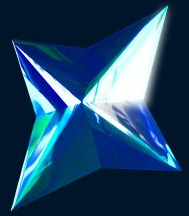
BRO Academy



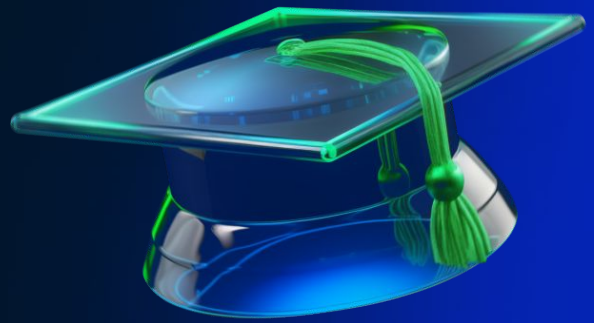
1

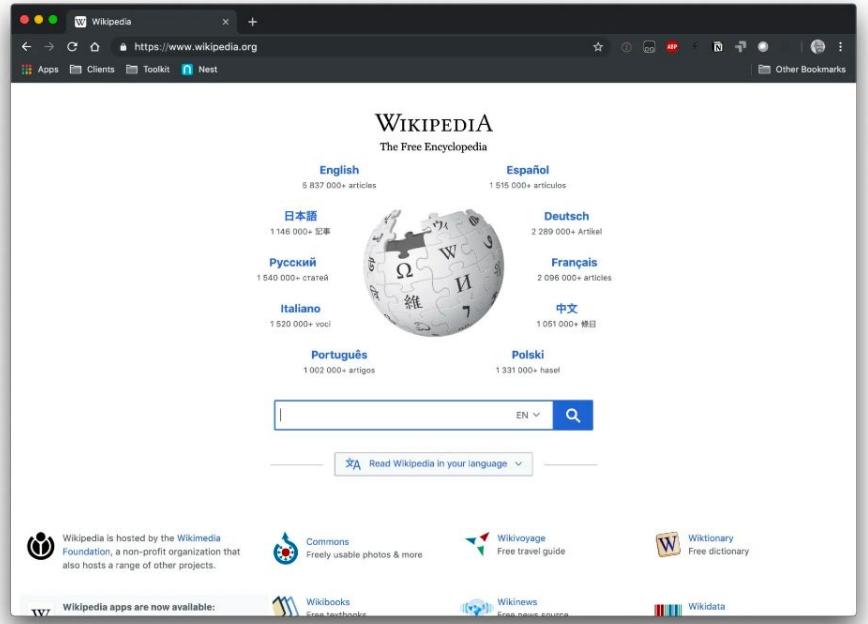
# Introduction to CSS

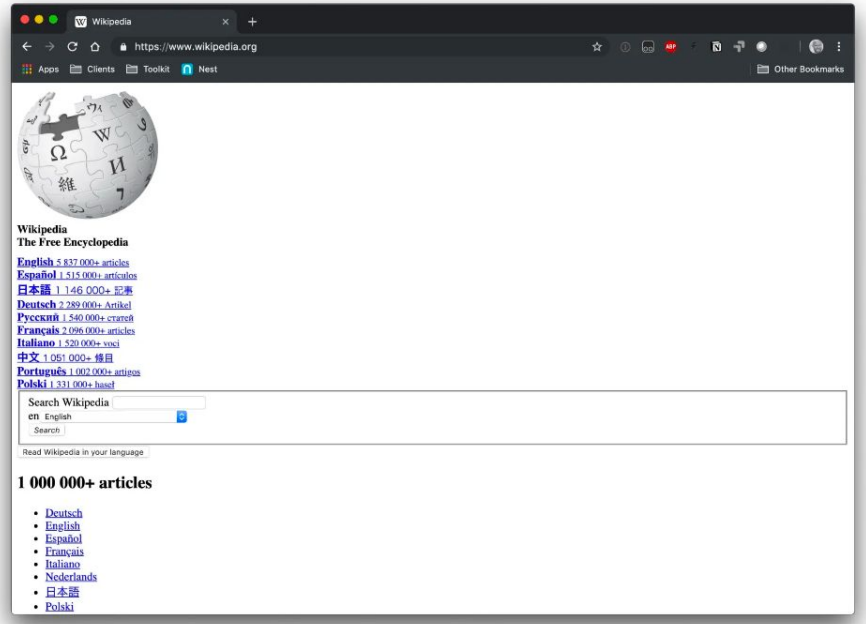
HTML course

# CSS: Lesson Plan

1. What is CSS
2. Where to write
3. How to write
4. How it will be applied




Wikipedia
https://www.wikipedia.org
Apps
Clients
Toolkit
Nest
Other Bookmarks
WIKIPEDIA
The Free Encyclopedia
English
5 837 000+ articles
Español
1 515 000+ artículos
日本語
1 146 000+ 記事
Deutsch
2 289 000+ Artikel
Русский
1 540 000+ статей
Français
2 096 000+ articles
Italiano
1 520 000+ voci
中文
1 051 000+ 條目
Português
1 002 000+ artigos
Polski
1 331 000+ haseł
EN
Read Wikipedia in your language
Wikipedia is hosted by the Wikimedia Foundation, a non-profit organization that also hosts a range of other projects.
Commons
Freely usable photos & more
Wikivoyage
Free travel guide
Wiktionary
Free dictionary
Wikipedia apps are now available:
Wikibooks
Wikinews
Wikidata


