



JONAS.IO  
SCHMEDTMANN



Subscribe here

BUILD  
RESPONSIVE  
REAL-WORLD  
WEBSITES WITH  
HTML AND CSS



Follow me here



@JONASSCHMEDTMAN

# SLIDES FOR THEORY LECTURES

(DON'T SKIP THEM, THEY ARE  
SUPER IMPORTANT 😎)





# TABLE OF CONTENTS: THEORY LECTURES (CLICK THE TITLES)

- 1 A High-Level Overview of Web Development
- 2 Watch Before You Start!
- 3 Introduction to HTML
- 4 Introduction to CSS
- 5 Working With Colors
- 6 CSS Theory #1: Conflicts Between Selectors
- 7 CSS Theory #2: Inheritance and Universal Selector
- 8 CSS Theory #3: The CSS Box Model
- 9 CSS Theory #4: Types of Boxes
- 10 CSS Theory #5: Absolute Positioning
- 11 The 3 Ways of Building Layouts
- 12 Using Floats
- 13 box-sizing: border-box
- 14 A Flexbox Overview
- 15 A CSS Grid Overview
- 16 Overview of Web Design and Website Personalities
- 17 Web Design Rules #1: Typography
- 18 Web Design Rules #2: Colors
- 19 Web Design Rules #3: Images and Illustrations
- 20 Web Design Rules #4: Icons
- 21 Web Design Rules #5: Shadows
- 22 Web Design Rules #6: Border-radius
- 23 Web Design Rules #7: Whitespace
- 24 Web Design Rules #8: Visual Hierarchy
- 25 Web Design Rules #9: User Experience (UX)
- 26 The Website-Personalities-Framework
- 27 Web Design Rules #10 - I: Elements and Components
- 28 Switching flex-direction to column
- 29 Vertical center with absolute position and transform
- 30 Web Design Rules #10 - II: Layout Patterns
- 31 The 7 Steps to a Great Website
- 32 Defining and Planning the Project (Steps 1 and 2)
- 33 Sketching Initial Layout Ideas (Step 3)
- 34 Responsive Design Principles
- 35 How Media Queries Work
- 36 How to Select Breakpoints



# **SECTION 01 – WELCOME AND FIRST STEPS**





# BUILD RESPONSIVE REAL-WORLD WEBSITES WITH HTML AND CSS

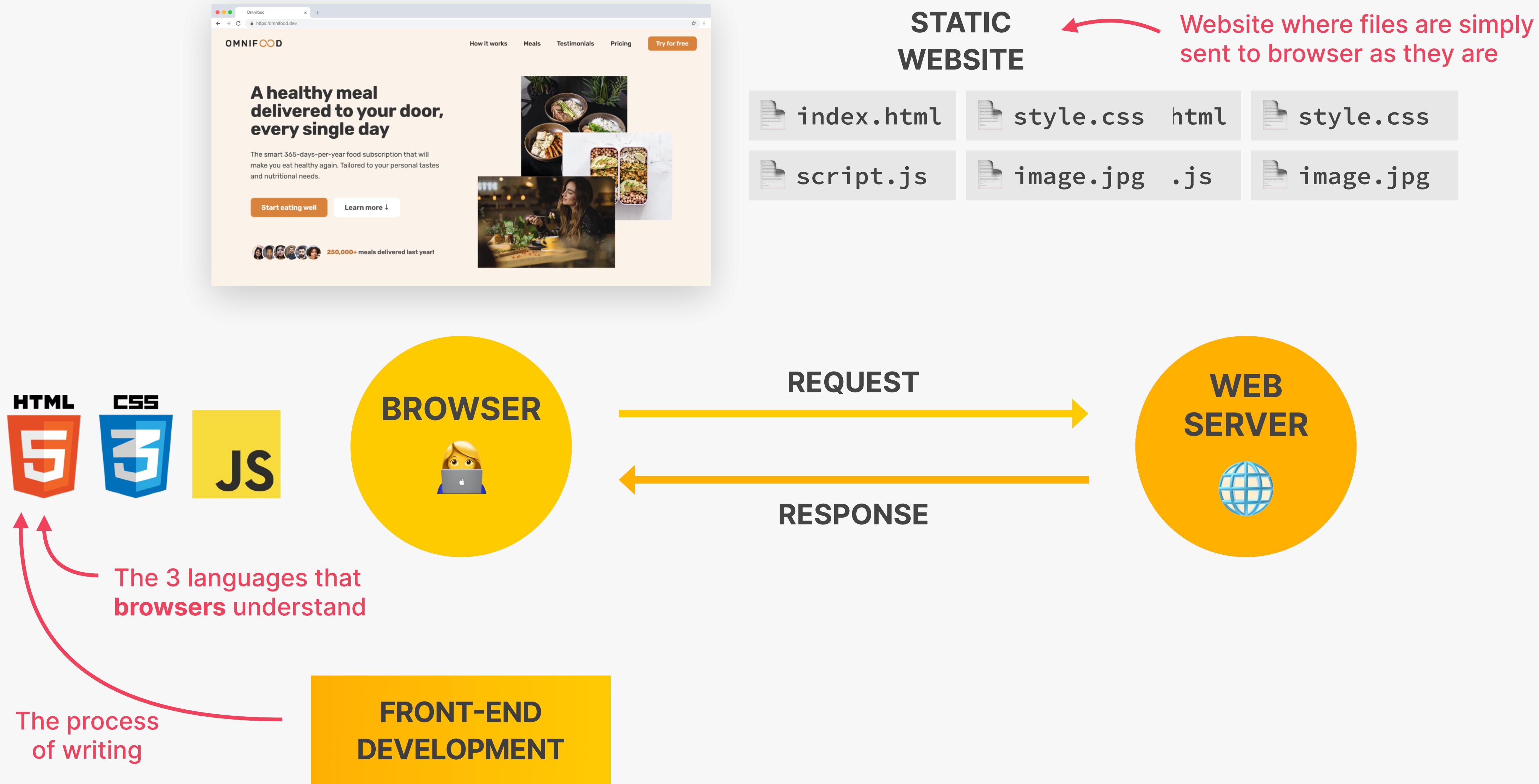
## SECTION

WELCOME AND FIRST STEPS

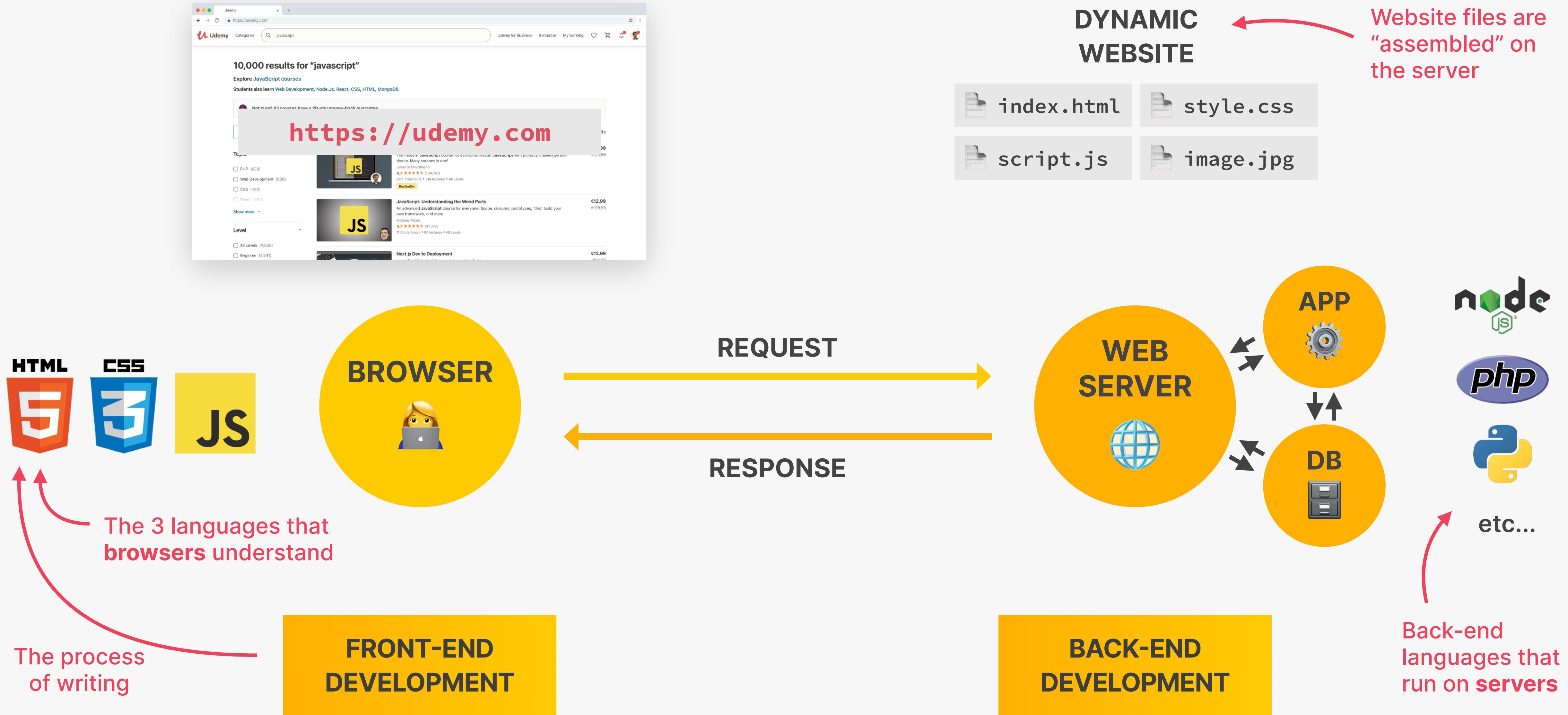
## LECTURE

A HIGH-LEVEL OVERVIEW OF WEB  
DEVELOPMENT

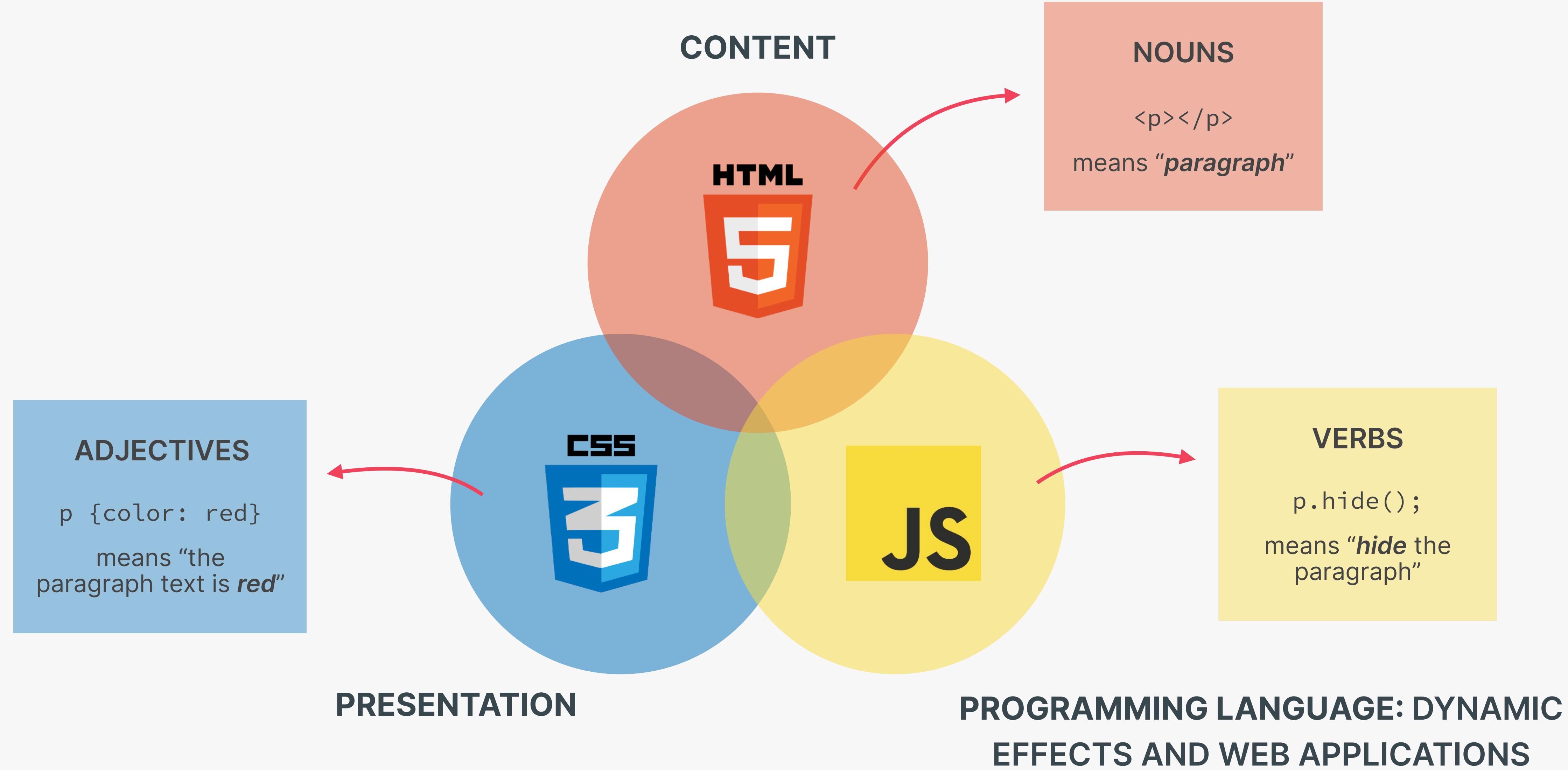
# FRONT-END VS. BACK-END DEVELOPMENT



# FRONT-END VS. BACK-END DEVELOPMENT



# THE 3 LANGUAGES OF THE FRONT-END







# BUILD RESPONSIVE REAL-WORLD WEBSITES WITH HTML AND CSS

## SECTION

WELCOME AND FIRST STEPS

## LECTURE

WATCH BEFORE YOU START!

