

CSE1500: Web - Lecture 2 - HTML & CSS

Lecture Summary

The WDT Team

Ujwal Gadiraju, Sagar Chethan Kumar, Ciprian Ionescu

Contents

1	HTML & CSS	2
1.1	Web Design	2
1.2	Hypertext Markup Language (HTML)	4
1.3	Cascading Stylesheets (CSS)	11
2	Sample Question Types	15
3	Extra Exercises	17
4	References and Further Reading	17

⚠ These summaries are not extensive overviews of lecture content and **DO NOT** cover all possible exam material. Rather, these serve as a refresher and capture **main points** for a given lecture.

Learning Goals

You should be able to do the following after the **lecture** and **assignment #1**:

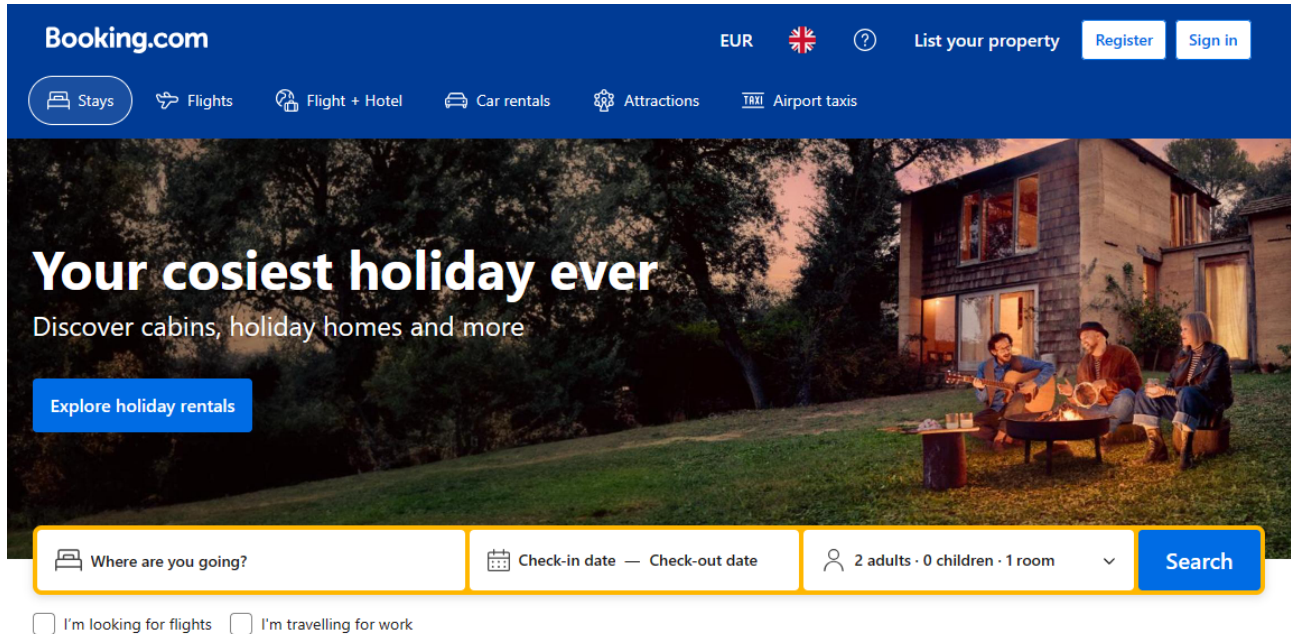
- **Analyse the requirements for a web application given a description of an application. (LO2)**
- **Create an application front-end with HTML and CSS based on a given static design. (LO3)**
- **Apply** Web design principles during the design stage of a Web app.
- **Position and style** HTML elements according to a given design of a web page.
- **Employ** HTML5 to create web pages.
- **Employ** CSS's data access/creation facilities.

