CSS Advanced!

Q1: Week 5

written by Airielle Randolf D. Dela Cruz

recall CSS:

CSS, or *Cascading Style Sheets*, is the language that we use to style the web!

The basic syntax of CSS involves selecting an element or elements using a `selector`, and defining their respective `properties`.

```
selector {
    property1: value1;
    property2: value2;
    property3: valueA valueB valueC;
}
```

recall CSS properties:

Some common CSS properties are listed below:

- `color` changes the text color of the selected element/s.
- `font-style`, `font-weight`, `font-size`, (etc.), are used to change the font properties of the selected element/s.
- `background-color` and `background-image` affect the background of the selected element/s.
- `margin` adds spacing *around* the selected element/s.
- `padding` adds spacing *within* the selected element/s.

recall CSS selection priority:

CSS properties are prioritized based on how specific the selector is. The more specific the selector, the higher the priority of properties.

For example, properties affecting ID selectors are prioritized over properties affecting class selectors, which themselves are prioritized over element selectors.

recall CSS selector syntax:

- Elements that belong to a class can be selected using: `.<class>`.
- Elements that have an ID can be selected using: `#<id>`
- All elements of the same type can be selected by their `element-name`.

```
.my-class {}
#my-id {}
section {}
footer {}
```

more CSS selector syntax

`*` selects all elements. It is also the least prioritized selector.

```
* {
    font-size: 2em;
}
```

more CSS selector syntax

`element.class` selects only the `elements` that belong in the specified `class`.

```
<p>Hello.</p>
<p class="red-text">Hi.</p>
<div class="red-text">Hey.</p>

<style>
    p.red-text {
        color: red;
    }
</style>
```

Hello.

Hi.

Hey.

more CSS selector syntax

`selector, selector` combines multiple selectors into one line. Properties apply to all.

```
html, p.red-text, #elem {
    color: red;
}
```

more CSS selector syntax

`ancestor element` selects elements that are nested within `ancestor`, no matter how deep.

```
<div>
    <section>
        <section>
            <p>Hi!</p> <!-- p is still technically inside div -->
        </section>
    </section>
</div>
```

```
div p {
    color: red;  /* 'Hi!' will be red */
}
```

more CSS selector syntax

finally, `ancestor > element` selects only the elements that are **direct children** of `ancestor`.

```
<div>
    <p> This will be colored. </p>
    <section>
        <p> This will not. </p>
    </section>
</div>
```

```
div > p {
    color: red;
}
```

truth be told...

remembering super specific CSS syntax is not too necessary. If you know how to select the elements that you want, using just the basics, it's often enough to get by.

```
.this-is-a-class {}

#this-is-an-id {}

this-is-an-element {}
```

let's get into CSS pseudo-classes.

these are not complicated, but can help spice up your HTML.

what ARE pseudo-classes?

CSS pseudo-classes refer to a specific **state** of an element. It could refer to an element specifically while it is hovered, or active, or offscreen, or what not.

what ARE pseudo-classes?

CSS pseudo-classes refer to a specific **state** of an element. It could refer to an element specifically while it is hovered, or active, or offscreen, or what not.

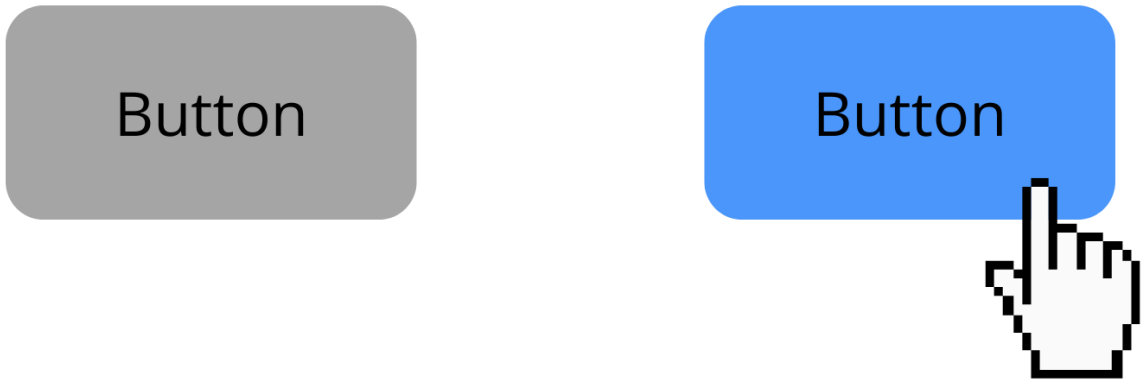
why should i care about pseudo-classes?

