

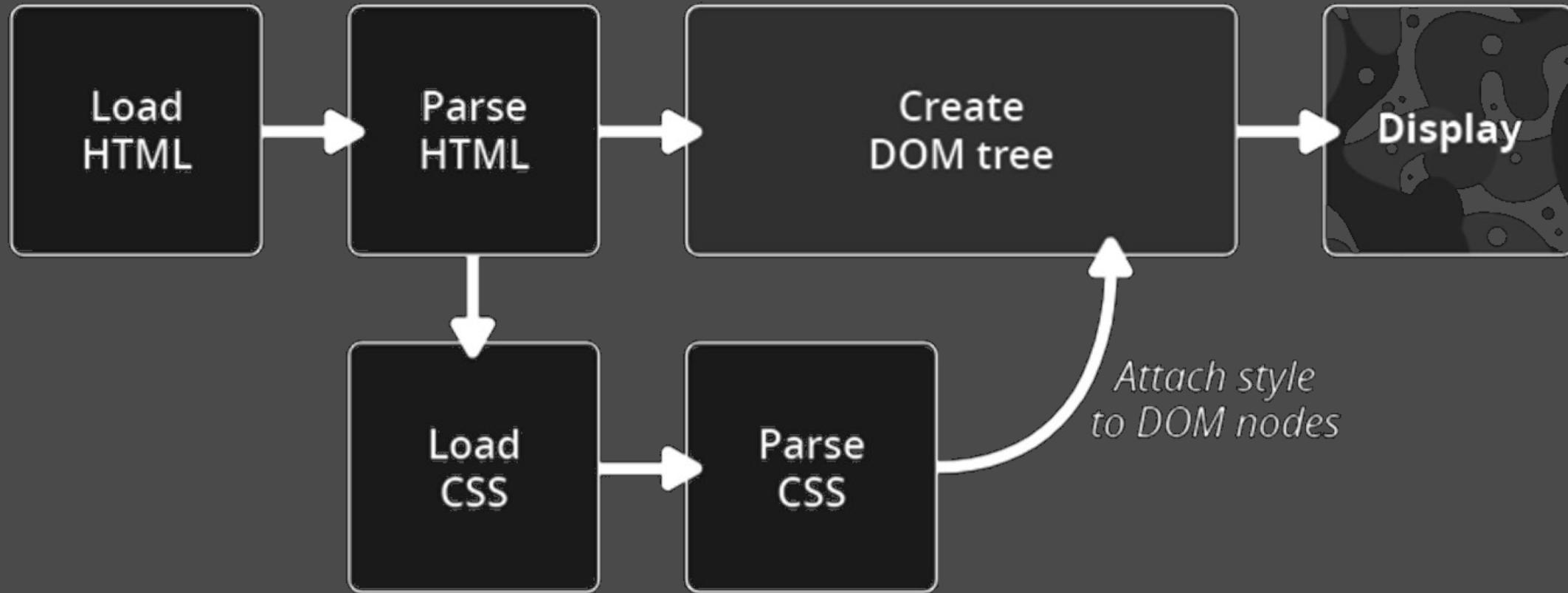
Jit Team & 3LO 2019-2020

{ CSS }

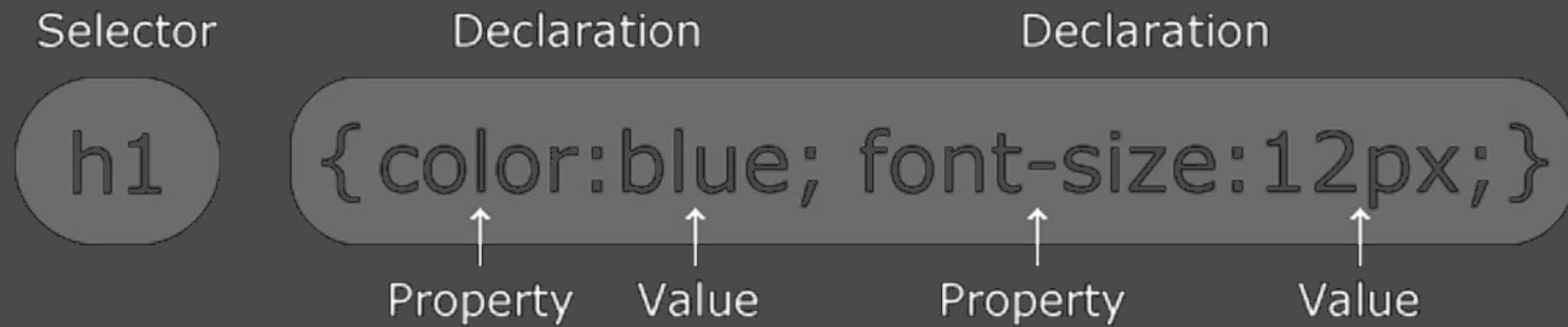
CSS cascading style sheets

Language describing how HTML elements should look like.

