Menti about HTML

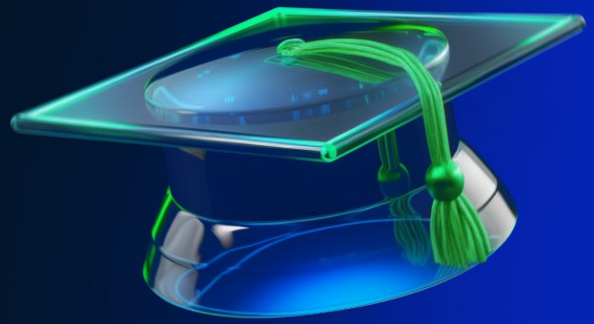
BRO Academy

Introduction to CSS

HTML course: Lesson 4

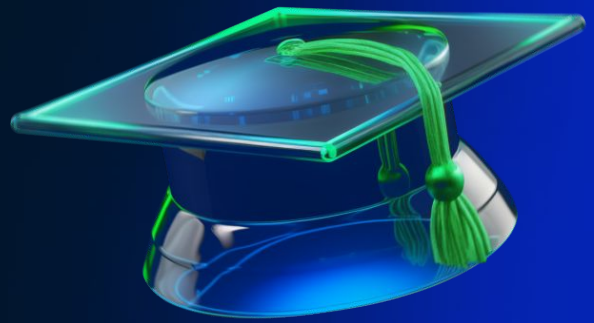
CSS: Real lesson Plan

1. CSS: specification, object model, render tree
2. Connecting styles: inline, internal, external
3. Rules, selectors, declarations, properties and values
4. Applying order: inheritance, cascading, specificity



