CSS

Loot at the different designs here:

https://www.csszengarden.com/
https://web.archive.org/web/20231010161409/http://csszengarden.com/

The content is the same, just the css is different. This shows how different an html page can look even if the content does not change.

Watch history of web design: https://www.youtube.com/watch?v=XYTwYmOjqQs

Nothing is built overnight, it took time to see the results that we have today, and it is still evolving. At some point Flash was created to a lot more than html.

Flash was discontinued because of security flaws. Html improved a lot to make almost everything that was possible with Flash.

CSS means Cascading Style Sheets

With CSS, you can specify the properties of font, text, spacing, layout, colors, ….

Create a css folder and save there a main.css file.

```
h1 {
        background-color: #b9fcff;
        text-align: center;
        font-size: 20px;
}
```

Load it in the index.html page in the header tag:
<link rel="stylesheet" href="css/main.css">

- h1 is what we called the selector.
- The block {} is called a declaration block.
- font-size: 20px; (or one of the lines inside {}), is called a declaration or a style
- font-size (or text-align, background-color, ….) is called a property, and it has a value related to the property (#b9fcff, center, 20px, …)

The selector + declaration block is called a css rule.

There are inline, internal and external CSS.

Inline CSS is when you specify the css rules directly inside an html element.

```
<h1 style="background-color: #b9fcff">
        <img src="img/movie-theater-icon.webp" width="100" alt="movie theater
image"/>  
```

An inline CSS is going to override any other css rules specified elsewhere.

Internal CSS is when you specify the css rules directly in the header like this:

```
<head>
        <style>
                h1 {
                        background-color: #d4cdff;
                        text-align: center;
                        font-size: 20px;
                }

        </style>
</head>
```

IMPORTANT: Inline CSS has priority over internal CSS.

External CSS is the .css file that we load into our html:

```
<link rel="stylesheet" href="css/main.css">
```

The priority between internal and external css is only defined by the place where it is loaded in the html.

If there is the same property defined several times over different css rules, the one which has been loaded last in the html will have priority over the ones. Only an inline CSS will overwrite any other rules.

In general you want to write your css rule into the external css file. But you know that you have the possibility to edit the css rules into the html directly if in some cases.

VSCodium:

On VSCodium, when you write a property and then :, VSCodium will give you all the options available that go with that property.

Also, you can check on the internet the documentation about the values you can set up, and the definition of the properties, what they do, learn more about them….

Text Style:

color (#ff8000): pick a color for your text
font-family: set up what font the html page wants to use. For example: sans-serif. For the value of font-family, we can specify multiple fonts, the browser is going to use the first it knows in the list.
font-size: change the size. There are various ways to specify the unit, for the moment we can just put pixel (px).
text-align: where you want to set up the element in comparison of the border of the element.
text-transform: with value for example uppercase (it will use only uppercase letters).
font-style: italic

Paragraph Style:

For the p element, where you have multiple lines, you can specify the line height:

line-height: 2 (no need to specify the unit). If we set a 2, it will mean 2 times the font size.

You can select different elements at the same time, and give them the same css rule.

CSS rule for multiple elements

```
h1, p, li {
  font-family: sans-serif
}
```

So, h1, p and li elements will have the same font-family. You only need to define it once, so when you want to change anything, you change it once, no 3 times.

Let set the p with this rule:

```
p{
  line-height:2;
  font-size: 22 px;
}
```

Let's add something in the footer (inside the body, at the end):

```
<footer>
  <p>
    Disclaimer: This website is under development.
  </p>
 </footer>
```

The disclaimer is inside a <p> element.

Above we specify that the font-size for p elements is 22 px.
But for the footer, I want it smaller. To make a rule different for the p, inside the footer, we are going to do:

```
footer p {
        font-size: 10px
}
```

Using a comma between elements, it means it is a list of html elements.

Using space, it is like a hierarchy. footer is the parent element and p the child element.
You can have multiple levels of hierarchy.