Wikipedia
https://www.wikipedia.org
Apps
Clients
Toolkit
Nest
Other Bookmarks
Wikipedia
The Free Encyclopedia
English 5 837 000+ articles
Español 1 515 000+ artículos
日本語 1 146 000+ 記事
Deutsch 2 289 000+ Artikel
Русский 1 540 000+ статей
Français 2 096 000+ articles
Italiano 1 520 000+ voci
中文 1 051 000+ 條目
Português 1 002 000+ artigos
Polski 1 331 000+ haseł
Search Wikipedia
en English
Search
Read Wikipedia in your language
1 000 000+ articles
Deutsch
English
Español
Français
Italiano
Nederlands
日本語
Polski

# Cascading style sheet

− is a style sheet language used for specifying the presentation and styling of a document written in a markup language such as HTML

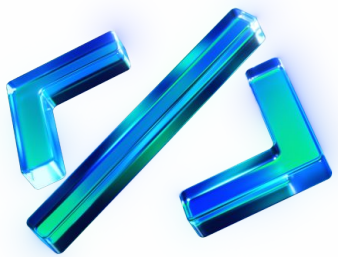
Describes how elements should be displayed on various media such as screens, paper, speech, or other media.

# Connecting styles

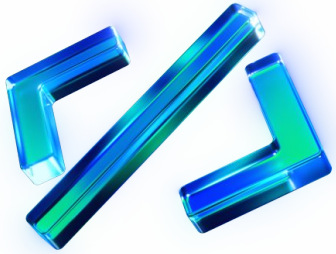
1. Inline
2. Embedded
3. External

# Inline styles

```
<p style="color: red; font-size: 14px;">text</p>
```

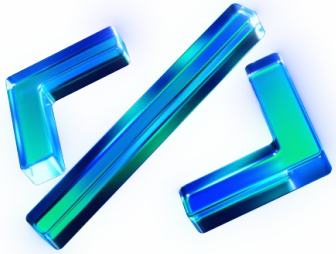


# Embedded styles

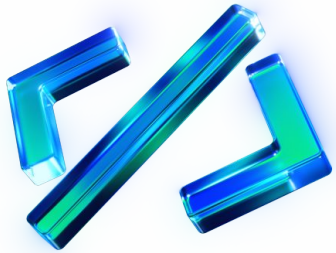


```
<head>
    <style>
        p {
            color: red;
            font-size: 14px;
        }
    </style>
</head>
```

# External styles



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



# Connecting styles

1. Inline
2. Embedded
3. External

# CSS Rules

**Selectors** are expressions that tell the browser which HTML element to apply the CSS properties defined within the style block.

**Inside a rule, selectors and declaration blocks are defined, consisting of properties and their values.**

Types of

# Selectors

- Tag Selector

```
h1 {}
```

- Class Selector

```
.main-heading {}
```

- ID Selector

```
#reasons-section {}
```

- Attribute Selector

```
a[href] {}
```

- Pseudo-classes

```
a:hover {}
```

- Pseudo-elements

```
p::first-line {}
```

- Universal Selector

```
* {}
```

- Grouped Selectors

```
h1, .heading {}
```

# Selector Combinators

| | | | | | |
|---|---|---|---|---|---|
| descendant combinator | **space** | `body article p` | elements are nested inside each other | `<body>`<br>`  <article>`<br>`  <p></p>`<br>`  </article>`<br>`</body>` | `<body>`<br>`  <section>`<br>`  <p></p>`<br>`  </section>`<br>`  <p></p>`<br>`</body>` |
| child combinator | **>** | `article > p` | an element is an immediate child | `<article>`<br>`  <p></p>`<br>`  <a></a>`<br>`</article>` | `<article>`<br>`  <div>`<br>`  <p></p>`<br>`  </div>`<br>`  <p></p>`<br>`</article>` |
| adjacent sibling combinator | **+** | `p + img` | an element that comes immediately after | `<article>`<br>`  <p></p>`<br>`  <img/>`<br>`  <a></a>`<br>`</article>` | `<article>`<br>`  <p></p>`<br>`  <a></a>`<br>`  <img/>`<br>`</article>` |
| general sibling combinator | **~** | `p ~ img` | an element located anywhere below in the code, inside a common parent | `<article>`<br>`  <p></p>`<br>`  <a></a>`<br>`  <img/>`<br>`</article>` | `<article>`<br>`  <p></p>`<br>`  <a>`<br>`  <img/>`<br>`  </a>`<br>`</article>` |

Selector

# Combinations

| | |
|---|---|
| **p {}** | `<p class="intro module"></p>`<br>`<div class="intro"></div>`<br>`<p class="default-text"></p>` |
| **p a { }** | `<p>`<br>`  <a></a>`<br>`</p>`<br><br>`<div>`<br>`  <a></a>`<br>`</div>` |
| **.intro {}** | `<p class="intro"></p>`<br>`<div class="intro"></div>` |
| **div.intro { }** | `<p class="intro"></p>`<br>`<div class="intro"></div>` |
| **.intro.module { }** | `<p class="module"></p>`<br>`<p class="module intro"></p>`<br>`<p class="intro module"></p>`<br>`<p class="intro"></p>` |

| | |
|---|---|
| **.intro .module { }** | `<p class="intro module"></p>`<br>`<div class="intro ~~module~~">`<br>`  <p class="module"></p>`<br>`</div>` |
| **.intro, .module { }** | `<p class="intro"></p>`<br>`<div class="module"></div>` |
| **.intro {} ≠ .Intro {}** | `<p class="intro"></p>`<br>`<div class="Intro"></div>` |
| `.nav ul li a { }` | **`.nav a { }`** |
| `/* comment */` | |
| Selectors are read from right to left<br>**nav ul li** {} | The rule applies to the **li** element, which is part of the **ul** element, which is inside the **nav** element. |