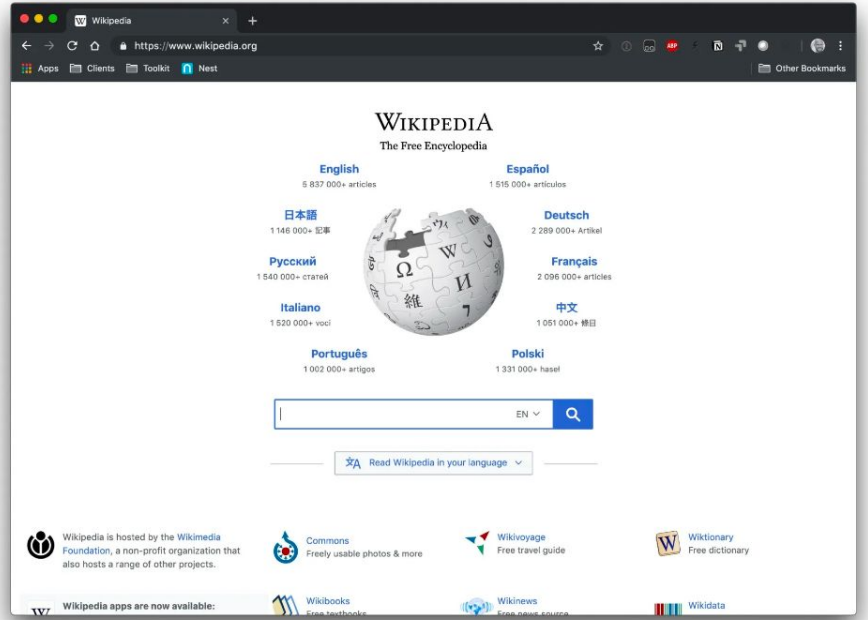
CSS: Lesson Plan

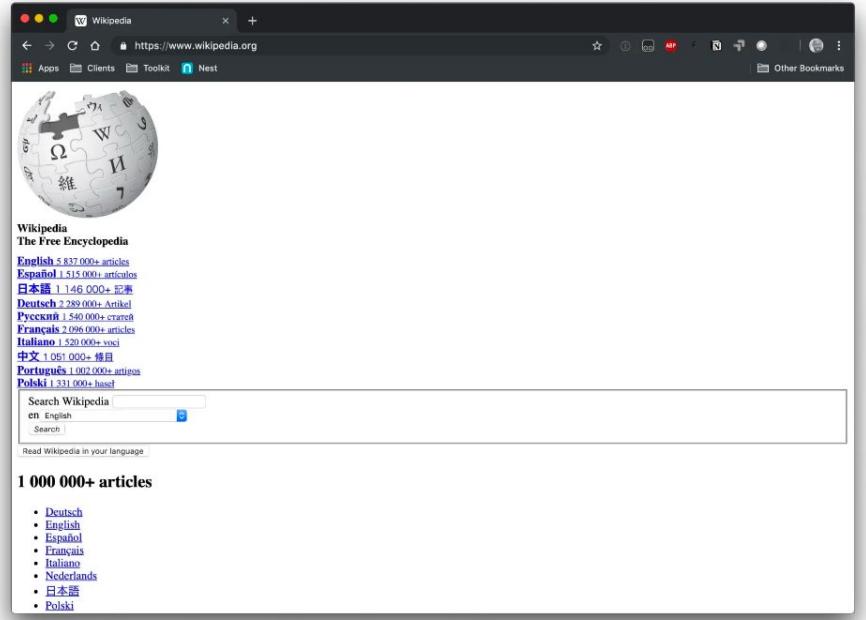
1. What is CSS
2. Where to write
3. How to write
4. How it will be applied



1

What


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.

2

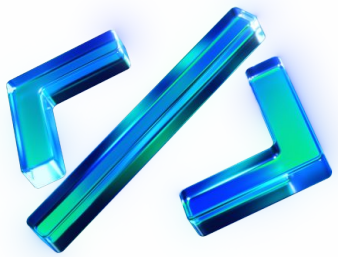
Where

Connecting styles

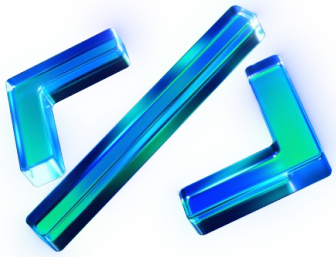
1. Inline
2. Embedded
3. External

Inline styles

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

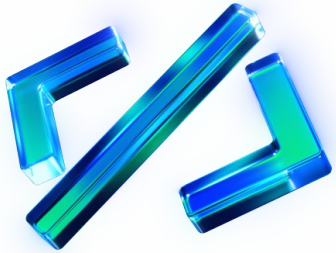


Embedded styles

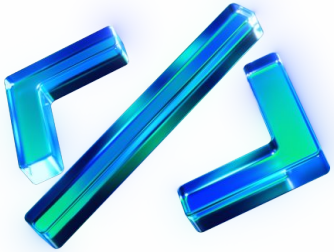


```
<head>
    <style>
        p {
            color: red;
            font-size: 14px;
        }
    </style>
</head>
```

External styles



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



3

How

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>`
				`  <article>`
				`  <p></p>`
				`  </article>`
				`</body>`
					`<body>`
					`  <section>`
					`  <p></p>`
					`  </section>`
					`  <p></p>`
					`</body>`

child combinator	**>**	`article > p`	an element is an immediate child	`<article>`
				`  <p></p>`
				`  <a></a>`
				`</article>`
					`<article>`
					`  <div>`
					`  <p></p>`
					`  </div>`
					`  <p></p>`
					`</article>`

adjacent sibling combinator	**+**	`p + img`	an element that comes immediately after	`<article>`
				`  <p></p>`
				`  <img/>`
				`  <a></a>`
				`</article>`
					`<article>`
					`  <p></p>`
					`  <a></a>`
					`  <img/>`
					`</article>`

general sibling combinator	**~**	`p ~ img`	an element located anywhere below in the code, inside a common parent	`<article>`
				`  <p></p>`
				`  <a></a>`
				`  <img/>`
				`</article>`
					`<article>`
					`  <p></p>`
					`  <a>`
					`  <img/>`
					`  </a>`
					`</article>`

Selector

Combinations

p {}	
p a { }	 <a> <a>
.intro {}	
div.intro { }	
.intro.module { }	

.intro .module { }	
.intro, .module { }	
.intro {} ≠ .Intro {}	
.nav ul li a { }	.nav a { }
/* comment */	
Selectors are read from right to left **nav ul li** {}	The rule applies to the **li** element, which is part of the **ul** element, which is inside the **nav** element.

