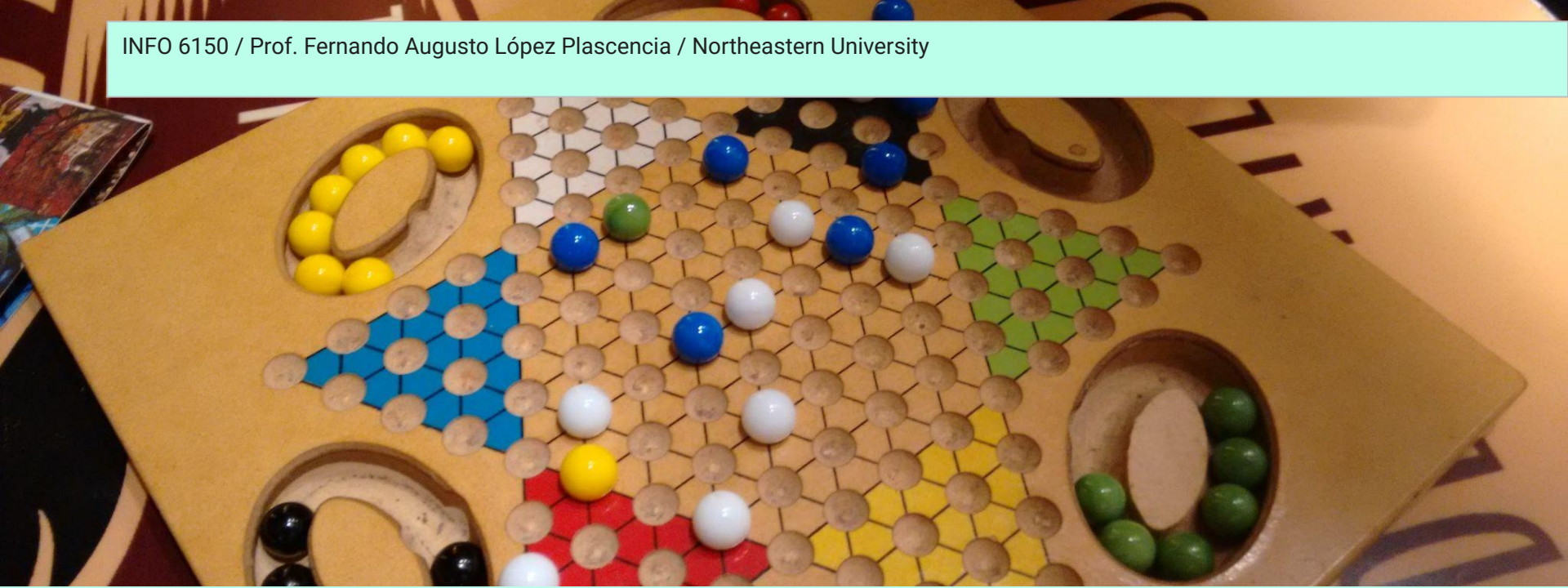


Week 4

# CSS Basics

# The website development process

- Discovery and planning
  - What is the problem?
  - What are we doing to address it?
- Development
  - Actually doing the things
- Deployment
  - Shipping
- Maintenance
  - Keeping up and measuring success

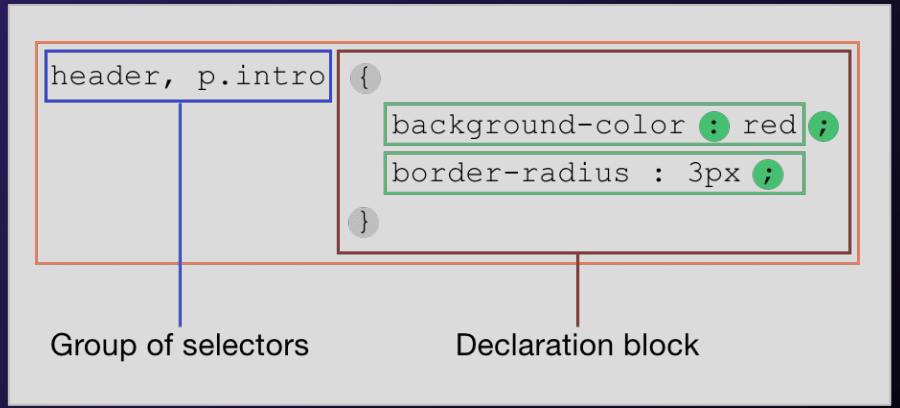
# In this lesson:

- The idea behind CSS
- Selectors
- Specificity
- Pseudo-selectors
- CSS units and how / where to use them
- CSS colors
- CSS typography and typography stacks
- The default display modes: block, inline, inline-block
- The box model

# CSS

Each declaration contains a **property** and a **value.**

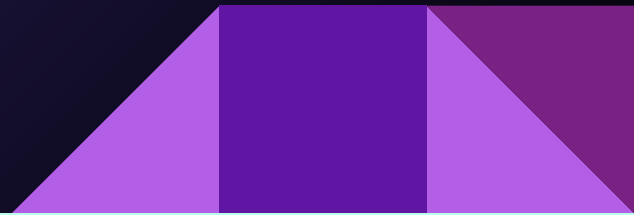
Declarations go inside **blocks -** curly braces {} . Before each block, we specify **selectors**, which refer to HTML elements.