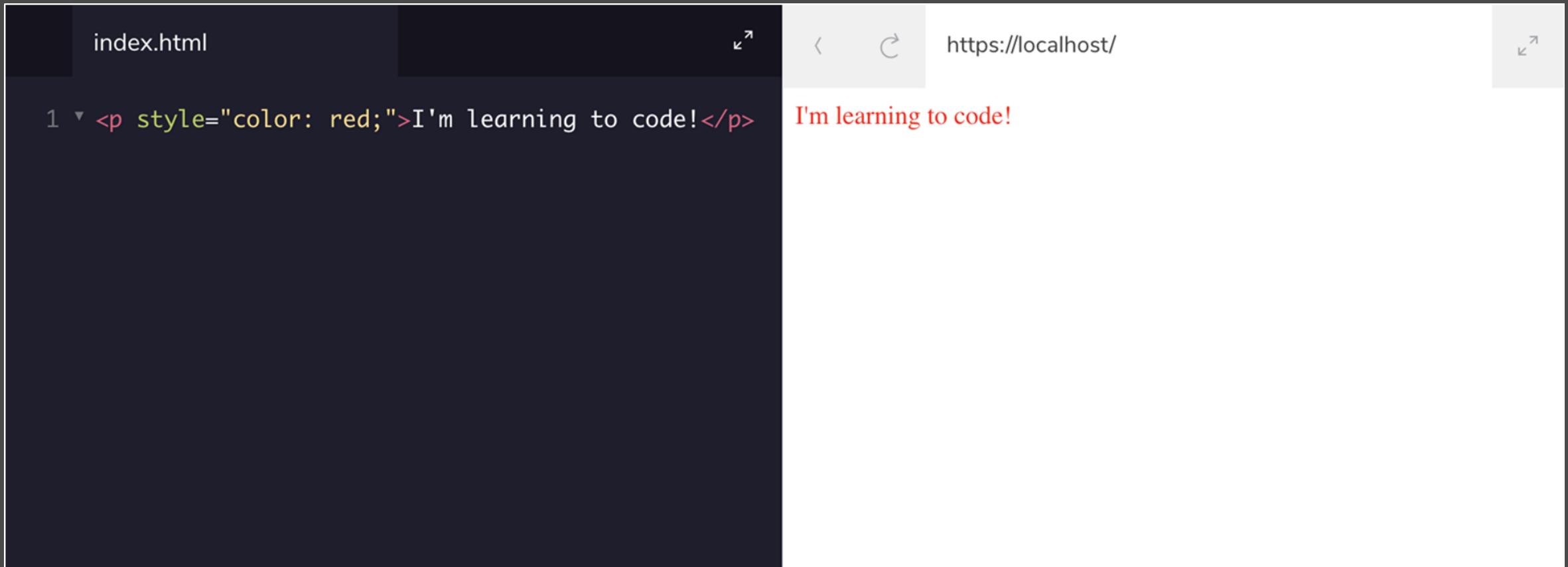
CSS behind the scenes



CSS syntax



CSS inline styles

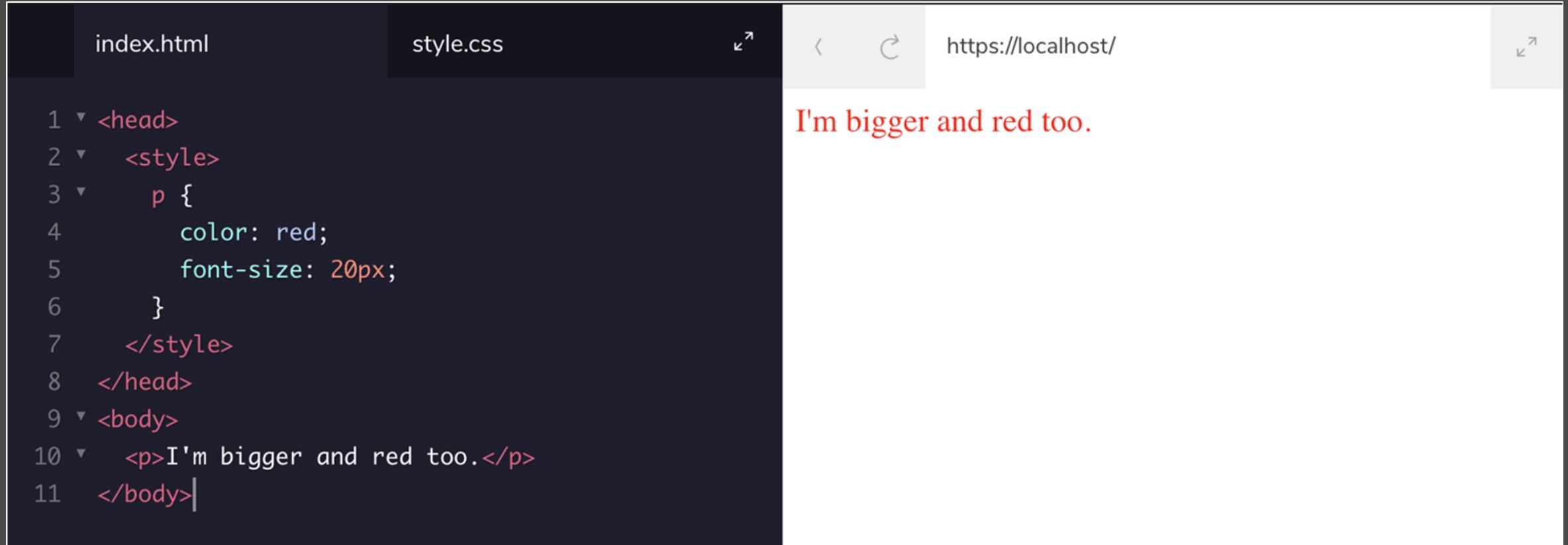


A screenshot of a web browser window. On the left, the file 'index.html' is open in a code editor. It contains a single line of HTML code:

```
1 <p style="color: red;">I'm learning to code!</p>
```

. On the right, the browser's address bar shows the URL <https://localhost/>. The page content is displayed as a single red text line: 'I'm learning to code!'.

CSS <style> tag



The screenshot shows a development environment with two code files and a browser preview.

index.html:

```
1 ▼ <head>
2 ▼   <style>
3 ▼     p {
4       color: red;
5       font-size: 20px;
6     }
7   </style>
8 </head>
9 ▼ <body>
10 ▼   <p>I'm bigger and red too.</p>
11 </body>|
```

style.css:

Browser Preview:

The browser window displays the URL <https://localhost/>. Inside, there is a single paragraph element with the text "I'm bigger and red too." The text is colored red and has a font size of 20px, demonstrating the effect of the CSS rule defined in the <style> tag.

CSS external styles

The image shows a code editor interface with two files open: `index.html` and `style.css`. The `index.html` file contains the following code:

```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <title>Hello World</title>
5     <link href=".style.css" type="text/css"
rel="stylesheet">
6   </head>
7   <body>
8     <p>I took style from external file.</p>
9   </body>
10  </html>
```

The `style.css` file contains the following CSS rules:

```
1 p {
2   font-family: Arial;
3   color: red;
4 }
```

A browser window is shown in the background, displaying the rendered HTML with the text "I took style from external file." in red font.

CSS selectors: tag name

```
p {
```

```
}
```

CSS selectors: class name

```
<h1 class="green bold"> ... </h1>
```

```
.green {  
    color: green;  
}  
  
.bold {  
    font-weight: bold;  
}
```

CSS selectors: ID name

```
<h1 id="large-title"> ... </h1>
```

```
#large-title {
```

CSS selectors: attribute name

```
<input type="text" />  
<input type="email" />
```

```
input[type="email"] {  
  color: tomato;  
}
```

CSS chaining selectors

```
<p class="special"></p>  
<h1 class="special"></h1>
```

```
h1.special {  
    color: tomato;  
}
```

CSS nested selectors

```
<ul class="main-list">
  <li></li>
  <li></li>
  <li></li>
</ul>
```

```
.main-list li {
  color: tomato;
}
```

CSS pseudo-classes

- `:link` – selects all unvisited links
- `:hover` – selects elements on mouse hover
- `:visited` – selects all visited links
- `:active` – selects an element while it is being selected by the user, for example, when the user is mid-click
- `:focus` – selects an element that is being focused via click or keyboard even, for example, tabbing through a website

CSS pseudo-elements

```
div:before {  
    content: "";  
}  
  
div:after {  
    content: "";  
}
```

The **:before** and **:after** pseudo-elements allow you to insert content before or after any HTML element that isn't self closing (like `` and `<input>`).

The content property is required but can be left blank.

These pseudo-elements can be treated and styled like any other element.



CSS combinator

- div > p – **parent selector** – selects all paragraphs where the parent is a div element
- div + p – **sibling selector** – selects all paragraphs that are placed immediately after a div element
- div ~ p – **sibling selector** – selects all paragraphs that are preceded by a div element

```
<p>Christmas shopping list:</p>
<div>
  <p>Fruit</p> ←
  <section>
    <ul>
      <li>Apple</li>
      <li>Banana</li>
      <li>Orange</li>
    </ul>
    <p>Toys</p>
    <ul> ←
      <li>Car</li>
      <li>Lego</li>
      <li>Excavator</li>
    </ul>
  </section>
</div>
<p>November, 2019</p>
```

```
div > p {
  background: red;
}

p ~ ul {
  background: blue;
}

div + p {
  background: magenta;
}
```