Let's add a header inside the body, at the top:

```
<header>
        <h1><img src="img/movie-theater-icon.webp" width="100" alt="movie theater
        image"/>  
          Movie Theater</h1>
        <nav>
                <a href="movie" target="_blank">Movie</a>
         </nav>
        <p>This is a paragraph inside our header</p>
</header>
```

Let's include the header inside an article element. The <article> tag specifies independent, self-contained content.

```
<article>
        <header>
                <p>
                        Welcome to our movie theater. Please be comfortable on your seat, take
        a snack and enjoy the movie
                <p>
        <header>
</article>
```

DO NOW: We want to set up the "Welcome to our movie theater., ….." to be italic

Setting styles in a hierarchical way is annoying sometimes. CSS developers have created a way to insert ids in elements, and apply the rules following the id of the element.

```
<article>
        <header>
                <p id="welcome">
                        Welcome to our movie theater. Please be comfortable on your seat, take
        a snack and enjoy the movie
                <p>
        </header>
</article>
```

We set up the id welcome in the p element.

Let's remove the rule article header p, and let create this one instead:

```
#welcome {
   font-style:italic;
}
```

We add an hashtag in front (#) to indicate it is an object's id

If you want to add comments in a css, you need to add /*... */

```
/*article header p {
   font-style:italic;
}*/
```

A shortcut is to do it is: CTRL + /

DO NOW: let's add an id on the footer p element and use that to set up the CSS rule

If you use the same id on the same page, the browser is going to apply the css rule to all the elements with that id. However, this is invalid (bad practices).

If we need to apply the same rule from an id to multiple elements, you may use a class. In general, we only use classes, because it is reusable. I just showed you the id, because it exists, but nobody uses it, it is better to use a class.

Let's add a class to the bullet point list:

```
<p>We can also have bullet points</p>
 <ul>
   <li class="bullet">First item</li>
   <li class="bullet">Second item</li>
   <li class="bullet">Third</li>
 </ul>
```

The class begins with a . in the css file:

```
.bullet {
   color: #6829b8;
   line-height:2;
}
```

Let's remove the bullet point that is displayed on your browser:

```
li {
  list-style: none;
}
```

We can set it up also like

```
ul {
  list-style: none;
}
```

But I only have one ul, so let's do it on the li.

DO NOW: Create another class .no-bullet that removes the bullet point and use that class on your li elements

Is it possible to apply rules from more than one class to the html elements?

DO NOW:

Try to replicate this "Movies with a lot of colors".
Replace the id="welcome" with a class="welcome".
Do not add any additional classes on the p element nor on the children elements you are going to introduce. You can use the element , it is going to be used to make a part of the text different from the rest.

Colors

Colors can be defined using a **combination of Red, Green and Blue.**
Each base color is associated with a value between 0 and 255.
On a html page, you can use colors like this:

rbg(red_value, green_value, blue_value)

Also, you can add rgb with a transparency ("alpha"):

rgba(red_vlaue, green_value, blue_value, alpha_value)

Alpha value between 0 and 1. With 0, we do not see anything (fully transparent), and with 1, it is like the rgb value.

We also have the **hexadecimal** notation. Instead of going from 0 to 255, we are going to write these numbers as hexadecimal values, so we are going from 0 to ff
https://www.eatyourbytes.com/decimal-binary-and-hexadecimal-numbers-from-0-to-255/

We do that for 3 colors and we add the symbol # in front. If you want to add transparency, you also add 2 characters from 00 to ff.

DO NOW: Find how you would write this #ff91f2 in rgb format and this one with the transparency #ff91f27F

Div element:

The div tag is used as a container for HTML elements.

Let's add a border to an element. We are going to use a div element.

```
<div class="aside">
<p>We can also have bullet points</p>
  <ul>
    <li class="bullet no-bullet">First item</li>
    <li class="bullet no-bullet">Second item</li>
    <li class="bullet no-bullet">Third</li>
  </ul>
 </div>

.aside {
  background-color: #f7f7f7;
  border: 5px solid #b9fcff;
}
```

You may notice that the value of the property border has 3 values: size, style and color. Sometimes you can assign multiple values to a property in one line.

If you want to assign only a top border and bottom border, it is possible:

```
.aside {
  background-color: #f7f7f7;
  border-top: 5px solid #b9fcff;
  border-bottom: 5px solid #b9fcff;
}
```

Let’s make the first element in the list look different. Let’s make it bold => font-weight: bold;

```
.first-li {
  font-weight: bold;
}

<li class="bullet no_bullet first-li">First item</li>
```

Pseudo class:

That works. But we can do it in a different way, let's do it with a pseudo class. A **CSS pseudo-class** is a keyword added to a selector that specifies a special state of the selected element(s).