pseudo-classes are great because they can give your HTML page a much more dynamic and interactive look.

CSS :hover

`selector:hover` specifies the properties of element/s only while they are being hovered.

```
button:hover {
    background-color: blue;
}
```



CSS :visited and :link

`selector:visited` applies specified properties to visited links.

```
a:visited {
    color: purple;
}
```

`selector:link` applies specified properties to unvisited links.

```
a:link {
    color: blue;
}
```

CSS :first-child and :last-child

Selects the first child and last child of an element, respectively.

```
<div>
    <p>1</p>
    <p>2</p>
    <p>3</p>
</div>
```

```
div:first-child {
    font-size: 20px;
}

div:last-child {
    font-size: 2px;
}
```

CSS Pseudo-elements

Similarly, pseudo-elements specify content within the element itself. It's not really worth going over them right now.

- The `::first-letter` pseudo-element changes the properties of the first letter.
- The `::before` and `::after` pseudo-element changes the properties of the space before and after the element, respectively.

let's try it! (~5 mins.)

- Go to your `my_page.html`.
- In one of the sections in your HTML, there should be a link to a website. If not, add a link to any website of your choice.
- Using `:hover`, `:link`, and `:visited`, make the link look different while it is unvisited, visited, or hovered.
- You can also apply other techniques, pseudo-classes, or even pseudo-elements. Be creative!

If you need any help, don't be afraid to ask!

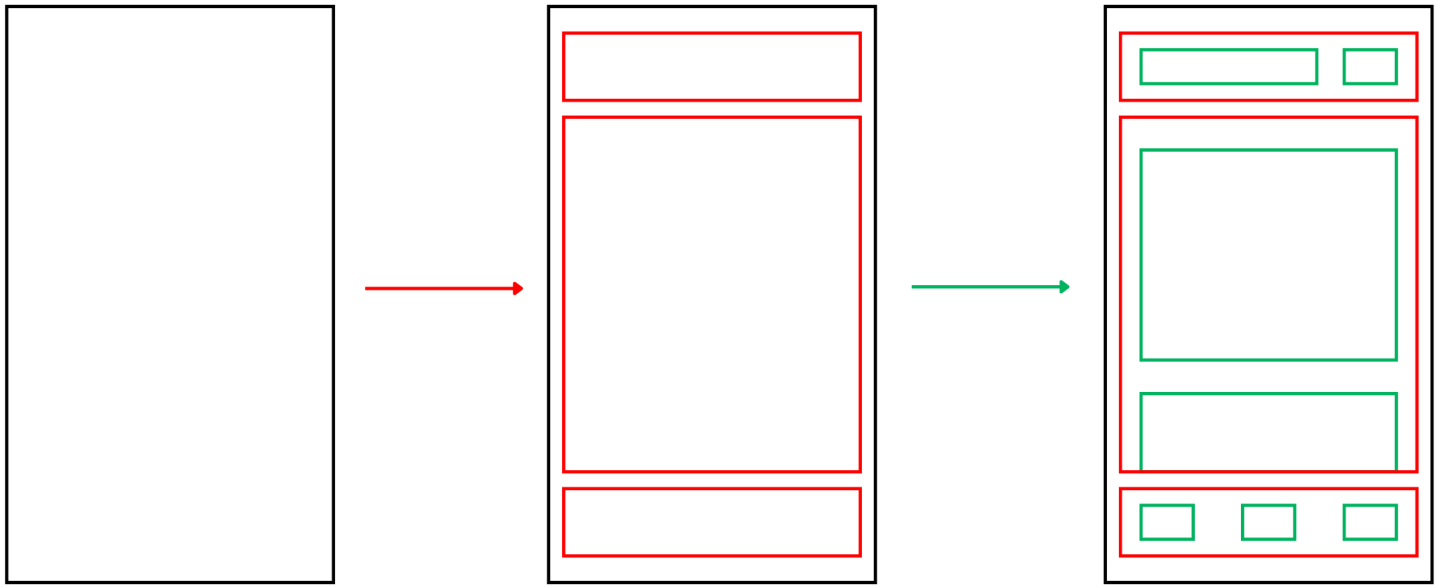
if you're done, take a deep breath...