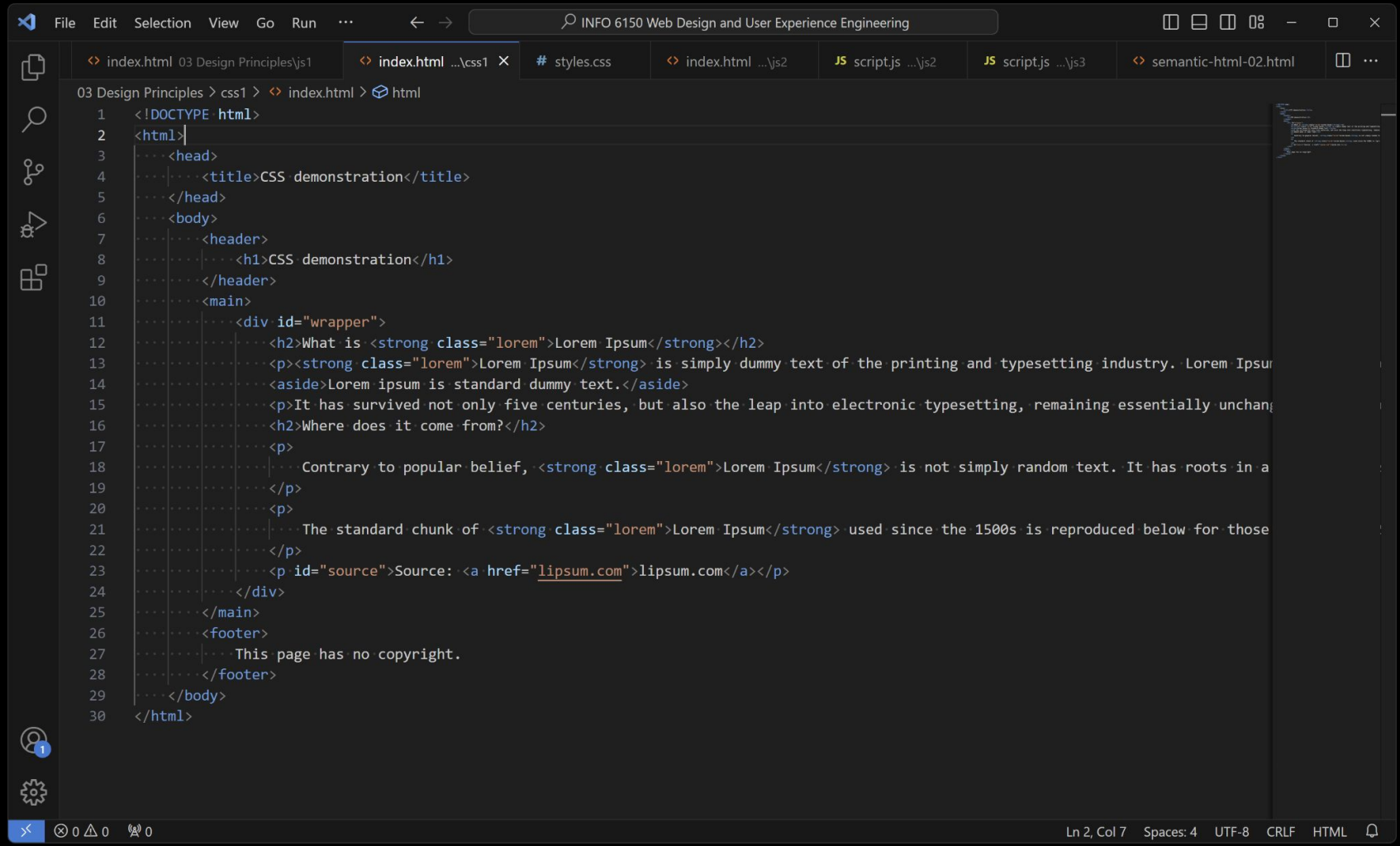


# The idea behind CSS

CSS: Cascading Style Sheets

- The styles of the parent elements cascade (fall) down to their children
- If two rules conflict, the last one in the sheet is the one that applies




File Edit Selection View Go Run
INFO 6150 Web Design and User Experience Engineering
index.html 03 Design Principles\js1
index.html ...\css1
styles.css
index.html ...\js2
script.js ...\js2
script.js ...\js3
semantic-html-02.html
03 Design Principles > css1 > index.html > html
<!DOCTYPE html>
<html>
<head>
<title>CSS demonstration</title>
</head>
<body>
<header>
<h1>CSS demonstration</h1>
</header>
<main>

<h2>What is Lorem Ipsum</h2>
Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsu
<aside>Lorem ipsum is standard dummy text.</aside>
It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchan
<h2>Where does it come from?</h2>

Contrary to popular belief, Lorem Ipsum is not simply random text. It has roots in a

The standard chunk of Lorem Ipsum used since the 1500s is reproduced below for those

Source: <a href="lipsum.com">lipsum.com</a>

</main>
<footer>
This page has no copyright.
</footer>
</body>
</html>
Ln 2, Col 7
Spaces: 4
UTF-8
CRLF
HTML

# CSS demonstration

## What is Lorem Ipsum

**Lorem Ipsum** is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book.

Lorem ipsum is standard dummy text.

It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.

## Where does it come from?

Contrary to popular belief, **Lorem Ipsum** is not simply random text. It has roots in a piece of classical Latin literature from 45 BC, making it over 2000 years old. Richard McClintock, a Latin professor at Hampden-Sydney College in Virginia, looked up one of the more obscure Latin words, consectetur, from a Lorem Ipsum passage, and going through the cites of the word in classical literature, discovered the undoubtable source. Lorem Ipsum comes from sections 1.10.32 and 1.10.33 of "de Finibus Bonorum et Malorum" (The Extremes of Good and Evil) by Cicero, written in 45 BC. This book is a treatise on the theory of ethics, very popular during the Renaissance. The first line of Lorem Ipsum, "Lorem ipsum dolor sit amet..", comes from a line in section 1.10.32.

The standard chunk of **Lorem Ipsum** used since the 1500s is reproduced below for those interested. Sections 1.10.32 and 1.10.33 from "de Finibus Bonorum et Malorum" by Cicero are also reproduced in their exact original form, accompanied by English versions from the 1914 translation by H. Rackham.

Source: lipsum.com

# Let’s review the basics

```
p {
    color: blue;
}

footer {
    font-size: 0.8rem;
}

main {
    width: 100%;
}

/* Everything with the class "lorem" */
.lorem {
    color: red;
}
```