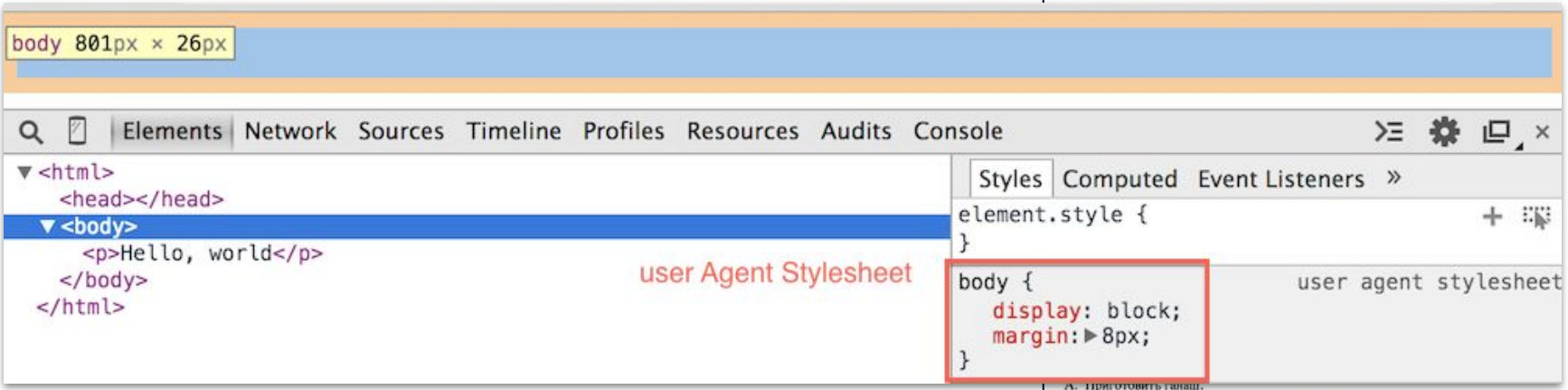
# Applying Styles from different sources

1. Browser Styles
2. User Styles
3. Author Styles

# Default Browser Styles

## User agent stylesheet

– styles present in any browser.




**Способ приготовления**

**Шаг 1**

**Как получить миндальную муку:**

- Миндальная мука можно приготовить следующим образом:
- В магазине, где продают орехи можно попросить, чтобы их смололи в муку (они это делают в кофемолке).
- Просеять ее.
- Миндаль должен быть очищенным, если вы делаете белые макаруны.
- Если шоколадные макаруны, то можно миндаль брать обычный.

**Как делать муку для макарун:**

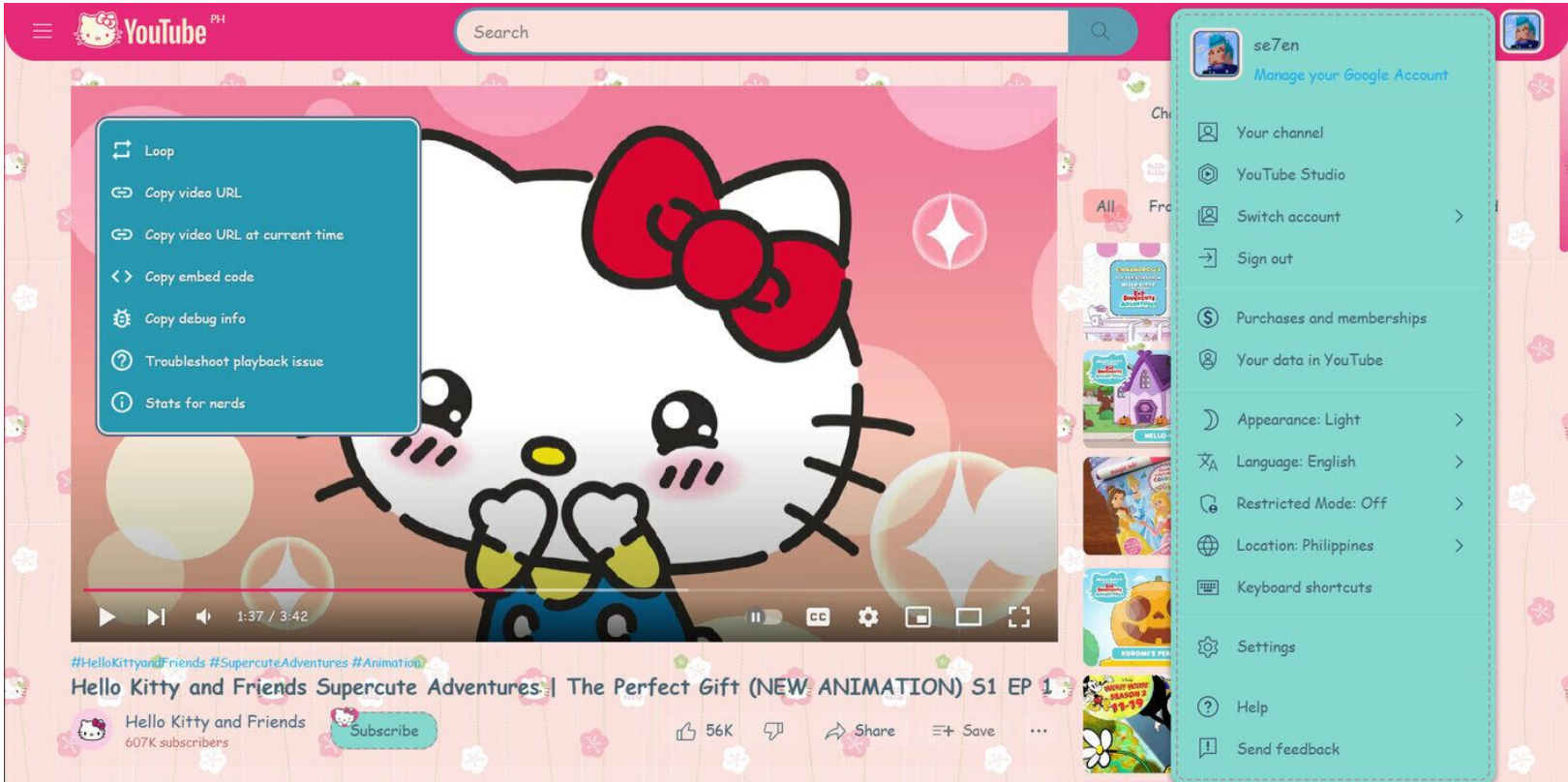
1. Нагреть духовку до 150 градусов.
2. Смешать миндальную муку, сахарную пудру и какао в комбайне, 2 минуты.
3. Противень застелить бумагой для выпечки высыпать сухую смесь на лист, просушить в духовке в течение 5 минут.
4. Просеять через очень мелкое сито.