# SOME QUICK CONSIDERATIONS BEFORE WE START...



恐慌表情符号 If this is your first time ever writing code, please don't get overwhelmed. It's 100% normal that you will not understand everything at the beginning. *Just don't think "I guess coding is not for me"!*



# SOME QUICK CONSIDERATIONS BEFORE WE START...



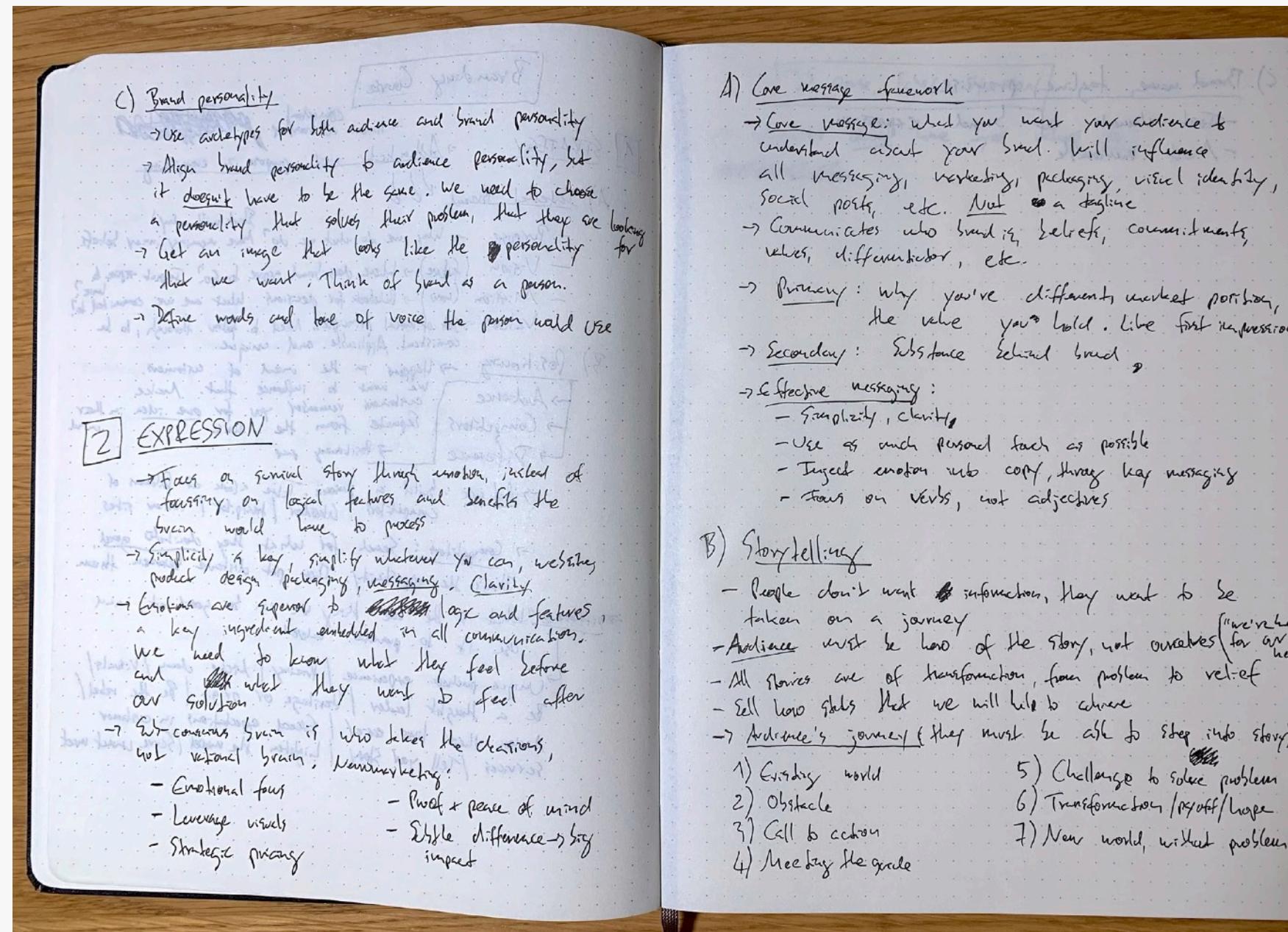
**You need to code along with me!** You will learn **ZERO** HTML and CSS skills by just sitting and watching me code. You really have to write code **YOURSELF!**



# SOME QUICK CONSIDERATIONS BEFORE WE START...



If you want the course material to stick, take notes. Notes on code syntax, notes on theory concepts, notes on everything!



Totally non-coding... Try to understand a single word 😂

# SOME QUICK CONSIDERATIONS BEFORE WE START...



🤓 **Try all the coding challenges!** Try to do your best, but if you get stuck for too long, watch the solution.  
**Don't beat yourself up if you can't figure it out!** Just rewatch the lectures that were covered in the challenge, try to understand them better, and move on.



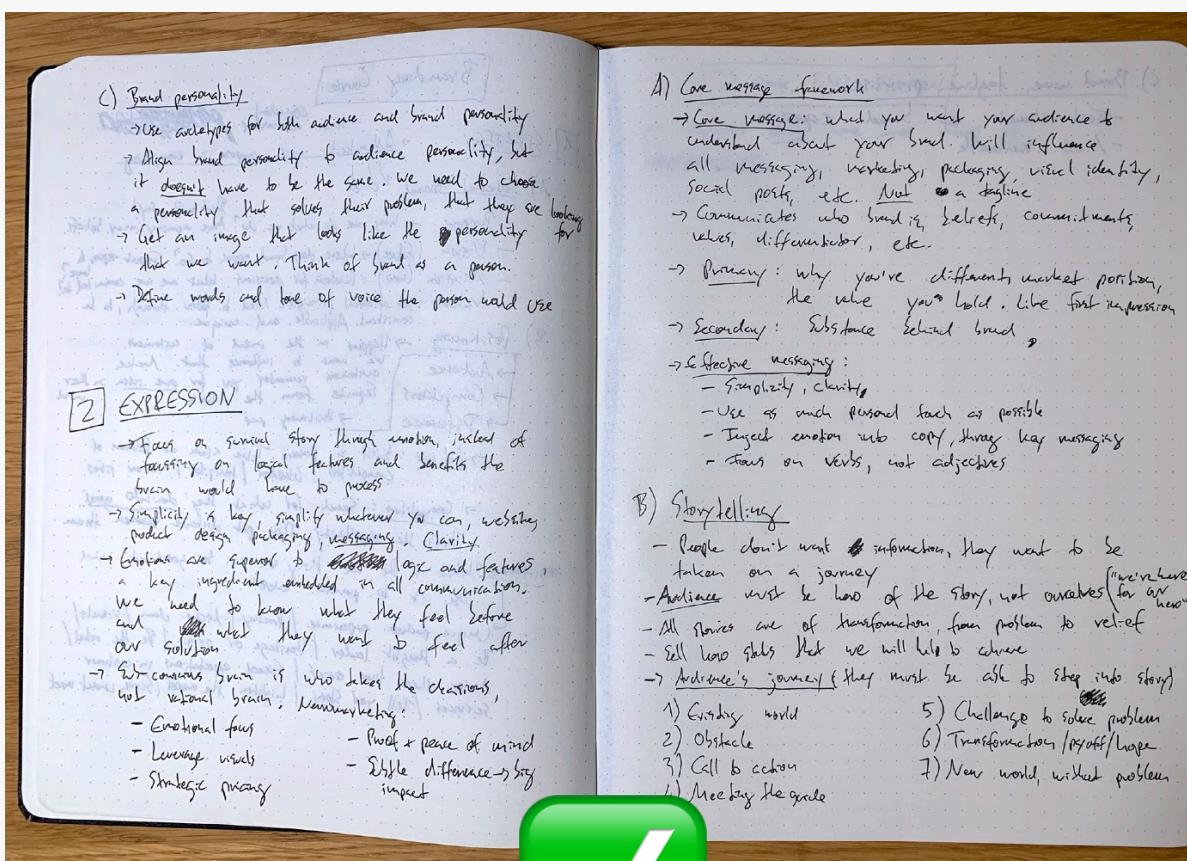
Watch for this sign!

PAUSE THE VIDEO  
FOR CHALLENGE

# SOME QUICK CONSIDERATIONS BEFORE WE START...



**Before moving on from a section, make sure that you understand exactly what was covered.** Take a break, review the code we wrote, review your notes, review the projects we built, and maybe even write some code yourself.



```
208 .chair-details li:not(:last-child) {  
209   /* margin-bottom: 24px; */  
210   margin-bottom: 16px;  
211 }  
212  
213 .chair-icon {  
214   width: 24px;  
215   height: 24px;  
216   stroke: #087f5b;  
217 }  
218  
219 .chair-price {  
220   display: flex;  
221   justify-content: space-between;  
222  
223   align-items: center;  
224   font-size: 20px;  
225 }  
226  
227 footer {
```



We couldn't live without these chairs anymore

"We couldn't live without these chairs anymore"

Our bestselling chairs

Chair Model	Description	Price	Add to Cart
The Laid Back	Leisure and relaxing, comfortable for 4h, vegan leather, weighs 16 kg	250€	ADD TO CART
The Worker Bee	Work, comfortable all day, vegan leather, organic cotton, weighs 22 kg	525€	ADD TO CART
The Chair 4/2	Leisure and relaxing, comfortable all day, organic cotton, weighs 80 kg	1450€	ADD TO CART

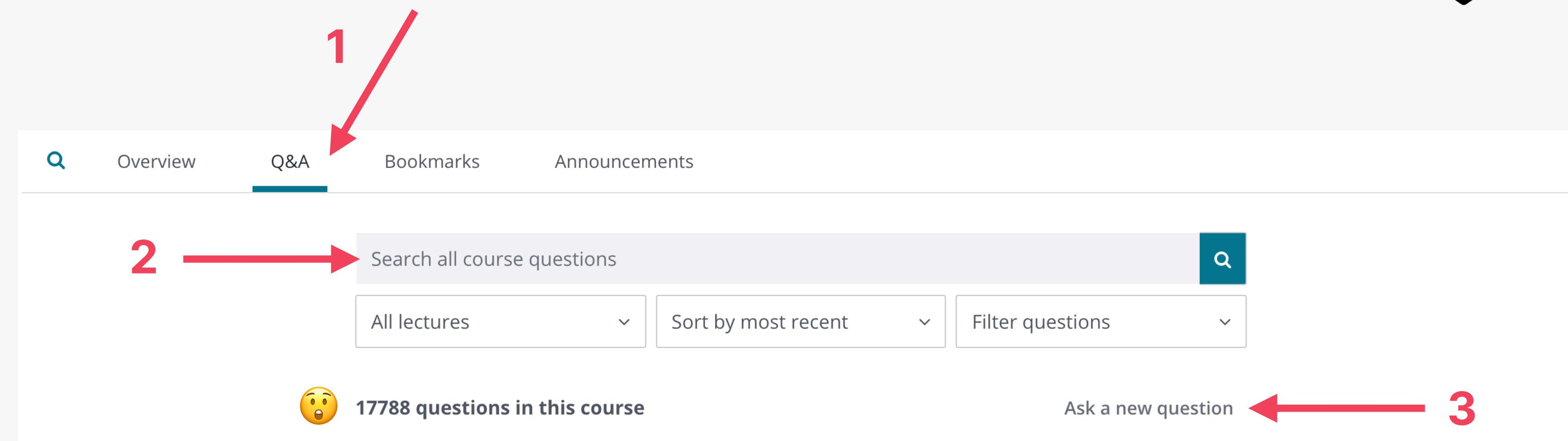
Copyright © 2027 by Jonas Schmedemann. Part of "Build a modern website with HTML and CSS" online course. Use for learning purposes only.



# SOME QUICK CONSIDERATIONS BEFORE WE START...

! If you have an error or a question, **start by trying to solve it yourself! This is essential for your progress.** If you can't solve it, check the Q&A section. If that doesn't help, you can **ask a new question**. Use a short description, and post code on [codepen.io](https://codepen.io).