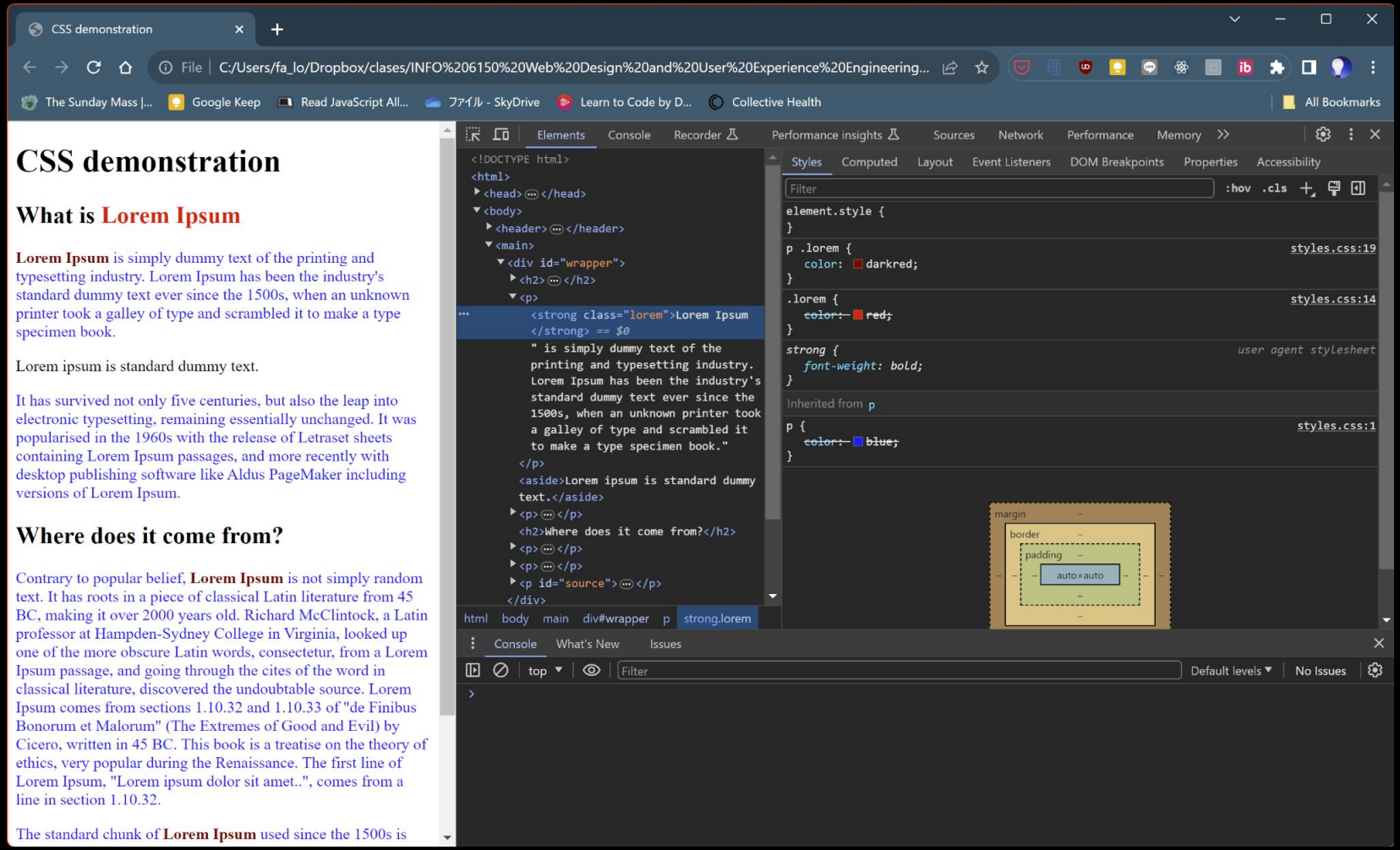
```
/* tags with lorem class, inside
paragraphs */
p .lorem {
    color: darkred;
}

/* a tag with the id "btn-contact".
   Ids should not repeat within a document
*/
#btn-contact {
    color: darkblue;
}
```

# Linking the css file

Use the link tag:

```
    <head>
        <title>CSS demonstration</title>
        <link rel="stylesheet" type="text/css" href="styles.css" />
    </head>
```

(also possible: write css directly inside the link tag. Not recommended though)


CSS demonstration
File | C:/Users/fa_lo/Dropbox/clases/INFO%206150%20Web%20Design%20and%20User%20Experience%20Engineering...
The Sunday Mass |...
Google Keep
Read JavaScript All...
ファイル - SkyDrive
Learn to Code by D...
Collective Health
All Bookmarks
CSS demonstration
What is Lorem Ipsum
Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book.
Lorem ipsum is standard dummy text.
It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.
Where does it come from?
Contrary to popular belief, Lorem Ipsum is not simply random text. It has roots in a piece of classical Latin literature from 45 BC, making it over 2000 years old. Richard McClintock, a Latin professor at Hampden-Sydney College in Virginia, looked up one of the more obscure Latin words, consectetur, from a Lorem Ipsum passage, and going through the cites of the word in classical literature, discovered the undoubtable source. Lorem Ipsum comes from sections 1.10.32 and 1.10.33 of "de Finibus Bonorum et Malorum" (The Extremes of Good and Evil) by Cicero, written in 45 BC. This book is a treatise on the theory of ethics, very popular during the Renaissance. The first line of Lorem Ipsum, "Lorem ipsum dolor sit amet..", comes from a line in section 1.10.32.
The standard chunk of Lorem Ipsum used since the 1500s is
Elements
Console
Recorder
Performance insights
Sources
Network
Performance
Memory
<!DOCTYPE html>
<html>
<head></head>
<body>
<header></header>
<main>

<h2></h2>

Lorem Ipsum == $0
" is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book."

<aside>Lorem ipsum is standard dummy text.</aside>

<h2>Where does it come from?</h2>

html body main div#wrapper p strong.lorem
Styles
Computed
Layout
Event Listeners
DOM Breakpoints
Properties
Accessibility
Filter
:hov .cls
element.style {
}
p .lorem {
color: darkred;
}
styles.css:19
.lorem {
color: red;
}
styles.css:14
strong {
font-weight: bold;
}
user agent stylesheet
Inherited from p
p {
color: blue;
}
styles.css:1
margin
border
padding
auto × auto
Console
What's New
Issues
top
Filter
Default levels
No Issues

# Cascading and specificity

What happens if two rules contradict themselves?

Two sets of rules govern the order of processing of the rules: cascading and specificity.

In general:

- User styles are more important than default styles.
- More **specific** rules have priority over more general rules.

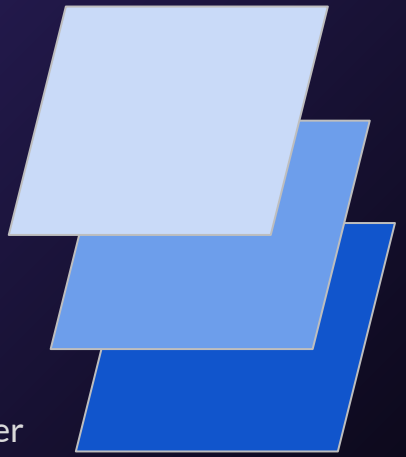
# Cascading

First, the browser will consider where the rules are coming from.