we have one last CSS topic.

CSS Positioning.

recall HTML hierarchy:

HTML elements can be placed inside other HTML elements, creating an easy-to-deal with hierarchy.



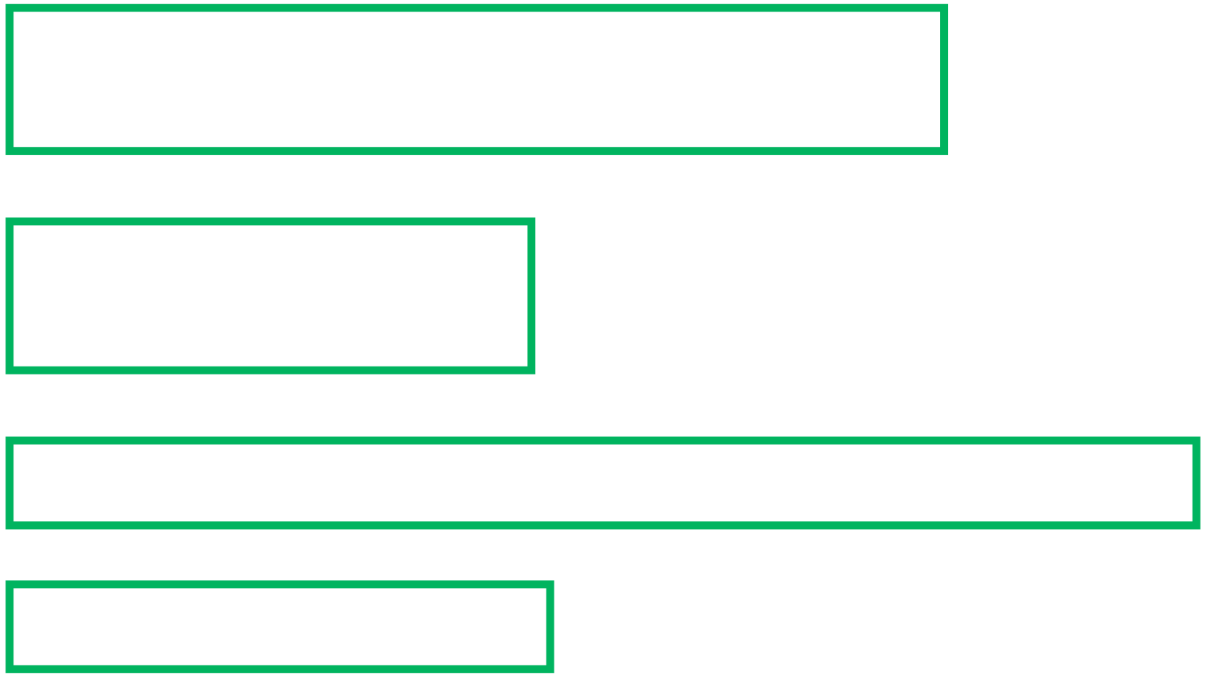
however!

HTML hierarchy *alone* can't give applications and webpages the dynamic layouts that they have. Everything will just be arranged vertically.