CSS multiple selectors

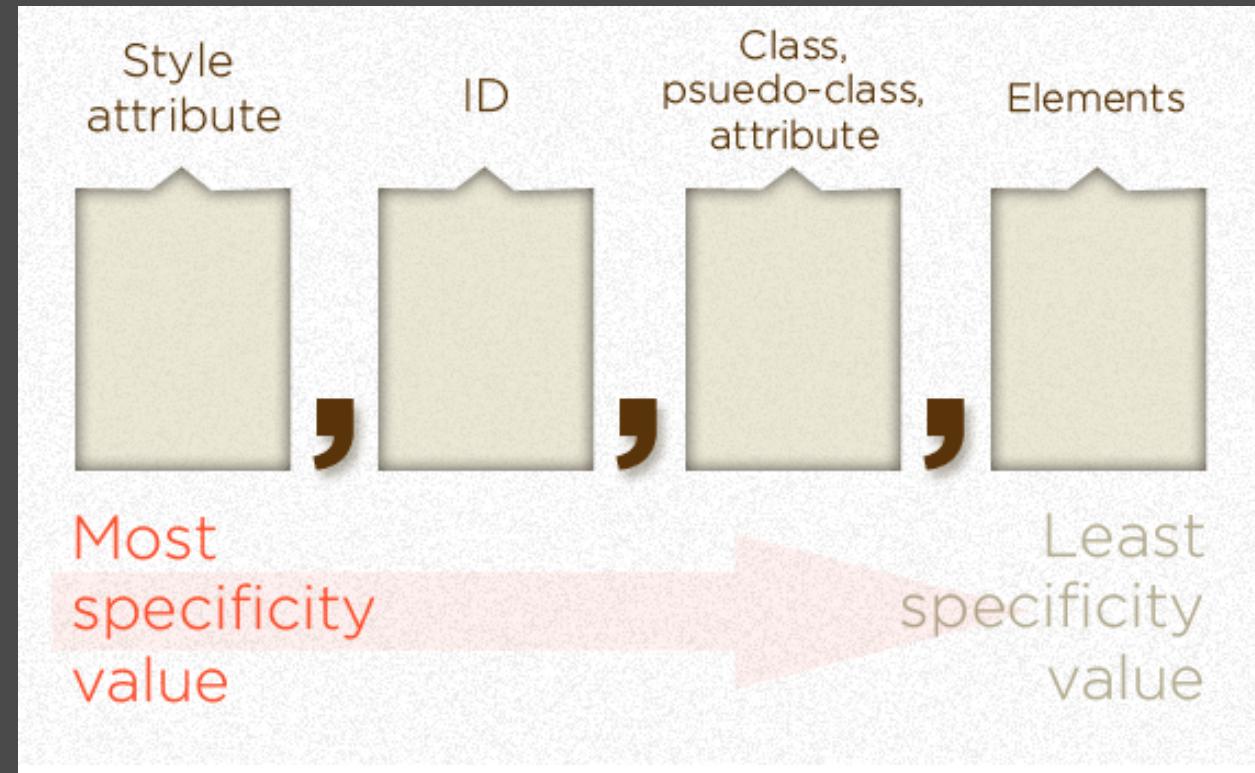
```
h1 {  
    font-family: Georgia;  
}  
  
.menu {  
    font-family: Georgia;  
}
```

=

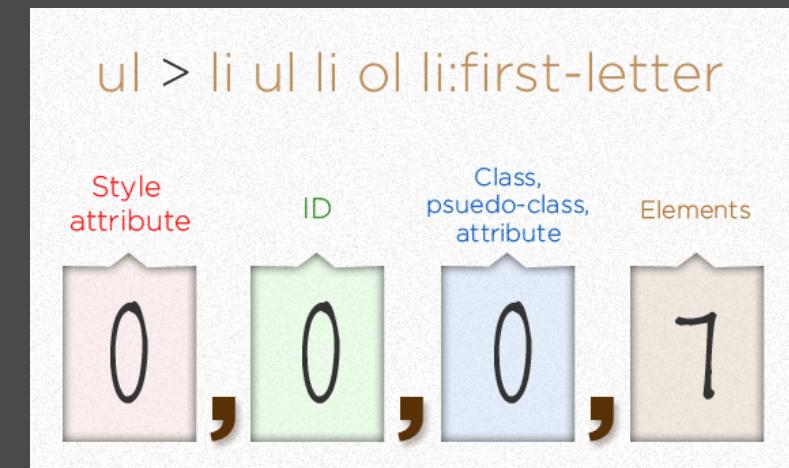
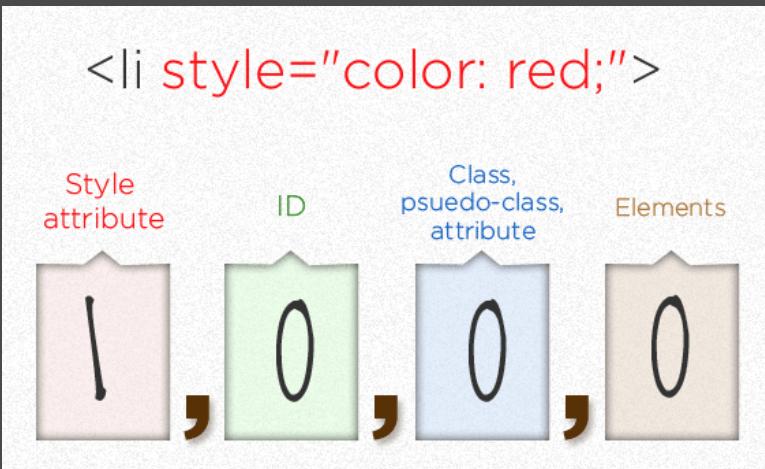
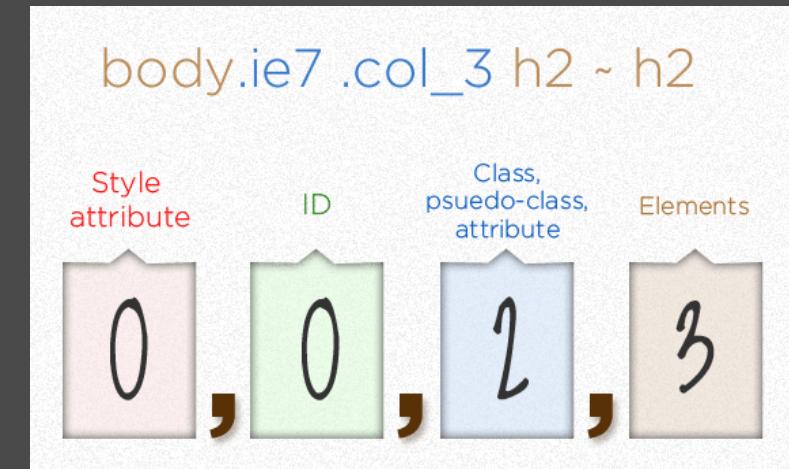
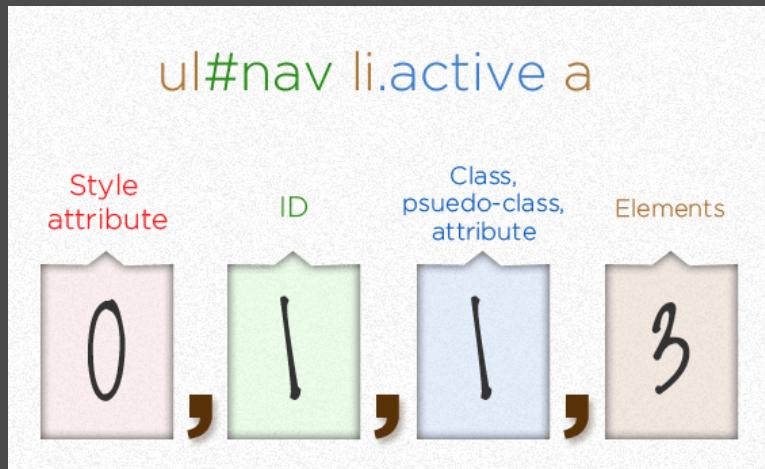
```
h1,  
.menu {  
    font-family: Georgia;  
}
```

CSS specificity

Specificity is the order by which the browser decides which CSS styles will be displayed.