1 HTML & CSS

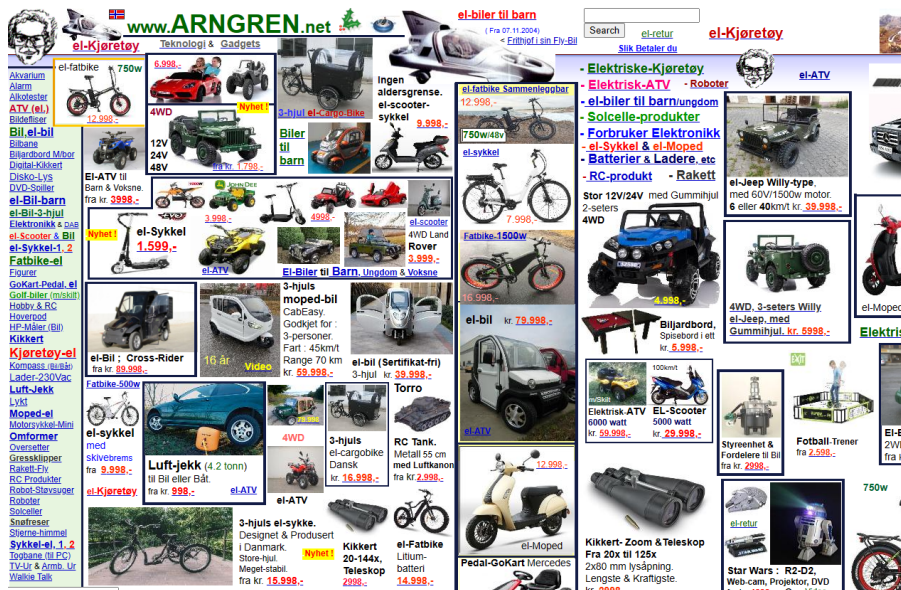
1.1 Web Design

Web design is **not** trivial. However, in general, there are a few guiding principles and approaches that we can employ that make our jobs as web developers a lot easier.

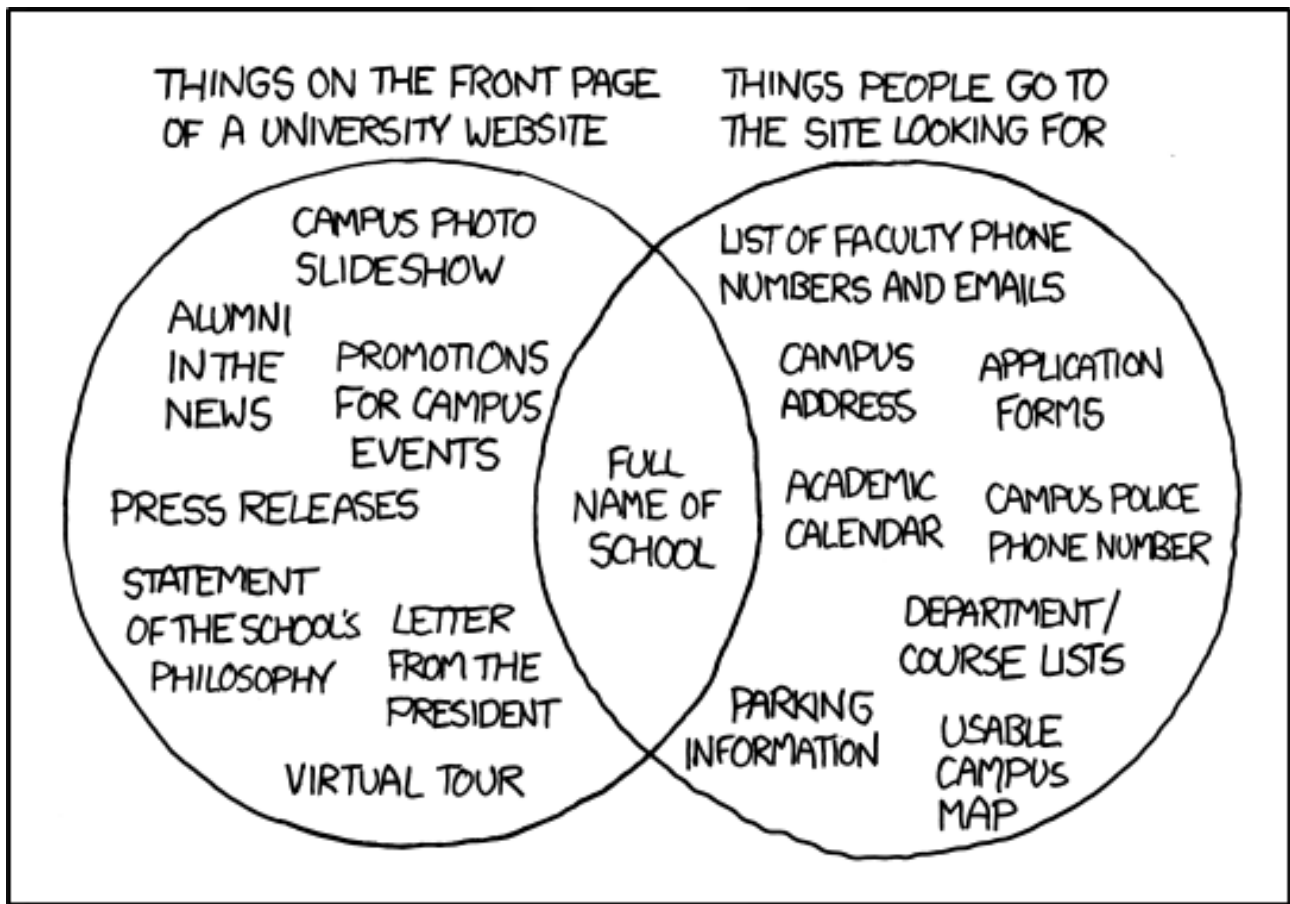
1. **Don't** make me **think**. How a webpage works should be self-evident (buttons, links, style standards, and clear user paths are some ways of achieving just that).