4

Apply

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

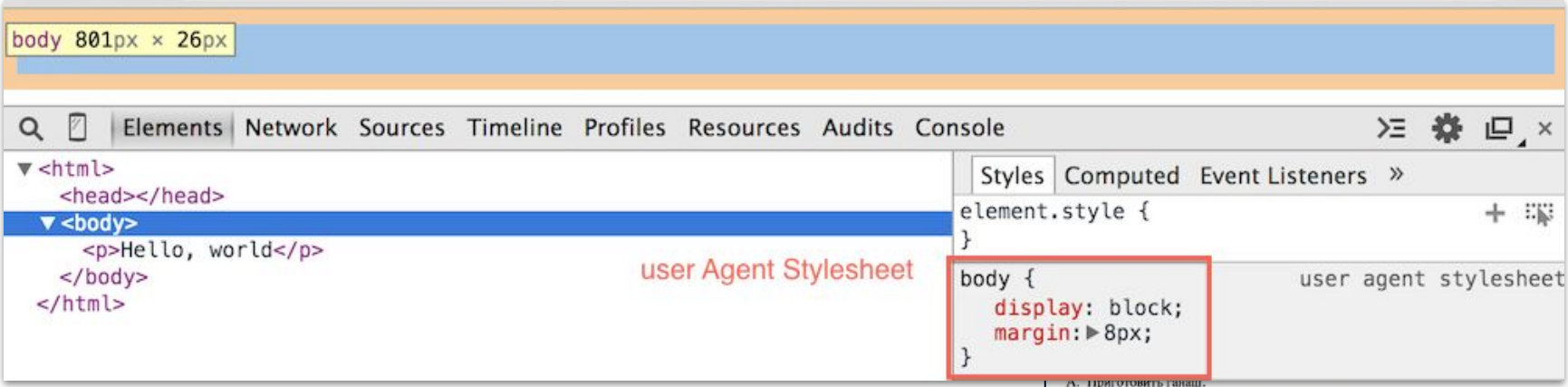
Как получить миндальную муку:

- 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.