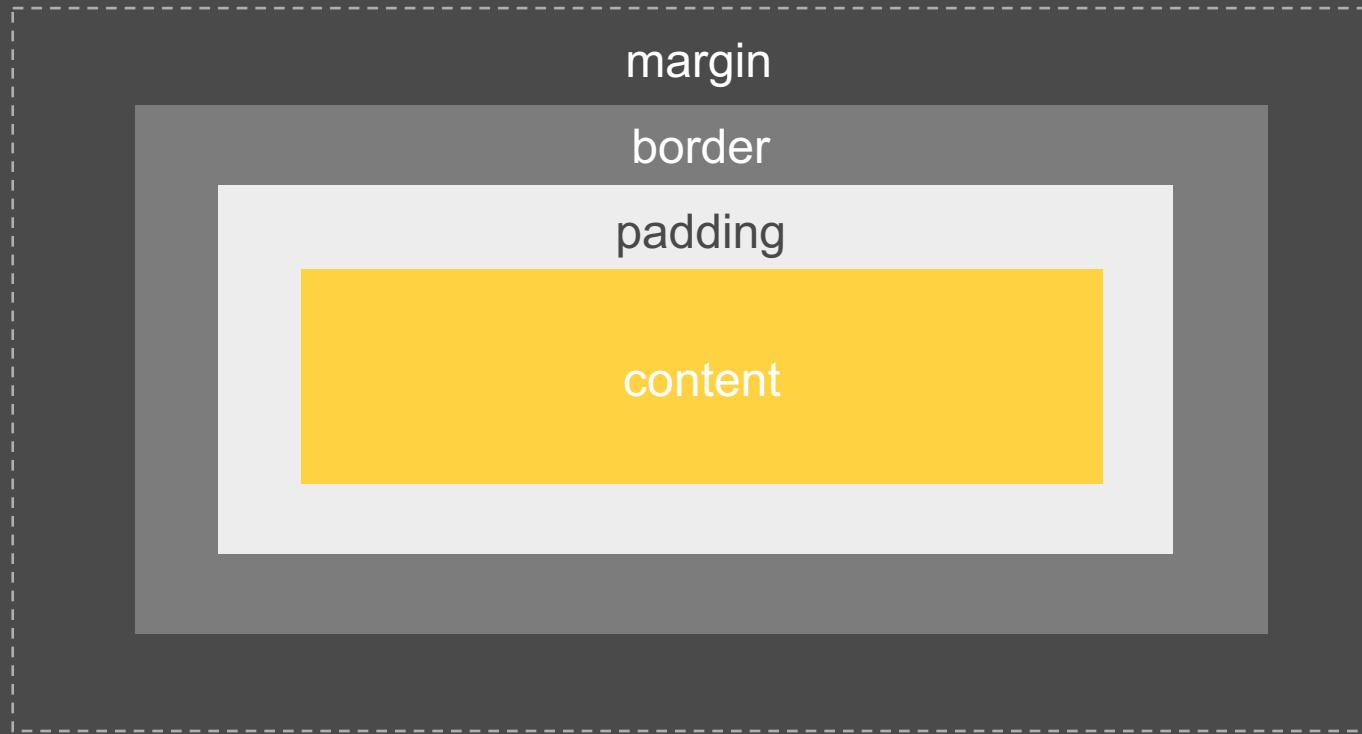
CSS specificity



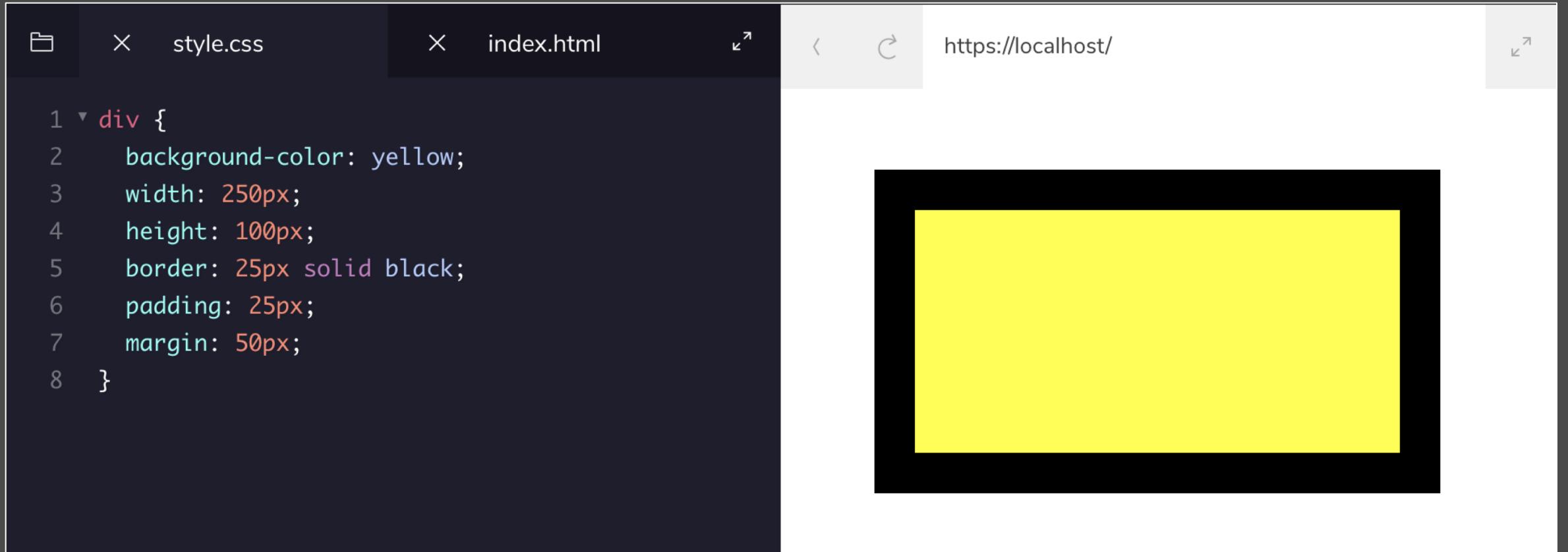
CSS basic properties

- `color: white, #FFFFFF, rgba(255,255,255,1);`
- `background-color: black, #000, rgb(0,0,0);`
- `font-family: "Times New Roman";`
- `font-size: 18px, 2em, 3rem;`
- `opacity: 0.5;`
- `background-image: url("./image.png");`
- `font-weight: 400, normal;`
- `text-align: right, center, left;`

CSS box model



CSS box model



```
1 ▼ div {  
2     background-color: yellow;  
3     width: 250px;  
4     height: 100px;  
5     border: 25px solid black;  
6     padding: 25px;  
7     margin: 50px;  
8 }
```