More important:
Rules set by the end user
(possible, but not common)

Next:
Rules set by the developer

Last:
Default rules set by the browser



# Cascading

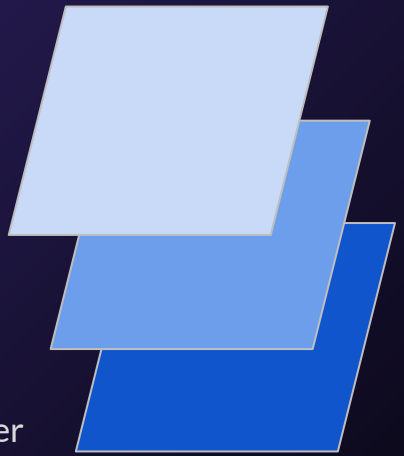
First, the browser will consider where the rules are coming from.

At this point, rules that do not apply are also discarded (eg. @media rules that apply only to print, or other window widths)

More important:
Rules set by the end user
(possible, but not common)

Next:
Rules set by the developer

Last:
Default rules set by the browser



# Specificity

If cascading rules are not enough to discern which styles go first, we use specificity.

Consider this p tag:

```
<html>
    <body>
        <main>
            <p id="unique" class="big green">Special text</p>
```

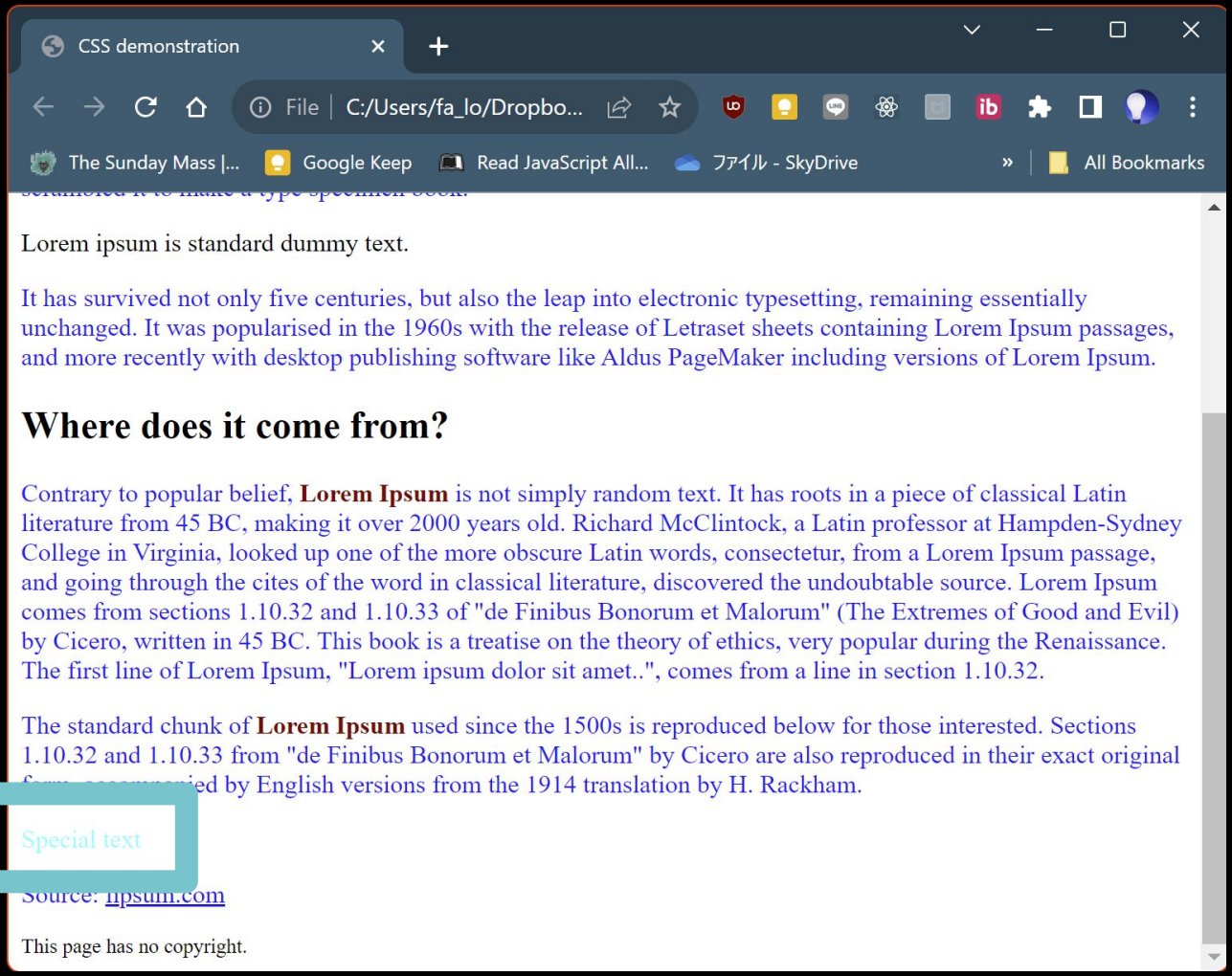
# Specificity

A tag can have two or more classes. In this case, both rulesets apply with the same priority.

```
<html>
    <body>
        <main>
            <p id="unique" class="big green">Special text</p>

/* both apply */
.big { font-size: 3rem; }
.green { color: green; }
```

# Specificity

```
/* in order of specificity. If we were to invert the order, the final color would still be cyan. */
p {
    color: blue;
}

.most {
    color: green;
}

p.most {
    color: yellow;
}

#unique {
    color: gray;
}

p#unique {
    color: cyan;
}
```




Lorem ipsum is standard dummy text.

It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.

## Where does it come from?

Contrary to popular belief, **Lorem Ipsum** is not simply random text. It has roots in a piece of classical Latin literature from 45 BC, making it over 2000 years old. Richard McClintock, a Latin professor at Hampden-Sydney College in Virginia, looked up one of the more obscure Latin words, consectetur, from a Lorem Ipsum passage, and going through the cites of the word in classical literature, discovered the undoubtable source. Lorem Ipsum comes from sections 1.10.32 and 1.10.33 of "de Finibus Bonorum et Malorum" (The Extremes of Good and Evil) by Cicero, written in 45 BC. This book is a treatise on the theory of ethics, very popular during the Renaissance. The first line of Lorem Ipsum, "Lorem ipsum dolor sit amet..", comes from a line in section 1.10.32.

