# Section 1 – First steps

## High-level overview of Web Development

Static website 🡪 Upon request to the server, the website assets are sent to browser as is

Dynamic website 🡪 Upon request to the server, the backend will assemble the website’s assets and then send to the browser

# Section 2 – HTML Fundamentals

## Structure of HTML document

<!DOCTYPE html> <=== Declare that this document is using HTML

<html> <=== A HTML element to contain all the stuff the HTML doc should have

<head></head>

<body></body>

</html>

## Text elements

|  |  |
| --- | --- |
| **<h1> to <h6>** | Helps to break up text content into sections  Has head from size h1 to h6 (Descending size)  Good practice to have only 1x h1 heading |
| **<p>** | Paragraph of text |
| **<b> OR <strong>** | b: bolding text content (OLD WAY OF BOLDING)  **strong: same as b (NEW WAY, makes more semantic sense)** |
| **<i> OR <em>** | i: italicise text content (OLD WAY)  **em: same as i (NEW WAY, makes more sematic sense)** |
| **<ol>** | Ordered list |
| **<ul>** | Unordered list |
| **<li>** | List item |

## Semantic tags

|  |  |
| --- | --- |
| **<header>** | Contains introductory content for a page (e.g., a banner), or a section of a page. |
| **<nav>** | Contains navigation content, such as a website navigation menu. |
| **<main>** | Contains the main content of the web page. |
| **<aside>** | Contains content that is tangentially related to the main content of the page (often this is presented in a sidebar). |
| **<footer>** | Contains the footer of a page, or of a section of a page. Typically, the footer contains information about the content, such as the author and a copyright statement. |
| **<article>** | Represents a self-contained composition in a document, page, application, or site, which is intended to be independently distributable or reusable |

## Semantic HTML

A way of writing HTML whereby each HTML element is writing with the appropriate tag with a proper meaning. This allows clear communication of each element’s meaning to the developer, browser, and users.

* Improve accessibility as text to speech programs make use of the HTML tags to recognize contents

# Section 3 – CSS Fundamentals

## CSS Rule

Many of these can be within 1 .css file

Diagram, timeline

Description automatically generated

## Inline vs Internal vs External CSS

|  |  |  |
| --- | --- | --- |
| **Inline** | **Internal** | **External** |
| CSS that are within the opening tag of a HTML element | CSS that are within the <head> of a .html file | CSS that are in its own isolated .css file and included into the .html file |
| Eg.  <h1 **style=”font-transform: uppercase”** /> | Eg.  <head>  **<style>**  **h1 {**  **font-transform: uppercase;**  **}**  **</style>**  </head> | Eg. **style.css**  **h1 {**  **font-transform: uppercase;**  **}**  index.html  <head>  <link href=”style.css” rel=”stylesheet”/>  </head> |

## Selectors

|  |  |  |
| --- | --- | --- |
| **Name** | **Syntax** | **Description** |
| Element Selector | h1 {…} | Select elements by its type |
| Class selector | .primary-header {…} | Select elements by its class |
| ID selector | #author {…} | Select elements by its ID |
| List selector | h1, h2, h3 {…} | Select elements that satisfying any selector included above |
| Descendent selector | article p {…} | Select the child element of an enveloping element |

## Pseudo-classes

Keywords that can be added to a selector to specify the state of the element you are selecting for

* Eg. *li:first-child*: Select a list item element that is the first child of its parent element

## Hyperlinks

When styling an anchor (ie. hyperlink) element, we must take care of its 4 states:

* :link, :visited, :hover, :active
  + :link -> Selects anchor that are actually a link (ie. has href attribute)
  + :active -> Selects anchor that are currently being clicked
* The element is expected to be styled differently for each state and the styling should be done in the above order

## Theory 1 – Conflicts between Selectors and Stylings

When there are >1 selectors selecting an element, all of them applies to it

For any rule that is in conflict (ie. font-size: 10px; and font-size: 15px;), resolve them with the following diagram:

Graphical user interface, text, application, chat or text message

Description automatically generated

* For selectors of differing priorities, the highest priority wins.
  + The more specific a selector, the higher priority the styling is
* For selectors of the same priority, the latest selector wins.

