, if an application cache is there, the browser loads the document and its associated resources straight from the cache, without accessing the network. Then browser checks to find out whether the manifest file has been updated on the server. If it has been updated, the browser downloads the new version of the manifest and the resources listed in it.

It is important to note that the manifest file itself should not be specified in the cache manifest file. If so, it will be quite difficult to notify the browser that a new manifest is available.

STEP 2 – Use the cache manifest file. After creating it, the web designer should upload the cache manifest file on the web server — making sure the web server is configured to serve the manifest files with the MIME type text/cache-manifest.

To make the cache manifest work, the web designer will need to enable it in the web pages, by adding the manifest attribute to the root <html> element, as shown by *Figure 99* below:

```
<!DOCTYPE html>
<html lang="en" manifest="example.appcache">
<head>
    <title>Using the Application Cache</title>
</head>
<body>
    <!--The document content will be inserted here-->
</body>
</html>
```

Figure 99 *– Making the cache manifest work (**Source:** https://www.tutorialrepublic.com/html-tutorial/html5-application-cache.php)*

If the user is online, the result for this code will be the following:

Figure 100 *– about_blank#blocked (**Source:** https://www.tutorialrepublic.com/html-tutorial/html5-application-cache.php)*

HTML5 Web Workers

This subtopic will be essentially useful for further learning on JS topics, as it will teach how to use HTML5 web worker to run JS code in the background. So, learners must come back if they experience any difficulties.

If one tries to perform intensive, time-consuming, and heavy calculations demanding tasks with JavaScript, it probably will freeze up the webpages and will prevent users

from doing anything until the job is done. Why? Well, because JS code always runs in the foreground. However, HTML5 has a new technology ('web worker') created to perform background work apart from other user-interface scripts, without impacting the performance of the page. Distinct from normal JS operations, web worker does not interrupt the user and the web page stays responsive because they are running the tasks in the background.

Web workers are specially useful for performing a time-consuming task. So, in the first, example, a simple JS task that counts from zero to 100 000 will be created *(name of the file should be worker.js)*, as follows on *Figure 101:*

```
1  var i = 0;
2  function countNumbers() {
3      if(i < 100000) {
4          i = i + 1;
5          postMessage(i);
6      }
7
8      // Wait for sometime before running this script again
9      setTimeout("countNumbers()", 500);
10 }
11 countNumbers();
```

Figure 101 *– Creating a JS task counting from 0 to 100 000 (**Source:** https://www.tutorialrepublic.com/html-tutorial/html5-web-workers.php)*

Note: For a better understanding, it is advisable to download the code from Fig. 101 and follow every step of this chapter.

So, now that the web worker file has been created, it is time to start the web worker from an HTML document that runs the code inside the file named "worker.js" in the background and gradually shows the result on the web page. It should be noted that the number in the right will always be growing until it reaches 100 000.

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4  <meta charset="utf-8">
5  <title>Using HTML5 Web Workers</title>
6  <script>
7      if(window.Worker) {
8          // Create a new web worker
9          var worker = new Worker("/examples/js/worker.js");
10
11         // Fire onMessage event handler
12         worker.onmessage = function(event) {
13             document.getElementById("result").innerHTML = event.data;
14         };
15     } else {
16         alert("Sorry, your browser do not support web worker.");
17     }
18 </script>
19 </head>
20 <body>
21     <div id="result">
22         <!--Received messages will be inserted here-->
23     </div>
24 </body>
25 </html>
```

26

Figure 102 *– Starting the web worker (**Source:** https://www.tutorialrepublic.com/html-tutorial/html5-web-workers.php)*

Now explaining what is going on the example from above, the statement **var worker = new Worker("worker.js");** creates a new web worker object, which is used to communicate with the web worker. When the worker posts a message, it triggers the **onmessage** event handler (line 14) that permits the code to receive messages from the web worker. The **event.data** element comprises the message sent from the web worker. For the record, the code that a worker runs is always stored in a separate JavaScript file to prevent web developer from writing the web worker code that makes an attempt to use global variables or directly open the elements on the web page.

It is also possible to **put an end to a running worker** in the middle of the operation, following the example below:

```html
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Start/Stop Web Worker in HTML5</title>
<script>
    // Set up global variable
    var worker;

    function startWorker() {
        // Initialize web worker
        worker = new Worker("/examples/js/worker.js");

        // Run update function, when we get a message from worker
        worker.onmessage = update;

        // Tell worker to get started
        worker.postMessage("start");
    }

    function update(event) {
        // Update the page with current message from worker
        document.getElementById("result").innerHTML = event.data;
    }

    function stopWorker() {
        // Stop the worker
        worker.terminate();
    }
</script>
</head>
```

Figure 103 *– Stopping the running worker (**Source:** https://www.tutorialrepublic.com/html-tutorial/html5-web-workers.php)*

The example above shows how to start and stop worker from a web page by simply clicking on HTML buttons.

HTML5 Server-Sent Events

This subchapter will be useful for understanding how to use the HTML5 server-sent events feature to make a unidirectional and permanent connection between a web page and a server.

HTML5 server-sent event is an innovative way for the web pages to communicate with a web server. Nevertheless, there are some circumstances where web pages need a longer-term connection to the web server, for example, on stock quotes on finance websites where price updated automatically or game tickers running on various sports websites. Such things are feasible with the HTML5 server-sent events, as it makes available for a web page to hold an open connection to a web server, in a way that the web server can send a new response mechanically at whatever time. At this point, there is no need to reconnect and run the same server script from the beginning repeatedly.

For a better understanding of the aforementioned concepts, a PHP[1] file named "server_time.php" and type the following script into it. This file will merely report the present time of the web server's built-in clock in regular intervals:

```php
<?php
header("Content-Type: text/event-stream");
header("Cache-Control: no-cache");

// Get the current time on server
$currentTime = date("h:i:s", time());

// Send it in a message
echo "data: " . $currentTime . "\n\n";
flush();
?>
```

Figure 104 *– server_time.php example (Source https://www.tutorialrepublic.com/html-tutorial/html5-server-sent-events.php)*

[1] A PHP file is a plain-text file which contains code written in the PHP programming language. Since PHP is a server-side (back-end) scripting language, the code written on it is executed on the server. In fact, a PHP file may contain plain text, HTML tags, or code as per the PHP syntax. PHP is often used to develop web applications that are processed by a PHP engine on the web server.

The first two line of the PHP script (*Fig. 104*) sets two crucial headers. Number one, it sets the MIME type to text/event-stream, which is needed by the server-side event standard. The second line informs the web server to turn off caching or else the output of the script may be cached.

Every message send through HTML5 server-sent events must start with the text data: followed by the actual message text and the new line character sequence (\n\n).

And lastly, the PHP flush() function has been used to ensure that the data is sent immediately, rather than buffered until the PHP code is complete.

Regarding on **how to process messages in a web page**, the EventSource object is used to receive server-sent event messages. In the example below, learners will see how a HTML document simply receives the current time reported by the web server and displays it to the web page visitors. For a better understanding, an HTML document named “demo_sse.htlm” will be created and furtherly placed in the same project

directory where “server_time.php” is located. The download of the following code can be done in the source’s link available below.

