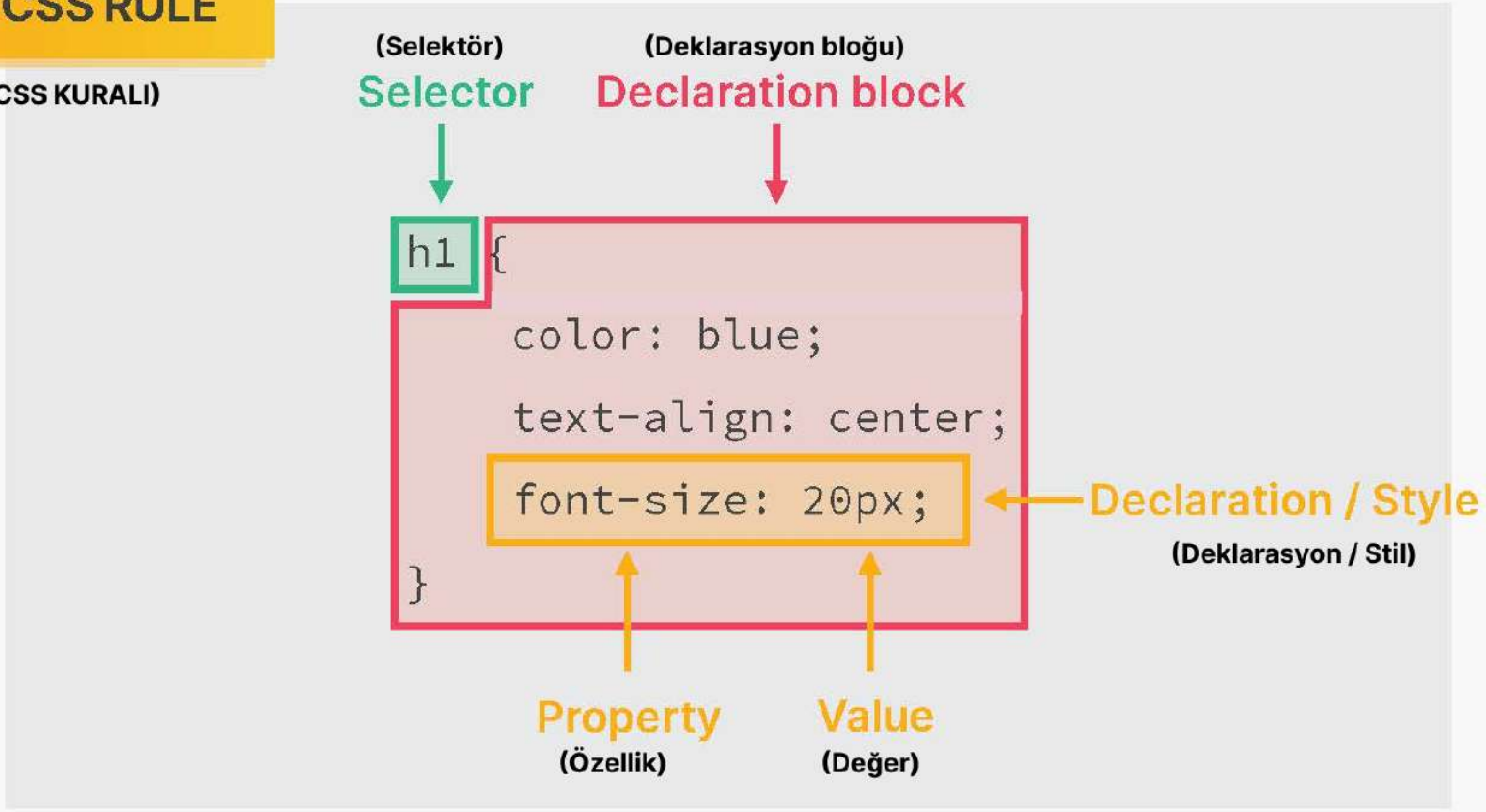
HOW WE **SELECT** AND **STYLE** ELEMENTS

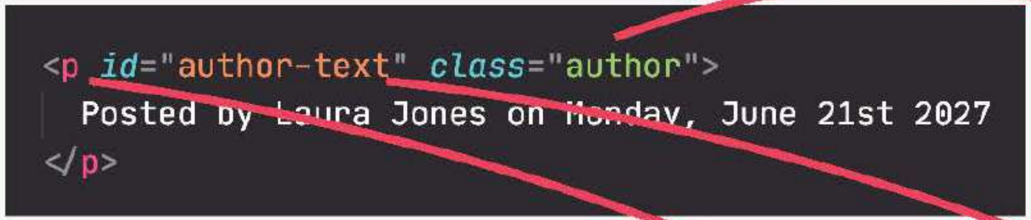
A CSS RULE

(CSS KURALI)