Let’s it put back the way it was

```
….
<li>First item</li>
….
<li class="bullet no-bullet">First item</li>
….
```

And add this:

```
li:first-child {
  font-weight: bold;
}
```

You can use also li:last-child, or li:nth-child(2) or li:nth-child(even)

Problem with pseudo classes

Let’s have this html code:

```
<div class="aside">
  <span>I am first</span>
  <p>We can also have bullet points</p>
  <ul>
    <li class="bullet no-bullet">First item</li>
    <li class="bullet no-bullet">Second item</li>
    <li class="bullet no-bullet">Third</li>
  </ul>
</div>
```

And then on the css:

```
div p:first-child {
  font-weight: bold;
}
```

You would have expected that the text “We can also have bullet points” will be bold. But no, it does not work this way. div p:first-child means that in all the div elements, when an element in the hierarchy is p, if that p is the first-element in his hierarchy level, then it will be bold.

This, is going to work:

```
<div class="aside">
  <p>We can also have bullet points</p>
  <ul>
    <li class="bullet no-bullet">First item</li>
    <li class="bullet no-bullet">Second item</li>
    <li class="bullet no-bullet">Third</li>
  </ul>
 </div>
```

Give a style to links with pseudo class:

```
a:link {
  color: #b9fcff;
}

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

You can also defined the style for a:hover, a:active

DO NOW: Apply styles a:hover, a:active

You know that you can check your styling with the dev tool. Open your dev tool, select an element, and on the right, you can see the style that has been applied on the element, and where is the source of the style.

You can edit the style directly from there if you would like, to test it in live, without modifying the html or css files. Once you reload the page, you will lose the modification you made though.

You can also unselect some of the style (just uncheck the checkbox)

You can also see there a :hov, if you click on it, you can select the behavior of your pseudo class styling selecting the state of the element.

CSS Conflicts:

What about conflicts when you have the same property rule (with different value) defined for an element, an id and a class (or multiple classes, remember we can have multiple classes for an object element).

I know we do not use id anymore, but it is there, so you know in case you see a css with an id at some point.

For example we have this:

My second paragraph. The browser sets up some space between 2 paragraphs.

With these css rules

```
p{
  line-height:2;
  font-size: 22px;
}

#conflict_id {
  font-size: 30px;
}

.conflict {
  font-size: 10px;
}
```

Go to your dev tool on the element object “My second paragraph….”
What does it say first? The rule for the element p selector, the class or the id?

What style goes second? And which one third?

The browser takes the one with higher priority (id) and then discards any other with the same property.
If you comment the id rule on the css file (select the rule and then do ctrl + /, it is going to automatically comment it)

Then, the rule for the class is applied.
What happens if we duplicate a rule with different values?