CSS media queries

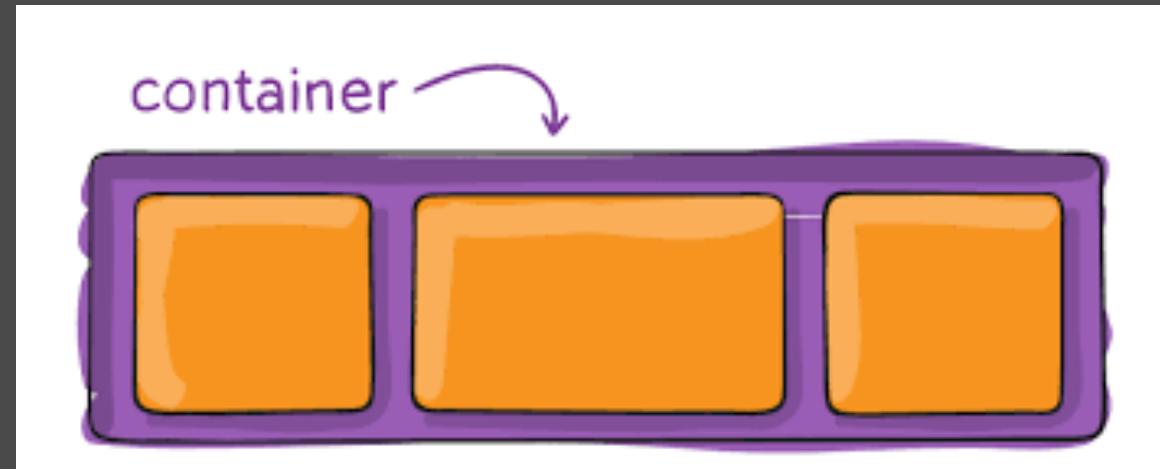


```
style.css
```

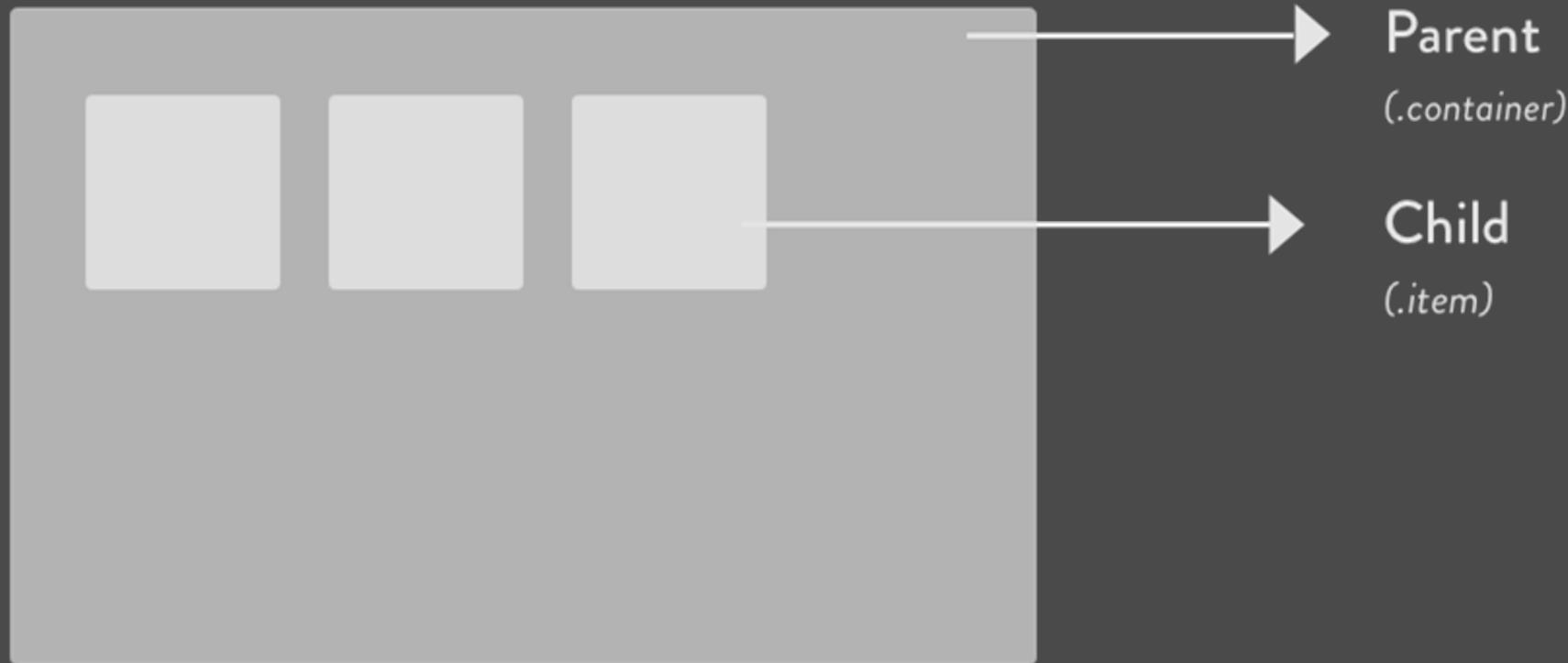
```
1  /* Large desktop */
2  @media (min-width: 1200px) {
3      /* CSS Classes */
4  }
5  /* Portrait tablet to landscape and desktop */
6  @media (min-width: 768px) and (max-width: 979px)
7  {
8      /* CSS Classes */
9  }
10 /* Landscape phone to portrait tablet */
11 @media (max-width: 767px) {
12     /* CSS Classes */
13 }
14 /* Landscape phones and down */
15 @media (max-width: 480px) {
16     /* CSS Classes */
17 }
```

CSS flexbox layout module

Makes creating responsive layouts easier



CSS flexbox anatomy



CSS flexbox properties

Flex **container** properties

- `flex-direction`
- `flex-wrap`
- `flex-flow`
- `justify-content`
- `align-items`
- `align-content`

Flex **item** properties

- `order`
- `flex-grow`
- `flex-shrink`
- `flex-basis`
- `flex`
- `align-self`

CSS flexbox properties

```
1 ▼ <div class="flex-container">  
2   ▼   <div>1</div>  
3   ▼   <div>2</div>  
4   ▼   <div>3</div>  
5   </div>
```

```
1 ▼ .flex-container {  
2   display: flex;  
3   flex-direction: column;  
4 }
```