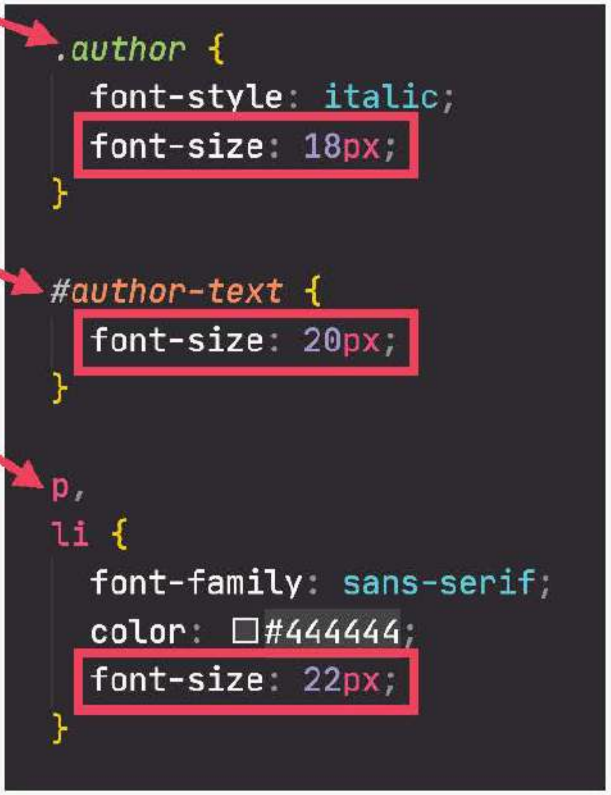


```
h1 {
    color: blue;
    text-align: center;
    font-size: 20px;
}
```

CONFLICTING SELECTORS AND DECLARATIONS



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



```css
.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?**

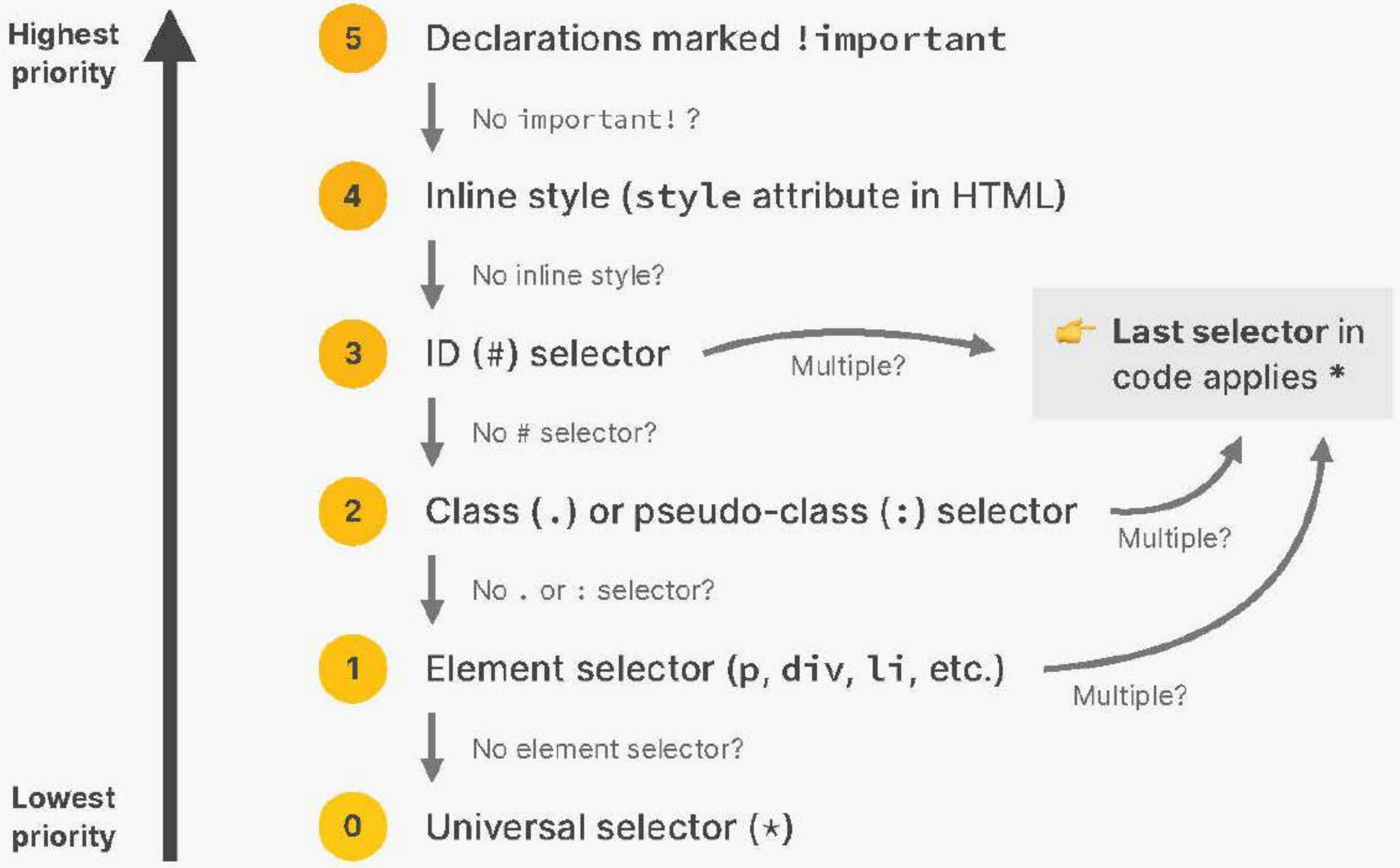
🤓 ***All of them. All rules and properties are applied!***

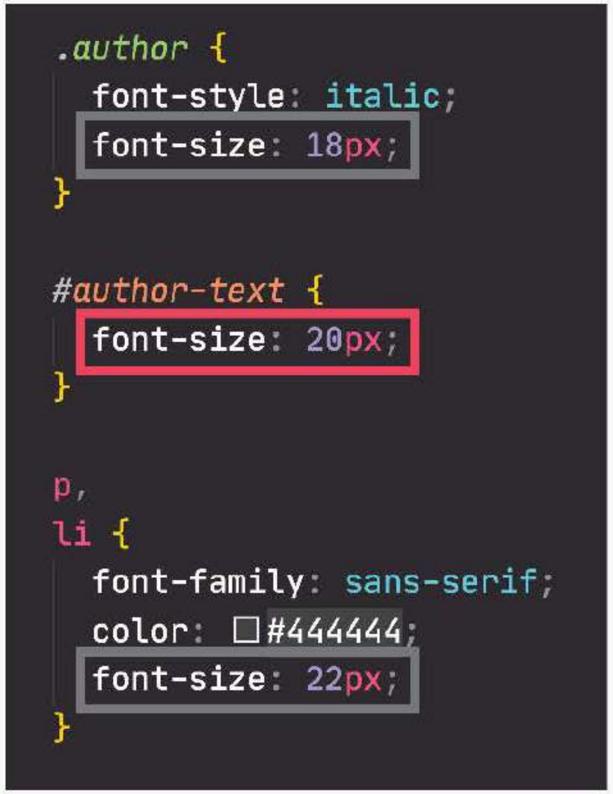
⬇

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

🤓 *Let's see how it works...*

RESOLVING CONFLICTING DECLARATIONS



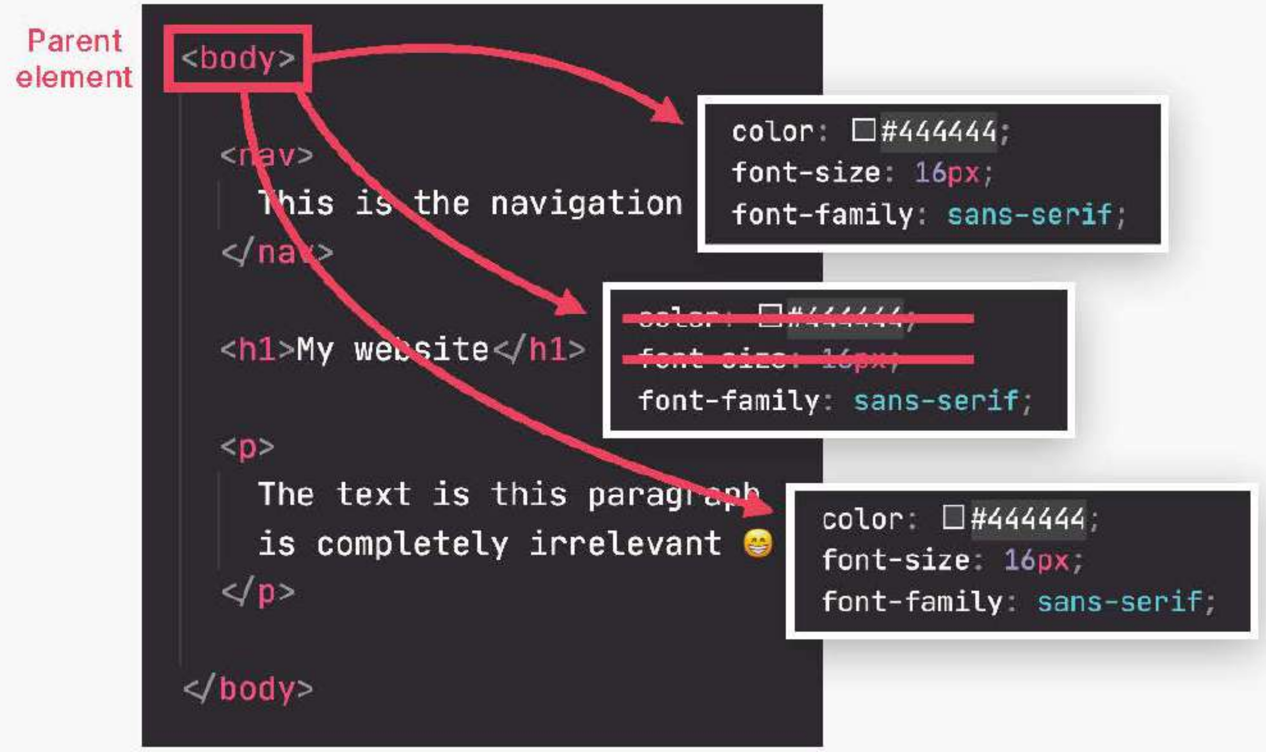