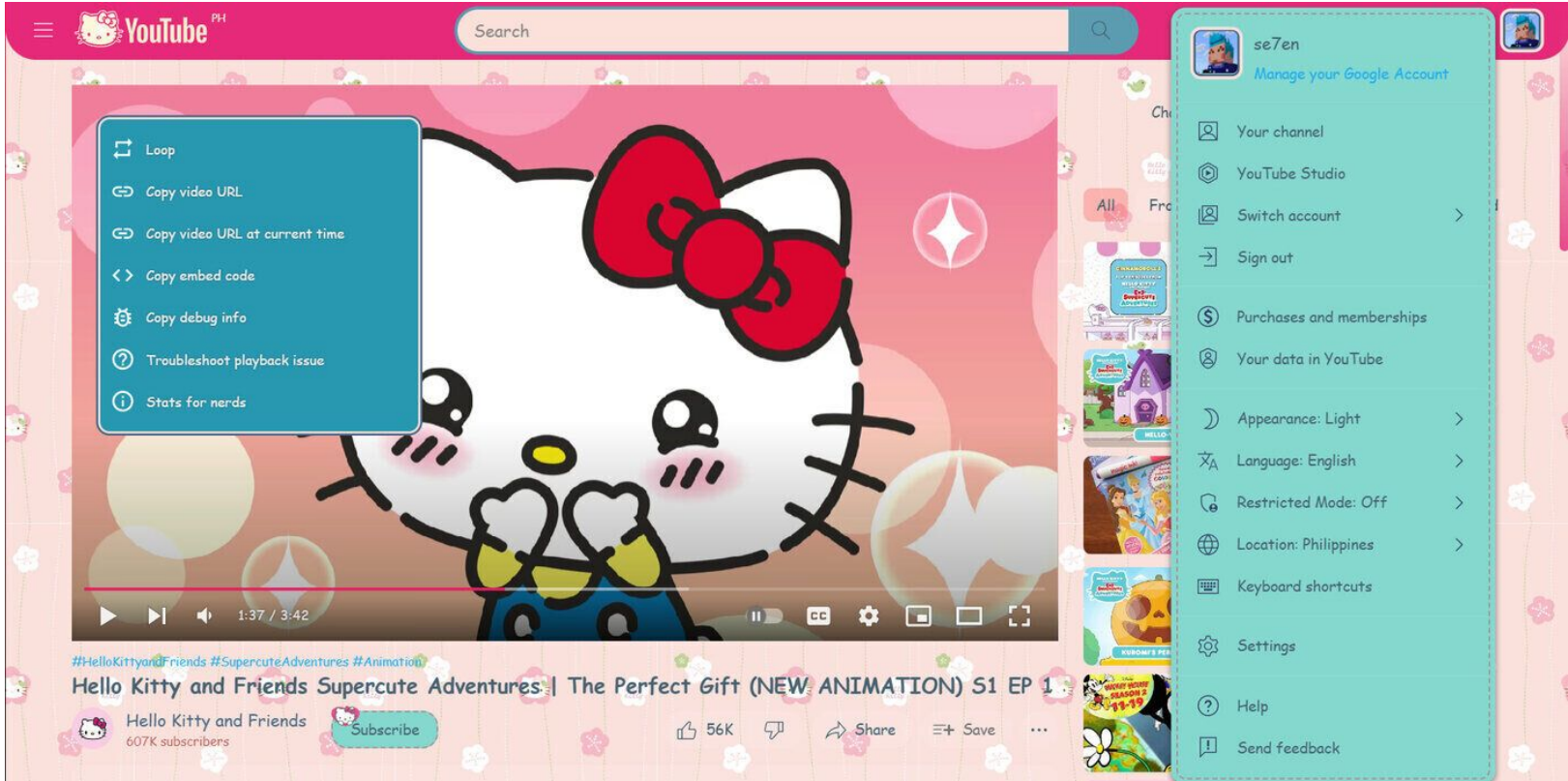
Шаг 2



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).

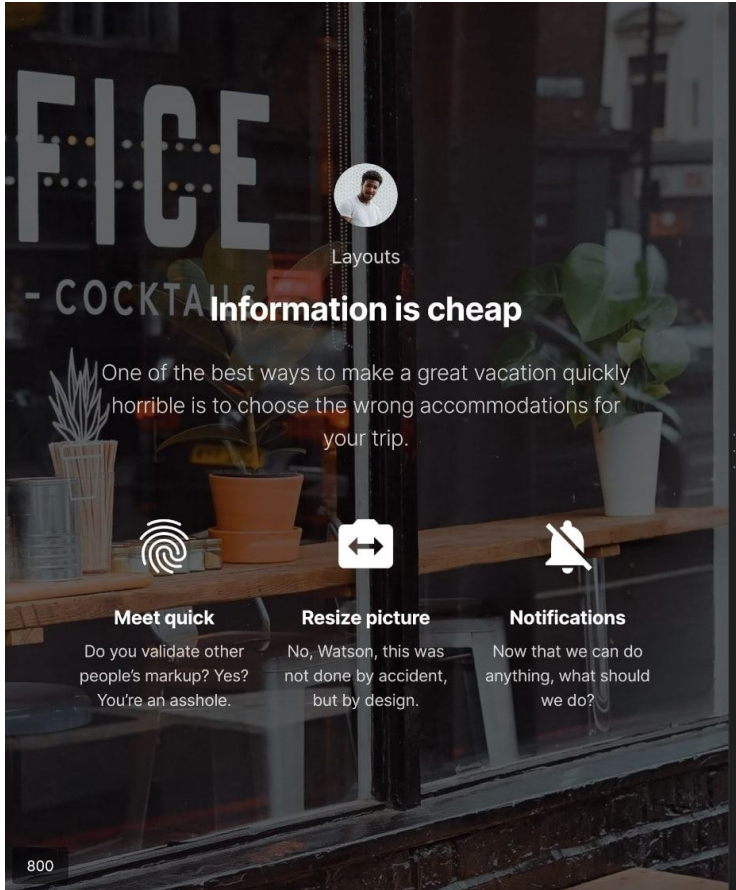
User styles

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



Author Styles

Site styles.