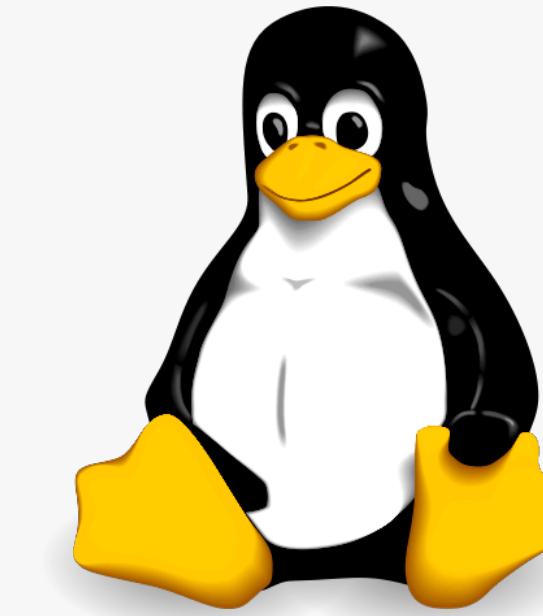
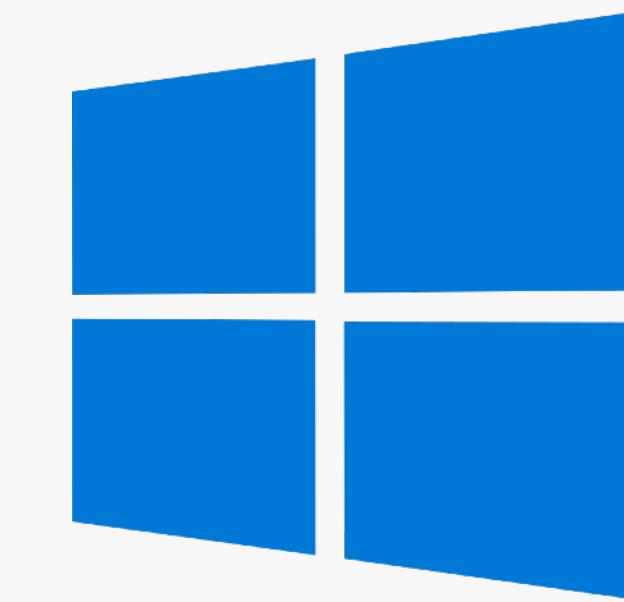
CODEPEN



# SOME QUICK CONSIDERATIONS BEFORE WE START...

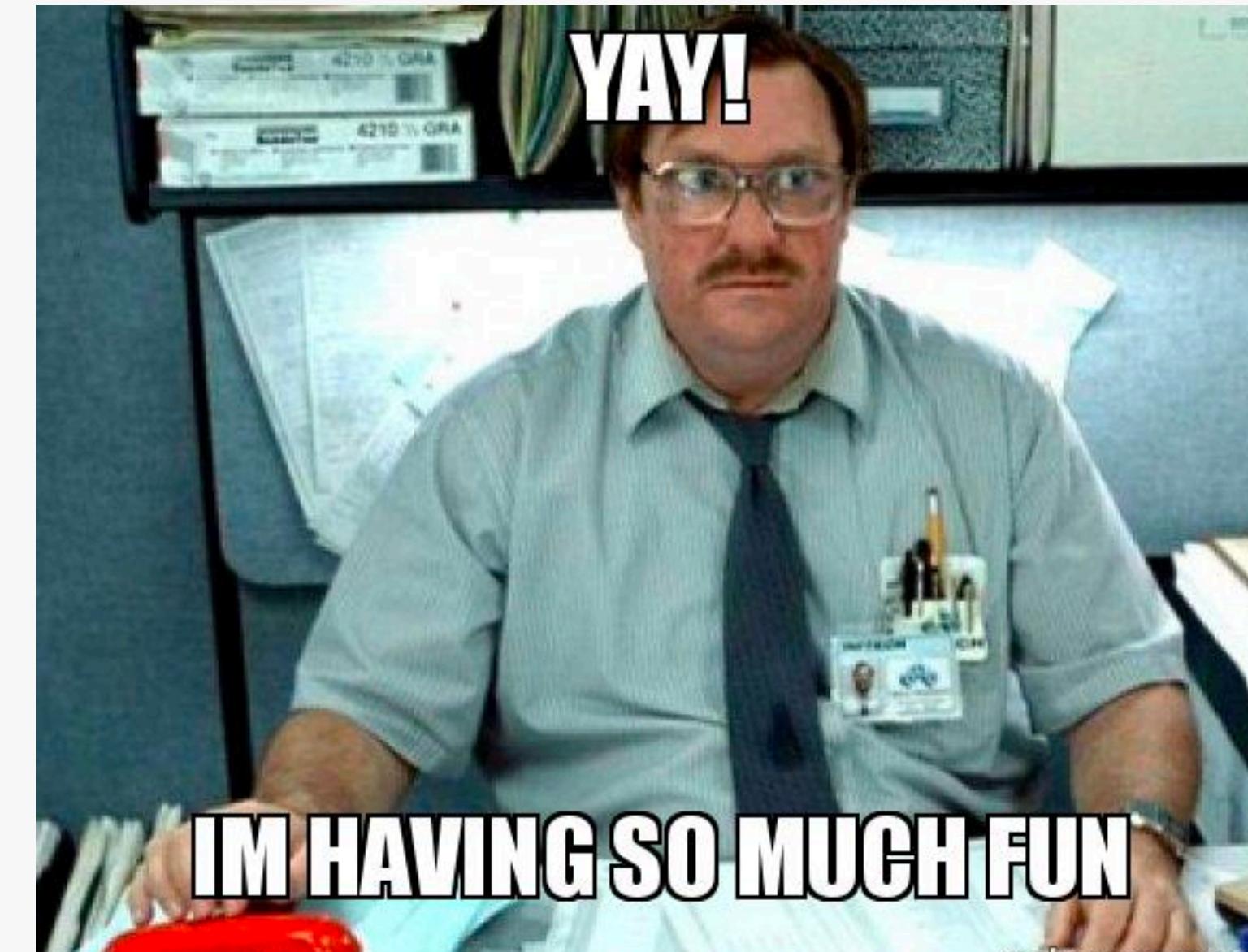


I recorded this course on a Mac, but everything works the exact same way on Windows or Linux. If something doesn't work on your computer, it's **NOT** because you're using a different OS.



# SOME QUICK CONSIDERATIONS BEFORE WE START...

😍 **Most importantly, have fun!** It's so rewarding to see something that **YOU** have built **YOURSELF!** So if you're feeling frustrated, stop whatever you're doing, and come back later!



And I mean **REAL** fun 😊



# **SECTION 02 –**

# **HTML FUNDAMENTALS**



# BUILD RESPONSIVE REAL-WORLD WEBSITES WITH HTML AND CSS

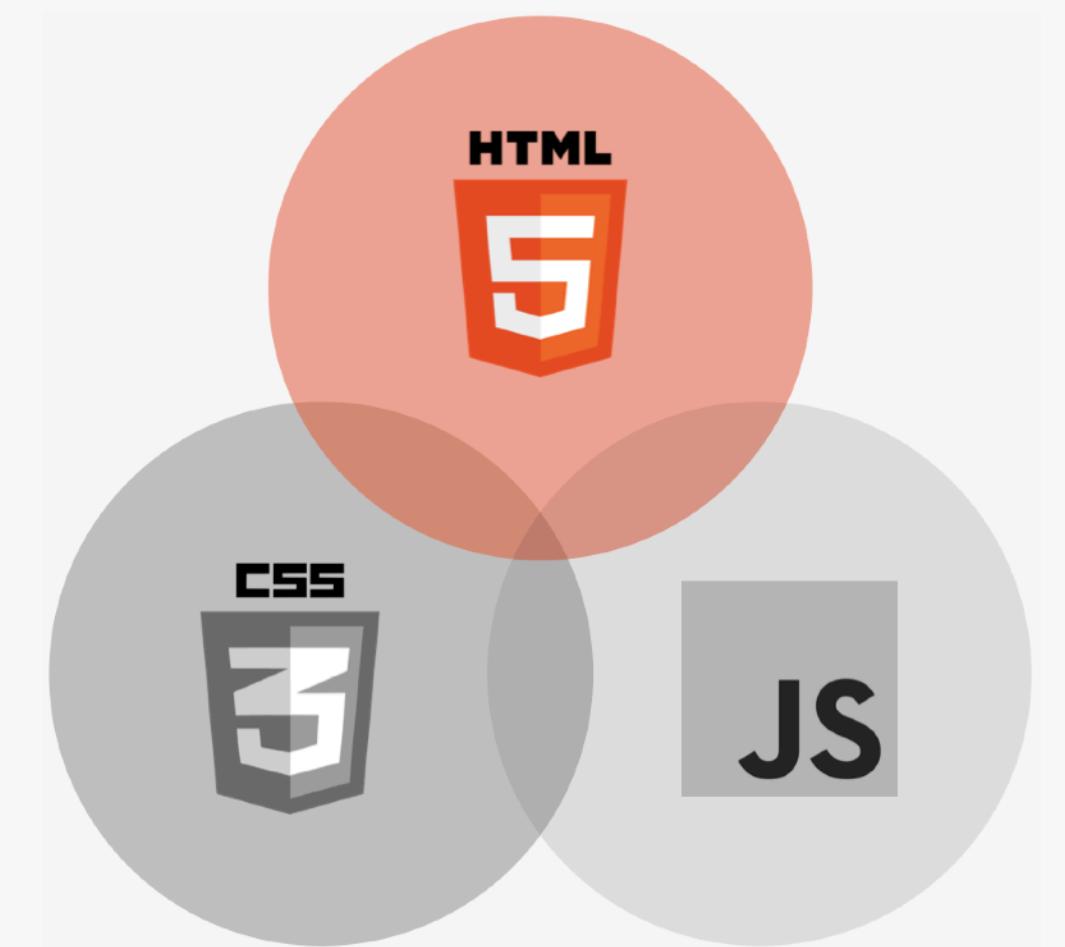
SECTION  
HTML FUNDAMENTALS

LECTURE  
INTRODUCTION TO HTML

# WHAT IS HTML?

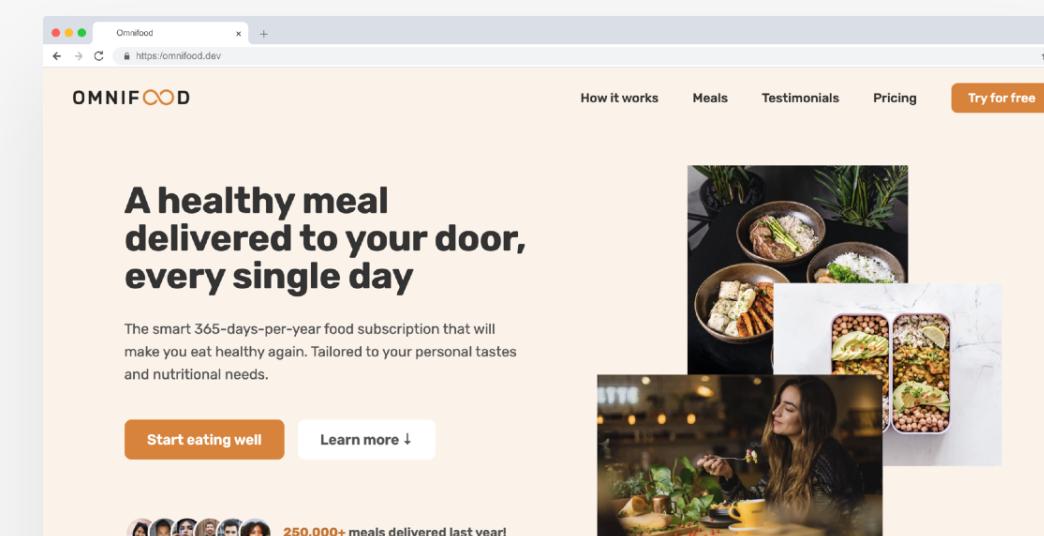
## HTML

- 👉 **HyperText Markup Language**
- 👉 HTML is a markup language that web developers use to **structure and describe the content** of a webpage (*not a programming language*)
- 👉 HTML consists of **elements** that describe different types of content: paragraphs, links, headings, images, video, etc.
- 👉 Web browsers understand HTML and **render HTML code as websites**