```
.author {
  font-style: italic;
  font-size: 18px;
}

#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

HOW INHERITANCE WORKS

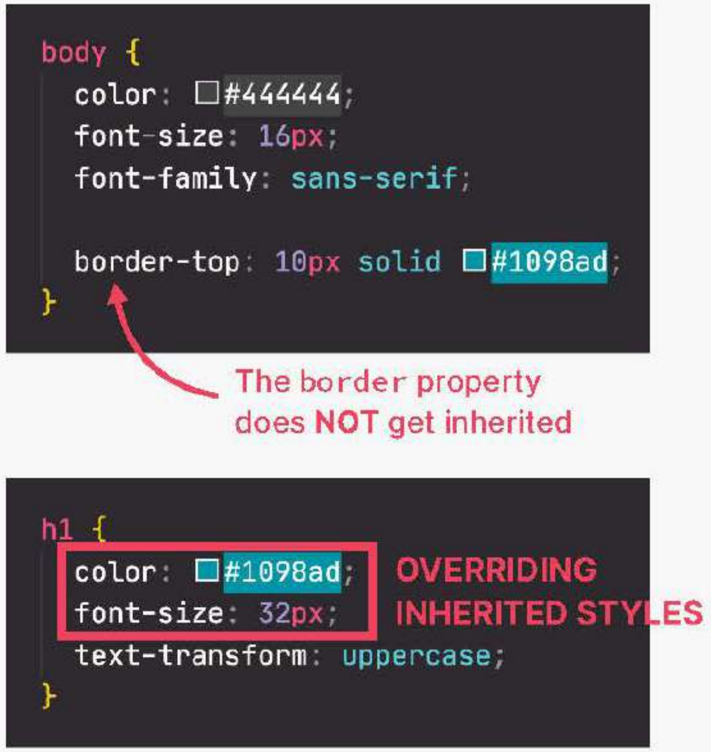


```html
<body>

  <nav>
    This is the navigation
  </nav>

  <h1>My website</h1>

  <p>
    The text is this paragraph
    is completely irrelevant 😁
  </p>

</body>
```

```css
color: #444444;
font-size: 16px;
font-family: sans-serif;
```

```css
~~color: #444444;~~
~~font-size: 16px;~~
font-family: sans-serif;
```

```css
color: #444444;
font-size: 16px;
font-family: sans-serif;
```



```css
body {
  color: #444444;
  font-size: 16px;
  font-family: sans-serif;

  border-top: 10px solid #1098ad;
}
```