**Шаг 2**

A. Приготовить ганаш.
B. Сливки подогреть, добавить поломанный на кусочки шоколад.
C. Размешать до полного растворения.
D. Дать остыть и поставить в холодильник (желательно на ночь).

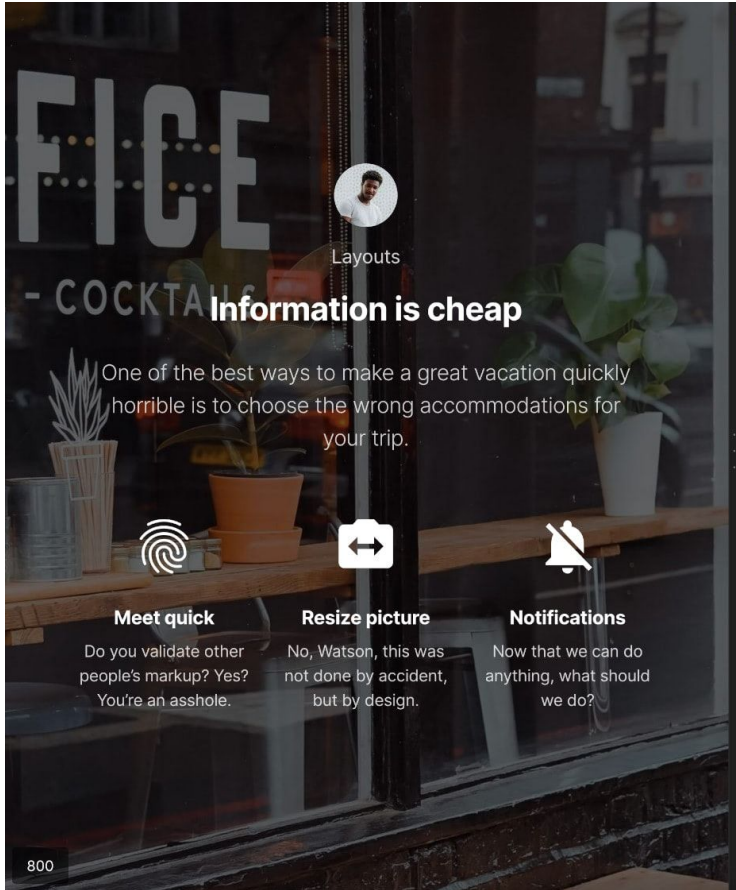
# User styles

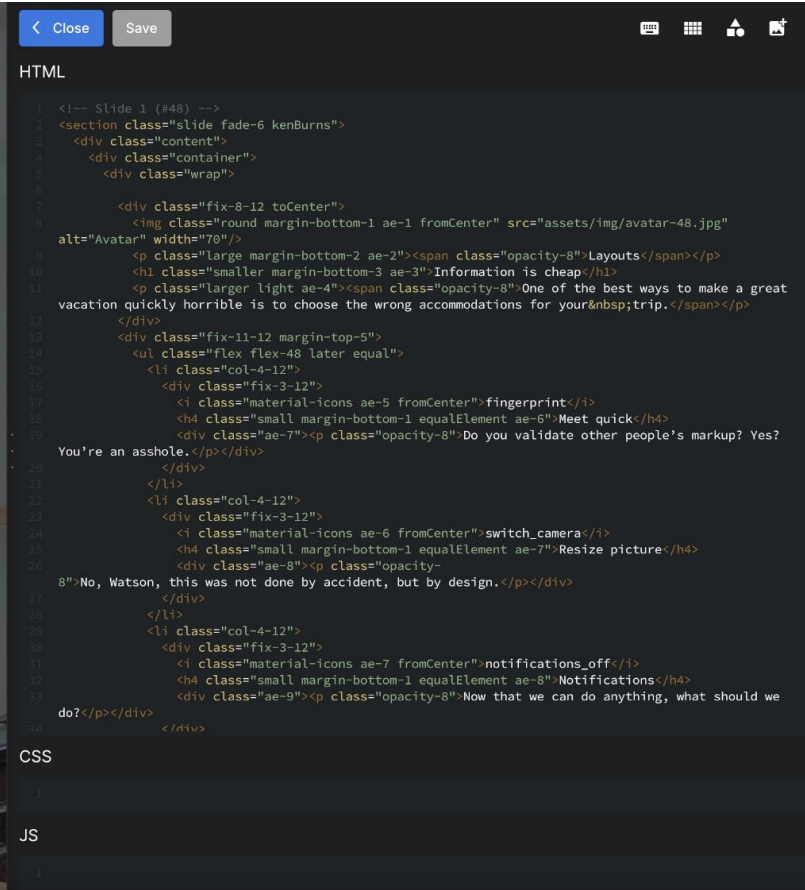
The user can customize fonts, colors, positions of links in the margins, and many other things.