```
<!DOCTYPE html>
<html lang="en">
<head>
<title>Using Server-Sent Events</title>
<script>
    window.onload = function() {
        var source = new EventSource("server_time.php");
        source.onmessage = function(event) {
            document.getElementById("result").innerHTML += "New time received
from web server: " + event.data + "<br>";
        };
    };
</script>
</head>
<body>
    <div id="result">
        <!--Server response will be inserted here-->
    </div>
</body>
</html>
```

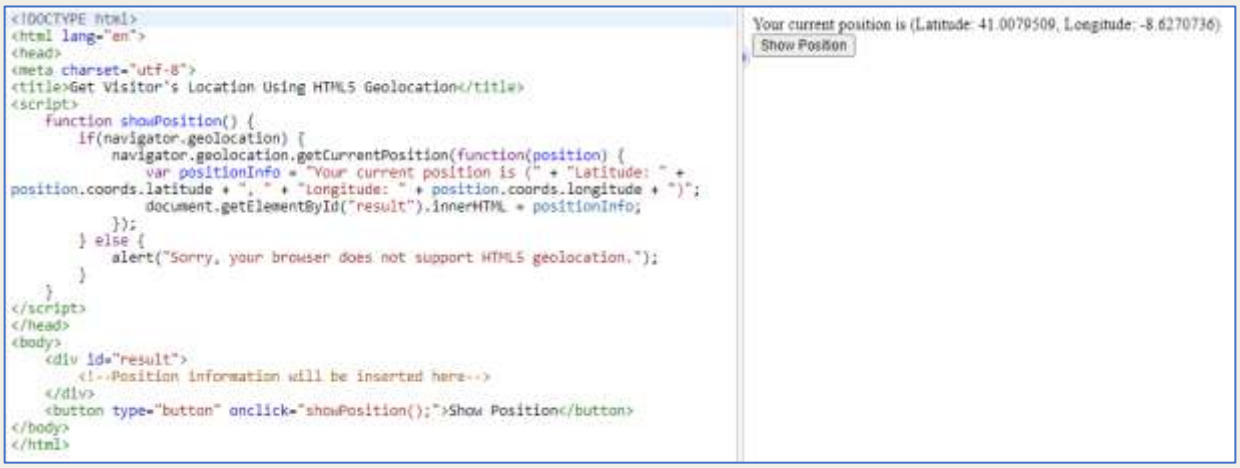
Figure 105 *– How to process messages in a web page (Source https://www.tutorialrepublic.com/html-tutorial/html5-server-sent-events.php)*

HTML 5 Geolocation

Through this subtopic, learners will get a few insights on how to use HTML5 geolocation feature for detecting user’s location. This feature lets the programmer find out the geographic coordinates (latitude and longitude) of the website visitor’s current location. It is especially useful for providing the best browsing experience for the visitor, since, for instance, this tool can show search results that are physically close to the user’s location.

Receiving the position information of the website visitor using the HTML5 geolocation API is not difficult. It exploits the three methods that are packed into the navigator.geolocation object — getCurrentPosition(), watchPosition() and clearWatch().

After the user agrees to let the browser tell the web server about his/her position (web browsers will not share the visitor location with a webpage unless the user agrees to do so), the geolocation process should occur as follows:




```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Get Visitor's Location Using HTML5 Geolocation</title>
<script>
    function showPosition() {
        if(navigator.geolocation) {
            navigator.geolocation.getCurrentPosition(function(position) {
                var positionInfo = "Your current position is (" + "Latitude: " +
position.coords.latitude + ", " + "Longitude: " + position.coords.longitude + ")";
                document.getElementById("result").innerHTML = positionInfo;
            });
        } else {
            alert("Sorry, your browser does not support HTML5 geolocation.");
        }
    }
</script>
</head>
<body>
    <div id="result">
        <!--Position information will be inserted here-->
    </div>
    <button type="button" onclick="showPosition();">Show Position</button>
</body>
</html>
```

Figure 106 *– The Geolocation feature process (**Source:** https://www.tutorialrepublic.com/html-tutorial/html5-geolocation.php)*

In case a user does not intend to share his/her location data with the website, the programmer can supply two functions when calling the getCurrentLocation() function. First function is called in case geolocation attempt is successful, whereas the second is called if the geolocation attempt fails.

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Handling the Geolocation Errors and Rejections</title>
<script>
    // Set up global variable
    var result;

    function showPosition() {
        // Store the element where the page displays the result
        result = document.getElementById("result");

        // If geolocation is available, try to get the visitor's position
        if(navigator.geolocation) {
            navigator.geolocation.getCurrentPosition(successCallback, errorCallback);
            result.innerHTML = "Getting the position information...";
        } else {
            alert("Sorry, your browser does not support HTML5 geolocation.");
        }
    };

    // Define callback function for successful attempt
    function successCallback(position) {
        result.innerHTML = "Your current position is (" + "Latitude: " + position.coords.latitude + ", " + "Longitude: " + position.coords.longitude + ")";
    }

    // Define callback function for failed attempt
    function errorCallback(error) {
        if(error.code == 1) {
            result.innerHTML = "You've decided not to share your position, but it's OK. We won't ask you again.";
        } else if(error.code == 2) {
            result.innerHTML = "The network is down or the positioning service can't be reached.";
        } else if(error.code == 3) {
            result.innerHTML = "The attempt timed out before it could get the location data.";
        } else {
            result.innerHTML = "Geolocation failed due to unknown error.";
        }
    }
</script>
</head>
<body>
    <div id="result">
        <!--Position information will be inserted here-->
    </div>
    <button type="button" onclick="showPosition();">Show Position</button>
</body>
</html>
```