Good layouts will depend on **CSS Positioning.**

HTML Flow

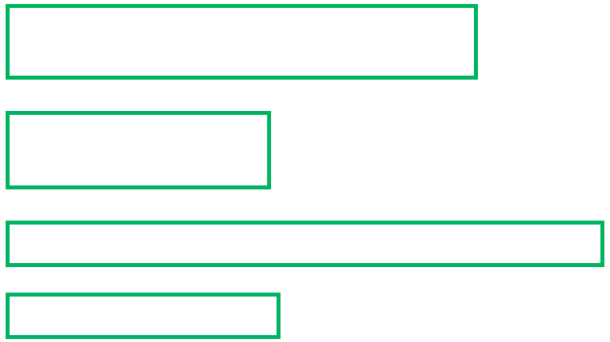
HTML elements follow what's called the **normal flow** of elements. That is, elements in an HTML page arrange themselves left-to-right and top-to-bottom, like text.



CSS static/default position

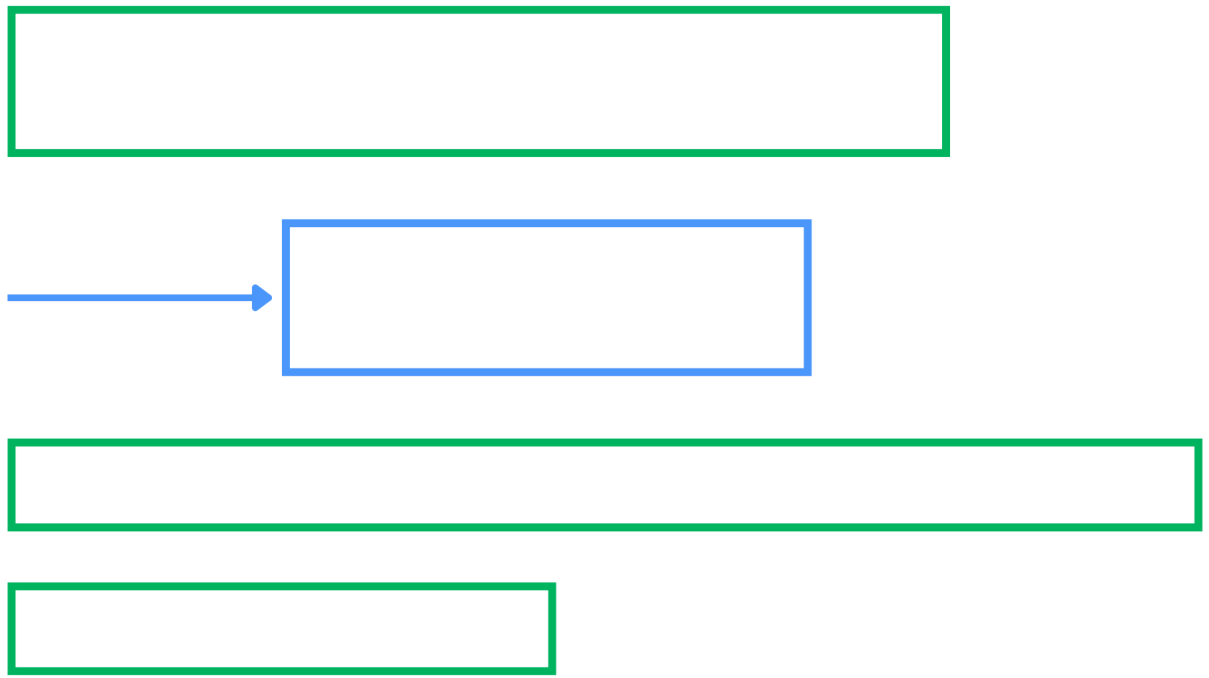
the `position` property of CSS determines the type of position elements will have. It can have multiple values. By default, the value is `position: static`.

Static positioning simply means that the element will follow the normal flow. It is the arrangement of HTML by default.