The standard chunk of **Lorem Ipsum** used since the 1500s is reproduced below for those interested. Sections 1.10.32 and 1.10.33 from "de Finibus Bonorum et Malorum" by Cicero are also reproduced in their exact original

Special text

Source: lipsum.com

# More specific

Use the style attribute in the html tag:

```
<p id="unique" class="most" style="color:darkgreen">Special text</p>
```

Note: this is not recommended. Use VERY sparingly.

# Even more specific!?

This trumps **everything else**. Use VERY SPARINGLY

```
#unique {
    color: purple !important;
}
```

# Even more specific!?

`#important` is so strong, it actually acts on the cascading!

#important on default rules (almost not used)

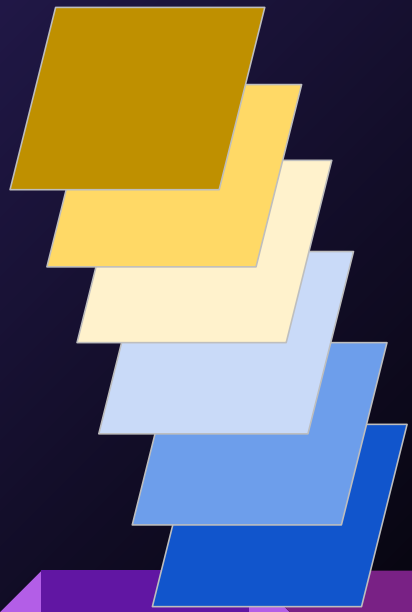
#important by the developer

#important by the end user

Rules set by the end user

Rules set by the developer

Default rules set by the browser



# Calculating specificities

To calculate the specificity of a complex selector, use the following table:

| | Number of #id | Number of .class | Number of tag type |
|---|---|---|---|
| Weight | | | |

The resulting combination of numbers can be used to compare with other rules.

# Calculating specificities

Example:

```
div.body section.first [id=”btn-subscribe”]
```

# Calculating specificities

Example:

```
div.body section.first [id=”btn-subscribe”]
```

**Attribute selector:**
Selects tags that have a specific attribute and value.

Another example: `input [type=”text”]` to select all `<input type=”text”>`

# Calculating specificities

Example:

```
div.body section.first [id=”btn-subscribe”]
```

| | Number of #id | Number of .class | Number of tag type |
|---|---|---|---|
| Weight | 0 | | |

There are no #id in the selector.

# Calculating specificities

Attribute selectors have the same weight as a .class

Example:

```
div.body section.first [id=”btn-subscribe”]
```

| | Number of #id | Number of .class | Number of tag type |
|---|---|---|---|
| Weight | 0 | 3 | |

# Calculating specificities

Example:

```
div.body section.first [id=”btn-subscribe”]
```

| | Number of #id | Number of .class | Number of tag type |
|---|---|---|---|
| Weight | 0 | 3 | 2 |

# Calculating specificities

Example:

```
div.body section.first [id=”btn-subscribe”]
```

| | Number of #id | Number of .class | Number of tag type |
|---|---|---|---|
| Weight | 0 | 3 | 2 |

So the final weight is 0-3-2. (You can think of this as “32”).

# Calculating specificities

Consider this:

```
#btn-subscribe {
   color: green;
}

div.body section.first [id=”btn-subscribe”] {
   color: red;
}
```

What is the `color` of the button?

# Calculating specificities

Consider this:

```
#btn-subscribe {
   color: green;
}

div.body section.first [id=”btn-subscribe”] {
   color: red;
}
```

What is the `color` of the button?

We calculated this specificity: it’s **0-3-2**

# Calculating specificities

Let’s calculate the specificity of `#btn-subscribe:`

| | Number of #id | Number of .class | Number of tag type |
|---|---|---|---|
| Weight | 1 | 0 | 0 |

The weight is 1-0-0.

# Calculating specificities

So we have:

```
#btn-subscribe {  /* 1-0-0: wins! */
   color: green;
}

div.body section.first [id=”btn-subscribe”] { /* 0-3-2 */
   color: red;
}
```

The color would be green.

More about specificities:
https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_cascade/Specificity

# A special case: inline styles

If a rule is set in the style attribute of an HTML tag, it gains a specificity level. Example:

```
<button id="btn-subscribe" style="color: yellow">Subscribe</button>
```

Here, you can think of the inline rule as having the weight 1-0-0-0. The color would be yellow, no matter what the CSS says.

(The only exception to this is using `#important`, since that applies to cascading.)

# More selectors: combinators

```
/* combinators */

/* The child combinator: >
   This means: all "a" tags that are direct
children of a "p" tag.
   For example, <p><a /><p>,
   but not <p><span><a /></span></p> */
p > a {
    font-size: 12px;
}

/* The next sibling combinator: +
   This means: the p tags that come right after
an img tag
   For example: <img /> <p></p>
   but not <img /> <span></span> <p></p>*/
img + p {
    border-top: 1px solid blue;
}
```

```
/* The subsequent sibling combinator: ~
   This means: the p tags after an img tag,
even if not immediately after it
   so both <img /> <p></p>
   and <img /> <span></span> <p></p>*/
img ~ p {
    border-bottom: 1px solid blue;
}
```

Hampden-Sydney College in Virginia, looked up one of the more obscure Latin words, consectetur, from a Lorem Ipsum passage, and going through the cites of the word in classical literature, discovered the undoubtable source. Lorem Ipsum comes from sections 1.10.32 and 1.10.33 of "de Finibus Bonorum et Malorum" (The Extremes of Good and Evil) by Cicero, written in 45 BC. This book is a treatise on the theory of ethics, very popular during the Renaissance. The first line of Lorem Ipsum, "Lorem ipsum dolor sit amet..", comes from a line in section 1.10.32.

The standard chunk of **Lorem Ipsum** used since the 1500s is reproduced below for those interested. Sections 1.10.32 and 1.10.33 from "de Finibus Bonorum et Malorum" by Cicero are also reproduced in their exact original form, accompanied by English versions from the 1914 translation by H. Rackham.