# Author Styles

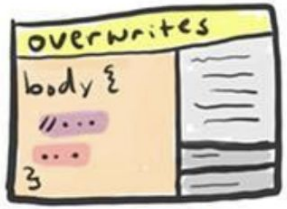
Site styles.



user agent

author styles




local user

least specific → most specific

# Reset and Normalize CSS

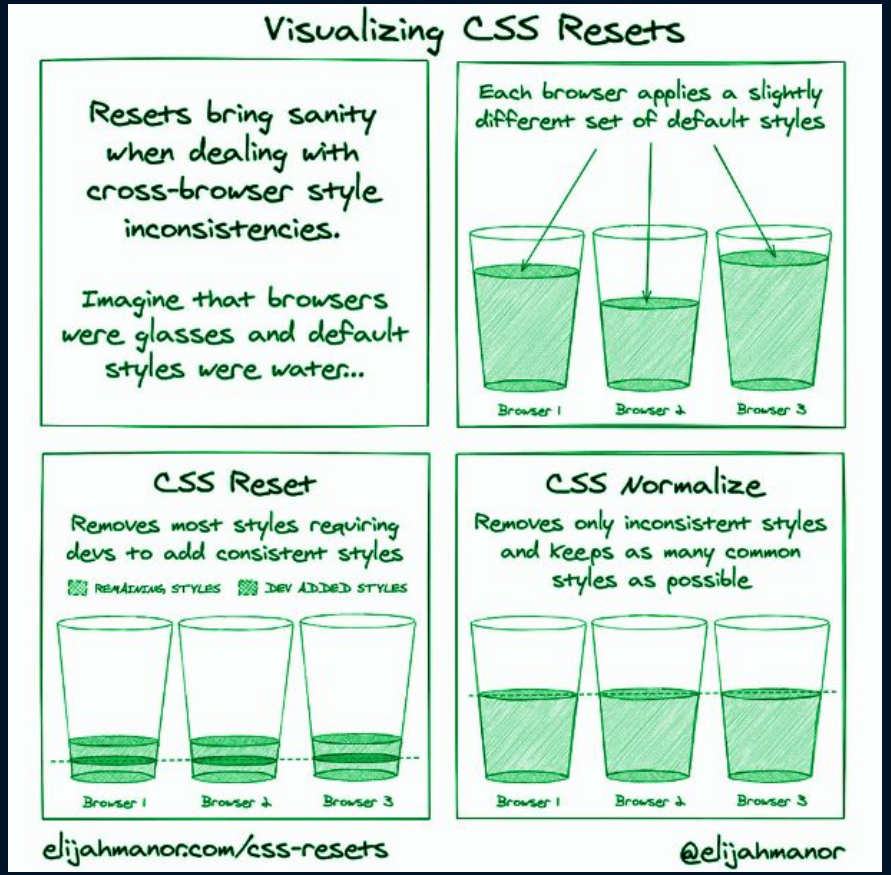
## normalize.css

~~reset.css~~

- Saves useful default styles
- Makes styles equal for each browser
- Makes default styles more accessible
- Includes comments and documentation

Connects to your style file:

```
<link rel="stylesheet" type="text/css"
href="https://necolas.github.io/normalize.css">
```



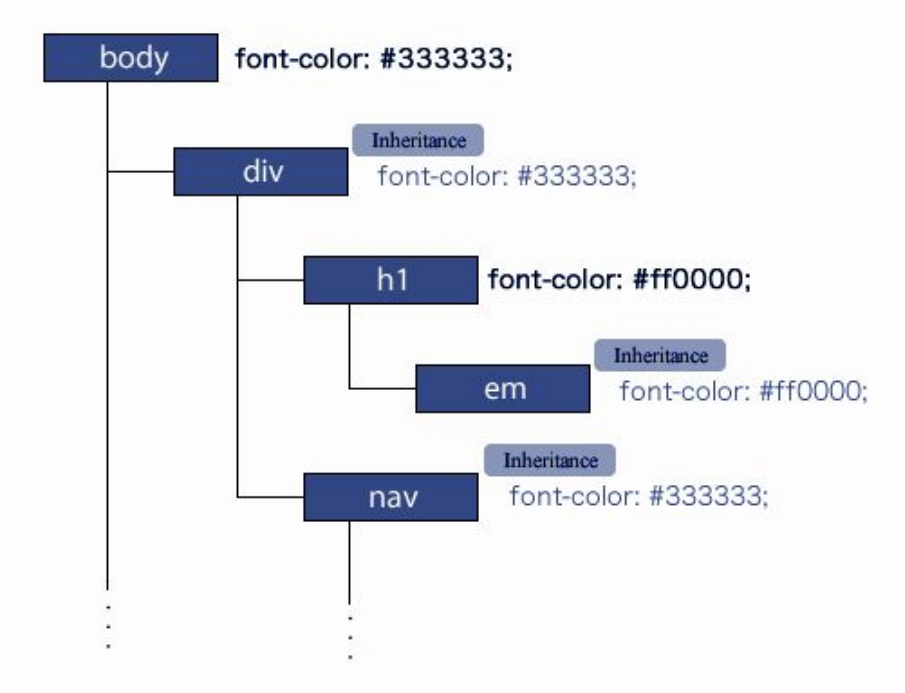
# Applying Styles from one source

1. Inheritance
2. Specificity
3. Cascading

Style

# Inheritance

+ Styles don't need to be applied to every element.
+ Styles don't need to define every property for each element.
+ CSS files can be smaller, easier to read, and load faster.
+ Only properties that simplify the developer's work are inherited.