## Learning points in Challenge #1

* Use **hexadecimal to represent colors**
* **text-transform**: uppercase
* **text-align** not align-text
* **text-decoration**: underline
* **Use classes whenever possible**, don’t be too general
* **list-style-type** to adjust the bullet point shapes (PUT IN LIST INSTEAD OF LIST ITEMS)
* **cursor: pointer** to change cursor to pointing finger when hovering a button

## Theory 2 – Inheritance and Universal Selector

### Inheritance

Certain properties from the parent elements are passed down to the child elements

* Mostly properties related to text
  + font-family, font-size, font-weight, font-style, color, line-height, letter-spacing, text-align, text-transform, text-shadow, list-style, etc.
* Inherited properties have the lowest priority, easily overwritten

Graphical user interface

Description automatically generated

This means that we **can propagate styling downwards in a similar fashion as event delegation** in JavaScript

## Universal Selector

\* {…}

Allow us to apply style to every element in the page but without any inheritance happening

* Useful for applying properties where it cannot be inherited
* Opposite of applying to *body* to let the properties be inherited by child elements

## Theory 3 – CSS Box Model

Defines how elements are displayed on a page and how they are sized. Each element on the page can be seen as **a rectangular box that has content, a border and space inside and outside of it**:

Diagram

Description automatically generated

* **Padding**: Spacing between the content and the border (Internal spacing)
* **Margin**: Spacing between the border and the nearby elements (External spacing)

### Box model – content-box

Diagram

Description automatically generated

* *Box-sizing: content-box*
* All the above values can be modified via CSS

### Box model – border-box

Defined width and height include the padding and borders of the box (ie. Whatever width and height that are defined, it will be the final width and height)

Diagram

Description automatically generated

* *Box-sizing: border-box;*
* Most CSS developers just set this to all elements using the \* universal selector

### Setting dimensions (width & height)

Width & height can be set to several units:

|  |  |
| --- | --- |
| **Px** | Pixels, doesn’t scale with window size |
| **%** | Percentage of the parent element’s height |
| **auto** | Scale automatically with the element’s content and other properties |

**Note**. Remember that these heights are just the defined dimensions not the final dimensions.

### Margins and Paddings

#### Global reset

A common practice to reset all paddings and margins to 0 using the universal selector

\* {padding: 0; margin: 0}

🡪 Provide a clean slate for the developers to work from and give explicit paddings and margins

#### Collapsing margins

When neighboring elements have margins that overlapped with one another, only the larger margins will be displayed and the smaller one will be ignored

* Eg. margin: 40px vs margin: 15px => margin: 40px will be displayed

## Trick to center a page’s content

1. Encapsulate the content into a blank container element like a <div>
2. Style the div to have
   1. Fixed width (Stop content from stretching)
   2. Auto left-right margin (Automatically balanced the left and right margin, giving the illusion of centering the container)

## Theory #4 – Types of boxes an element can be in

An element can be displayed in 3 different types of blocks:

Graphical user interface, application, Word

Description automatically generated

* By default, an element uses either block-level or inline boxes
* Can be modified but in normal use cases, shouldn’t modify

## Theory #5 – Absolute positioning

An element has 3 forms of positioning: Normal, Relative, and Absolute

Graphical user interface, application

Description automatically generated Diagram

Description automatically generated

* **Normal position**: Default, position according to the flow of the document
* **Relative position:** 
  + Position according to the flow of the document
  + Can apply offset relative to itself to adjust its position (top, right, bottom, left)
* **Absolute position:**
  + Remove the element from the standard flow of positioning
  + Has no impact on surrounding elements
  + Can apply offset relative to closest positioned ancestor or the initial containing block (top, right, bottom, left)

## Pseudo-element

Elements that don’t exists in the HTML but that we can still select and style in CSS

**Examples**

::first-child, ::first-letter, ::before (Creates an element as the first child for the selected element), ::After (Similar to ::before but as the last child for the selected element)

# Extracts from the Advanced CSS Course

## What happens to HTML and CSS upon page load

Diagram

Description automatically generated

When the HTML and CSS are loaded, they are parsed to form their respective object models

* When parsing a CSS, there are mainly 2 processes
  + Resolving conflicting CSS declarations (Cascading)
  + Processing final CSS values

## Cascading

A process of combining multiple stylesheets together and resolving the conflicts it has between different CSS rules and declarations when they are applied to an element

* Ie. Declarations from a CSS rule conflicts with declarations from another CSS rule, we need cascade to decide which declaration the element will use

### Sources of CSS declarations

|  |  |
| --- | --- |
| **Author** | Declarations written by the developers within the .css files |
| **User** | Declarations written by the users outside of the .css files  (eg. Declaring in Chrome devtools) |
| **Browser** | Default declarations that a browser made as a baseline |

### Cascading flow

A screenshot of a computer

Description automatically generated with low confidence

#### Importance

* Author declarations & User declarations are basic declarations without any !important

#### Specificity

Specificity of an element is determine based on how many different types of selectors were used in the CSS rule:

**(# of inline styles, # of IDs, # of classes/pseudo-classes/attribute, # of elements/pseudo-elements)**

We then compare the # of selectors of each type between the conflicting declarations. The declaration with higher number of more specified selector wins the conflict.