Close Save

HTML

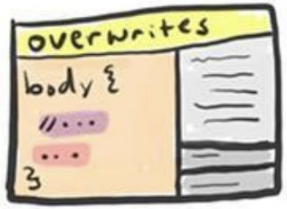
```
<!-- Slide 1 (#48) -->
<section class="slide fade-6 kenBurns">
  <div class="content">
    <div class="container">
      <div class="wrap">

        <div class="fix-8-12 toCenter">
          <img class="round margin-bottom-1 ae-1 fromCenter" src="assets/img/avatar-48.jpg" alt="Avatar" width="70"/>
          <p class="large margin-bottom-2 ae-2"><span class="opacity-8">Layouts</span></p>
          <h1 class="smaller margin-bottom-3 ae-3">Information is cheap</h1>
          <p class="larger light ae-4"><span class="opacity-8">One of the best ways to make a great vacation quickly horrible is to choose the wrong accommodations for your trip.</span></p>
        </div>
        <div class="fix-11-12 margin-top-5">
          <ul class="flex flex-48 later equal">
            <li class="col-4-12">
              <div class="fix-3-12">
                <i class="material-icons ae-5 fromCenter">fingerprint</i>
                <h4 class="small margin-bottom-1 equalElement ae-6">Meet quick</h4>
                <div class="ae-7"><p class="opacity-8">Do you validate other people's markup? Yes? You're an asshole.</p></div>
              </div>
            </li>
            <li class="col-4-12">
              <div class="fix-3-12">
                <i class="material-icons ae-6 fromCenter">switch_camera</i>
                <h4 class="small margin-bottom-1 equalElement ae-7">Resize picture</h4>
                <div class="ae-8"><p class="opacity-8">No, Watson, this was not done by accident, but by design.</p></div>
              </div>
            </li>
            <li class="col-4-12">
              <div class="fix-3-12">
                <i class="material-icons ae-7 fromCenter">notifications_off</i>
                <h4 class="small margin-bottom-1 equalElement ae-8">Notifications</h4>
                <div class="ae-9"><p class="opacity-8">Now that we can do anything, what should we do?</p></div>
              </div>
```

CSS

JS

user agent

author styles



local user

least specific → most specific

Reset and Normalize CSS

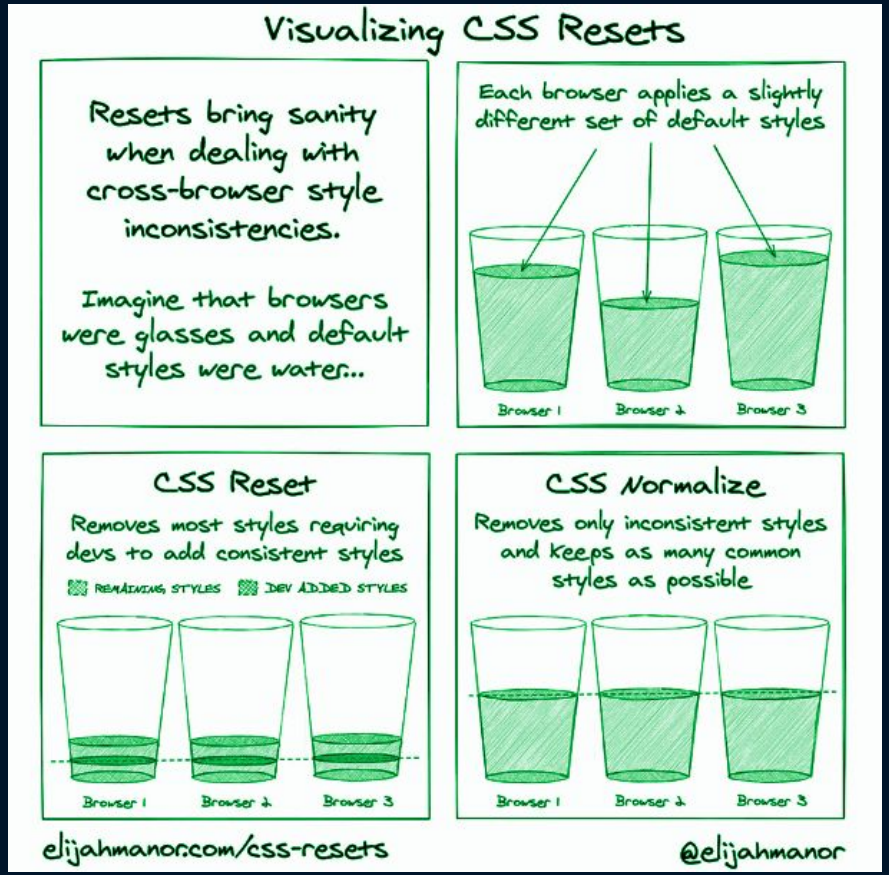
normalize.css

~~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">
```



Practice:

- What color will be applied to the header