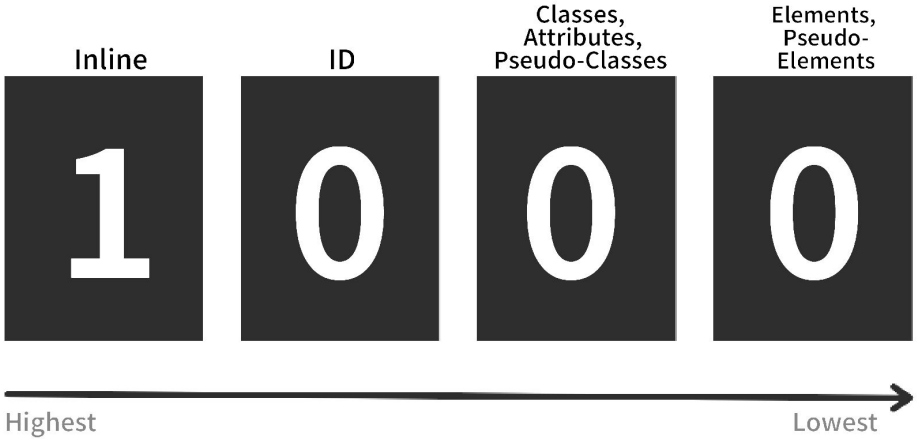

* – check manual to find inherited styles

# Specificity of selectors

Determines which styles will be applied to an element based on the selector's weight.

Avoid overusing weight; try to create lightweight selectors.

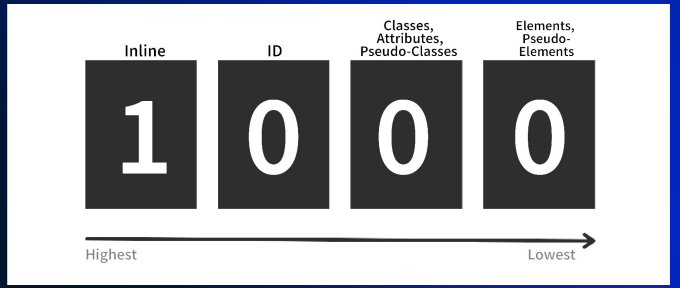
Selector nesting should be no more than three levels.




**specificity = weight**

universal selector * has no specificity weight (0)

# Practice: Calculating specificity




**#header h1 span a { }**

a = 0 x inline styles
b = 1 x ID
c = 0 x classes
d = 1 x elements
Specificity = 0,1,0,1

**.intro ::first-letter { }**

a = 0 x inline styles
b = 0 x IDs
c = 1 x class
d = 1 x pseudo-element
Specificity = 0,0,1,1

**a[href^="http:"] { }**

a = 0 x inline styles
b = 0 x IDs
c = 1 x attribute selector
d = 1 x element
Specificity = 0,0,1,1

# Practice: Calculating specificity

**#header h1 span a { }**

a = 0 x inline styles
b = 1 x ID *(#header)*
c = 0 x classes
d = 3 x elements *(h1,span,a)*
Specificity = 0,1,0,3

**.intro ::first-letter { }**

a = 0 x inline styles
b = 0 x IDs
c = 1 x class *(.intro)*
d = 1 x pseudo-element *(::first-letter)*
Specificity = 0,0,1,1

**a[href^="http:"] { }**

a = 0 x inline styles
b = 0 x IDs
c = 1 x attribute selector *([href^="http:"])*
d = 1 x element *(a)*
Specificity = 0,0,1,1

# Practice:

- What color will be applied to the text

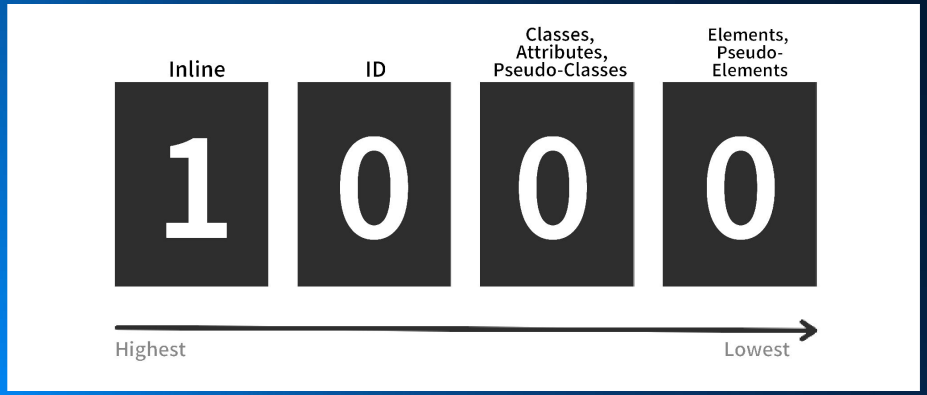
```
<div id="container">
    <nav class="nav">
        <p class="intro">
```

```
.nav p { color: yellow; }

p { color: blue; }

div#container p { color: purple; }

p.intro { color: orange; }
```