Eg.

Text

Description automatically generated

# Section 4 – Layouts: Floats, Flexbox and Grid

## 3 ways of building layouts

1. Floats
2. Flexbox
3. Grid

## Floats

A CSS property to remove an element from the normal flow, but allow other elements to wrap around it (Unlike absolute positioning)

Note. If all elements in a container float, the container element will **collapse**

### Clear float

A CSS property that set whether an element must be moved below floating elements that precede it

* Can be applied to floating or non-floating elements
* Element that is being cleared must be a block element
* Can be applied to a pseudo-element if there isn’t any element to keep the container element stretched

## Flexbox

A set of related CSS properties for building 1-dimensional layouts

* Based on the idea of distributing empty space in a container element to its child elements
* Make floats obsolete

### Components of a flexbox

Diagram

Description automatically generated

* *flex-direction* determines which axis will be the main axis and cross axis
  + Affects justify and alignment of items

### Properties

Graphical user interface, text, application

Description automatically generated

#### Flex shorthand

Text

Description automatically generated

* *flex: 1 ⬄ flex: 1 1 0 ⬄* All flex-items has the same size regardless of content

## Grid

A set of CSS properties for building 2-dimensional layouts

* Based on the idea of dividing a container element into rows and columns that can be filled with child elements
* Not meant to replace flexbox
  + 1D layout: flexbox
  + 2D layout: grid

### Components of Grid

Graphical user interface, application

Description automatically generated with medium confidence

* Unlike flexbox, the grid’s axis cannot be changed
* Gutters ⬄ Row gap or Column gap
* Each grid line is labelled with a positive and negative line number, both count from opposing directions

Graphical user interface

Description automatically generated

### Properties

Graphical user interface, text, application

Description automatically generated

#### Fr & auto

fr 🡪 A proportion value that we can give to distribute free space across a grid’s rows or columns

auto 🡪 Specify that a row/column will take the exact amount of space needed for its content

#### Repeat(…)

For grid-template-columns and grid-template-rows, if we have columns/rows with same distribution, we can declare them one-shot as follows:

#### Spanning a grid item across a row or column

# Section 5 – Web design rules and framework

## Good vs Bad design

Graphical user interface, application, Teams

Description automatically generated

## Web design ingredients

Graphical user interface, website

Description automatically generated

## Website personalities

Refer to the lecture slides for examples of each website personality

Text

Description automatically generated

## Typography

The art and technique of arranging type to make written languages legible, readable, and appealing when displayed

### Serif vs Sans-serif typefaces

|  |  |
| --- | --- |
| **Serif typefaces** | **Sans-serif typefaces** |
|  |  |
| * Gives a traditional/classic look and feel * Conveys trustworthiness * Good for long text | * Gives a modern look and feel * Clean and simple * Easier to use for beginner designer |

### Guidelines

#### Typefaces

1. Use only good and popular typefaces to play it safe
   * Can refer to the lecture slides or websites like Google Fonts and Font Squirrel
2. Stick to 1 or 2 typefaces per page
3. Choose the right typeface according to the website’s personality and message

#### Font sizes and weights

1. Limit your choices by referring to Type Scale tools or some pre-defined range

Eg.