The border property
does NOT get inherited

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

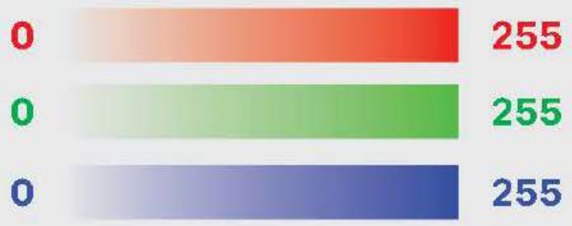
OVERRIDING INHERITED STYLES

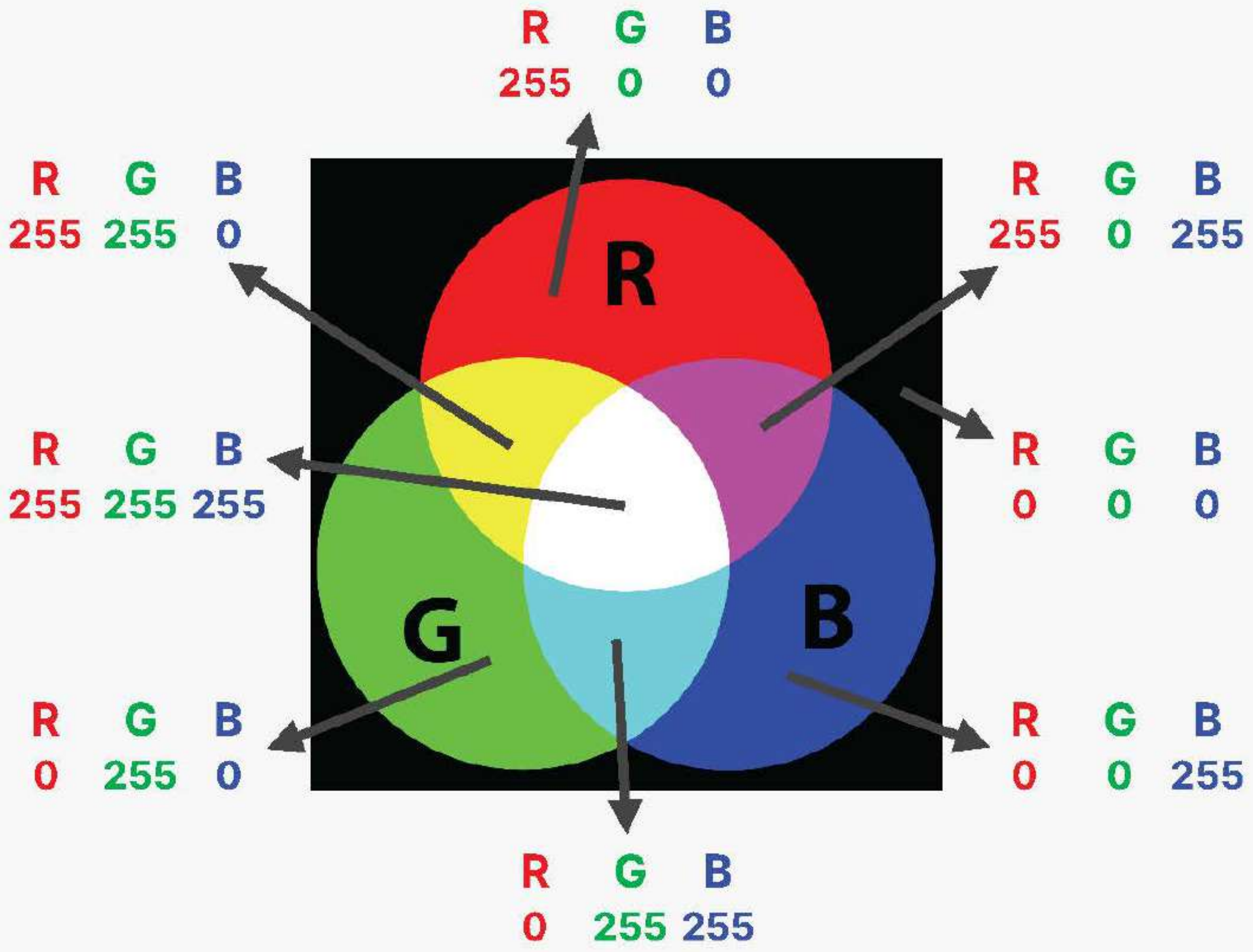
☝️ Not all properties get inherited. It's mostly ones **related to text**: `font-family`, `font-size`, `font-weight`, `font-style`, `color`, `line-height`, `letter-spacing`, `text-align`, `text-transform`, `text-shadow`, `list-style`, etc.