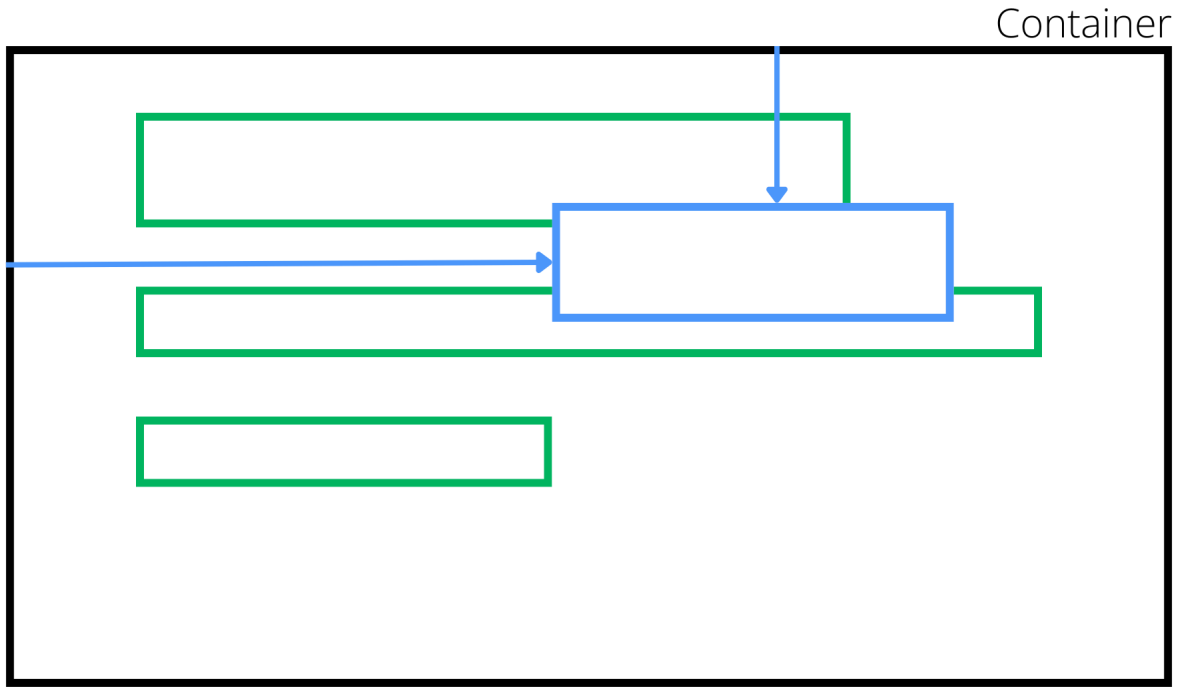
CSS relative position

`position: relative` can shift the position of an element away from its normal flow position.