SPACING SYSTEM (px)

2 / 4 / 8 / 12 / 16 / 24 / 32 / 48 / 64 / 80 / 96 / 128

FONT SIZE SYSTEM (px) from type-scale.com

10 / 12 / 14 / 16 / 18 / 20 / 24 / 30 / 36 / 44 / 52 / 62 / 74 / 86 / 98

1. For “normal” text, use font size between 16px and 32px
2. For long text (eg. blog post), use font size of at least 20px
3. For headlines, can use font size of 50px or more with a weight of 600 or more
   * Depending on the website’s personality

#### Readability

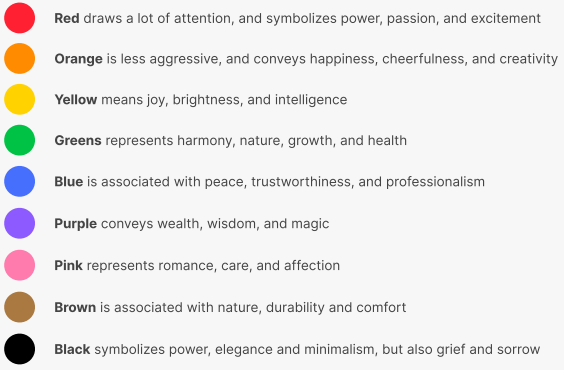
1. Use < 75 characters per line
2. The smaller/longer the text is, the larger the line height needs to be
   * For “normal” text, use a line height between 1.5 and 2.
   * For “big” text, use a line height below 1.5
3. Decrease the letter spacing in headlines if it looks unnatural
4. For short titles, experiment with all caps, small, and bold fonts with increased letter-spacing
5. Usually, don’t justify text
6. Centering text should only be for smaller text blocks

## Colors

### Guidelines

#### Choosing the right color

1. Pick a main color that match the website’s personality



1. Use a tone of the main color by referring to tools like OpenColor, TailwindCSS, and Flat UI

Calendar

Description automatically generated with medium confidence

#### Establishing a color system

1. Pick a grey color for the color palette
2. Pick an accent (secondary) color for the color palette through tools like Coolors and Palleton
3. Improve the color palette’s diversity by picking lighter and dark versions of the picked colors
   * Tints: Lighter versions of the colors
   * Shades: Darker versions of the colors

Chart, bubble chart

Description automatically generated

#### When and how to use colors

1. Use your main color to draw attention to the most important elements on the page

Graphical user interface, application

Description automatically generated

1. Use your accent colors to make components or sections stand out

Graphical user interface, text, application

Description automatically generated

1. Can try to use images and illustrations with supporting/consistent colors to the palette

Graphical user interface

Description automatically generated

#### Colors and typography

1. On dark colored backgrounds, can use a tint of the background for text color

Graphical user interface, text, application

Description automatically generated

1. Text should usually not be completely black, can try a lighter version to be less heavy
2. Establish enough contrast between text and background colors
   * At least 4.5:1 for normal text and 3:1 for large text (Can use tools like Coolors)

## Images and Illustrations

### Guidelines

#### Choosing images

1. Choose 1 of 4 image types: **Product photos, Storytelling photos, Illustrations, Patterns**

Graphical user interface

Description automatically generated A person in a green shirt

Description automatically generated with low confidence

Graphical user interface

Description automatically generated with low confidence A picture containing graphical user interface

Description automatically generated

1. Choose images that are **relevant and support your website’s message and story**
2. Prefer **original** images or **original-looking stock** images
   * Tools: Unsplash, Pexels, DrawKit, unDraw

Graphical user interface, application

Description automatically generated

#### Using images

1. Use images that **show real people to trigger user’s emotions**

Graphical user interface

Description automatically generated

1. **Crop** the image to fit your message if necessary
2. Can try to **combine** photos, illustrations, and patterns for better effect

A picture containing text, person

Description automatically generated

#### Handling text on images

1. Method 1: **Darkening/Brightening** **the image** completely/partially using a gradient

A picture containing text, different, screenshot, mountain

Description automatically generated

1. Method 2: Placing the text in the image’s **neutral area**

A person smiling with a car in the background

Description automatically generated with medium confidence

1. Method 3: Placing the text in **a box over the image**

A picture containing text, outdoor, house

Description automatically generated

#### Technical details

1. Image’s dimension should be **2x of their displayed size** to support high-res screens
2. Images should be **compressed** for a lower file size and better load performance
   * Tool: Squoosh
3. Ensure that images that are placed side-by-side have the **same dimensions / aspect ratios**

## Icons

### Guidelines

#### Picking icons

1. Use an icon pack
   * Resources: Phosphor icons, Ionicons, IconS8 etc
2. Use **only 1 icon pack** and not mix the icons from different icon packs
3. Use **SVG icons or icon fonts** instead of bitmap image icons (eg. .jpg, and .png)

Graphical user interface, application

Description automatically generated

1. Choose an icon pack that **fit the website’s personality**
   * Refer to the typography’s roundness, weight, and filled/outlined