2. Minimize clutter and "fluff"

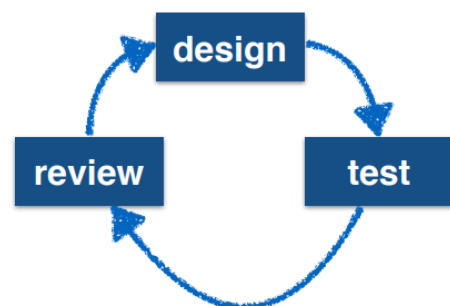


3. Can't make it self-evident? Make it **intuitive**. Minimize the cognitive load of the users.



4. Test **often** and **early**. (Creation of an account, forgot password, **queries**, deletions...)

Usability test: give a user a **typical task** and observe her



1.2 Hypertext Markup Language (HTML)

HTML, short for Hypertext Markup Language, serves as the language that dictates how a web browser should present a webpage. Unlike a programming language, HTML functions as a markup language, incorporating codes that define the formatting of a page without including procedural code.

Let's take a look at a simple example:

```
<p>
I love CSE1500! Ever since I started the course
I can't stop thinking about relational databases and
functional dependencies! Now I am doing the web segment of
the course and I can barely hold all
my <b>excitement</b> in!
</p>
```

Typical Rendering:

I love CSE1500! Ever since I started the course I can't stop thinking about relational databases and functional dependencies! Now I am doing the web segment of the course and I can barely hold all my **excitement** in!

HTML consists of tags, such as `<p>`, filled with plain text. We have opening tags like the `<p>` tag which begins a paragraph and closing tags like `</p>` which ends a paragraph. Similarly, `` starts bold text and `` ends it.

In HTML, almost every opening tag has a closing tag. There are a few exceptions called self-closing tags, but the overwhelming majority of tags must be closed. Additionally, some tags may have attributes, such as the `face` attribute of the `` tag. If an attribute value contains a space, we must enclose it in quotation marks:

Font Face Examples

```
<font face=arial>
<font face="arial narrow">
```

Many HTML tags exist and are regularly used. However, for the sake of brevity, both in this course and in this summary, we will **only** cover the main ones. You can find a list of most tags [here](#).

HTML describes structure and content, which in many cases can be partially described with the main following tags:

<code><h1></code> - <code><h6></code>	Headings
<code><p></code>	Paragraph
<code></code>	Group elements on a single line
<code><div></code>	Group elements across lines
<code><article></code> <code><section></code> <code><nav></code>	Group elements in semantically meaningful ways
<code></code> <code><video src=""></code> <code><audio src=""></code>	Multimedia
<code></code> <code></code>	Unordered List (bullet points)
<code></code> <code></code>	Ordered (numbered) List
<code><table></code> <code><tr></code> <code><td></code>	Tables with rows and cells
<code><form></code> <code><input></code>	Forms with user input.

Table 1: HTML Tags and Descriptions

Often we will nest this HTML to section off parts of a webpage and the logic related to it.