Figure 106 *– Applying two functions to the getCurrentLocation() function (**Source:** https://www.tutorialrepublic.com/html-tutorial/html5-geolocation.php)*

There are many interesting functions to explore with geolocation data, as showing the user's location on Google Maps. Based on the latitude and longitude data retrieved through the HTML5 geolocation feature, this example shows the reader's current location. This simply shows a static image showing user's location, although an interactive Google maps with dragging, zoom in/out and other features, as this example shows.

All the aforementioned examples have been based on the getCurrentPosition() method. Nevertheless, the geolocation function has another technique –watchPosition() that allows to track the visitor's movement by returning the updated position as the location changes. watchPosition() has the same input parameters as getCurrentPosition(). Yet, watchPosition() may activate the success function multiple times — when it gets the

location for the first time, and again, every time it spots a new position, as this example suggests.

HTML5 Drag and Drop

It is a common procedure in the online daily routine to drag and drop an element to another location in a website. The HTML5 Drag and Drop feature allows to do so, and any element can be dragged and dropped. This w3schools example sets a simple drag and drop example learners may try to get more familiar with this concept. Even though the code seems difficult to understand, it is quite simple and logic:

- Firstly, to make an element draggable, the draggable attribute must be true:

```
<img draggable="true">
```

- Then, it should be specified what should happen once the element is dragged. In the w3schools example given above, the ondragstart attribute calls a function (drag (event)) that specifies what data will be dragged. The dataTransfer.setData() process sets the data type and the value of the dragged data:

```
function drag(ev) {
 ev.dataTransfer.setData("text",ev.target.id);
}
```

In this example, the data type is “text”, being the value the id of the draggable element (“drag1”).

- The ondragover event stipulates where the dragged data can be dropped. By default, data/elements are unable to be dropped in other elements. To permit a drop, the web designer should prevent the default handling of the element, by calling the event.preventDefault() technique for the ondragover event:

```
event.preventDefault()
```

- As soon as the dragged data is dropped, a drop event happens. In the example given previously, the ondrop attribute calls a function, drop(event):

```
function drop(ev) {
  ev.preventDefault();
  var data=ev.dataTransfer.getData("text");
  ev.target.appendChild(document.getElementById(data));
}
```

✓ Call preventDefault() to prevent the browser default handling of the data (default is open as a link on drop);

✓ The programmer gets the dragged data with the dataTransfer.getData() technique. This technique will return any data that was set to the same type in the setData() technique;

✓ The dragged data is the id of the dragged element ("drag1")

✓ The programmer should also attach the dragged element into the drop element.

HTML5 References

For a detailed, comprehensive list of elements concerning **HTML5 Tags/Elements, HTML5 Global Attributes**, **HTML5 Event Attributes, HTML5 Language Codes, HTML5 Character Entities**, **HTTP Status Codes, HTML5 Color Picker,** and other useful references, learners should refer to the following link (HTML5 References section).