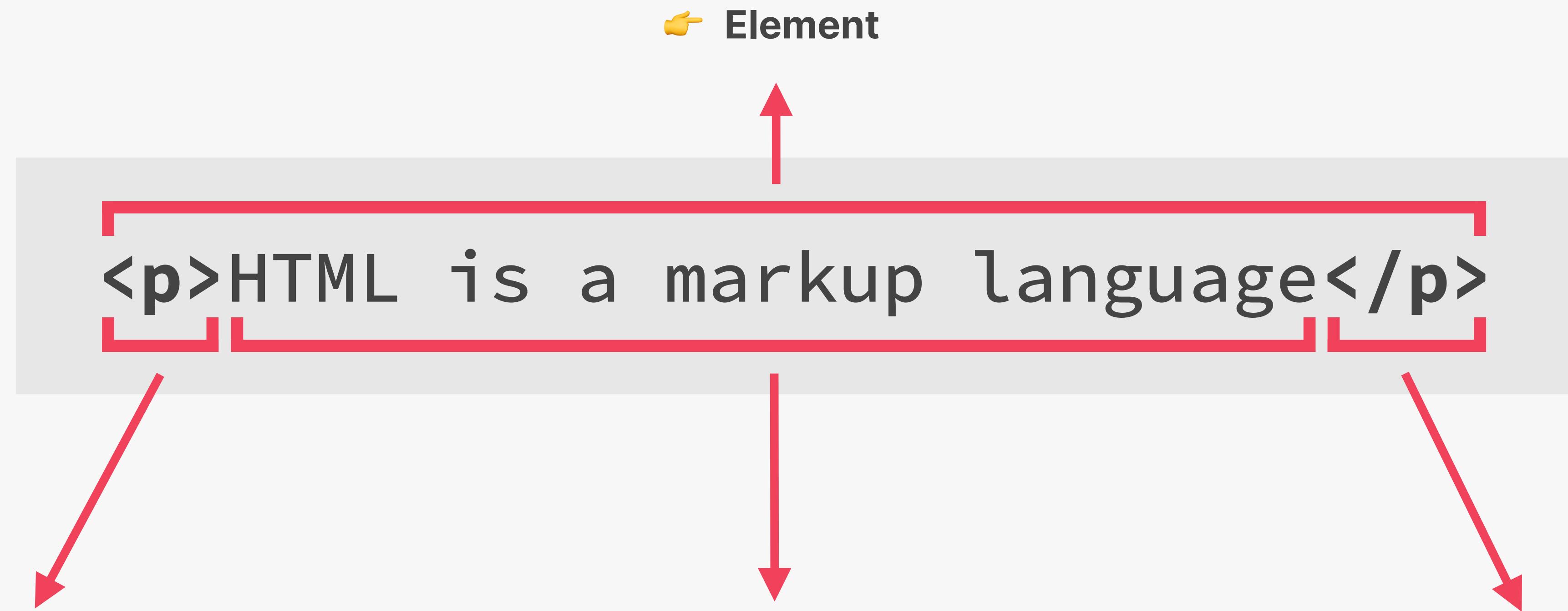


A screenshot of a code editor window titled "index.html - index.html". The code is as follows:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <link rel="stylesheet" href="style.css" />
  </head>
  <body>
    <main class="container">
      <div class="poll">
        <form action="#" method="POST" class="poll_form">
          <fieldset>
            <legend class="poll__question">What's your favorite programming language?</legend>
            <ul class="poll__answers">
              <li><input type="radio" name="radio" id="option-0" class="poll_answer-radio"/>
                <label for="option-0" class="poll_answer-label">PHP</label>
              </li>
              <li><input type="radio" name="radio" id="option-1" class="poll_answer-radio"/>
                <label for="option-1" class="poll_answer-label">JavaScript</label>
              </li>
              <li><input type="radio" name="radio" id="option-2" class="poll_answer-radio"/>
                <label for="option-2" class="poll_answer-label">Ruby</label>
              </li>
            </ul>
          </fieldset>
        </form>
      </div>
    </main>
  </body>
</html>
```



# ANATOMY OF AN HTML ELEMENT



👉 **Opening tag:** Name of the element, wrapped in < and >

👉 **Content:** Content of the element, in this example text. But it might be another element (**child element**). Some elements have **no content** (e.g. <img>)

👉 **Closing tag:** Same as opening tag, but with a /. When element has no content, it's omitted



# SECTION 03 – CSS FUNDAMENTALS



# BUILD RESPONSIVE REAL-WORLD WEBSITES WITH HTML AND CSS

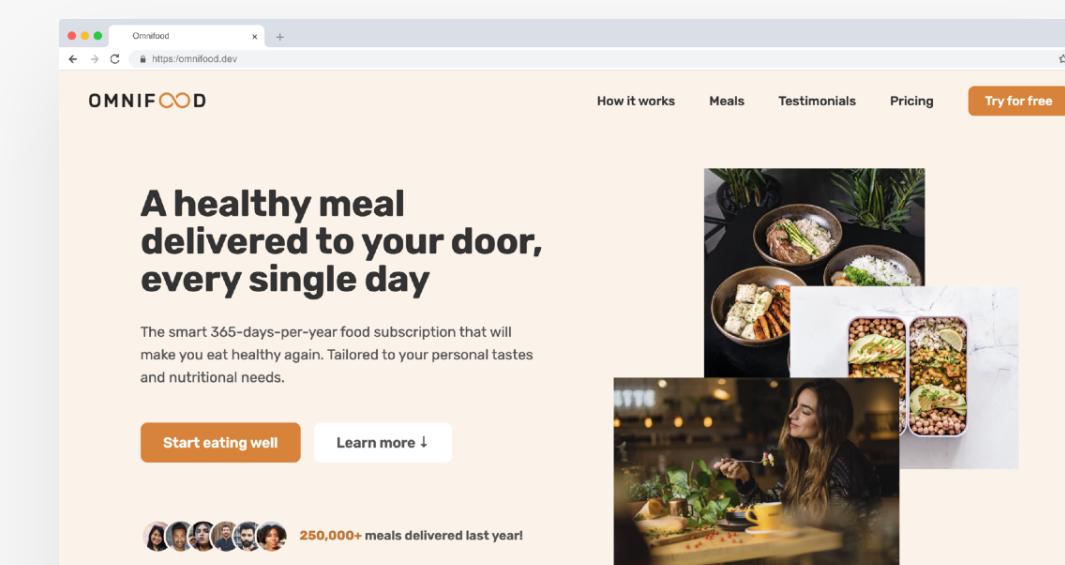
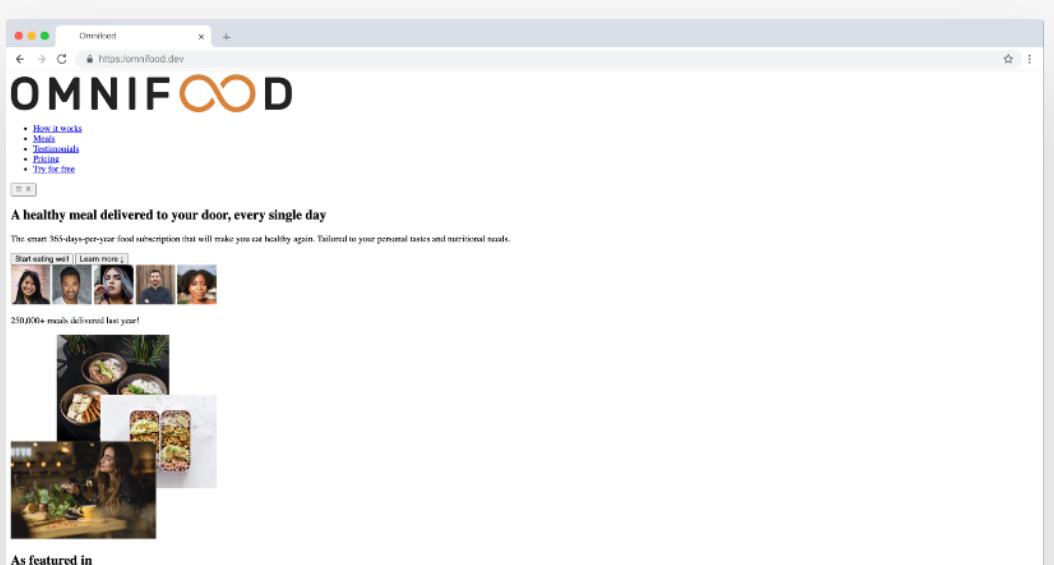
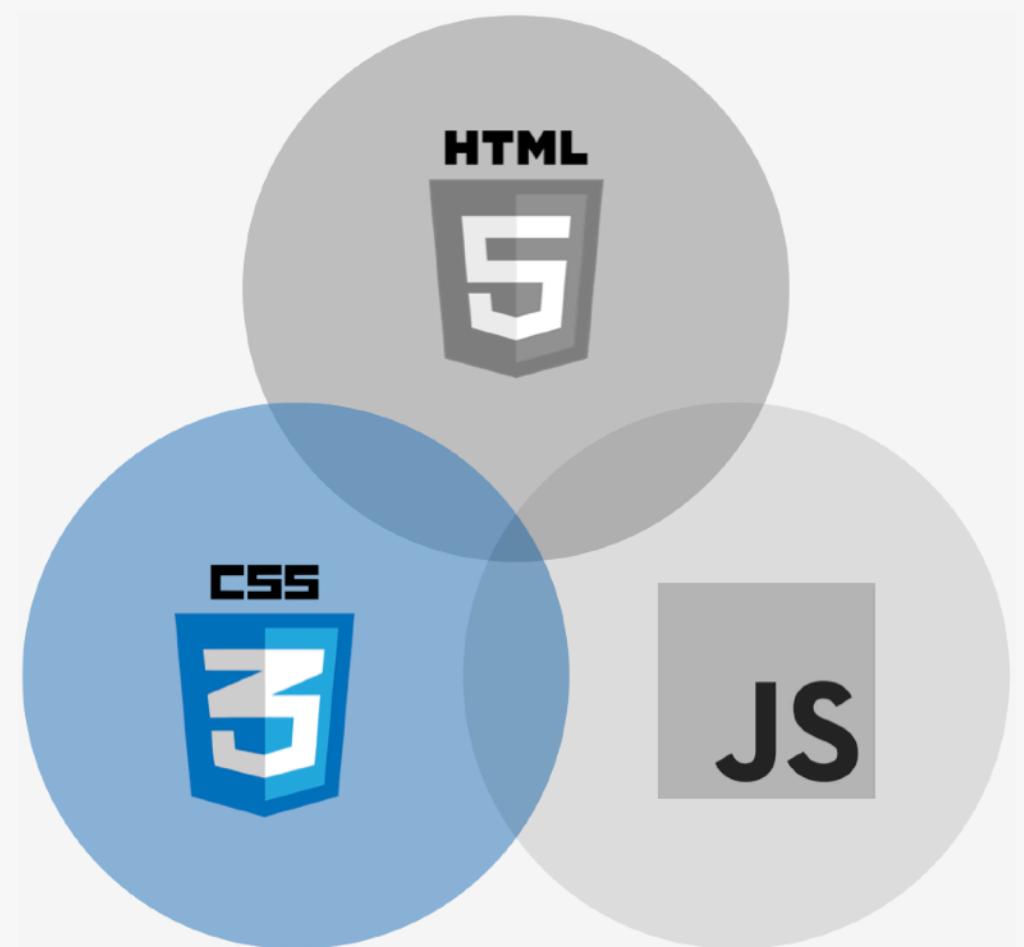
SECTION  
CSS FUNDAMENTALS

LECTURE  
INTRODUCTION TO CSS

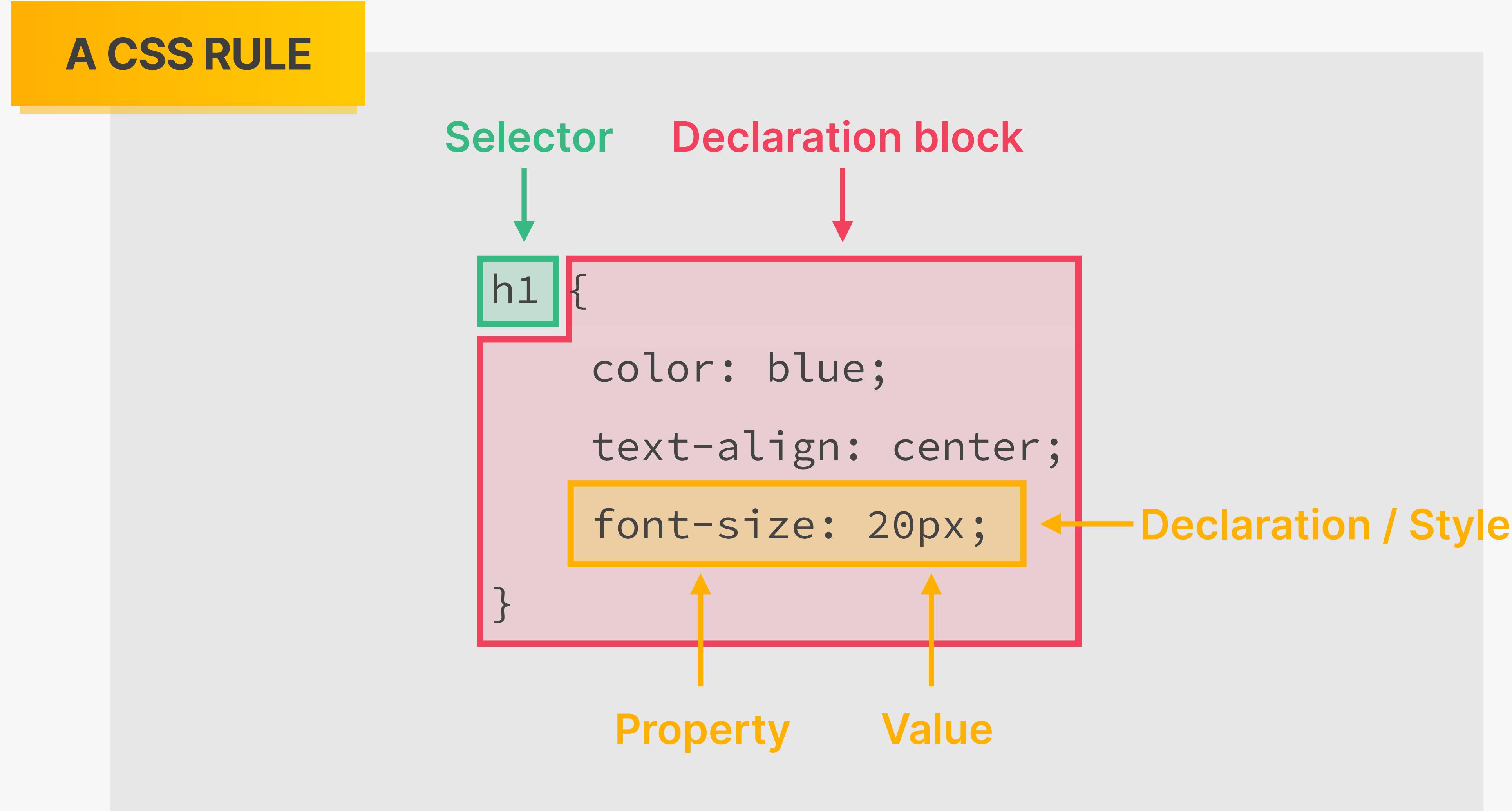
# WHAT IS CSS?

CSS

- 👉 Cascading Style Sheets
- 👉 CSS describes the **visual style and presentation** of the **content written in HTML**
- 👉 CSS consists of countless **properties** that developers use to format the content: properties about font, text, spacing, layout, etc.



# HOW WE SELECT AND STYLE ELEMENTS







# BUILD RESPONSIVE REAL-WORLD WEBSITES WITH HTML AND CSS

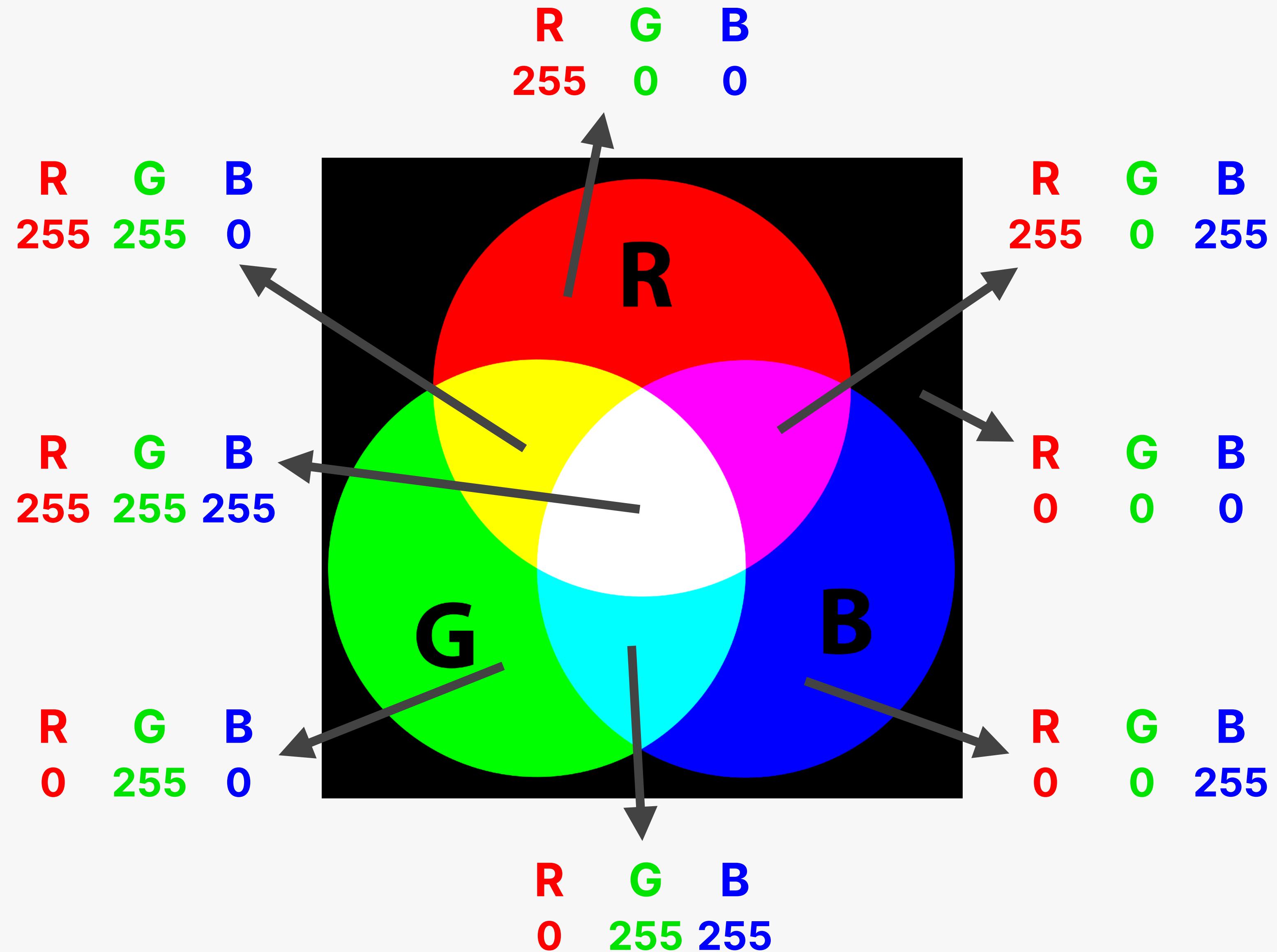
SECTION  
CSS FUNDAMENTALS

LECTURE  
WORKING WITH COLORS

# THE RGB MODEL

👉 **RGB Model:** Every color can be represented by a combination of **RED**, **GREEN** and **BLUE**

👉 Each of the 3 base colors can take a value between **0** and **255**, which leads to 16.8 million different colors



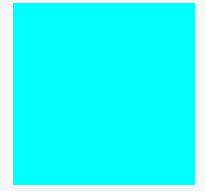
# DEFINING COLORS IN CSS

1

## RGB / RGBA NOTATION

- 👉 Regular RGB model