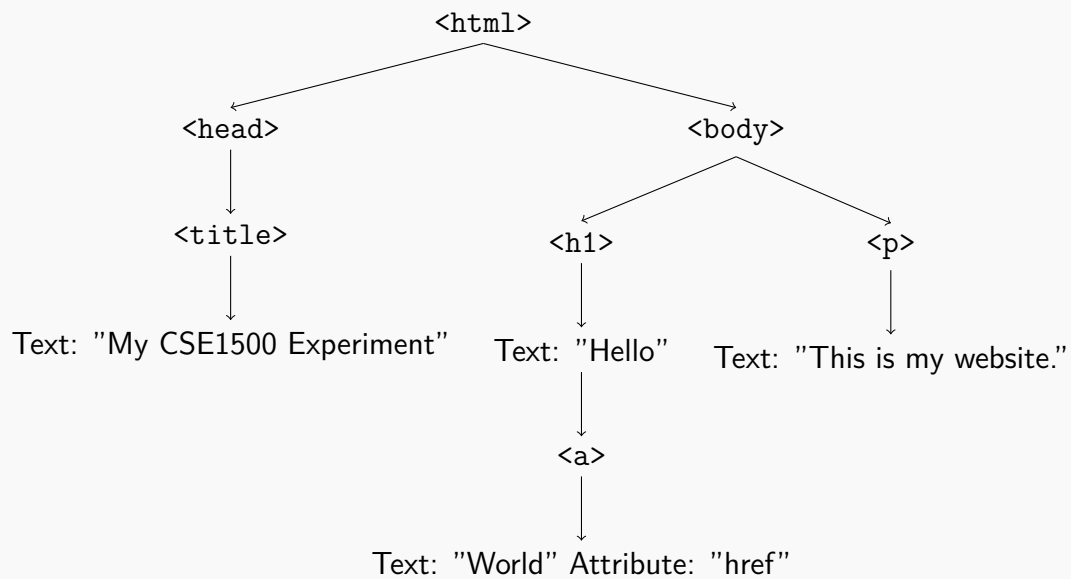
CSE1500 Website Example

```
<html>
  <head>
    <title>My CSE1500 Experiment</title>
  </head>
  <body>
    <h1>Hello
      <a href="https://www.tudelft.nl">World</a>
    </h1>
    <p>This is my website.</p>
  </body>
</html>
```

The HTML **Document Object Model** (DOM) is a an Object Model for HTML that defines individual HTML elements as *objects*, for which properties, methods, and events can be handled. Often times we can see a tree representation for the hierarchical nature of the DOM.

In our example the DOM representation would look like this:

CSE1500 DOM Example



When working on **web assignment #1** keep the DOM Tree in mind so that locating elements in your HTML code will be easier when you need to apply operations on them.

Lists and Tables:

In HTML we commonly encounter unordered and ordered lists as previously seen, which can be defined as follows:

HTML Lists

```
<h2>Unordered List</h2>
<ul>
  <li>Item 1</li>
  <li>Item 2</li>
  <li>Item 3</li>
</ul>

<h2>Ordered List</h2>
<ol>
  <li>First item</li>
  <li>Second item</li>
  <li>Third item</li>
</ol>
```

Unordered List

- Item 1
- Item 2
- Item 3

Ordered List

1. First item
2. Second item
3. Third item

In addition to the list we have other attributes, like type, start, and reversed that can change the appearance of a list. Can you tell what the following code produces?

List Attributes Experiment

```
<h2>Unordered List with Attributes</h2>
<ul type="circle">
  <li>Item 1</li>
  <li>Item 2</li>
  <li>Item 3</li>
</ul>

<h2>Ordered List with Attributes</h2>
<ol start="5" type="A" reversed>
  <li>First item</li>
  <li>Second item</li>
  <li>Third item</li>
</ol>
```

Links and Anchors:

The `<a>` tag defines a hyperlink, which is used to link from one page to another. The essential attribute of the `<a>` element is the `href` attribute, which indicates the link's destination. The `<a>` `target` attribute can specify how the link is opened.

Links and Anchors

```
<p>The <a href="https://www.example.com" target = "_top">example
link</a> defines a hyperlink, which is used to link from one page
to another.</p>
```

Images:

Similarly the `` tag can be used to embed images into an HTML page and with its `attributes` you can specify its visual properties.

Images

```

```

Tables: HTML tables organize data in rows and columns. They use `<table>`, `<tr>` (table row), and `<td>` (table data) tags for structure. Headers are defined with `<th>` in `<thead>`.

There are also footers that are defined with `<tfoot>`.