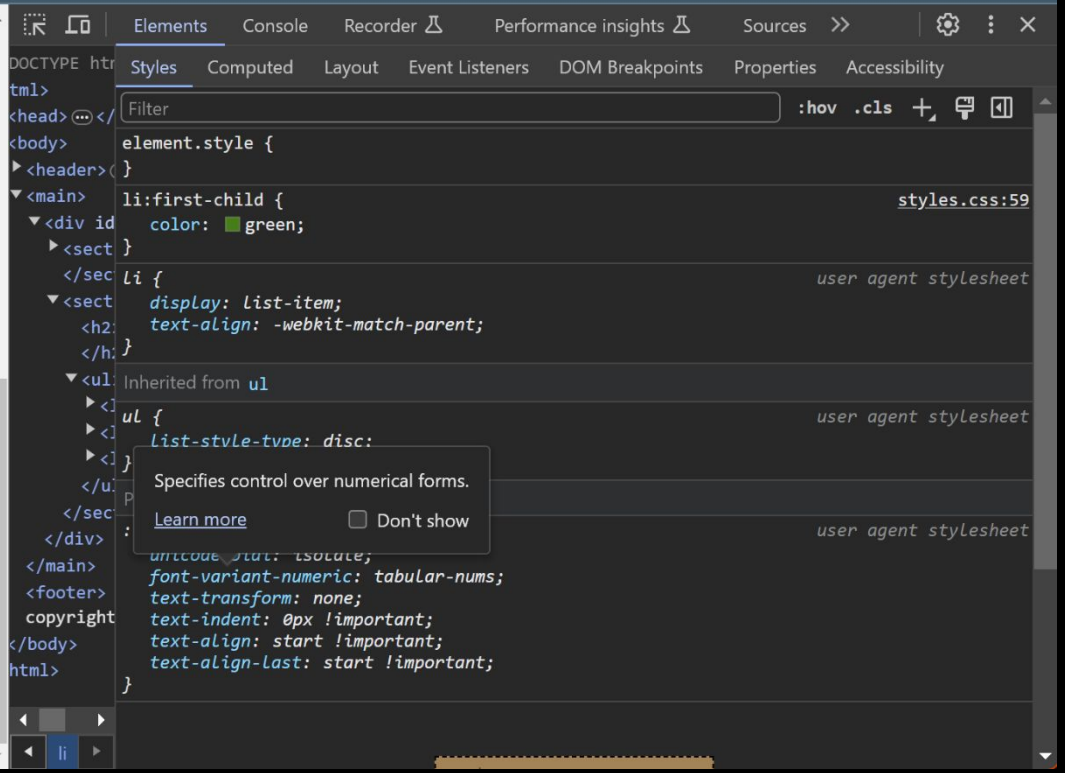
Special text

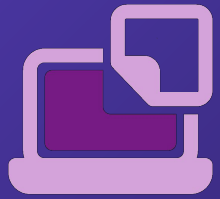
Source: lipsum.com

## Some African Countries

- Algeria
- Burkina Faso
- Cameroon

This page has no copyright.





Created by HideMaru
from Noun Project

# Quiz time!

## With these rules:

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

section p {
    color: blue;
}
```

## What color will be applied to the “something” text below? Why?

```
<section>
    <p>Something</p>
</section>
```

Icon by HideMaru for Noun Project

# Neat tricks using pseudo-selectors

Pseudo-selectors are powerful! They can:

- Create more “virtual” content to create interesting css effects without adding html (`::before`, `::after`)
- Create dynamic selections that cannot be done in other ways
  - `::first-line` will always select only a first rendered line, no matter the window width
  - `::selection` styles the portion of text selected by the mouse cursor
  - `:first-child`, `:first-of-kind`, `:even`, `:odd`, `:nth-of-type()`, etc.
- Style states of an element (`:hover`, `:visited`, etc.)

These are only a few examples. There are more!

# Pseudo-selectors

Test this in any page. What happens when you hover on a link? What happens after you visited the link and go back to your page?

```
/* :hover and :visited are a type of pseudo-selector called
   pseudo-classes. They allow us to style states of an element. */

/* when we hover an <a>, it's in a "hover" state */
a:hover {
    font-weight: bold;
}

/* when we have already visited a link, it's in a "visited" state */
a:visited {
    font-weight: normal;
    font-style: italic;
}
```

# Wait, what is “the state of an element”?

Example: a box is either closed or open. Each one of this is a state.

A closed box can become open by opening it.

An open box can become closed by closing it.

In other words:

- The **state** of an object represents how it is in a specific moment
- The object can **change state**

Elements that the user interacts with, such as links and inputs, have state. We can use CSS to change the appearance of the element depending on its state.

# Now, let's talk about declarations.

Many possible attributes can be styled. Some simple and fun ones:

```
#sample {
    background-color: yellowgreen;
    background-image: url(images/smile.png);
    background-repeat: no-repeat;
    background-position: top left;
    background: yellowgreen no-repeat top left url(images/smile.png); /* the same, in a single row */

    border: 5px solid green;
    color: blue;

    font-family: 'Courier New', Courier, monospace;
    text-align: right;

    height: 2rem;
    width: 100%; /* the default */
}
```

# CSS demonstration 2

Here is a div. Image by FontAwesome.

Another div.

This page has no copyright.

# Let's talk units

Many ways to specify **sizes.** Some common ones:

| | |
|---|---|
| `px` | "Pixels" but not really |
| `rem` | 1 rem = the height of the font globally. Always use this instead of em |
| `em` | 1 em = the height of the font in this element. Rules are weird; don't use. |
| `%` | Percentage relative to the parent |
| `vh, vw` | Viewport height, viewport width. 100vh = the height of the browser window |

# CSS colors

Many ways to specify them. Most common ones:

| | |
|---|---|
| `#rrggbb` | Red, green, blue, in hexadecimal (base 16) numbers - from 00 to FF. Example: #ff0000 = red; #00ffff = cyan. |
| `#rgb` | Abbreviated form; less colors here. Ex.: #f00 = red. |
| `rgb()` | Red, green, blue, in parts from 0 to 255. Example: rgb(255, 0, 0) = red. |
| `rgba()` | Same but also specifies transparency from 0 to 1. rgba(0, 0, 0, 0.75) = black, but with a bit of transparency. 0 is transparent; 1 is opaque. |
| `name` | Named colors. Use for examples only |

# CSS typography

CSS uses “font stacks”: specify many fonts in order of priority:

```
font-family: 'Courier New', Courier, monospace;
```

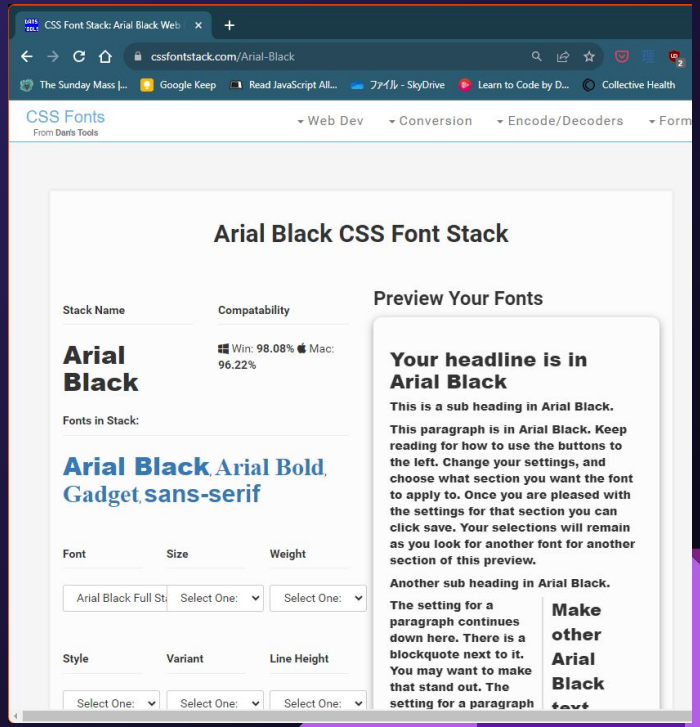
If the first one is not available in the user’s system, the next one is picked.

It’s good practice to end with one of the “generic” fonts such as  sans-serif, serif, monospace, or cursive. Full list here.

# CSS typography: Font stack generators

Use a tool such as [cssfontstack.com](cssfontstack.com) to create font stacks for you.

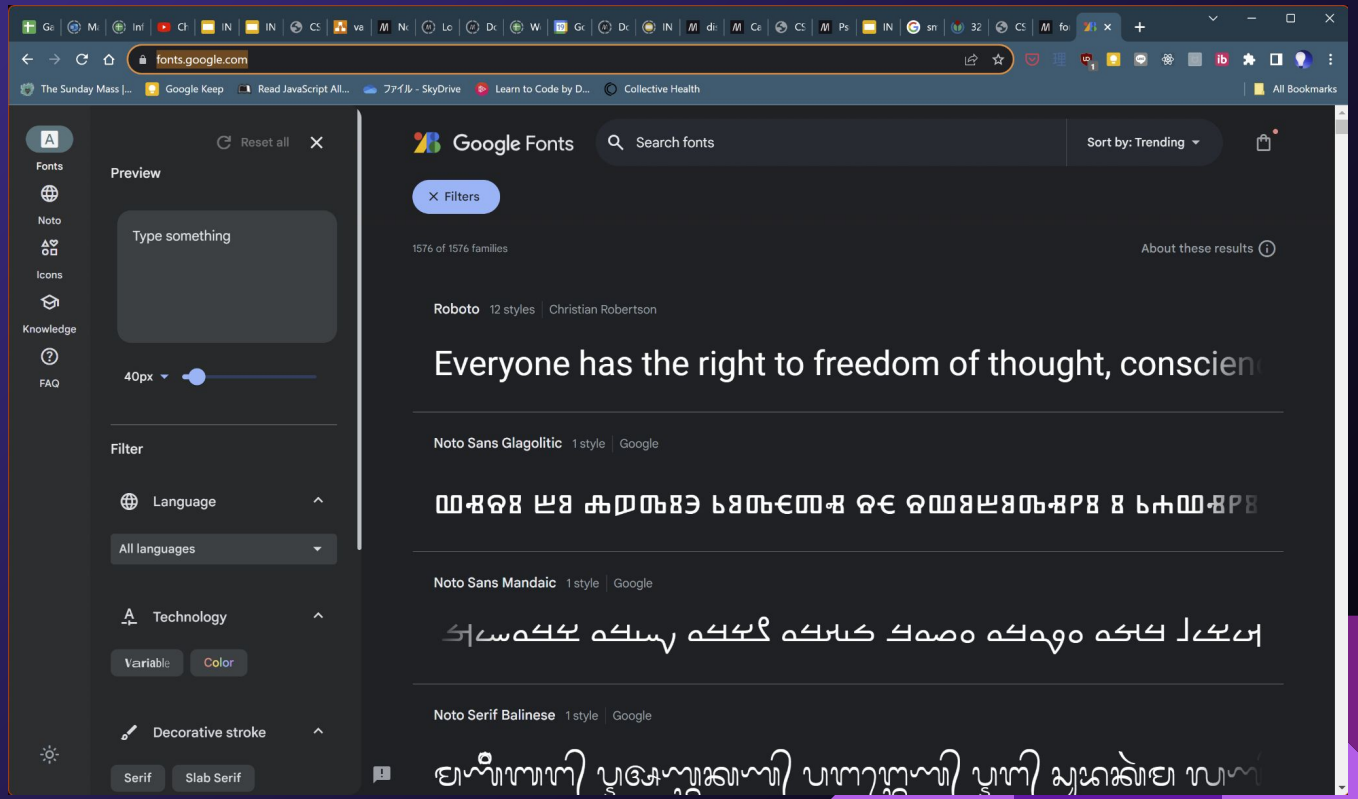
If you import a font from Google Fonts, it will also come with a proper font stack.



# Web fonts

You can also use a font from a provider such as Google Fonts (free) or paid alternatives.

Select your font(s), then copy and paste the provided code.



# Layout

Layout refers to the general “shape” of the page.

To understand how to create a layout, we need to understand **display modes.**

Display modes are set with the css `display` attribute.

# Default display modes

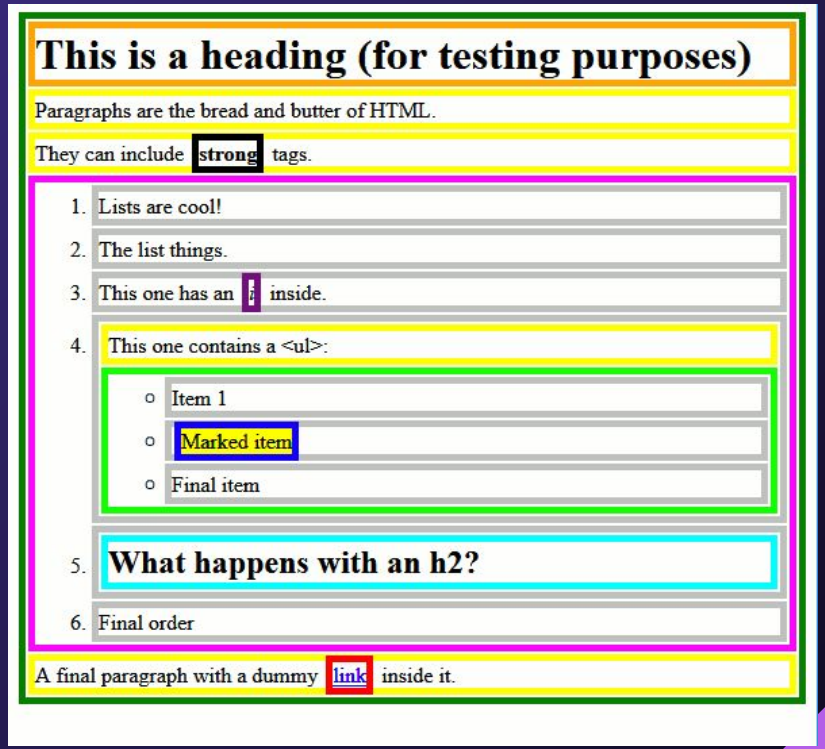
Every tag has, by default, either a block or an inline display mode.

- `block`: creates a rectangle. By default occupies all of the available width.
  - Basic building block of layouts.
  - Example: body, div, p, article
- `inline`: goes with the flow of text.
  - Examples: span, strong, img; also, text outside any tag

Blocks can contain inline and other blocks. Eg. an article can contain text

Inline should only contain inline; eg. a  can contain text, but not a 

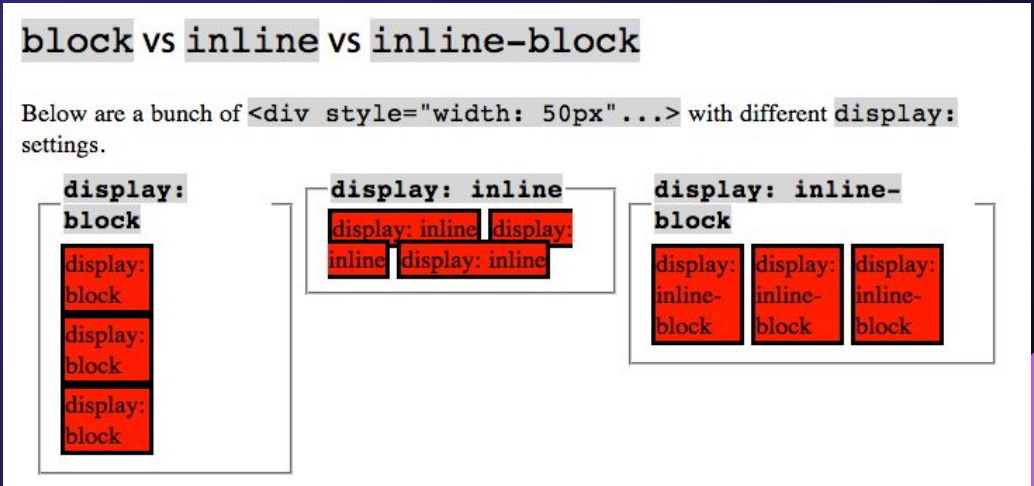