#### When to use icons

1. Use icons to **provide visual assistance** to the text

A screenshot of a computer

Description automatically generated with low confidence

1. Use icons to **present product feature blocks**

Graphical user interface, text, application

Description automatically generated

1. Use icons to **represent actions**, and label with text (Unless no space or icon is 100% clear)
   * Don’t mix label and unlabeled icons together

Graphical user interface, application

Description automatically generated

1. Use icons as **bullet points**

Graphical user interface, text, application

Description automatically generated

#### How to use icons well

1. To keep icons **neutral**, use **same color** as text. To **draw more attention**, use **different color**

A green screen with white text

Description automatically generated with medium confidenceGraphical user interface

Description automatically generated with medium confidence

1. Icons must **make sense and fit the text/action**
2. **Don’t enlarge the icons further** than what they **were designed for**
   * Can enclose them in a larger shape to make it “larger”
   * Icons with a lot of details and thin lines are to be used in a larger size

Graphical user interface, application, Teams

Description automatically generated

Graphical user interface, text, application

Description automatically generated

## Shadows

By adding more shadow, the further away the element seems from the interface:

A picture containing graphical user interface

Description automatically generated

**Note**. In webpages, we assume the light source comes from top-down. Most elements’ shadows will only offset vertically.

### Guidelines

1. Shadows is not a must, only use them if it **makes sense for the website’s personality**
   * The more serious the website less, the less shadow the website has

Graphical user interface, application, Teams

Description automatically generated

1. Use shadows **in small doses**, don’t use it in every element
2. Keep the shadow to be **lighter** than darker
3. Use **small shadows for smaller elements** to stand out

Graphical user interface, text, application

Description automatically generated

1. Use **medium shadows for larger areas/sections** to stand out more

Graphical user interface, application, chat or text message

Description automatically generated

1. Use **large shadows for elements that are floating** above the interface

Graphical user interface, application

Description automatically generated

1. Experiment with **shadow strengths** to visualize interactions (eg. Click and Hover button)

Graphical user interface, text, application, chat or text message

Description automatically generated

1. Experiment with **glows (ie. colored shadows)** for interesting visuals

A picture containing text

Description automatically generated

### CSS property

*box-shadow: HORI\_OFFSET VERT\_OFFSET BLURNESS SCALE COLOR*

*text-shadow: HORI\_OFFSET VERT\_OFFSET BLURNESS COLOR*

## Roundness (Border-radius)

### Guidelines

1. Use roundness to **increase the playfulness/fun** of the design, making it **less serious**

Graphical user interface, application, website

Description automatically generated

1. Roundness of the elements should **match the roundness of the typefaces**

Graphical user interface, application

Description automatically generated

1. Use roundness on **buttons, images, icons, sections** etc

### CSS properties

*border-radius: AMOUNT\_IN\_PX;*

* A large enough border-radius can completely circle-fy the corners
* Individual corners of the element can be circle-fy or straighten:

*border-top-left-radius: AMOUNT\_IN\_PX;*

*border-top-right-radius: AMOUNT\_IN\_PX;*

*border-bottom-left-radius: AMOUNT\_IN\_PX;*

*border-bottom-right-radius: AMOUNT\_IN\_PX;*

## Whitespace

### What it does

* The right amount helps to make design look **clean, modern, and polished**
* Helps to **communicate how different piece of information are related** to one another
* **Implies invisible relationships** between the elements of a layout

### Guidelines

#### Where to use whitespace

1. Use **tons of whitespace between sections**
2. Use **lots of whitespaces between groups of elements**
3. Use **whitespace between elements**
4. Within groups of elements, **use whitespace instead of lines** to separate them

Graphical user interface, application

Description automatically generated

#### How much whitespace to use

1. The more some elements/groups of elements **belong together, the closer they should be**
   * Law of proximity

Graphical user interface, application, Teams

Description automatically generated

1. **Start with excessive amount** of whitespace, then cut down from there
2. **Match other design choices**. If you have big elements (eg. Text or icons), more whitespace

Graphical user interface, application

Description automatically generated

1. When it comes to the specific amount of spacing, try **use a hard rule** (eg. Multiples of 16px)

Chart, bar chart

Description automatically generated

## Visual Hierarchy

Establishes which elements/components of a design are the most important ones by drawing attention to them using a combination of position, size, colors, spacing, borders, and shadows

* Defines a “path” to guide the users through the page

### Guidelines

#### Fundamentals

1. Position **important elements closer to the top of the page**, where they get more attention
   * **User’s attention flow down** the page and its components