Header 1 (Includes Col 1,2)		Header 2
Row 1, Col 1	Row 1 & 2, Col 2 (rowspan)	Row 1, Col 3
Row 2, Col 1		Row 2, Col 3
Row 3, Col 1	Row 3, Col 2	Row 3, Col 3
Footer 1		Footer 2

This example can be replicated with the following code:

Table Example

```
<table>
  <thead>
    <tr>
      <th colspan="2">Header 1 (Includes Col 1,2)</th>
      <th>Header 2</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>Row 1, Col 1</td>
      <td rowspan="2">Row 1 & 2, Col 2 (rowspan)</td>
      <td>Row 1, Col 3</td>
    </tr>
    <tr>
      <td>Row 2, Col 1</td>
      <td>Row 2, Col 3</td>
    </tr>
    <tr>
      <td>Row 3, Col 1</td>
      <td>Row 3, Col 2</td>
      <td>Row 3, Col 3</td>
    </tr>
  </tbody>
  <tfoot>
    <tr>
      <td colspan="2">Footer 1</td>
      <td>Footer 2</td>
    </tr>
  </tfoot>
</table>
```


Forms: Forms in HTML have various elements that enable us as web developers to collect user input. Such user input is sent to a server for processing. A more comprehensive description on forms can be found [here](#). This is an example of forms that has a variety of mediums for user input:

Text Input:

Password Input:

Email Input:

Number Input:

Date Input:

 ☐

Checkbox

☐

Radio Option 1

☐

Radio Option 2

Select:

Textarea:

Submit

Note, the example above has used some (sub-optimal) CSS to make the form a bit more compact. The code that produced this output is on the following page.

Form Example

```
<form action="/submit" method="post">
  <label for="text-input">Text Input:</label>
  <input type="text" id="text-input" name="text_input"
  placeholder="Enter text">

  <label for="password-input">Password Input:</label>
  <input type="password" id="password-input" name="password_input"
  placeholder="Enter password">

  <label for="email-input">Email Input:</label>
  <input type="email" id="email-input" name="email_input"
  placeholder="Enter email">

  <label for="number-input">Number Input:</label>
  <input type="number" id="number-input" name="number_input"
  placeholder="Enter number">

  <label for="date-input">Date Input:</label>
  <input type="date" id="date-input" name="date_input">

  <label for="checkbox-input">
    <input type="checkbox" id="checkbox-input"
    name="checkbox_input"> Checkbox
  </label>

  <label for="radio-input1">
    <input type="radio" id="radio-input1" name="radio_input"
    value="option1"> Radio Option 1
  </label>
  <label for="radio-input2">
    <input type="radio" id="radio-input2" name="radio_input"
    value="option2"> Radio Option 2
  </label>

  <label for="select-input">Select:</label>
  <select id="select-input" name="select_input">
    <option value="option1">Option 1</option>
    <option value="option2">Option 2</option>
    <option value="option3">Option 3</option>
  </select>

  <label for="textarea-input">Textarea:</label>
  <textarea id="textarea-input" name="textarea_input" rows="4"
  placeholder="Enter text"></textarea>

  <button type="submit">Submit</button>
</form>
```

1.3 Cascading Stylesheets (CSS)

CSS is the language we use to style an HTML document and is used to describe how HTML elements should be displayed. However, after seeing all the wonderful things HTML can do with all its tags and attributes, you might be asking yourself (or maybe not at all) do we even need CSS? To convince yourself, take a look at the (intentionally) drastic differences CSS can make to a webpage:



Figure 1: Mystery Webpage 1

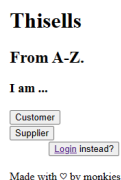


Figure 2: Mystery Webpage 2

Can you guess which one used CSS? You can easily check the impact of CSS on any webpage by right-clicking on a webpage and selecting **Inspect** to remove the CSS code. As shown above, CSS, when used properly, can help promote responsive layouts, intuitive element groupings, and flexible design choices.

CSS Syntax:

A CSS rule describes presentation and consists of declarations which are property-value pairs.