```
rgb(0, 255, 255)
```



- 👉 RGB with transparency ("alpha")

```
rgba(0, 255, 255, 0.3)
```

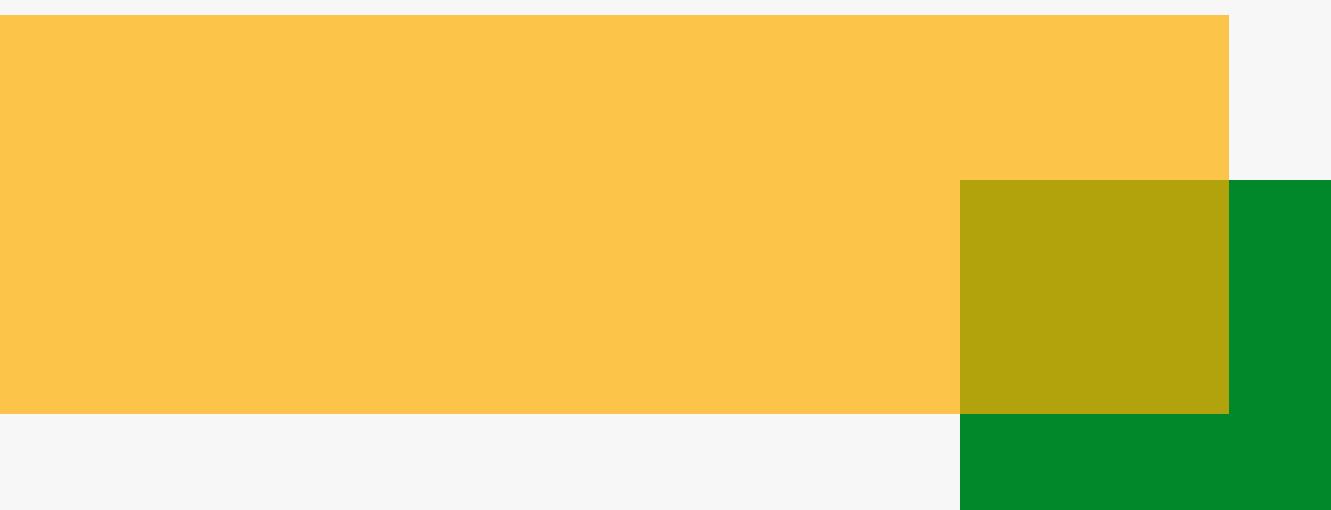


```
#f4b33f
```

```
rgb(244, 179, 63)
```



```
rgba(244, 179, 63, 0.7)
```



2

## HEXADECIMAL NOTATION

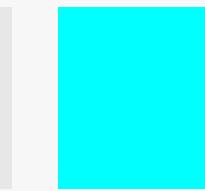
- 👉 Instead of using a scale from 0 to 255, we go from **0** to **ff** (255 in hexadecimal numbers)

```
#00ffff
```

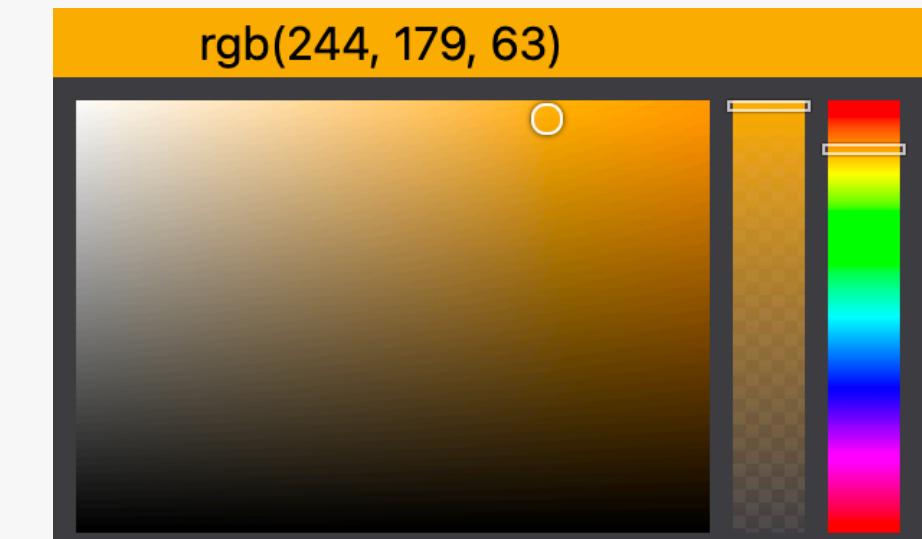


- 👉 Shorthand, when all colors are identical pairs

```
#0ff
```



💡 In practice, we mostly use **hexadecimal** colors, and **rgba** when we need transparency

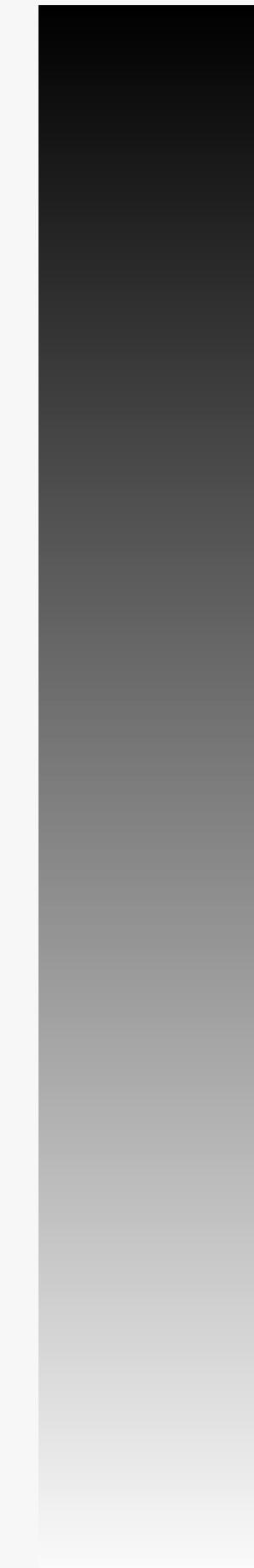


👉 Color picker in VS Code

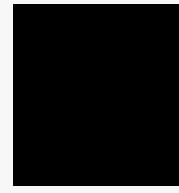
# SHADES OF GREY

👉 When colors in all 3 channels are the same, we get a **grey color**

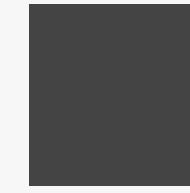
👉 There are 256 pure grays to choose from



`rgb(0, 0, 0) / #000000 / #000`



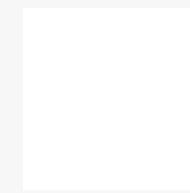
`rgb(69, 69, 69) / #444444 / #444`



`rgb(183, 183, 183) / #b7b7b7`



`rgb(255, 255, 255) / #ffffff / #fff`







# BUILD RESPONSIVE REAL-WORLD WEBSITES WITH HTML AND CSS

SECTION  
CSS FUNDAMENTALS

LECTURE  
CSS THEORY #1: CONFLICTS  
BETWEEN SELECTORS

# CONFLICTING SELECTORS AND DECLARATIONS

```
<p id="author-text" class="author">  
  Posted by Laura Jones on Monday, June 21st 2027  
</p>
```