# Practice:

- What color will be applied to the text

| | |
|---|---|
| `.nav p { color: lime; }` | 0,0,1,1 |
| `p { color: blue; }` | 0,0,0,1 |
| **`div#container p { color: purple; }`** | **0,1,0,2** |
| `p.intro { color: green; }` | 0,0,1,1 |

Cascading

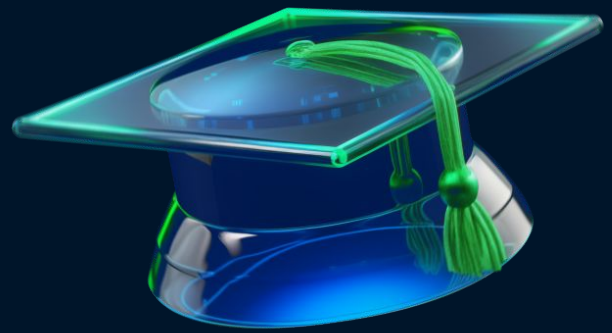
# Styles

When selectors have the same weight, the one defined later will be applied

```
/* green will be applied */
h1 {
  background-color: red;
}
h1 {
  background-color: green;
}
```

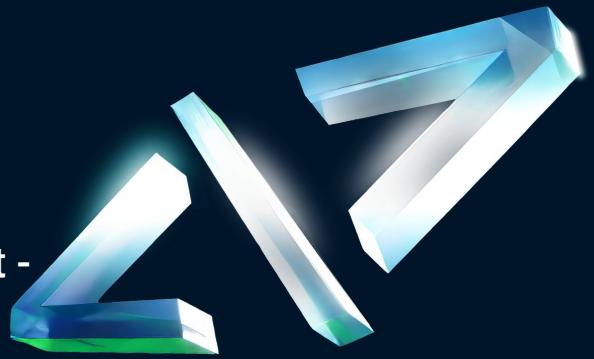
# Summary

1. CSS syntax
2. What selectors are
3. The order in which styles are applied
4. How to link styles to a page
5. The purpose of a style normalizer



# Homework

1. Complete one of the courses to reinforce the theory.
2. Add a font using Google Fonts.
3. Write basic styles for the marked-up blocks of the layout using Emmet right away:
    - font size / Line height / Font type / Font weight - font-*
    - Text color
    - Background color
4. Optional: Connect `normalize.css` to Binabox project as a separate file.



# Quality Criteria for HTML Course

❤️Mandatory for passing the course

💛Required for the highest grade

💚Optional

❤️ 2.1. Single CSS File.

💛 2.2. Include Normalize.css

❤️ 2.3. All fonts used in the design are connected to the pages.

❤️ 2.4. Provide fallback fonts and family types at the end of the font list.

❤️ 2.5. Do not use !important in CSS.

❤️ 2.6. Do not use #id for styling.

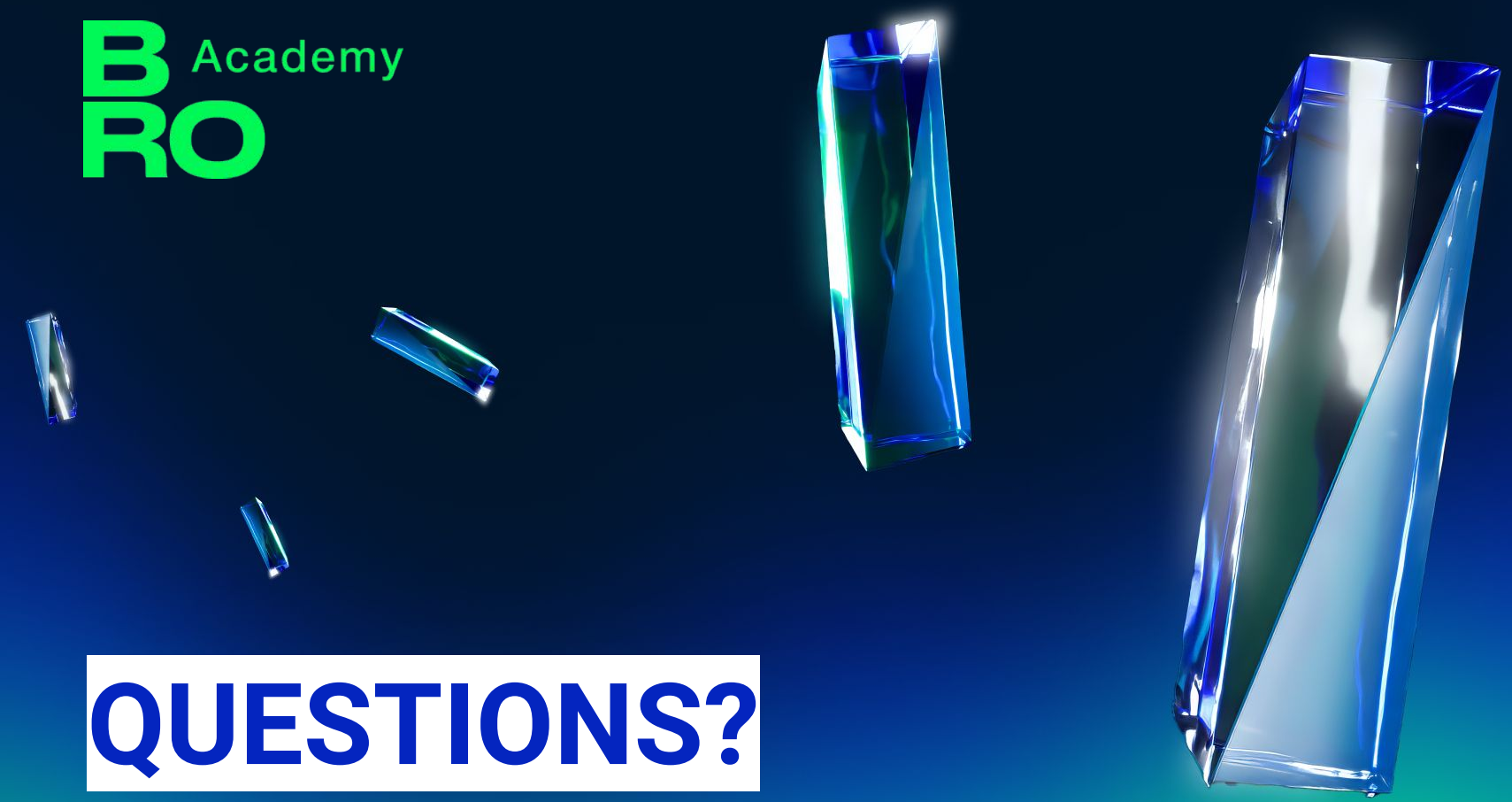
💛 2.7. Avoid nesting selectors more than two levels deep.