```
p{
  line-height:2;
  font-size: 22px;
}

.conflict {
  font-size: 10px;
  color: blue;
}


.conflict {
  font-size: 15px;
}
```

Yes, higher priority is the one who has been stated last. The first class .conflict has a color, and that property works because it is not specified in the second .conflict class. The rule for the blue comes in second place, but because the first rule does not specify anything for the color, the color blue is applied in the second one.

What about inheritance?

An html element, it will inherit the rule from its parent (and then its grandparent ... .and so on) unless there is a specific rule for a certain html element as we learned before.

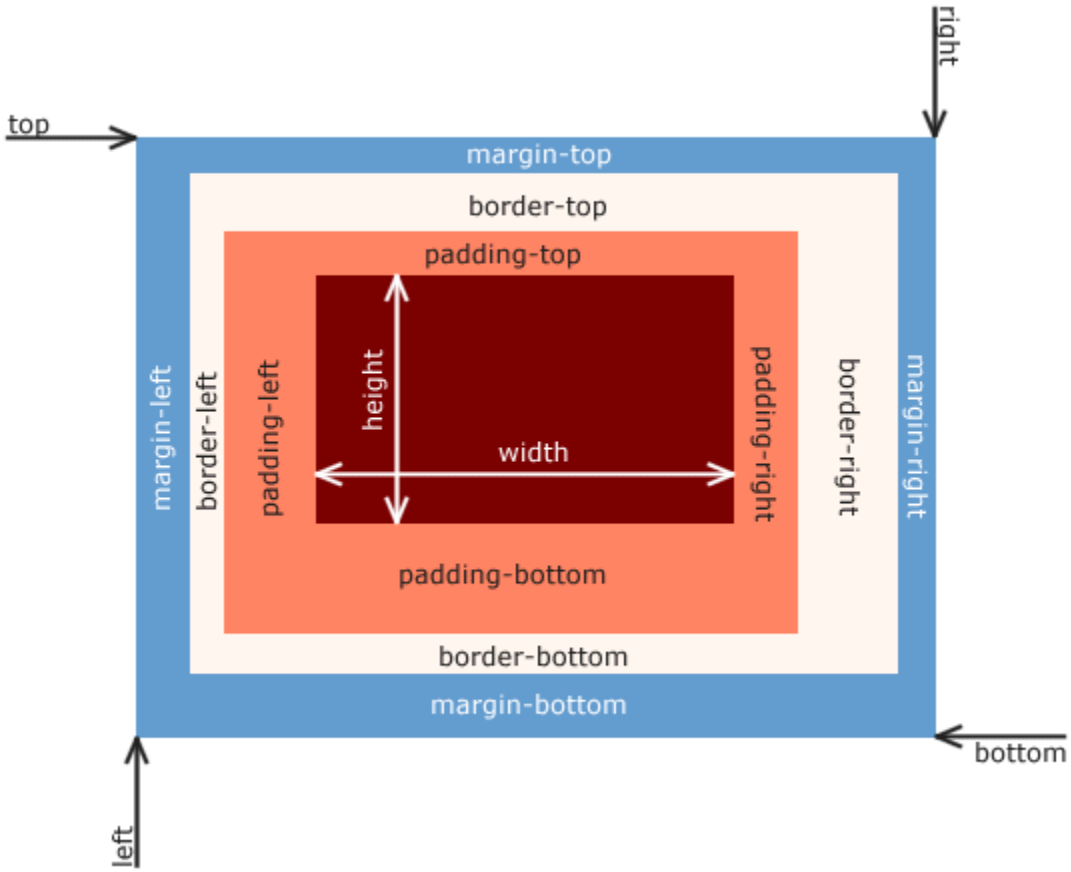
If you remove all the font-size property you have for a p selector in your css and add this:

```
body {
  font-size: 20px;
}
```

On your dev tool, go over the first element p, on the style section, you see at the bottom “Inherited from body” and the font-size that has been applied.
Of course, not all properties are going through inheritance. If in the body selection, you add border-top, you are not going to see a border-top on all its children and grandchildren ….
Otherwise, it is going to be annoying.

It is mostly the ones related to text that are going to be inherited: font-family, font-size, font-weight, font-style, color, line-height, letter-spacing, text-align, text-transform, text-shadow, list-style, ….

CSS Model Box



This is the model box in CSS.

The **content** (text, images, …) are in the **red box.** You can specify the height and width of those elements (we will see that later).

Around the content, you can have a **border** (that you can see). And you can define a space, called **padding, between the content and the border.**

Content, padding and border are the space inside the html element.

And then you have the **margin**, which is a space that is **outside the html element.**

We have not defined any of these parameters yet (except the border). When you do not specify anything, it is going to get the default values.

Go to the dev tool, pick an element, and on the right side, you can see the model box on the layout tab.

If you pick a p, ul or ol element, you can see the default values are the same: 20px for top and bottom margins, 0 for left and right margins, and for the padding all around. For ol and ul, it has 40 for padding left, this is the space for digits or bullets. These are the default values for each element. If you check a div, the default value for margin all around is 0.

If you apply something in the background (like an image or a color), it is going to fill all the space of the element: content + padding.

So,

- **Final element width** is: left border + left padding + width + right padding + right border
- **Final element height:** top border + top padding + width + bottom padding + bottom border

Also, you can notice that by changing the width of your browser, the width of the content is modified. For the moment, our elements are taking all the width available.

Let's go over our css class aside, in our html. We have our bullet points inside (but we do not see them though).

We left our class like this:

```
.aside {
  background-color: #f7f7f7;
  border-top: 5px solid #b9fcff;
  border-bottom: 5px solid #b9fcff;
}
```

Let's add a padding of 20px