# Default display modes




Source: complete-concrete-concise.com

# Default display modes

- Inline-block:
  - Internally behaves like a block (eg. it can contain other blocks or inline),
  - Externally, goes with the flow of text




Source: this article on medium.com

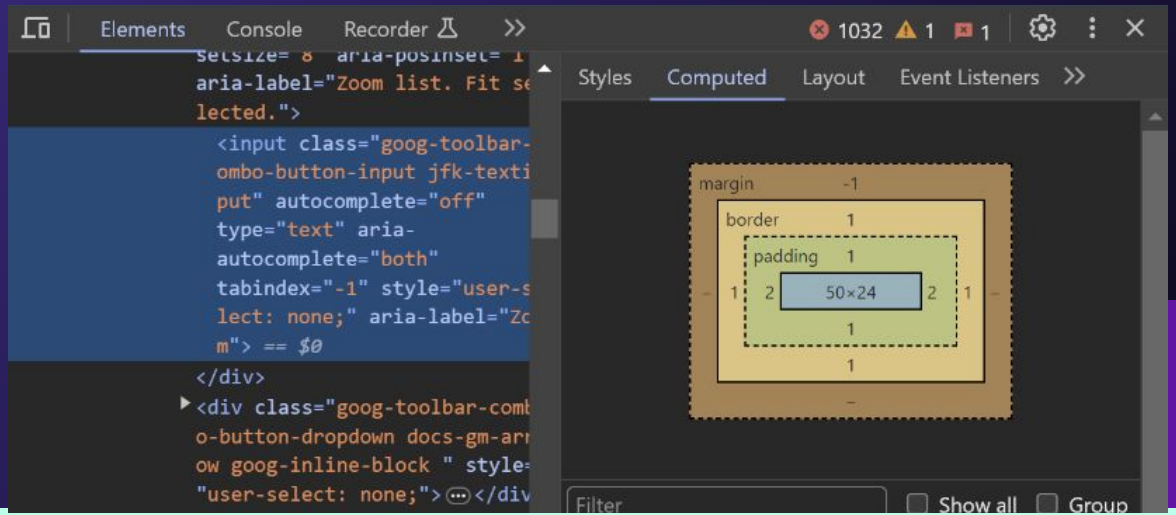
# Other display modes

- display: none - use to not render an element.
- Others, less used, that we will not mention

We will cover display: flex and display: grid later on.

# The box model

- Padding: the space around a box = space between the border and other boxes
- Border
- Margin: the space between the border and the internal contents



# The box model

**Did you set a `width` and things are misbehaving? Read on!**

By default, `width` applies to the **content** of a box.


Screenshot from MDN

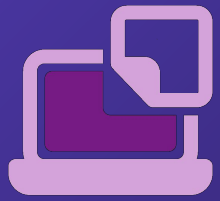
# The box model

Change this by doing:

```
box-sizing: border-box;
```

With border-box,  width applies to contents + padding. This is more consistent with most people's expectations.


Screenshot from MDN



Created by HideMaru
from Noun Project

# A quick exercise

- Ask the AI to create a sample HTML + CSS. Study the files.
- Then, copy the HTML to a new file, then try to make the CSS by yourself *without looking at the generated code.*
    - Focus on text styles, images, tables, and other elements studied so far.

Icon by HideMaru for Noun Project

# In the next lesson...

We'll complete our study of CSS and learn how to make page layouts.