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)
```

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

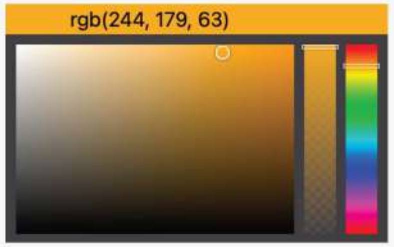
```
#0ff
```

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

```
#f4b33f
```

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

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



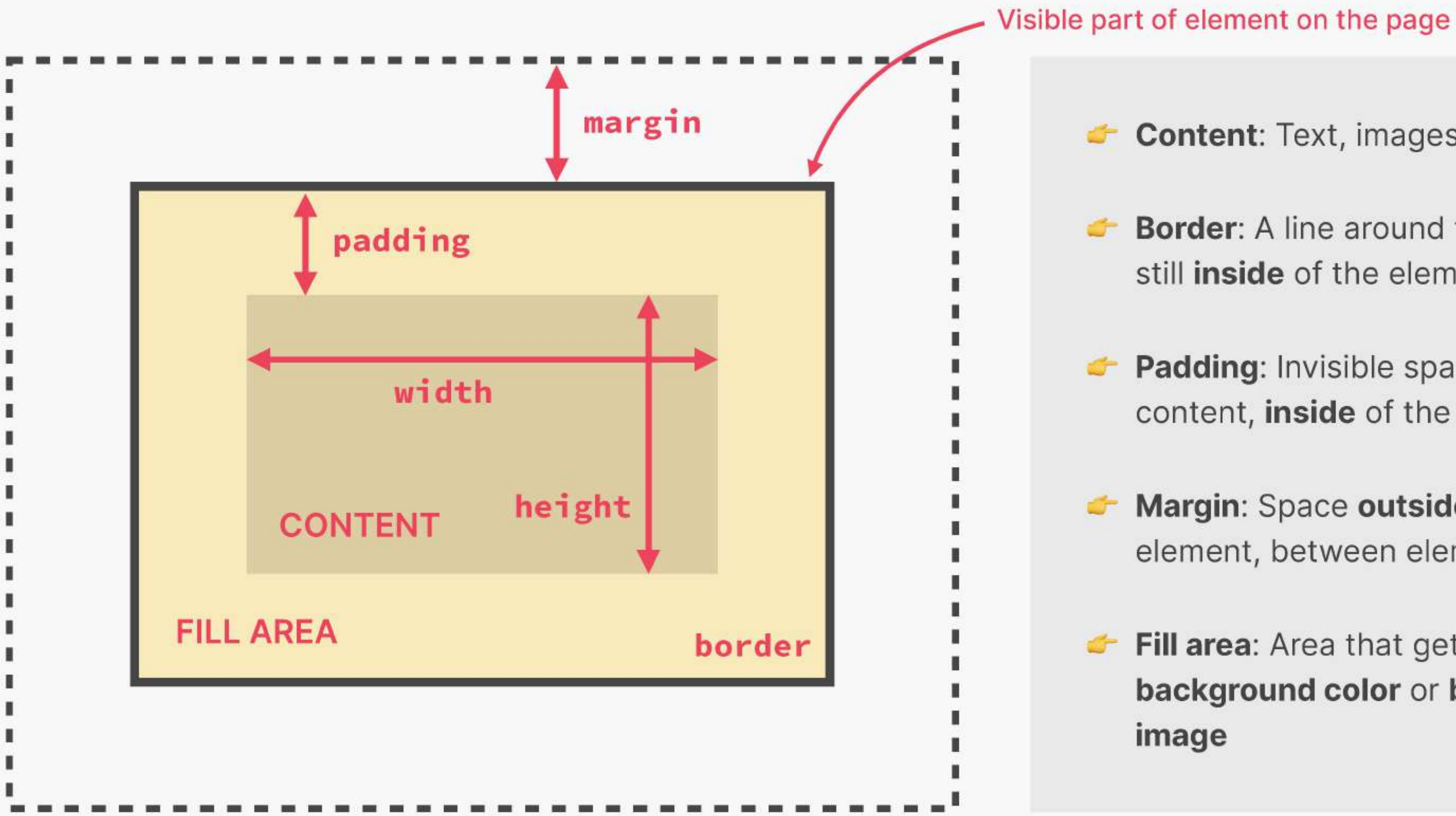
- Color picker in VS Code

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

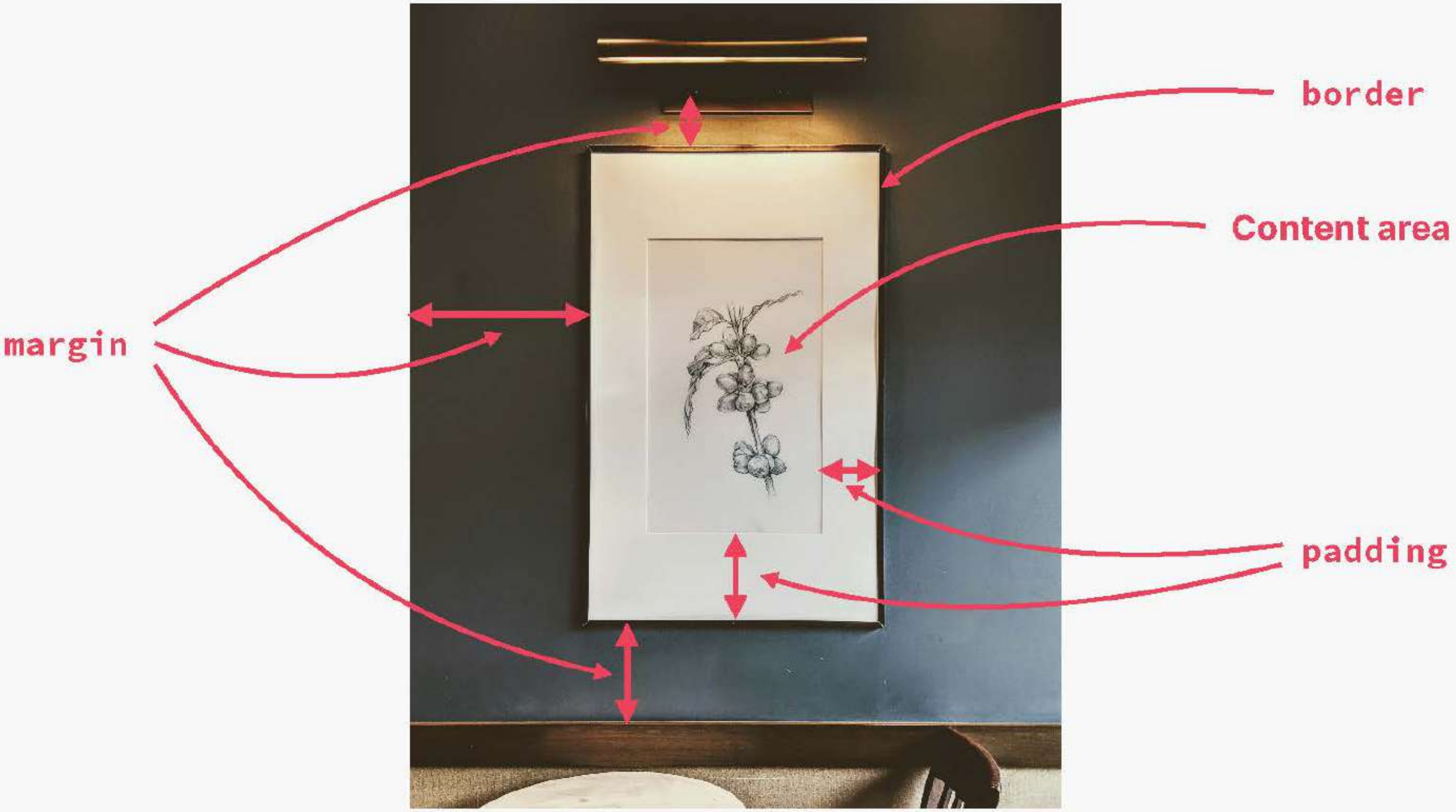
```
rgb(0, 0, 0) / #000000 / #000
rgb(69, 69, 69) / #444444 / #444