```
/* user-agent style sheet */

h2 { color: black; }
```

```
/* user style sheet */

h2 { color: green; }

/* author style sheet */

h2 { color: blue; }
```

5

Apply

Applying Styles from one source

1

Inheritance

2

Specificity

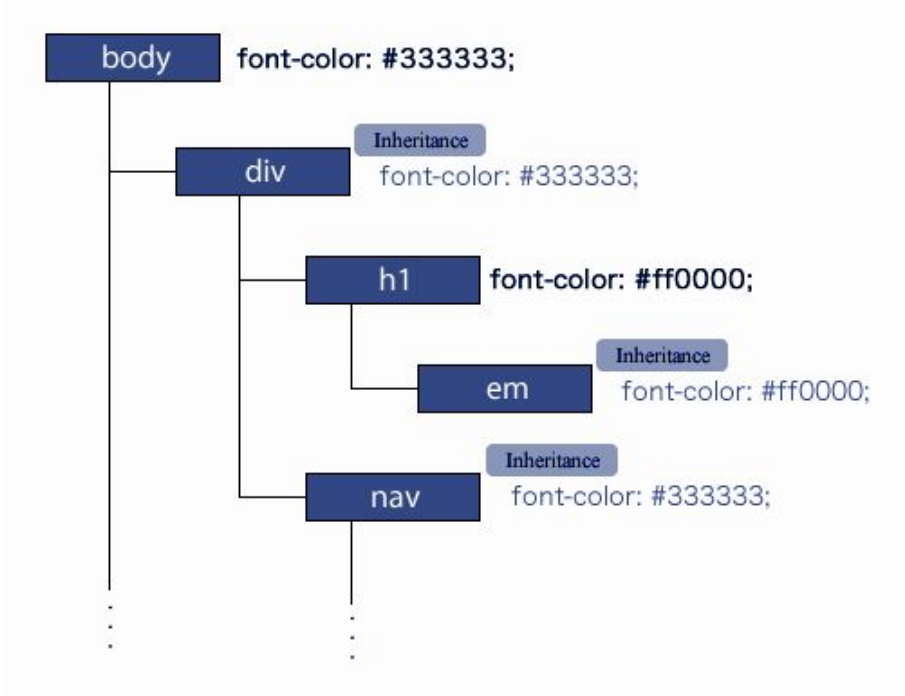
3

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

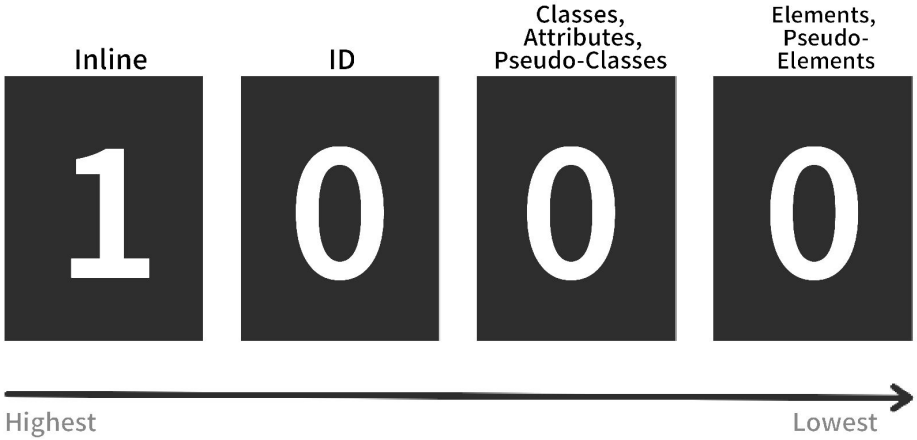
Specificity of

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)

What will be applied?

	important	inline styles	id	class, attribute, pseudo-class	elements, pseudo-elements	
	10000	1000	100	10	1	
`.card a:hover`	0	0	0	10+10	1	21
`.card .heading:before`	0	0	0	10*2	1	21
`body #block > .card`	0	0	100	10	1	111
`.card .heading .highlight + a[href]`	0	0	0	10*3+10	1	-
`body main #block .card h2 ~ a`	0	0	100	10	1*4	114
`.default-link {...!important}`	10000	0	0	10	0	10010
`<a style="...">`	0	1000	0	0	0	1000
`<a style="...!important">`	10000	1000	0	0	0	11000

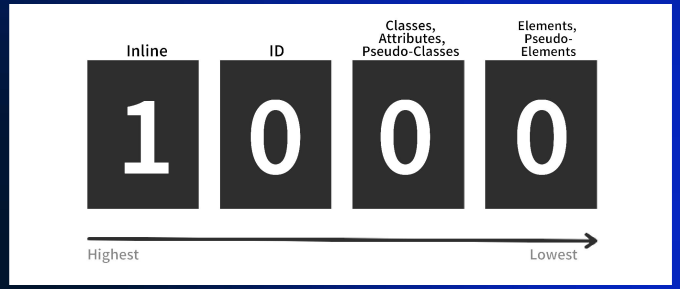
Specificity of

selectors

Examples