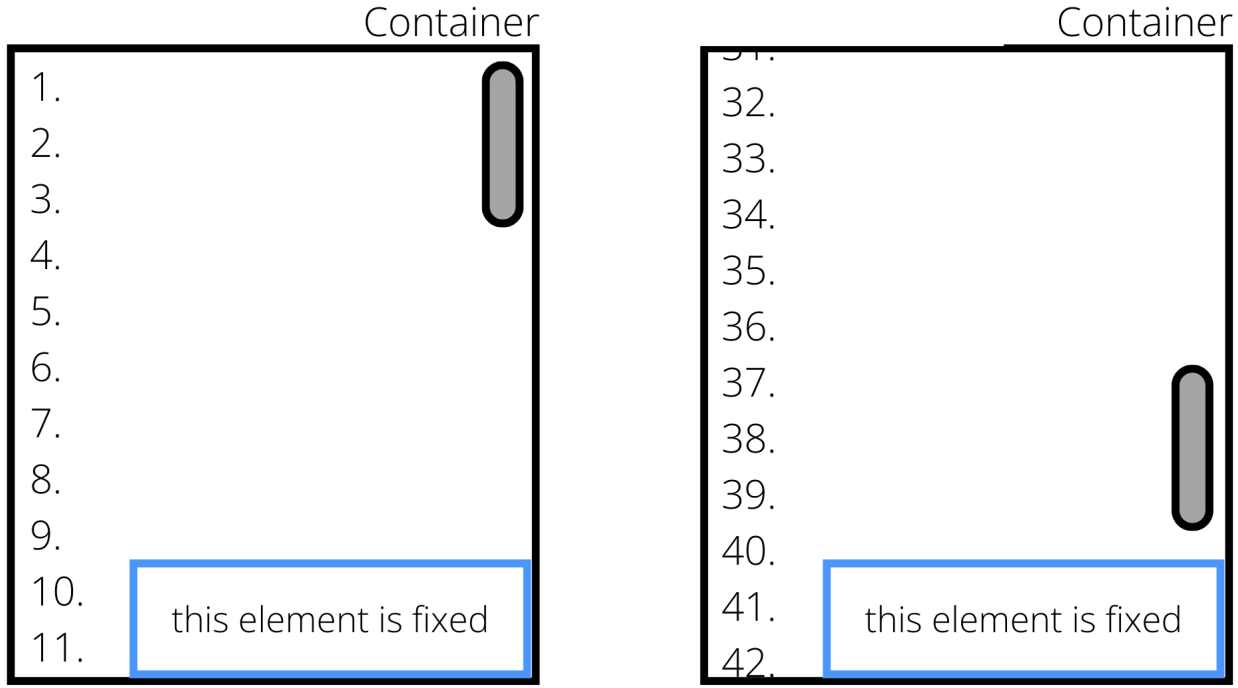
CSS absolute position

`position: absolute`, on the other hand, shifts the position of an element *based on the boundaries of its container* (unless the container has `static` positioning).



CSS fixed position

`position: fixed` keeps an element fixed on the screen based on its container, *regardless of user scroll or window view.*



CSS positioning

the `position` property only specifies the type of position the element will have. To actually move the element, we use `left`, `top`, `right`, and `bottom` properties.

> Note that the directional properties move an element AWAY from that direction, not TOWARDS that direction.

```
section {
    position: relative;
    left: 50px;  /* moves AWAY from left and TO the right */
    bottom: 50px;  /* moves AWAY from bottom and TO the top */
}
```

positions vs. margins. what's the difference?

`margin` adds tangible space around an element. If you were to add a margin to the bottom of an element, all of the elements below will be "pushed" downwards too.

On the other hand, `position` strictly moves the element's position. It does not affect the position of other elements, unless it has `absolute` or `fixed` position, in which case the element's spot in the normal flow is freed and can be taken by other elements.

kaya pa ba? 😅

that should be the finish line for CSS... but...

but the truth is...

this is only the tip of the iceberg when it comes to HTML and CSS. there is still so much more to be learned and discovered!

so, if you find yourself stuck, always remember: there is always Google by your side. you can always freely look up references and guides for these topics. and there's always these Reboot slides available for you!

if it makes you feel better, even professionals forget how to center a .

so, to recap:

CSS Selector Syntax:

- `*` selects all the elements in the page.
- `element.class` selects all elements of type `element` within `class`.
- `ancestor element` selects all elements that are nested within `ancestor`.
- `ancestor > element` selects all elements that are direct children of `ancestor`.

CSS Pseudo-classes:

- `:hover` defines the properties of the selector while hovered.
- `:link` defines the properties of an unvisited link.
- `:visited` defines the properties of a visited link.

so, to recap:

CSS Pseudo-classes:

- `:first-child` and `:last-child` defines the properties of the first and last child of the selector, respectively.

CSS Positioning:

- `position: static` is the default position and refers to elements positioned by HTML's normal flow.
- `position: relative` refers to elements that are being repositioned based on their original position in normal flow.

so, to recap:

CSS Positioning:

- `position: absolute` refers to elements that are being repositioned based on the boundaries of their container / ancestor.
- `position: fixed` refers to elements who have a fixed position on-screen, regardless of user scroll or window view.

Free Style (get it?) (~10-20 mins.)

For your final CSS exercise:

- Go to your `my_page.html`
- With everything that you've learned from HTML and CSS, GO CRAZY!!!!!!
- Reposition everything! Add colors to everything! Add more elements if you want! Have fun!
- The best looking site will be chosen as the **WINNER** *(no JavaScript allowed, for those who are advanced!)*.

If you need help, don't be afraid to ask!