rgb(183, 183, 183) / #b7b7b7
rgb(255, 255, 255) / #ffffff / #fff
```

THE **CSS BOX MODEL**



- 👉 **Content**: Text, images, etc.
- 👉 **Border**: A line around the element, still **inside** of the element
- 👉 **Padding**: Invisible space around the content, **inside** of the element
- 👉 **Margin**: Space **outside** of the element, between elements
- 👉 **Fill area**: Area that gets filled with **background color** or **background image**

ANALOGY FOR THE CSS BOX MODEL

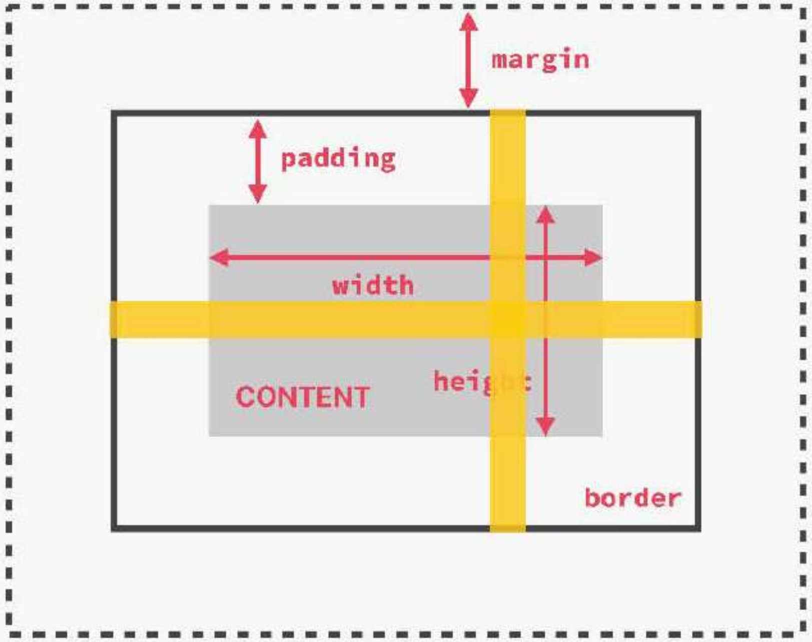


ELEMENT **HEIGHT** AND **WIDTH** CALCULATION