```
.author {  
  font-style: italic;  
  font-size: 18px;  
}
```

```
#author-text {  
  font-size: 20px;  
}
```

```
p,  
li {  
  font-family: sans-serif;  
  color: #444444;  
  font-size: 22px;  
}
```

🤔 There are **multiple selectors** selecting the same element. **Which one of them applies?**

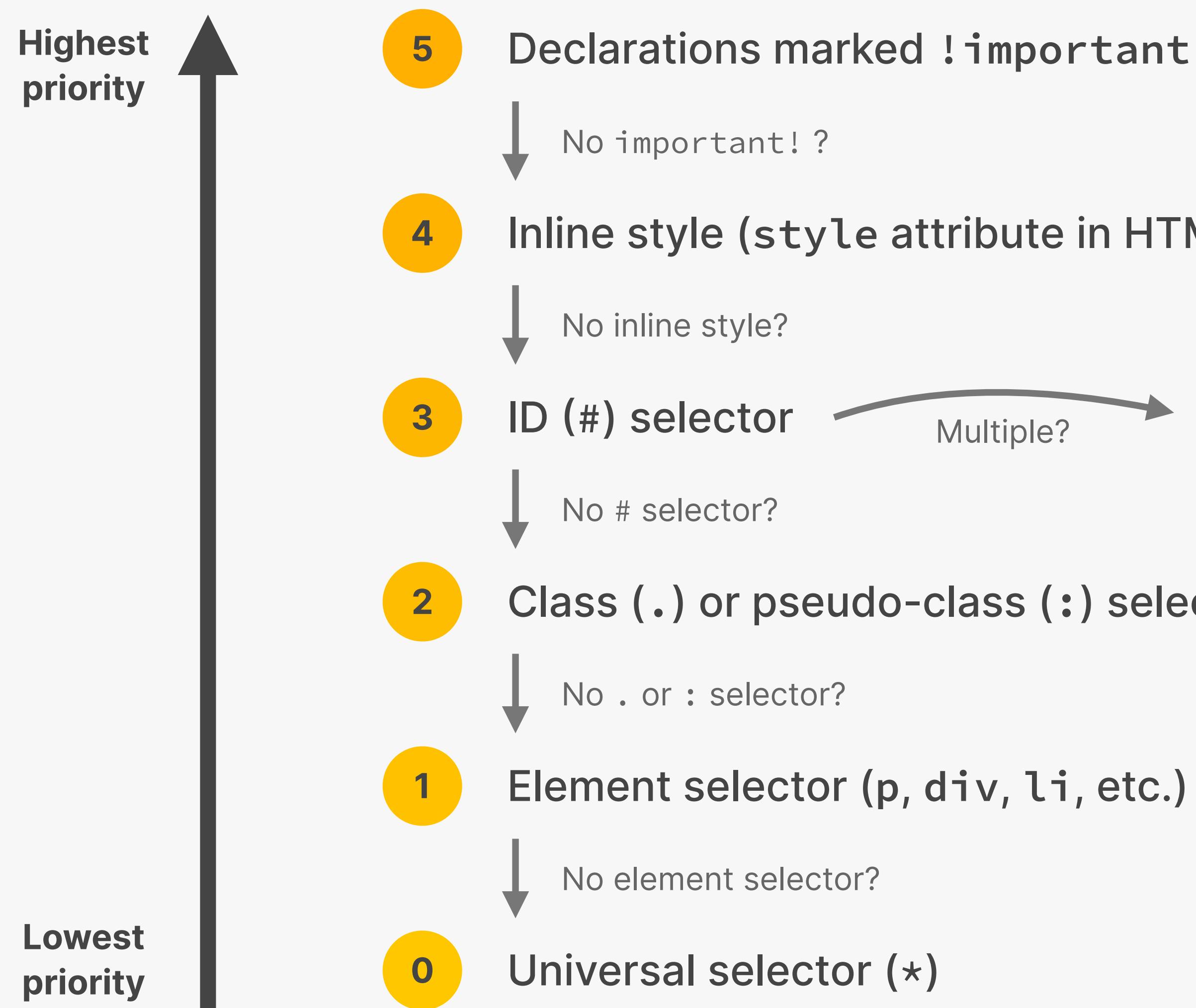
🤓 **All of them. All rules and properties are applied!**



🤔 But there are **conflicting font-size declarations!** Is it 18px, or 20px, or 22px?

🤓 **Let's see how it works...**

# RESOLVING CONFLICTING DECLARATIONS



```
.author {  
    font-style: italic;  
    font-size: 18px;  
}  
  
#author-text {  
    font-size: 20px;  
}  
  
p,  
li {  
    font-family: sans-serif;  
    color: #444444;  
    font-size: 22px;  
}
```

👉 There is an ID selector (`#author-text`), so **for the conflicting `font-size` property, this is the selector that applies**



\* It's a bit more complicated in reality



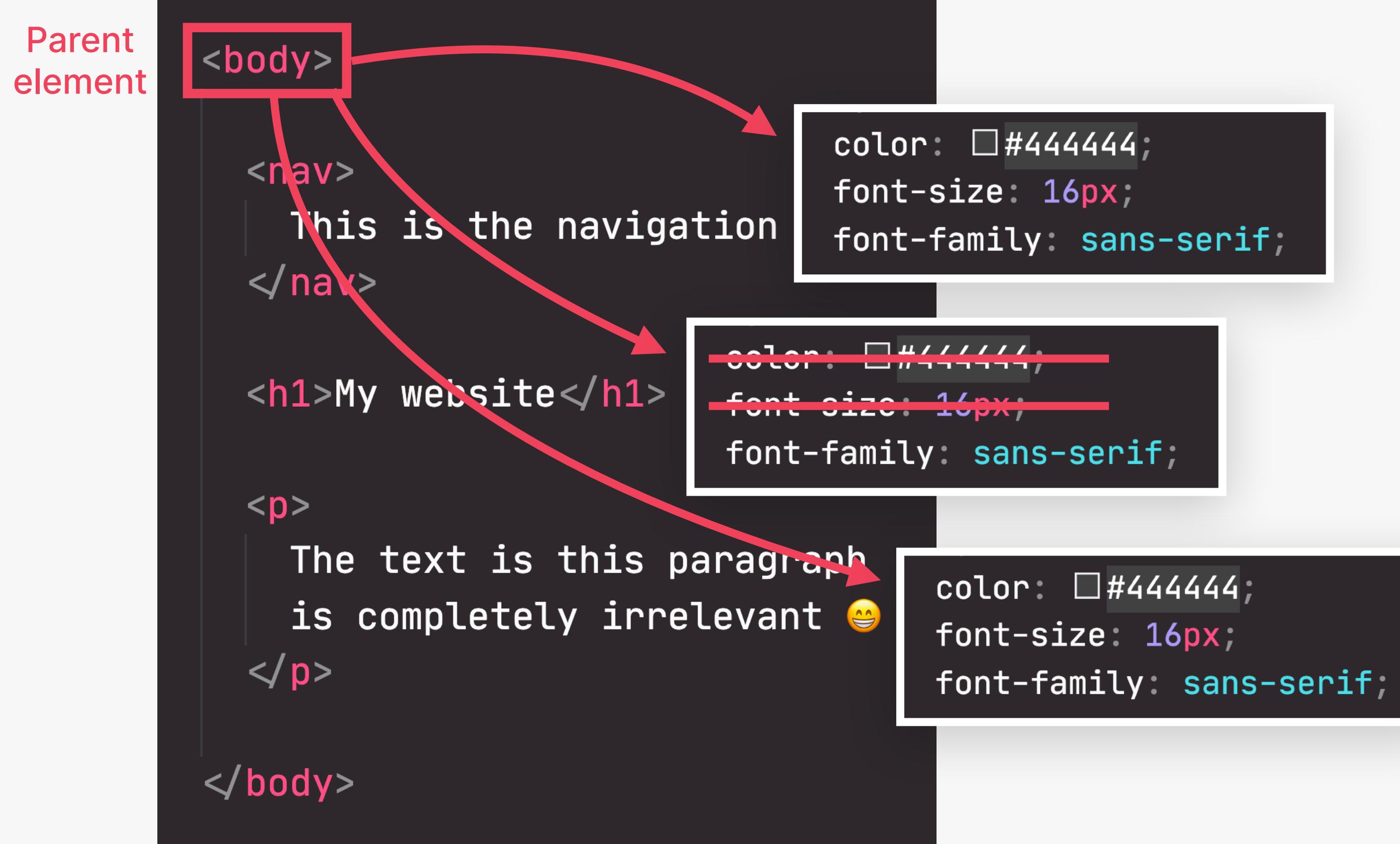


# BUILD RESPONSIVE REAL-WORLD WEBSITES WITH HTML AND CSS

SECTION  
CSS FUNDAMENTALS

LECTURE  
CSS THEORY #2: INHERITANCE  
AND THE UNIVERSAL SELECTOR

# HOW INHERITANCE WORKS



```
body {  
  color: #444444;  
  font-size: 16px;  
  font-family: sans-serif;  
  
  border-top: 10px solid #1098ad;  
}
```

The border property does NOT get inherited

```
h1 {  
  color: #1098ad; OVERRIDING  
INHERITED STYLES  
  font-size: 32px;  
  text-transform: uppercase;  
}
```

- 👉 Not all properties get inherited. It's mostly ones **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.



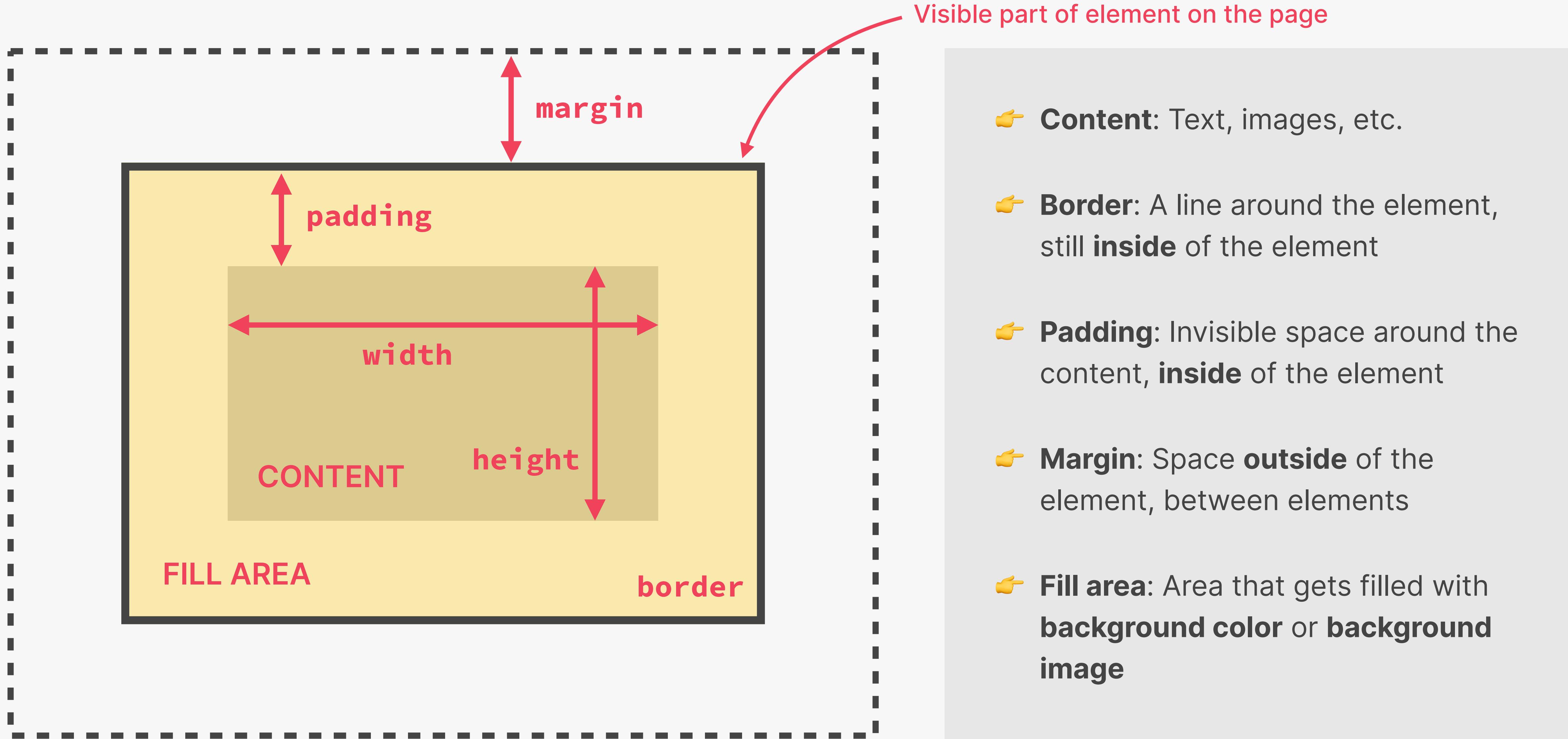


# BUILD RESPONSIVE REAL-WORLD WEBSITES WITH HTML AND CSS

SECTION  
CSS FUNDAMENTALS

LECTURE  
CSS THEORY #3: THE CSS BOX  
MODEL

# THE CSS BOX MODEL



# ANALOGY FOR THE CSS BOX MODEL



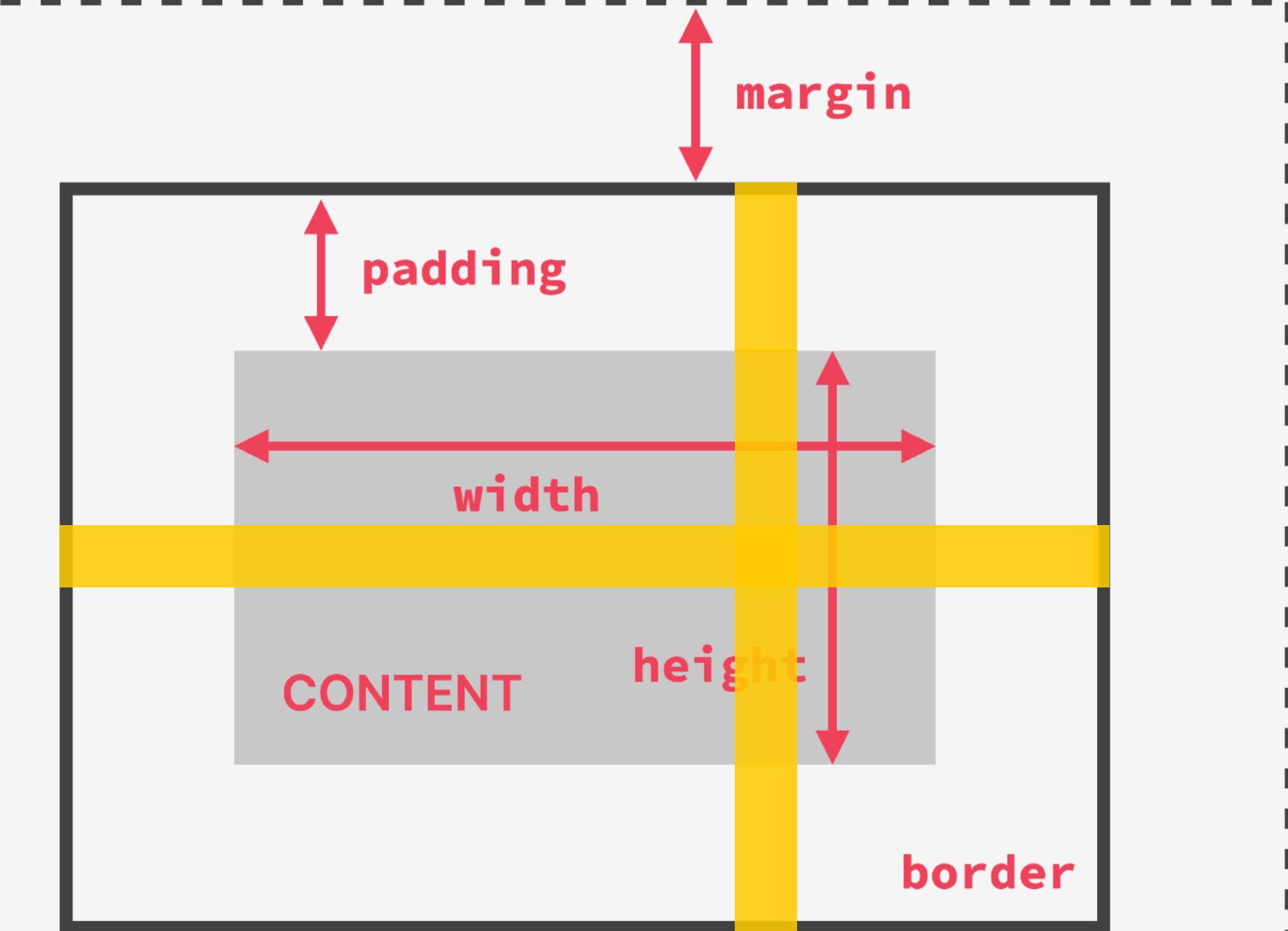
# ELEMENT HEIGHT AND WIDTH CALCULATION

**Final element width** = left border + left padding + width + right padding + right border

**Final element height** = top border + top padding + height + bottom padding + bottom border

👉 We can specify all these values using CSS properties

👉 This is the **default behavior**, but we can change it







# BUILD RESPONSIVE REAL-WORLD WEBSITES WITH HTML AND CSS

SECTION  
CSS FUNDAMENTALS

LECTURE  
CSS THEORY #4: TYPES OF  
BOXES

# BLOCK-LEVEL ELEMENTS

- 👉 Elements are formatted visually as **blocks**
- 👉 Elements occupy **100% of parent element's width**, no matter the content
- 👉 Elements are **stacked vertically** by default, one after another
- 👉 The box-model **applies as showed** earlier

**Default elements:** body, main, header, footer, section, nav, aside, div, h1-h6, p, ul, ol, li, etc.

**With CSS:** `display: block`

## The Basic Language of the Web: HTML



Posted by **Laura Jones** on Monday, June 21st 2027