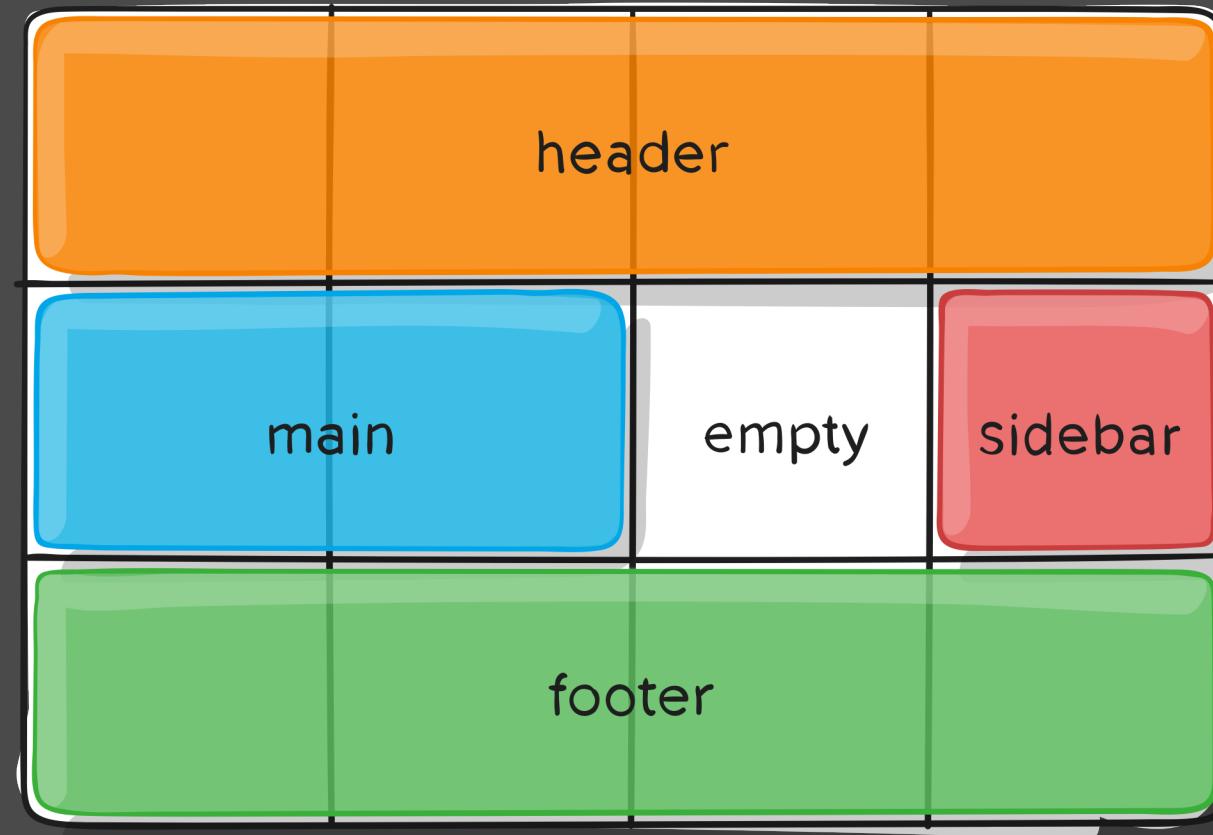
1

2

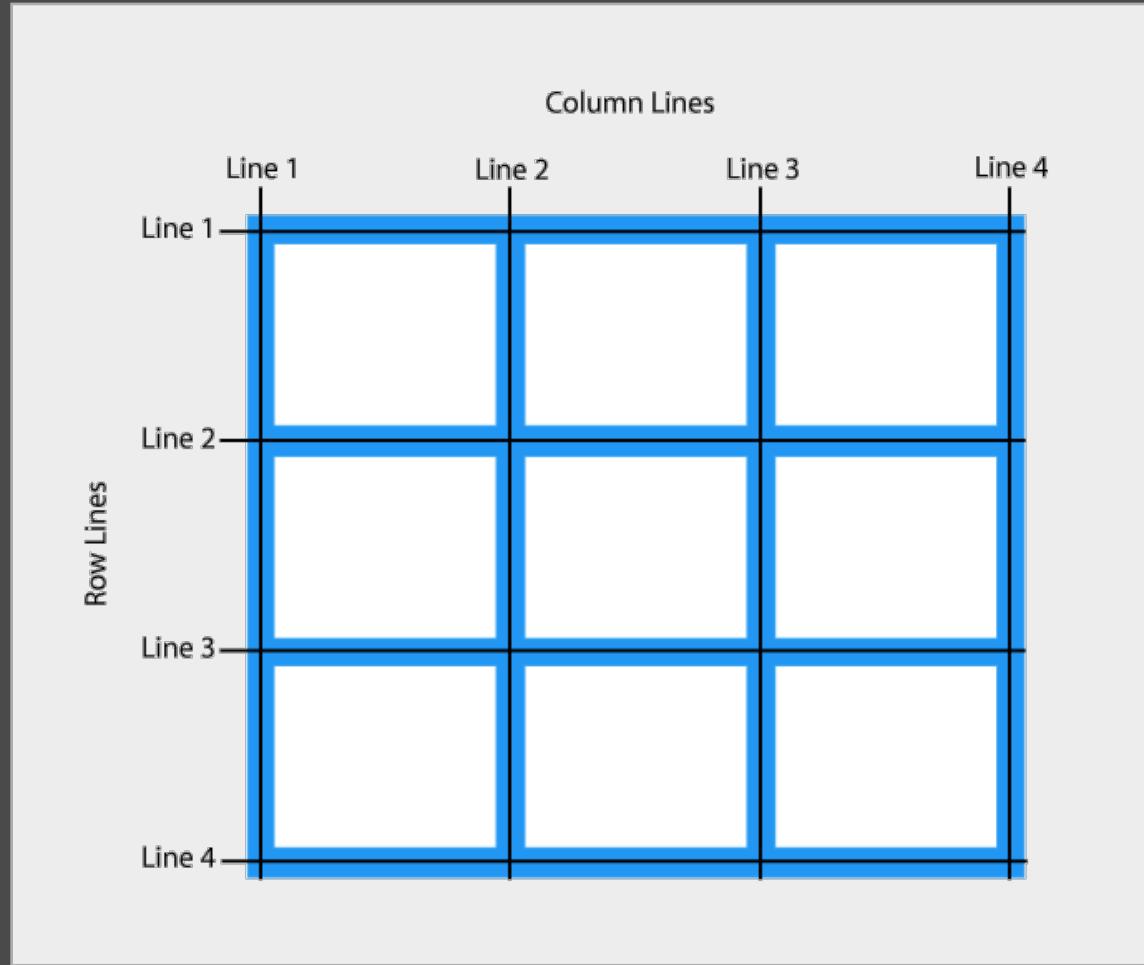
3

Grid Layout Module

Makes creating layouts easier



Grid anatomy



Grid properties

Grid **container** properties:

- Grid-template-columns
- Grid-template-rows
- Grid-template-areas
- Grid-gap
- Justify-items (horizontally)
- Justify-content
- Align-items (vertically)
- Align-content
- Grid-auto-columns
- Grid-auto-rows
- Grid-auto-flow

Grid **item** properties:

- Grid-column
- Grid-row
- Grid-area
- Justify-self
- Align-self

CSS debugging

Styles Computed Event Listeners »

Filter :hover .cls +

```
element.style {  
}  
  
.container { (index):32  
    display: grid;  
    height: calc(100% - 93px);  
    box-sizing: border-box;  
    padding-bottom: 25px;  
    max-width: 330px;  
    margin: auto;  
}  
  
main { user agent stylesheet  
    display: block;  
}  
  
Inherited from html  
html { user agent stylesheet  
    color: -internal-root-color;  
}  
  

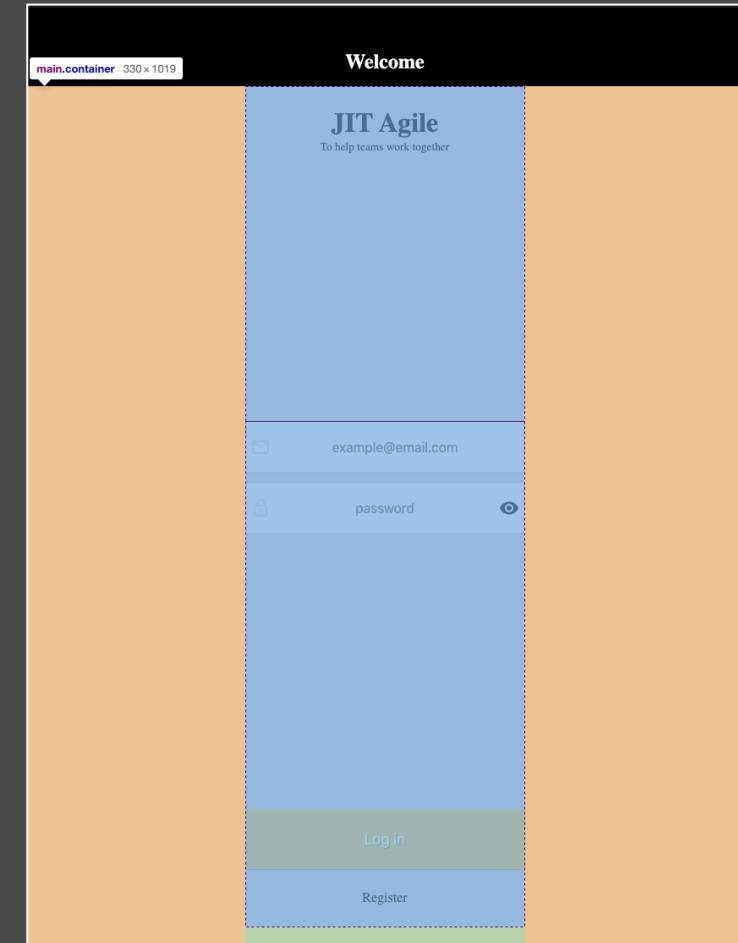
```