💛 2.8. Avoid styling tags directly

❤️ 2.10. Use consistent units for element sizes and positioning.

💛 2.11. Colors should be in a consistent format (hex or rgba).

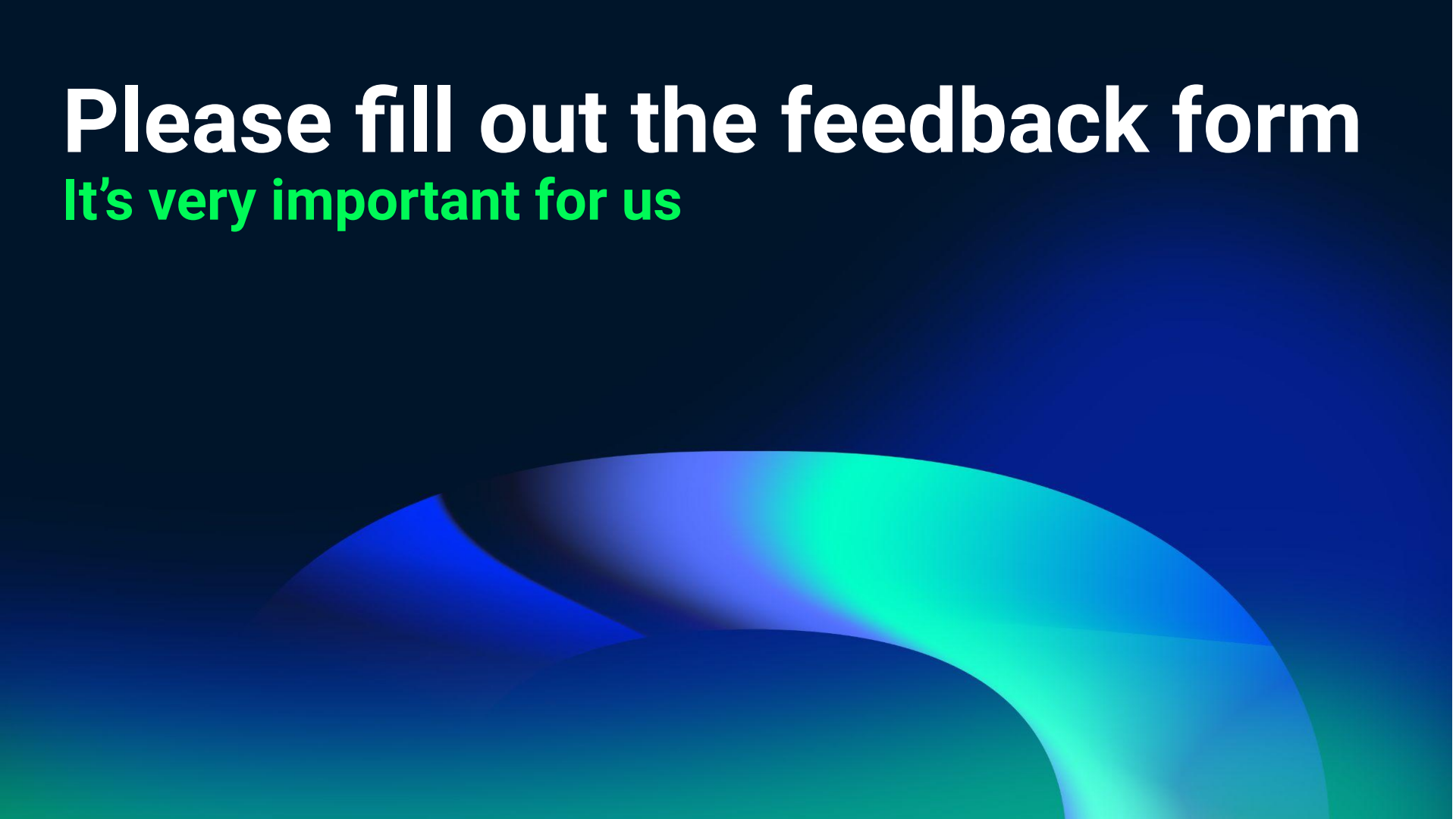
BRO Academy



# QUESTIONS?

# Please fill out the feedback form

## It’s very important for us



THANK YOU!
Have a good evening!