Graphical user interface, application

Description automatically generated

1. Be mindful of **image** usage, as they can **draw a lot of attention**
   * Smaller images can be used to draw lesser attention

A picture containing text, screenshot

Description automatically generated

1. Use **whitespace** strategically to **separate and emphasize elements** of a page

Graphical user interface, application, website, Teams

Description automatically generated

#### For text elements

1. Use **font size, font weight, color, and whitespace** to **convey importance** of text

Graphical user interface, application, Teams

Description automatically generated

1. Can **emphasize important** texts (eg. Titles, sub-titles, links, buttons, data points, icons) and **de-emphasize less important** texts (eg. Labels, secondary/additional information)

Graphical user interface, application, Teams

Description automatically generated

#### Between components

1. **Emphasize important** components using **background color/shadow/border** or **de-emphasize the less important** components

Graphical user interface, application, Teams

Description automatically generated

Graphical user interface, application, Teams

Description automatically generated

1. Can emphasize components like **Testimonials, Call-to-action sections, Highlight sections, Preview cards, Forms, Pricing tables, Important rows/columns in tables** etc

## User experience (UX)

The overall experience the user has while interacting with a product

* Guided by the aim of aligning the user’s goal for visiting the webpage and the business’s goals for creating it

### Guidelines

#### Usability

1. **Don’t reinvent or use overly complicated layouts**. Use layouts that users are familiar with
2. Make **call-to-action** the **most prominent** element and make the **text descriptive**

**Graphical user interface, application

Description automatically generated**

1. **Animation** should have a **purpose** and be **fast** (ie. 200 – 500ms)
2. In forms, **labels and fields should be aligned** in a single vertical line to make it easier to scan

Graphical user interface, application, Teams

Description automatically generated

1. Offer user with **effective feedbacks** for all actions (eg. Form errors, Form success)
2. **Place user actions in context** where they will create the effect (ie. Law of locality)

A screenshot of a computer

Description automatically generated with medium confidence

#### Website content

1. Keep the **headline descriptive** and **keyword-focused**

Graphical user interface, text, application, Teams

Description automatically generated

1. Only **include relevant information** and **cut out fluff** to keep the content clear and efficient
2. Use **simple and clear words**, avoid technical jargon that might confuse the users
3. **Break up the content** with sub-headings, images, block quotes, bullet points etc

Graphical user interface, text, application

Description automatically generated

## Website personalities framework

Diagram

Description automatically generated

Graphical user interface, application

Description automatically generated

Graphical user interface, application, website

Description automatically generated

Graphical user interface, application, Word

Description automatically generated

Graphical user interface, application

Description automatically generated

Graphical user interface, text, application, website

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Graphical user interface, application

Description automatically generated

### Combining playfulness and boldness in website’s personality

Diagram

Description automatically generated

## Elements, Components, and Layouts

A webpage is assembled using 1 or more layout areas that are combined from components which are composed of elements and smaller components

Graphical user interface, website

Description automatically generated

### Examples (Non-exhaustive)

Graphical user interface, application

Description automatically generated

* For visual examples, refer to the lecture slides

### Notes from Accordion

Graphical user interface, text, application, email

Description automatically generated

* Center an element by setting its horizontal margin to be *auto*
  + Eg. *margin: 100px auto;*
* Insert gaps between the children elements using flexbox
  + Eg. *display:* flex; *| flex-direction: column; | gap: 12px;*
* Add a new class to customize how an element look in another state
  + Eg. Add *open* class and having its styling allow us to style how the element look when it is opened

### Notes from Carousel

Graphical user interface, website

Description automatically generated

* When there is an **absolute positioned element**, there should be **a parent element that is set to positioned relatively**
* ***transform*** – A CSS property that allow us **to rotate, skew, scale or translate a transformable element** (eg. *translate(50%, 10%))*
  + **In relative to the element itself**
  + The transformation does not affect the element’s box model (ie. Transform the appearance but not influencing the document flow)
* Can remove a HTML button’s default bevel border with *border: none*
* *border-radius: WIDTH\_OR\_HEIGHT\_OF\_ELEMENT* to make an element **circular**
* *left/right/top/bottom* properties **position the element’s left/right/top/bottom** to the specified amount of the parent’s container
  + **In relation to the parent element**
* *cursor: pointer* to change the cursor into a finger that is going to click