Styles Computed Event Listeners »

margin -
border -
padding -
257 - 330 x 994 - 257 -
- - - - -

Filter Show all

- box-sizing border-box
- color □rgb(0, 0, 0)
- display **grid**
- height 1019px
- margin-bottom 0px
- margin-left 257px
- margin-right 257px
- margin-top 0px
- max-width 330px
- padding-bottom 25px
- width 330px

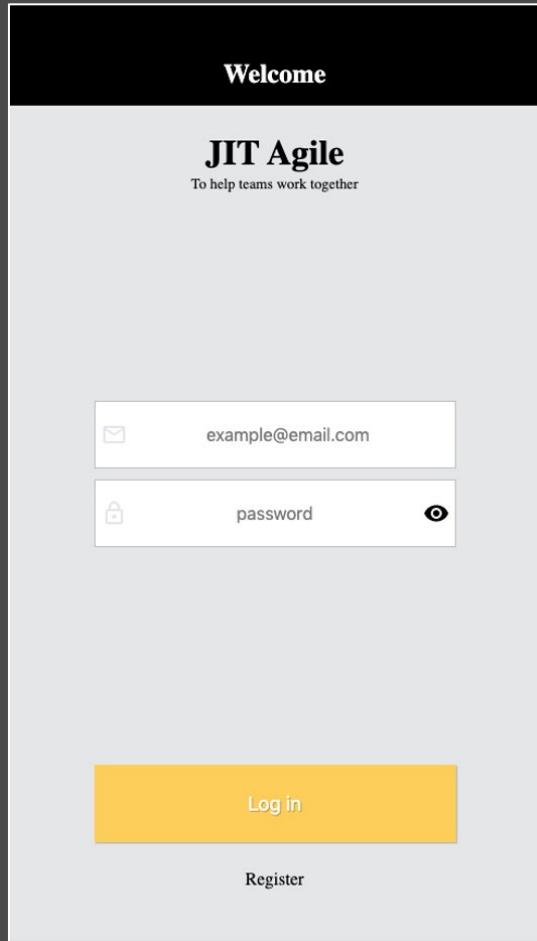


CSS where to go from here?

- css-tricks.com
- flexboxfroggy.com
- flexbox.io
- cssgridgarden.com
- cssgrid.io



Exercise #1 login page



Exercise #1 primary header



Welcome

It should:

- span the entire width of the viewport
- have increased vertical padding ~ 3:1 ratio
- have centered, white text on black background – use hexcode or rgb colors

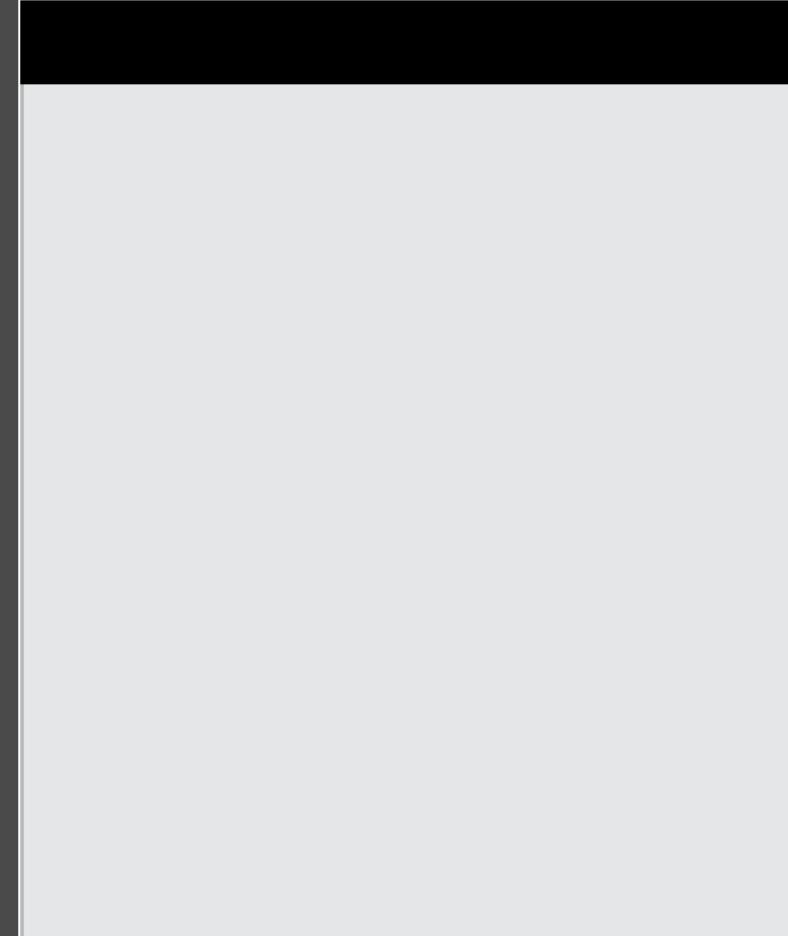
CSS properties you'll need for this task: background, padding, margin, color, text-align

Exercise #1 background

It should:

- span the entire width and height of the viewport
- Background colour set to `#e5e6e8`

CSS properties you'll need for this task: `background`,
`height`



Exercise #1 secondary header

It should:

- be aligned horizontally on the page
- Have a 25 px space from primary header

CSS properties you'll need for this task: margin, text-align

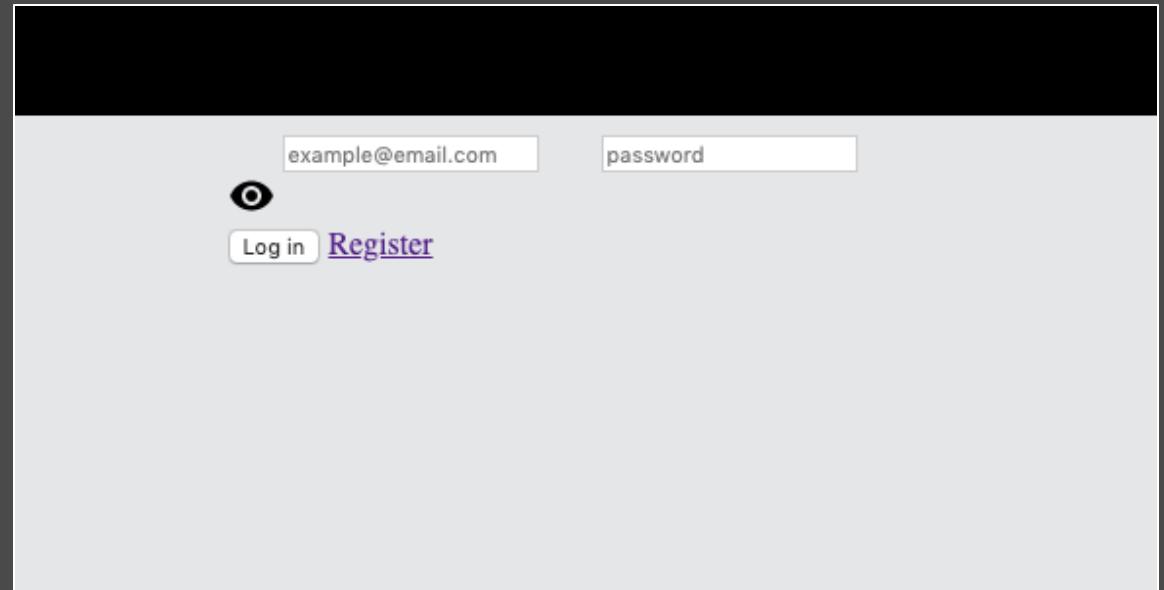


Exercise #1 form

It should:

- be centered on the page
- Span 100% of the viewport width up to 330 pixels

CSS properties you'll need for this task: margin, width, max-width



The image shows a screenshot of a web browser displaying a login form. The form is centered on the page and spans most of the available width. It features a black header bar at the top. Below the header, there are two input fields: one for email with the placeholder "example@email.com" and another for password. Underneath the input fields are two radio buttons. To the right of the radio buttons is a "Log in" button and a "Register" link. The entire form is set against a light gray background.