```
.aside {
  background-color: #f7f7f7;
  border-top: 5px solid #b9fcff;
  border-bottom: 5px solid #b9fcff;
  padding: 20px;
}
```

Same as the border, if we do not specify the side, it is going to be the value for the 4 sides (left, top, right, bottom). You see the content inside moved. The border (left, top, right) did not move, and the bottom border moved because the space inside the element is now bigger.
Now, there is more space between the content and the border.

There is shortcut if you want to put different padding for the sides:
padding: 25px 50px 75px 100px;

https://www.w3schools.com/css/css_padding.asp

If the `padding` property has four values:

- **padding: 25px 50px 75px 100px;**
  - top padding is 25px
  - right padding is 50px
  - bottom padding is 75px
  - left padding is 100px

If the `padding` property has three values:

- **padding: 25px 50px 75px;**
  - top padding is 25px
  - right and left paddings are 50px
  - bottom padding is 75px

If the `padding` property has two values:

- **padding: 25px 50px;**
  - top and bottom paddings are 25px
  - right and left paddings are 50px

If the `padding` property has one value:

- **padding: 25px;**
  - all four paddings are 25px

Let's take the class bullet, and remove the line-height we had before.

```
.bullet {
  color: #6829b8;
}
```

This time we are going to add a margin, to have space between elements.

```
.bullet {
  color: #6829b8;
  margin: 20px;
}
```

In the dev tool, let's inspect the element. We see the margin for each line in yellow. You see that the yellow of the margin-bottom is superposing on top of the margin-top of the following line. With padding, you are not going to see that. Paddings are going to request their own space. Margins only say they need to have this space free so that other element margins can also use it.

In the div with the class aside, let's set the width.

```
.aside {
  background-color: #f7f7f7;
  border-top: 5px solid #b9fcff;
  border-bottom: 5px solid #b9fcff;
  width: 800px;
}
```

With a very long sentence inside:

```
<a href="https://www.google.com/maps">google map</a>
  <p>We can also have bullet points with long sentences, long super long, so that the browser will display in multiple lines</p>
  <ul>
```

At this moment the browser will set the width and will not make it smaller even if you set the browser with a smaller width.

Let constrain the height also

```
.aside {
  background-color: #f7f7f7;
  border-top: 5px solid #b9fcff;
  border-bottom: 5px solid #b9fcff;
  width: 800px;
  height: 100px;
}
```

It is screwing everything up. The border will be set exactly where it is, the content is going to be displayed anyway. But this element and the ones below will be superposing. So, be careful when you specify the width and height of your element.

Let’s remove the height because it is messing up our web page.

Do not forget when we specify the width in the css, it is going to be the size of the content and not the width of the whole element.

You can also set the width or height with a percentage.
Let's try this over the image (just to test it):

```
.picture_test {
  width: 10%
}
```

```
<header>
    <h1><img class="picture_test" src="img/movie-theater-icon.webp" alt="movie theater
image"/>  
      Movie Theater</h1>
```

When you are going to change the width of the browser, the picture will change its width too (10% of the current width of the browser)

Center the page

Let's create a div as the first element of the body, and we put everything inside, with a class called container.

```
body>
  <div class="container">
….
  </div>
</body>
```

It does not look good to have a web page where the content is too wide, unless you really need it. Let's set a width of 800px:

```
.container {
  width: 800px;
}
```

Our eyes are not good at looking at things horizontally. Instead, a vertical configuration is better for our eyes as the newspaper configuration. Check The Spectator (stuy newspaper).

Now, let's center the content on our webpage. Setting up the margin on the left and right as auto will make the browser calculate the same margin on the left and right.

```
.container {
  width: 800px;
  margin: 0 auto;
}
```

And voila, it is done!

Type of boxes

All the html elements do not behave the same in terms of spacing.

Block-level boxes: Div and p tag elements are going to take all the width they are allowed to. These elements take the width of the parent element no matter what.

Inline boxes: If you check elements like a (hyperlink) tag, they are just going to take the space they need for their content.

Because block-level boxes take up all the space horizontally, you cannot have two block boxes side to side. While, with Inlines boxes, yes, you will be able to fit it side by side if the total content of the 2 inlines boxes is smaller than the width of the parent element.

Example of block-element: body, main, header, footer, section, nav, aside, div, h1-6, p, ul, ol, li, ….
=> display: block (see dev tool, layout tab, on the bottom)
=> the box model applies as we saw it previously

Example of inline elements: a, img, strong, em, button, …
=> display: inline (you can check dev tool)
=> height and width does not apply for these elements in the box model (you may try it, you will see that the browser does not care about it). Paddings and margins only apply horizontally.

Example: Take an a element and add a padding and a margin:
<a href="https://www.google.com/maps"
style="padding:20px;margin:20px;height:200px;width:600px">google map

As you can see, margin and padding have no effect vertically. Only horizontally.
Also, the height and width have no effect, in the box model you can see the height and width are defined exclusively by the content.

Convert inline into block:

We can convert an <a> element (inline display), into a block display.
<a href="https://www.google.com/maps"
style="padding:20px;margin:20px;height:200px;width:600px**;display: block**">google map

Let’s put it back as it was before the changes:
google map

Convert block into inline:

<p style="display:inline">