```
<body>
 <main>
   <section id="block">
     <article class="card">
       <h2 class="heading">
           <span class="highlight"></span>
        </h2>
       <a class="default-link">
```

Practice: Calculating specificity



#header h1 span a { }

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

.intro ::first-letter { }

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

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

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

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

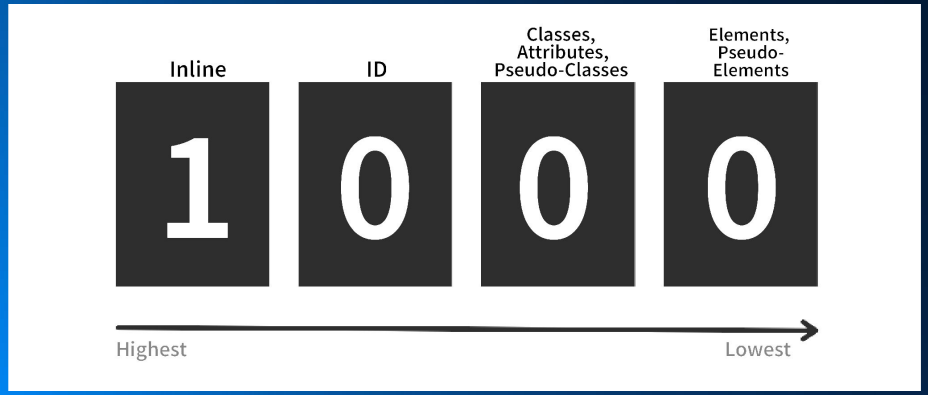
a[href^="http:"] { }

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