Exercise #1 form inputs

- Each input and icon(s) should be inside a flex container
- Flex containers should all be on a separate line
- Flex container should have a solid border, 1 pixel thick -
#B0B1B5 - and white background
- Input should take up all free space and have no border
- Text inside inputs should be centered and 16 pixels in size
- Space between input and icon(s) should be 5 pixels

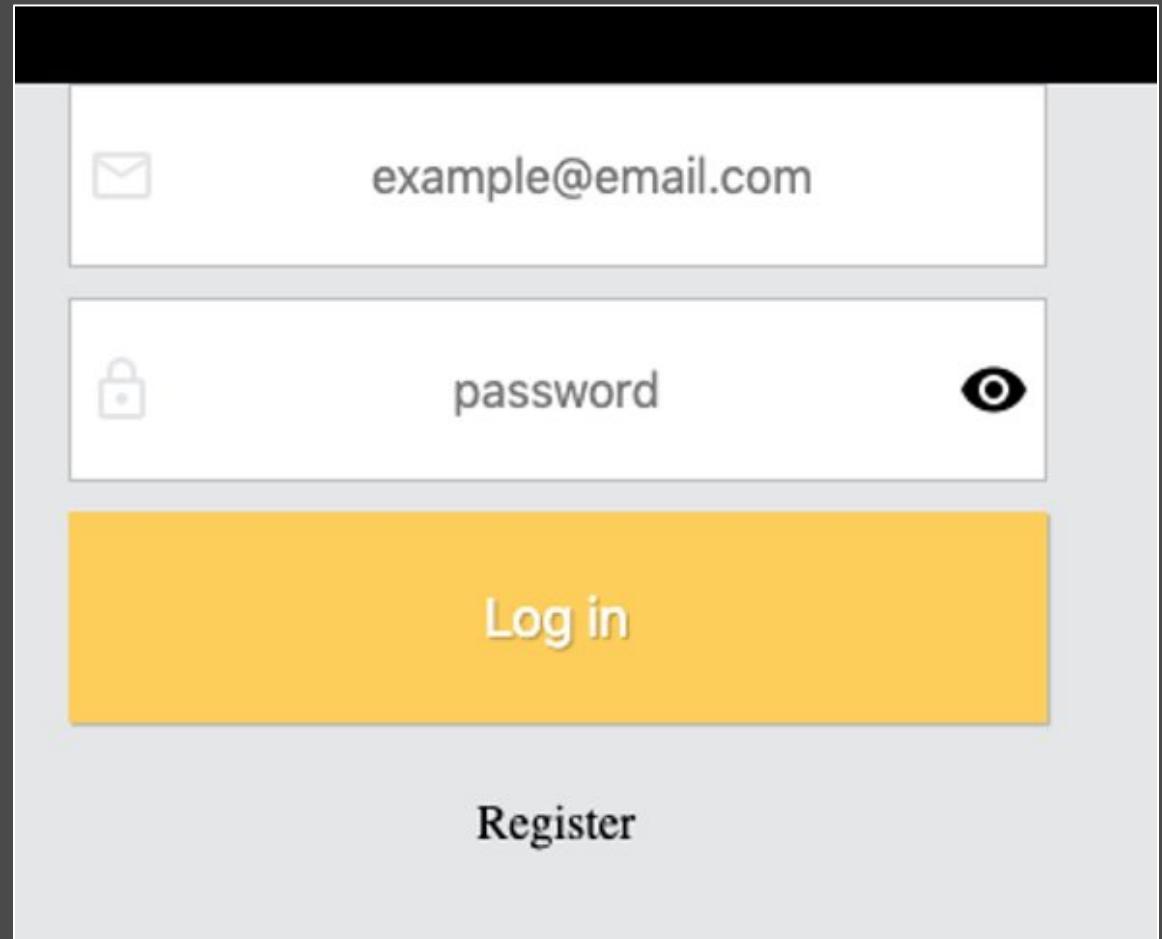
The image shows a user interface for a login or sign-up page. At the top is a black header bar. Below it is a light gray main area containing two input fields. The first input field for 'example@email.com' has an '@' symbol icon to its left. The second input field for 'password' has a lock icon to its left and an eye icon to its right for password visibility. Below these fields are two buttons: 'Log in' and 'Register'.

CSS properties you'll need for this task: **display**, **border**, **margin**,
padding, **background**, **flex**, **text-align**, **font-size**

Exercise #1 buttons

- Put both buttons inside a **flex container**
- Each button should be on a separate line
- Register button should be plain black text, middle-aligned with no underline
- Both buttons should have top and bottom padding of 25 pixels
- Login button should have white text – 18 pixels in size – on yellow background – **#ffcf51** – and no border
- Button and text have shadow with **x:1 y:1 blur:1** values and **rgba(0,0,0,0.24)** color

CSS properties you'll need for this task: **display**, **flex-direction**, **text-decoration**, **text-align**, **color**, **padding**, **background**, **border**, **box-shadow**, **text-shadow**, **font-size**

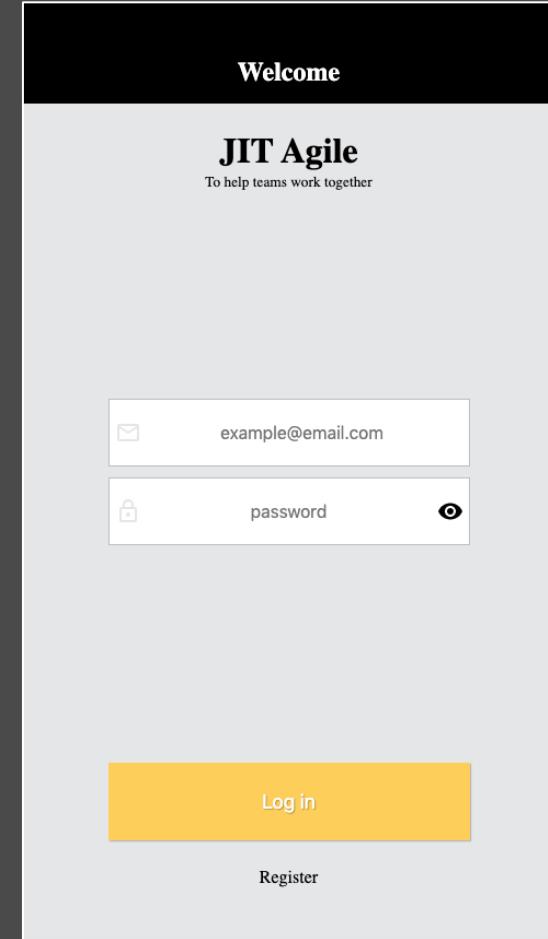


Exercise #1 layout

- Turn `.container` into a grid container
- Calculate its height based on the height of its parent minus the height of the primary header
- Form should have height 100%
- Form should be a grid container
- Grid items should have space around

CSS properties you'll need for this task:

`display`, `height + CSS calc`, `justify-items`



Exercise #2 register page