```
-->
<!--HEADER BOXED FONT WHITE TRANSPARENT...
<div class="header-black-bg"></div>
<!--NEED FOR TRANSPARENT HEADER ON MOBILE...
▶ <header id="nav" class="header header-1">
  <!--FEATURES 7 HALF IMG-->
  ▶ <div class="page-section bg-gray-light cleartfix">
    ::before
    ▶ <div class="fes7-img-cont col-md-1">
      <div class="fes7-img" style="background-image: url('https://...');">
      </div>
    ▶ <div class="container">□</div>
    ::after
  </div>
```

All modern websites and web applications are built using three *fundamental* technologies: HTML, CSS and JavaScript. These are the languages of the web.

In this post, let's focus on HTML. We will learn what HTML is all about, and why should learn it.

### What is HTML?

*Lorem ipsum dolor sit amet consectetur adipisicing elit. Quam recusandae reprehenderit vitae ratione veritatis corrupti sit ut vero, dolores nulla exercitationem eos quod iusto incident, perferendis alias tenetur. Est, vel!*

In HTML, each element is made up of 3 parts:

1. **The opening tag**
2. **The closing tag**
3. **The actual element**

You can learn more at the [MDN Web Docs](#).

# INLINE ELEMENTS

- 👉 Occupies only the space **necessary for its content**
- 👉 Causes **no line-breaks** after or before the element
- 👉 Box model applies in a different way: **heights and widths do not apply**
- 👉 **Paddings and margins** are applied **only horizontally** (left and right)

**Default elements:** a, img, strong, em, button, etc.

**With CSS:** display: inline

## The Basic Language of the Web: HTML



Posted by **Laura Jones** on Monday, June 21st 2027