```
    This is a paragraph. <b>This is bold text using the b element. </b>.
    <strong>This is with the strong element. </strong><br>
    <em>This is an italic text.</em>
</p>

  <p id="conflict_id" class="conflict" style="display:inline">
        My second paragraph. The browser set up a little bit of space between 2 paragraphs
  </p>
```

We converted the 2 paragraphs into inline elements. We see the second paragraph is next to the previous one (horizontal level).

Inline-block Box:

The inline-block box looks like inline from the outside, but behaves like block-level on the inside.
=> as a inline-block: occupies only content's space and cause no line-breaks
=> as a block-level box: box-model applies as it is shown
You can set it up as display: inline-block

It takes the best of both worlds (inline and block).

```
<a href="https://www.google.com" target="_blank" style="display:
inline-block;padding-top:10px">Link to google</a>
```

With the inline-block, we will keep the inline behavior: a element will be next of the previous horizontal element (if it is not a block-level element). Also, you will be able to give it all the properties of the box-model (padding and margin all around, and height and width work).

That's it for the boxes!!!

Positioning

The normal flow of element positioning is the css rule => **position: relative**
Elements are displayed according to their order in the html code, taking into consideration their box type. We say that the elements in this position mode, are "in flow"

You can also set up an element with an **absolute position**.
An element will be able to overlap other "in flow" elements, and you use the top, bottom, left, or right edge to position exactly the element on the web page.

Css rule - position: absolute
Let's create an element and set it up with absolute positioning.

Let's write this at the end of the body, before the footer:

```
<img src="img/stuy_logo.png"/>
```

This is normal positioning, it is “in flow”.

```
<img src="img/stuy_logo.png" style="position:absolute;top:50px;left:50px;"/>
```
The image is close to the top-left corner now. If you adjust the width of the browser to be smaller, the image is going to overlap other elements of the page.
Let's set it up on the bottom right.

```
<img src="img/stuy_logo.png" style="position:absolute;bottom:50px;right:50px;"/>
```

If you can scroll down, you can see that the picture is not exactly at 50px bottom right.

On the css file, we need to set up the body to have position:relative. It **makes absolute positioning relative to the entire body element (including off-screen parts), rather than just the initially visible part of the page**. See the definition of the initial containing block in the CSS specification.

```
body {
  font-size: 20px;
  position:relative
}
```

Now the picture is really at 50px bottom right (relative to the body, you will see in the next step why the element closer in the parenthood which have the position:relative is important)

If you insert the img element inside the container element, and you set up the css for the container with position:relative, the picture will be at 50px bottom right relative to the container element.

```
.container {
  width: 800px;
  margin: 0 auto;
  position:relative
}
```

If you want an **element to be always visible** even when you scroll down/up, use the value **position:fixed.**

```
<img src="img/stuy_logo.png" style="position:fixed;bottom:50px;right:50px;"/>
```

Pseudo elements

Do you remember the pseudo classes? Pseudo classes focus on a css rule of the whole element, while pseudo elements are going to focus on a part of the element.

We have used this:

h1::first-letter or p::first-line,

For example we can do this:

<h2>I like to watch movies.</h2> (to put inside the html code)

```
h2::after{
  content: "Yes really"
}
```

You see you are going to see Yes really right after “I like seeing movies”... yeah css is funny.

::after means after the content that has been specified, apply the following css rules.
Let's change the format a little bit to make it look very different.

```
h2 {
  position:relative;
}

h2::after{
  content: "Yes really";
  background-color: yellow;
  color: blue;
  font-size: 16px;
  font-weight: bold;
  display: inline-block;
  padding: 5px;
  position: absolute;
  top: -5px;
  right:-5px
}
```

In this example, you can see many of the properties we have learned.
It has position:relative, position: absolute, display: inline-block, padding.
You see you can define absolute positioning with negative value.