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

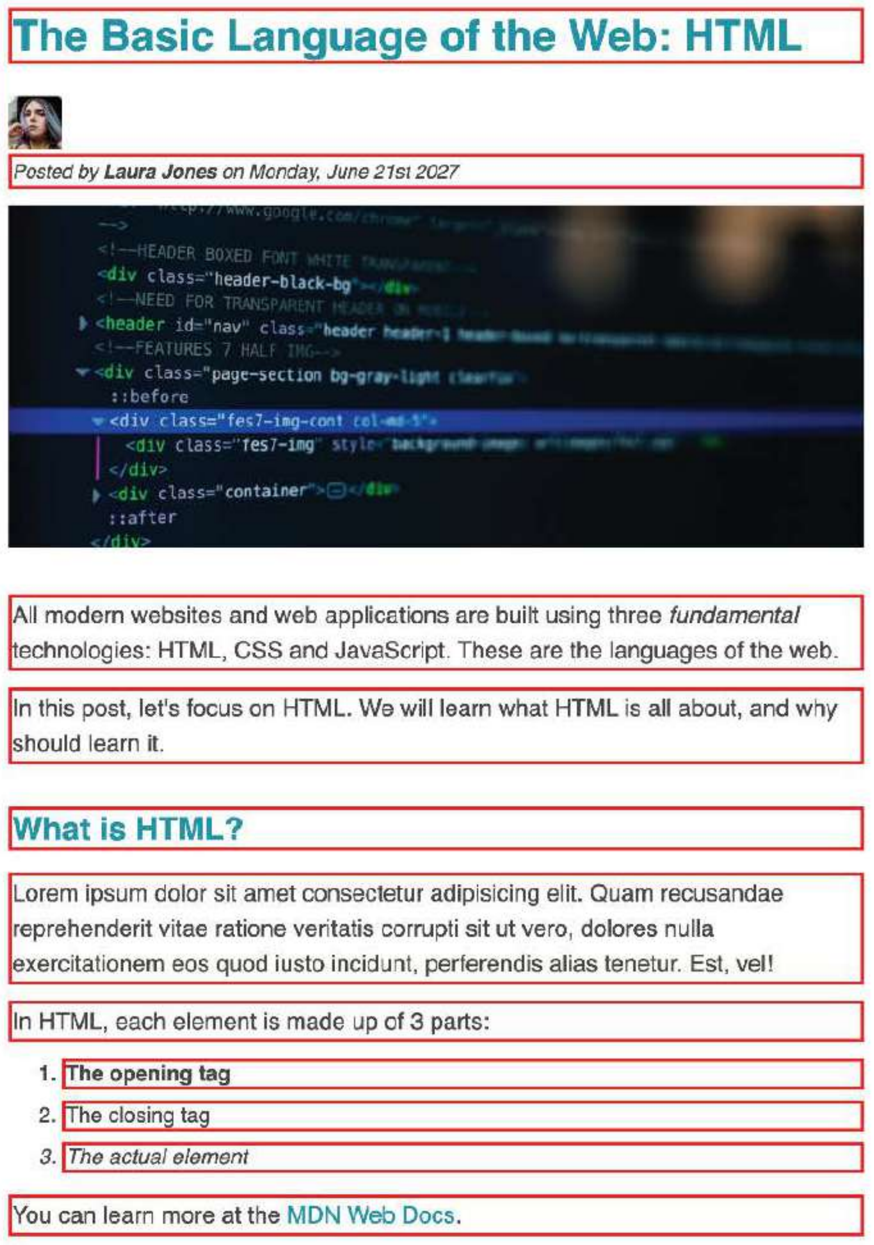


BLOCK-LEVEL ELEMENTS

- 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*

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 incidunt, 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*

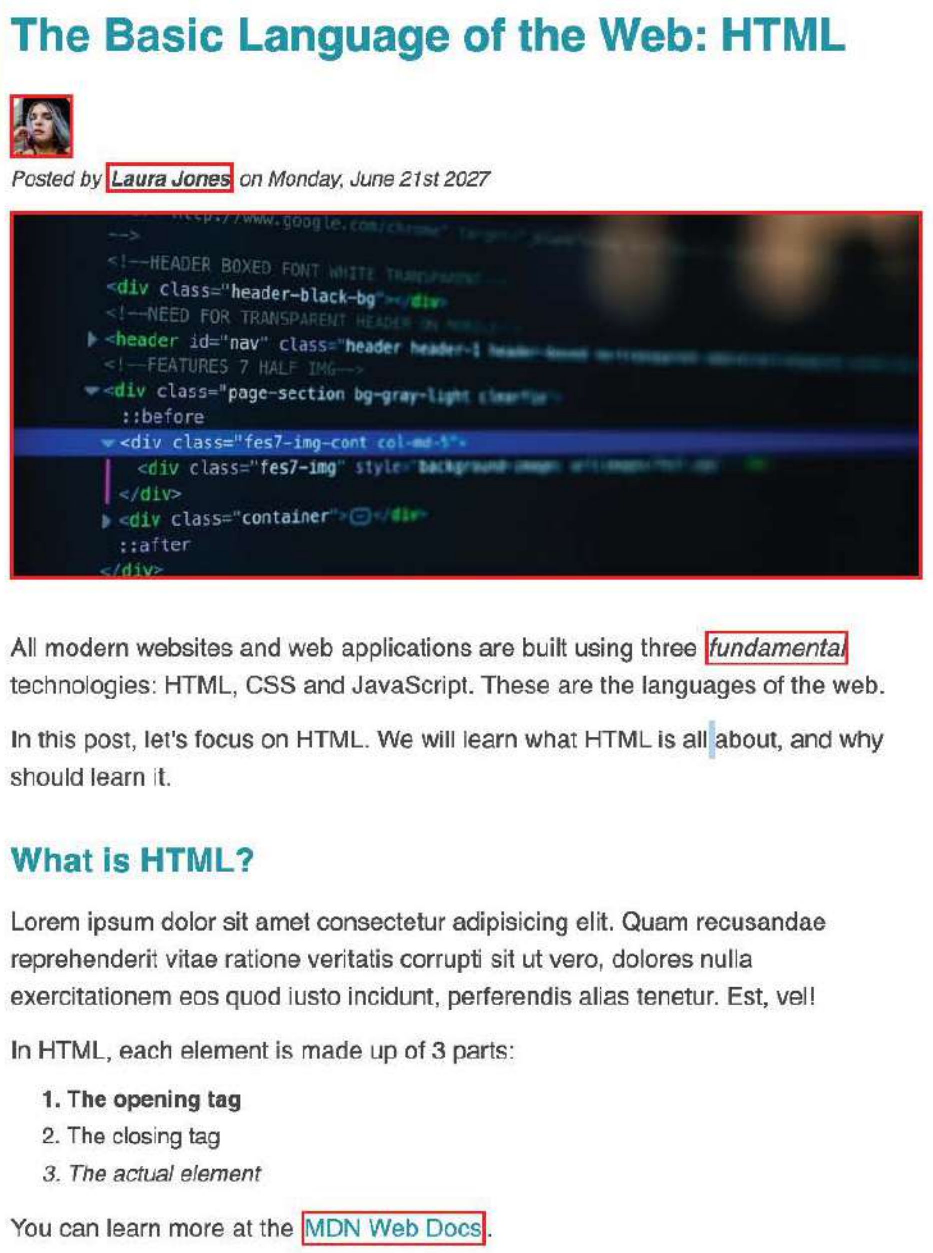
You can learn more at the MDN Web Docs.

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
<!--FEATURES 7 HALF IMG-->
<div class="page-section bg-gray-light clearfix">
  ::before
  <div class="fes7-img-cont col-md-5">