```
-->
<!--HEADER BOXED FONT WHITE TRANSPARENT-->
<div class="header-black-bg"></div>
<!--NEED FOR TRANSPARENT HEADER ON MOBILE-->
▶ <header id="nav" class="header header-1 header-transparent">
  <!--FEATURES 7 HALF IMG-->
  ▶ <div class="page-section bg-gray-light clear">
    ::before
    ▶ <div class="fes7-img-cont col-md-5">
      <div class="fes7-img" style="background-image: url('https://www.google.com/chrome%20transparent%20header%20mobile%20background%20image.jpg');"></div>
    <div class="container">□</div>
    ::after
  </div>
```

All modern websites and web applications are built using three **fundamental** technologies: HTML, CSS and JavaScript. These are the languages of the web.

In this post, let's focus on HTML. We will learn what HTML is all **about**, and why you should learn it.

### What is HTML?

Lorem ipsum dolor sit amet consectetur adipisicing elit. Quam recusandae reprehenderit vitae ratione veritatis corrupti sit ut vero, dolores nulla exercitationem eos quod iusto incident, preferendis alias tenetur. Est, vel!

In HTML, each element is made up of 3 parts:

1. The opening tag
2. The closing tag
3. The actual element

You can learn more at the [MDN Web Docs](#).

# SUMMARY: INLINE, BLOCK-LEVEL AND INLINE-BLOCK BOXES

## BLOCK-LEVEL BOXES

- 👉 Elements formatted visually as blocks
- 👉 100% of parent's width
- 👉 Vertically, one after another
- 👉 Box-model applies as showed

## INLINE-BLOCK BOXES

- 👉 Looks like inline from the **outside**, behaves like block-level on the **inside**
- 👉 Occupies only content's space
- 👉 Causes no line-breaks
- 👉 Box-model applies as showed

display: inline-block

## INLINE BOXES

- 👉 Occupies only content's space
- 👉 Causes no line-breaks
- 👉 Box model is different: heights and widths do not apply
- 👉 Paddings and margins only horizontal (left and right)





# BUILD RESPONSIVE REAL-WORLD WEBSITES WITH HTML AND CSS

SECTION  
CSS FUNDAMENTALS

LECTURE  
CSS THEORY #5: ABSOLUTE  
POSITIONING

# NORMAL FLOW VS. ABSOLUTE POSITIONING

## NORMAL FLOW

- 👉 Default positioning
- 👉 Element is “**in flow**”
- 👉 Elements are simply laid out according to their order in the HTML code

**Default positioning**  
`position: relative`

## ABSOLUTE POSITIONING

- 👉 Element is removed from the normal flow: “**out of flow**”
- 👉 No impact on surrounding elements, might overlap them
- 👉 We use top, bottom, left, or right to offset the element from its **relatively positioned container**

`position: absolute`

# UNDERSTANDING ABSOLUTE POSITIONING





# SECTION 04 – LAYOUTS: FLOATS, FLEXBOX, AND CSS GRID FUNDAMENTALS



# BUILD RESPONSIVE REAL-WORLD WEBSITES WITH HTML AND CSS

## SECTION

LAYOUTS: FLOATS, FLEXBOX, AND  
CSS GRID FUNDAMENTALS

## LECTURE

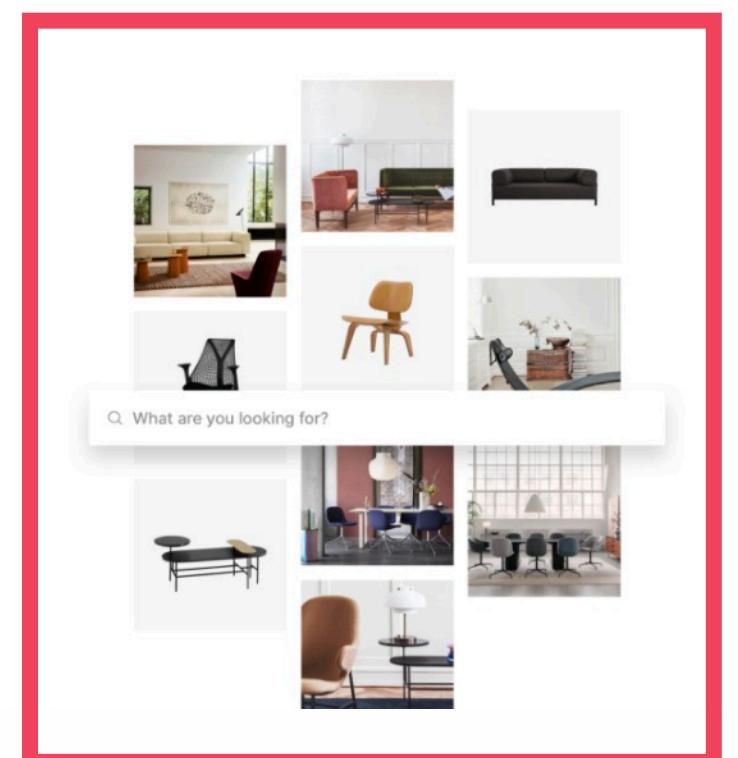
THE 3 WAYS OF BUILDING  
LAYOUTS

# WHAT DOES “LAYOUT” MEAN?

## LAYOUT

- 👉 Layout is the way text, images and other content is placed and arranged on a webpage
- 👉 Layout gives the page a visual structure, into which we place our content
- 👉 **Building a layout:** arranging page elements into a visual structure, instead of simply having them placed one after another (normal flow)

The screenshot shows the homepage of Clippings, a platform for interior professionals. At the top, there's a navigation bar with links for 'Platform', 'For', 'Projects', 'Book a demo', 'Sign up', and 'Log in'. The main headline reads 'The new way for interior professionals to buy furniture'. Below the headline are two buttons: 'Sign up' and 'Book a demo →'. To the right of the headline is a photograph of a modern interior space featuring a round table, a chair, and some shelving.



We work with...  
Interior designers



Find furniture for every type of project

- 650+ brands
- Trade pricing
- Source from anywhere
- Free samples

650+ brands  
Browse millions of products from the world's leading brands.

Trade pricing  
See trade pricing and lead times right away. No need to request a quote.

Source from anywhere  
Can't find it on Clippings? Add items from any website with the Clip Tool.

Free samples  
Get free fabric, wood, marble and rug samples.

# PAGE LAYOUT VS. COMPONENT LAYOUT

## PAGE LAYOUT

The new way for interior professionals to buy furniture

Sign up Book a demo →

Find furniture for every type of project

650+ brands

Trade pricing

Source from anywhere

Free samples

We work with... Interior designers

Hospitality  
The Silo Restaurant, London by Nina+Co

650+ brands

Browse millions of products from the world's leading brands.

Trade pricing

See trade pricing and lead times right away. No need to request a quote.

Source from anywhere

Can't find it on Clippings? Add items from any website with the Clip Tool.

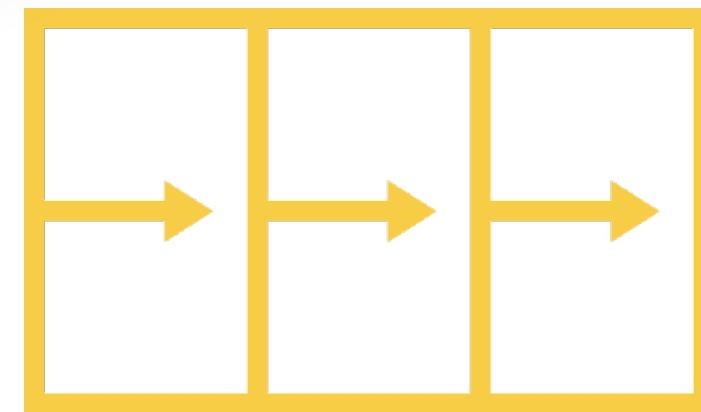
Free samples

Get free fabric, wood, marble and rug samples.

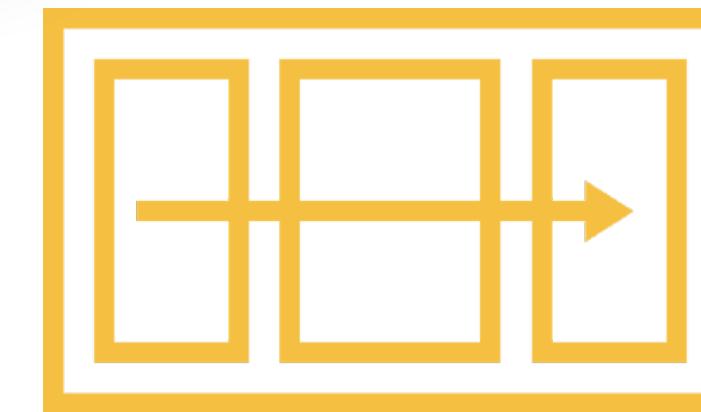
## COMPONENT LAYOUT

# THE 3 WAYS OF BUILDING LAYOUTS WITH CSS

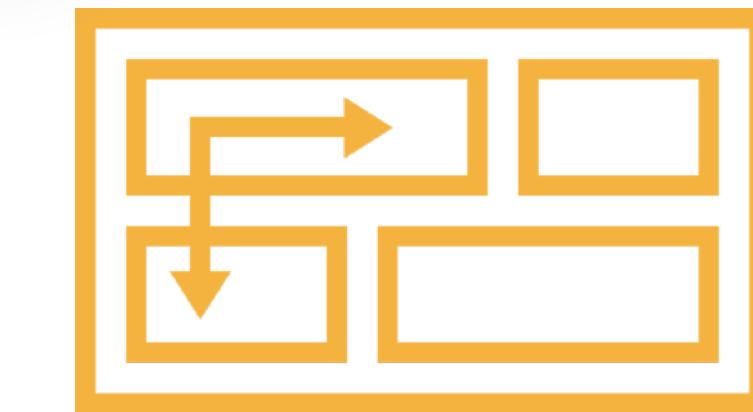
1



2



3



## FLOAT LAYOUTS

The **old way of building layouts** of all sizes, using the `float` CSS property. Still used, but getting outdated fast.

## FLEXBOX

Modern way of laying out elements in a **1-dimensional row** without using floats. Perfect for **component layouts**.

## CSS GRID

For laying out element in a fully-fledged **2-dimensional grid**. Perfect for **page layouts and complex components**.





# BUILD RESPONSIVE REAL-WORLD WEBSITES WITH HTML AND CSS

## SECTION

LAYOUTS: FLOATS, FLEXBOX, AND  
CSS GRID FUNDAMENTALS

LECTURE  
USING FLOATS

# ABSOLUTE POSITIONING VS. FLOATS

## NORMAL FLOW

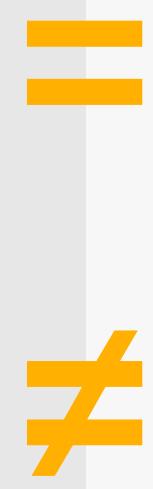
- 👉 Default positioning
- 👉 Element is “**in flow**”
- 👉 Elements are simply laid out according to their order in the HTML code

**Default positioning**  
`position: relative`

## ABSOLUTE POSITIONING

- 👉 Element is removed from the normal flow: “**out of flow**”
- 👉 No impact on surrounding elements, might overlap them
- 👉 We use top, bottom, left, or right to offset the element from its **relatively positioned container**

`position: absolute`



## FLOATS

- 👉 Element is removed from the normal flow: “**out of flow**”
- 👉 Text and inline elements will wrap around the floated element
- 👉 The container will **not** adjust its height to the element

`float: left`  
`float: right`





# BUILD RESPONSIVE REAL-WORLD WEBSITES WITH HTML AND CSS

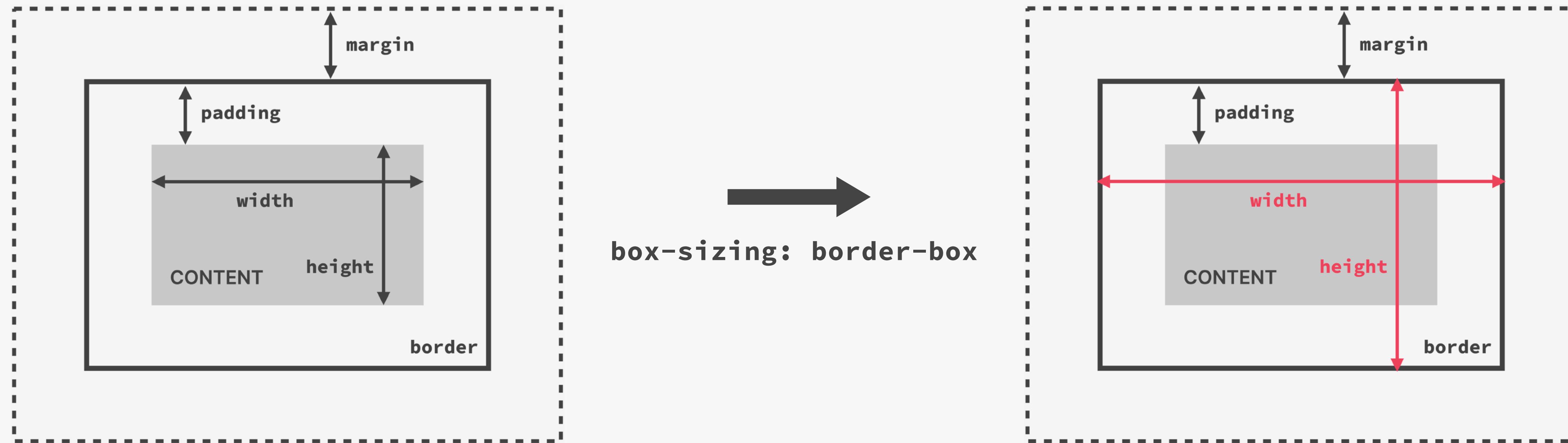
## SECTION

LAYOUTS: FLOATS, FLEXBOX, AND  
CSS GRID FUNDAMENTALS

## LECTURE

BOX-SIZING: BORDER-BOX

# THE BOX MODEL WITH BOX-SIZING: BORDER-BOX



**Final element width** = ~~right border + right padding + width + left padding + left border~~

**Final element height** = ~~top border + top padding + height + bottom padding + bottom border~~





# BUILD RESPONSIVE REAL-WORLD WEBSITES WITH HTML AND CSS

## SECTION

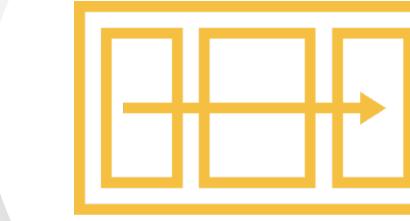
LAYOUTS: FLOATS, FLEXBOX, AND  
CSS GRID FUNDAMENTALS

## LECTURE

A FLEXBOX OVERVIEW

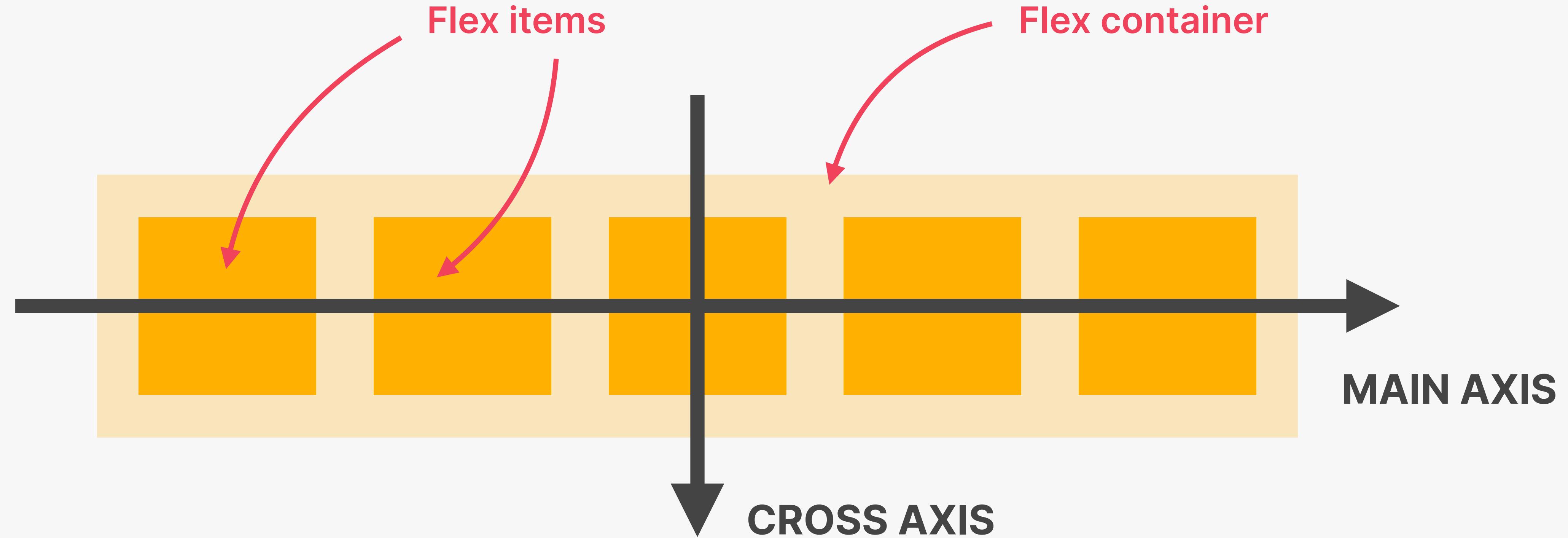
# WHAT IS FLEXBOX?

## FLEXBOX



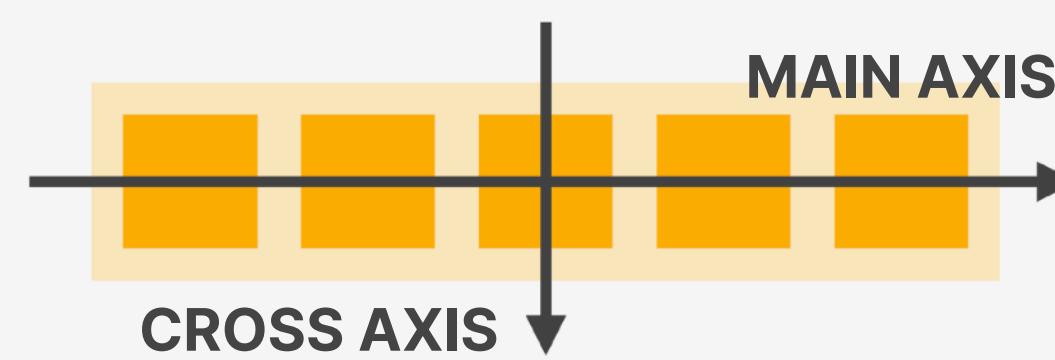
- 👉 Flexbox is a set of related **CSS properties** for **building 1-dimensional layouts**
- 👉 The main idea behind flexbox is that empty space inside a container element can be **automatically divided** by its child elements
- 👉 Flexbox makes it easy to automatically **align items to one another** inside a parent container, both horizontally and vertically
- 👉 Flexbox solves common problems such as **vertical centering** and creating **equal-height columns**
- 👉 Flexbox is perfect for **replacing floats**, allowing us to write fewer and cleaner HTML and CSS code

# FLEXBOX TERMINOLOGY



`display: flex`

## FLEX CONTAINER



## FLEX ITEMS

1 `gap: 0 | <length>`

👉 To create **space between items**, without using margin

2 `justify-content: flex-start | flex-end | center | space-between | space-around | space-evenly`

👉 To align items along main axis (**horizontally**, by default)

3 `align-items: stretch | flex-start | flex-end | center | baseline`

👉 To align items along cross axis (**vertically**, by default)

4 `flex-direction: row | row-reverse | column | column-reverse`

👉 To define which is the **main axis**

5 `flex-wrap: nowrap | wrap | wrap-reverse`

👉 To allow items to **wrap into a new line** if they are too large

6 `align-content: stretch | flex-start | flex-end | center | space-between | space-around`

👉 Only applies when there are **multiple lines** (flex-wrap: wrap)

1 `align-self: auto | stretch | flex-start | flex-end | center | baseline`

👉 To **overwrite align-items** for individual flex items

2 `flex-grow: 0 | <integer>`

👉 To allow an element **to grow** (0 means no, 1+ means yes)

3 `flex-shrink: 1 | <integer>`

👉 To allow an element **to shrink** (0 means no, 1+ means yes)

4 `flex-basis: auto | <length>`

👉 To define an item's width, **instead of the width property**

5 `flex: 0 1 auto | <int> <int> <len>`

👉 **Recommended** shorthand for flex-grow, -shrink, -basis.

6 `order: 0 | <integer>`

👉 Controls order of items. -1 makes item **first**, 1 makes it **last**





# BUILD RESPONSIVE REAL-WORLD WEBSITES WITH HTML AND CSS

## SECTION

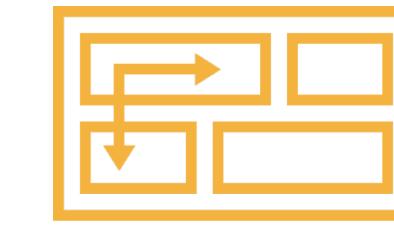
LAYOUTS: FLOATS, FLEXBOX, AND  
CSS GRID FUNDAMENTALS

## LECTURE

A CSS GRID OVERVIEW

# WHAT IS CSS GRID?

## CSS GRID



- 👉 CSS Grid is a set of **CSS properties** for **building 2-dimensional layouts**
- 👉 The main idea behind CSS Grid is that we **divide a container element into rows and columns** that can be filled with its child elements
- 👉 In two-dimensional contexts, CSS Grid allows us to write **less nested HTML** and **easier-to-read CSS**
- 👉 CSS Grid is **not meant to replace flexbox!** Instead, they work perfectly together. Need a **1D** layout? Use flexbox. Need a **2D** layout? Use CSS Grid.