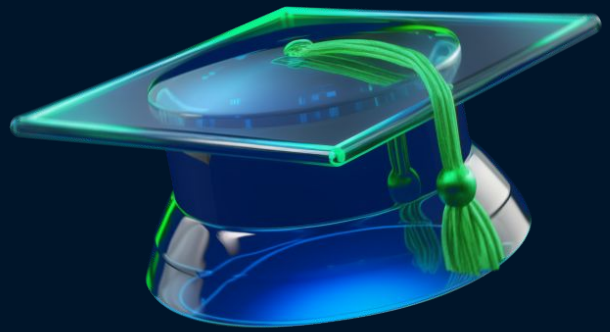
When selectors have the same weight, the one defined later will be applied

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

Summarizing

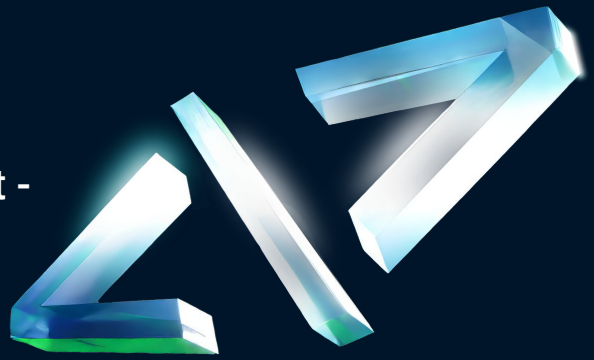
Summary

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. Add a font using Google Fonts.
2. 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
3. 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.

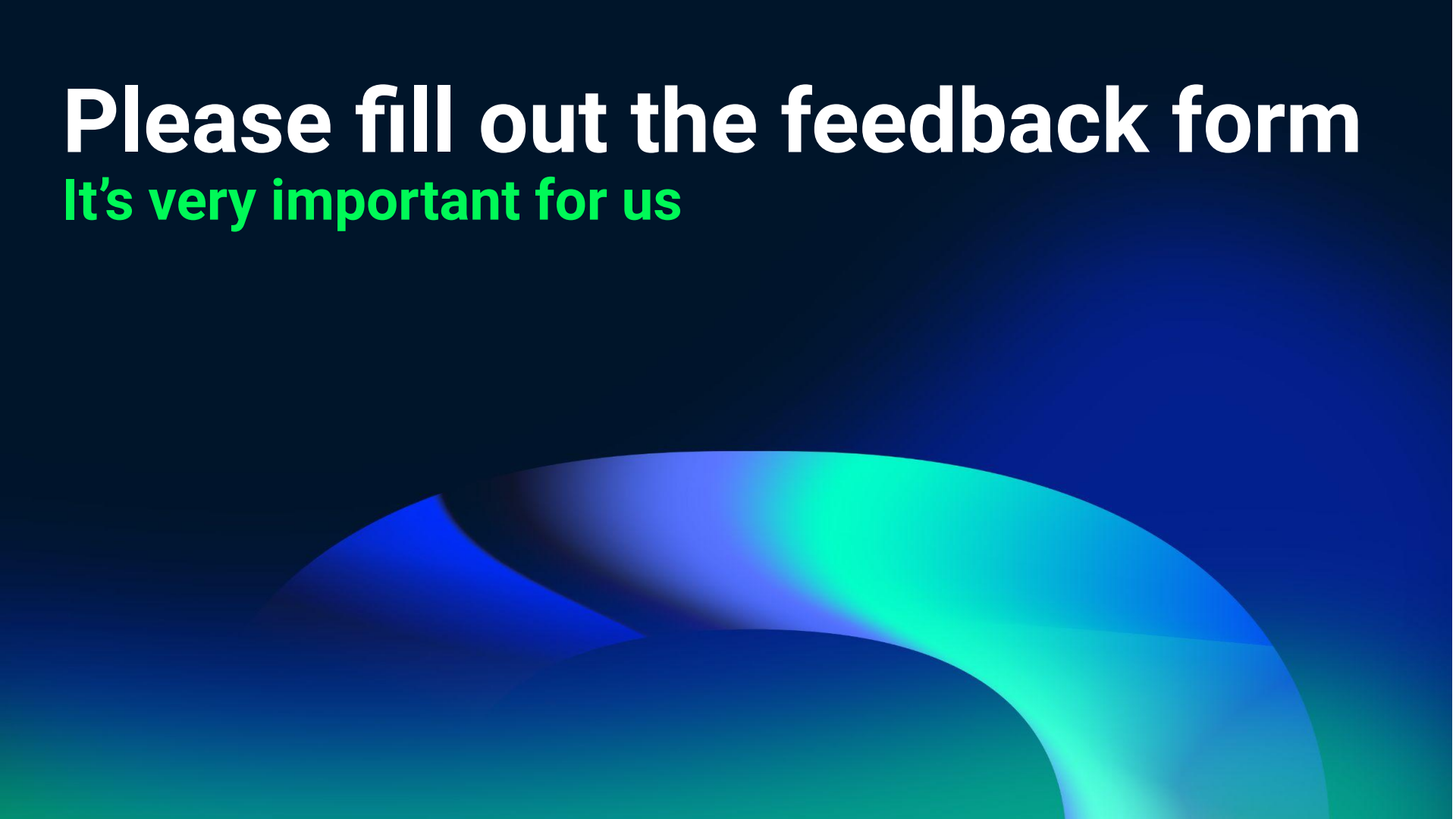
💛 2.11. Colors should be in a consistent format (hex or rgba).

BRO Academy

QUESTIONS?

Please fill out the feedback form

It’s very important for us



THANK YOU!

Have a good evening!