    <div class="fes7-img" style="background-image
  </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 incidunt, 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.

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)

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

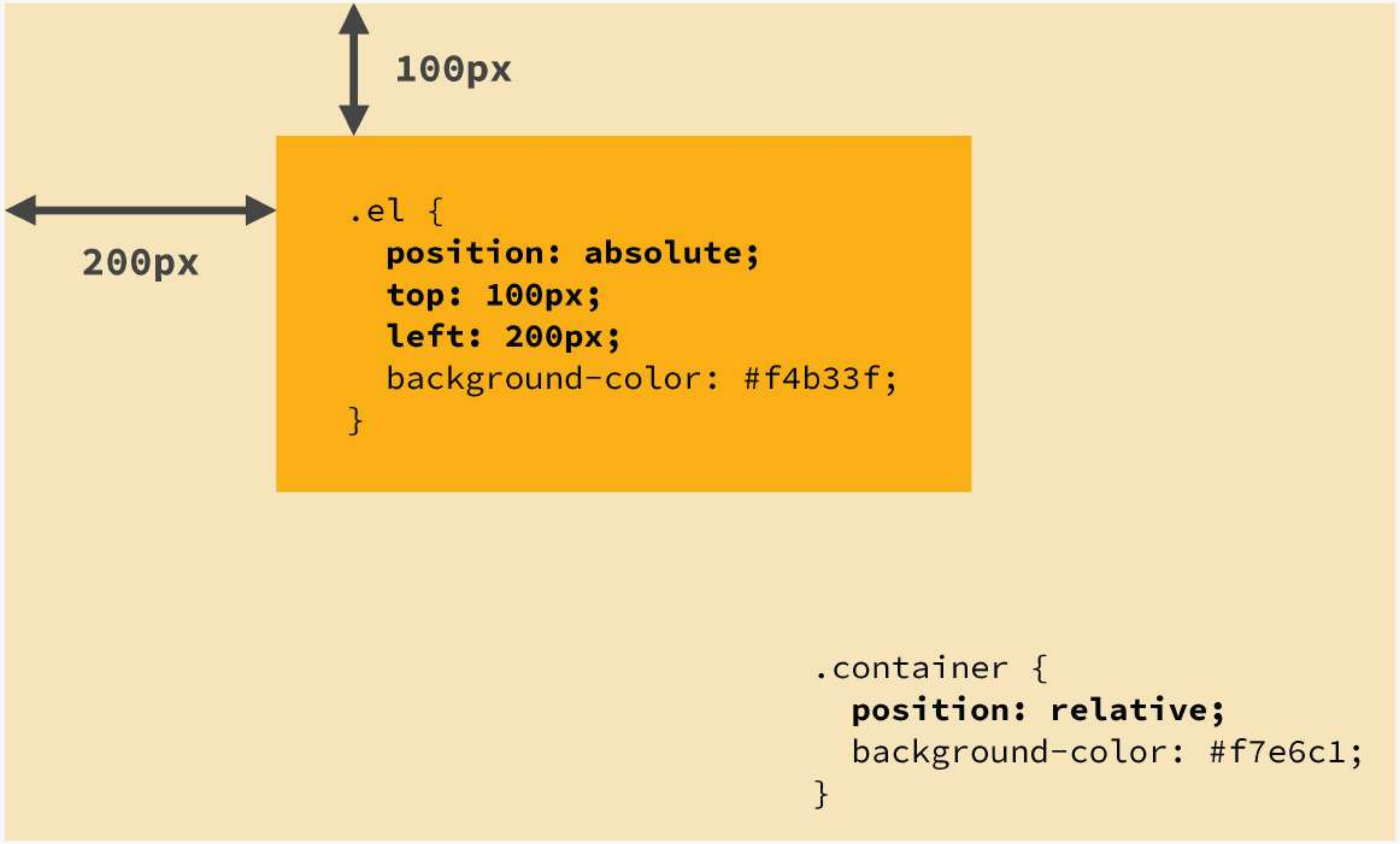
```
Default positioning
position: relative
```

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



```
.el {
  position: absolute;
  top: 100px;
  left: 200px;
  background-color: #f4b33f;
}
```

```
.container {
  position: relative;
  background-color: #f7e6c1;
}
```