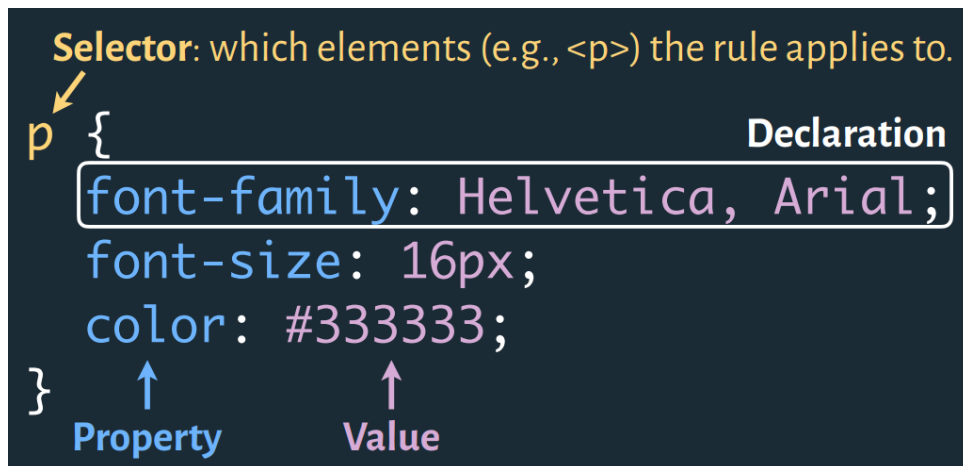


Figure 3: CSS Rule Explained

In general, we can make use of a stylesheet in 3 main ways: inline (directly in the style attribute of a HTML Tag), internal (defining a `<style>` in the HTML), and external (a dedicated CSS file).

Methods of CSS Integration

```
<!-- Inline CSS -->
<div style="property: value;">Content</div>

<!-- Internal CSS -->
<style>
  selector {
    property: value;
  }
</style>

<!-- External CSS -->
<link rel="stylesheet" type="text/css" href="styles.css">
```

CSS Selectors:

There are many different ways to select the elements which a CSS rule will be applied to. An overview of code examples can be found on the following page.

CSS Selectors

```
/* Universal selector: selects all elements */
* {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
}

/* element Selects elements by tag (e.g., p, div, h1, ul, li) */
p {
    font-size: 16px;
    line-height: 1.5;
}

/* [attr="val"] Selects elements by attribute name (and value) */
a[href="http://tudelft.nl"] {
    color: #007bff;
    text-decoration: none;
}

/* #id Selects elements by their id attribute (ex: id="header") */
#header {
    /* Styles for an element with */
    background-color: #f2f2f2;
    padding: 10px;
}

/* .class Selects elements by their class attribute
(ex: class="photo") */
.photo {
    border-radius: 50%;
    overflow: hidden;
}

/* Pseudo-classes */
.button:hover {
    background-color: #4caf50;
    color: #fff;
}

input[type="checkbox"]:checked {
    border: 2px solid #4caf50;
}

input:focus {
    outline: 2px solid #007bff;
}
```

Box Model:

The CSS box model is a box that wraps around every HTML element. It consists of: content, padding, borders and margins. The image below illustrates the box model:

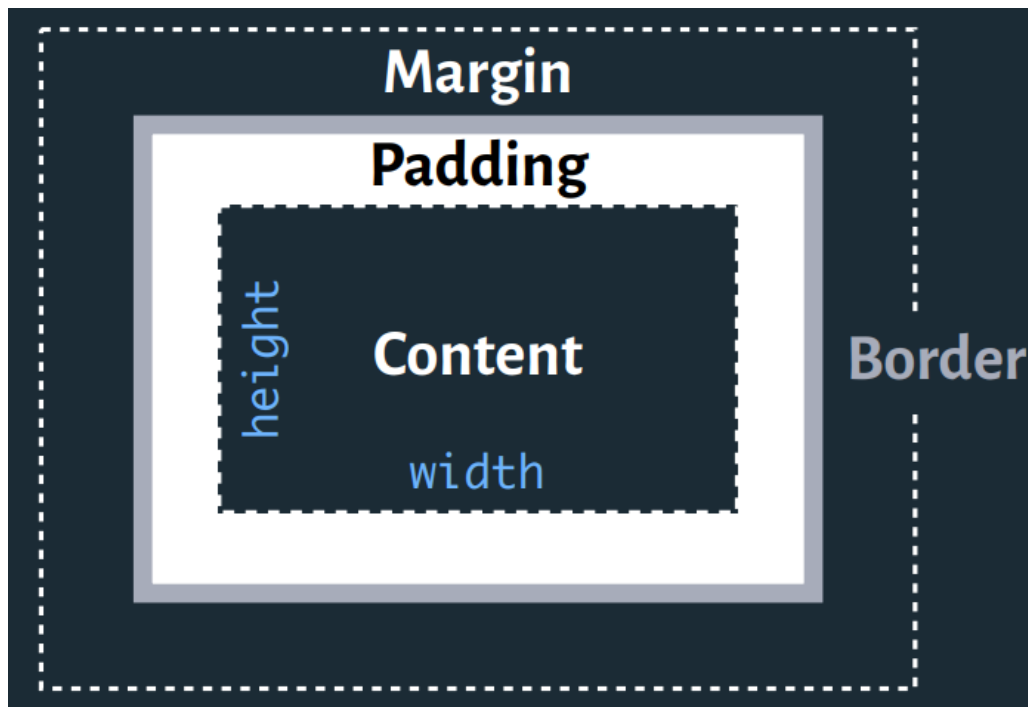


Figure 4: CSS Box Model

Text and images are visible within the box's **content**. **Padding** creates a transparent space around the content. The **border** surrounds both the padding and content. **Margin** clears an area beyond the border, and it is also transparent. Try testing out what the CSS code below produces!

Box Model Experiment

```
div {  
  width: 300px;  
  border: 15px solid green;  
  padding: 50px;  
  margin: 20px;  
}
```

Flexbox:

CSS has many display types but one of the most commonly used ones is `display: flex`. Through the *primary* and *secondary* axes we can achieve various arrangements of elements that enable responsive display. The best way to get familiar with this is practice: [Flexbox Froggy](#).

Grid:

Similarly, CSS Grid is best learnt in practice and experimenting with some layouts. Follow this [resource](#) and observe what effects certain changes bring on your local IDE while reading it.

2 Sample Question Types

(1) Change the following code to reach the target table.

```
1 <table border="1" cellpadding="5" cellspacing="0">
2   <thead>
3     <tr>
4       <th>Header 1</th>
5       <th>Header 2</th>
6       <th>Header 3</th>
7     </tr>
8     <tr>
9       <th>Header 4</th>
10      <th>Header 5</th>
11      <th>Header 6</th>
12    </tr>
13  </thead>
14  <tbody>
15
16 </tbody>
17  <tfoot>
18    <tr>
19      <td>Table Footer</td>
20    </tr>
21  </tfoot>
22 </table>
```

Header 1	Header 2	Header 3
Header 4	Header 5	Header 6
Table Footer		



Header 1	Header 2		Header 3
	Header 4	Header 5	Header 6
Table Footer			

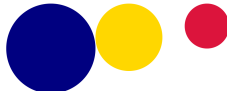
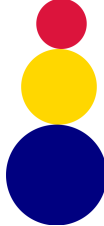
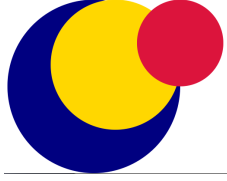

Figure 5: Source and target tables

(2) Consider the HTML snippet below. What will the rendering of this code look like?

```

1  <!DOCTYPE html>
2  <html>
3  <head>
4    <style>
5      div {
6        width: 200px;
7        height: 200px;
8        /* border-radius setting leads to a circle */
9        border-radius: 200px;
10       background-color: navy;
11       position: absolute;
12       bottom: 0;
13     }
14     div::before {
15       content: '';
16       width: 150px;
17       height: 150px;
18       border-radius: 100px;
19       position: absolute;
20       bottom: 50px;
21       background-color: gold;
22     }
23     div::after {
24       content: '';
25       width: 100px;
26       height: 100px;
27       border-radius: 50px;
28       position: absolute;
29       bottom: 100px;
30       background-color: crimson;
31     }
32   </style>
33 </head>
34 <body><div></div></body>
35 </html>

```

- a. 
- b. 
- c. 
- d. 

3 Extra Exercises

If you are feeling confident on the topic of **HTML and CSS** check out the following tasks¹:

- [Flexbox Froggy](#) (Try all exercises!)
- [W3Schools HTML Practice](#)

4 References and Further Reading

Here are some useful resources for learning more about **HTML and CSS**:

- [Dr. Philip Greenspun's HTML Notes](#)
- [Interactive HTML/CSS practice from UWash](#)

¹Note: None of these exercises are mandatory nor provide a bonus.