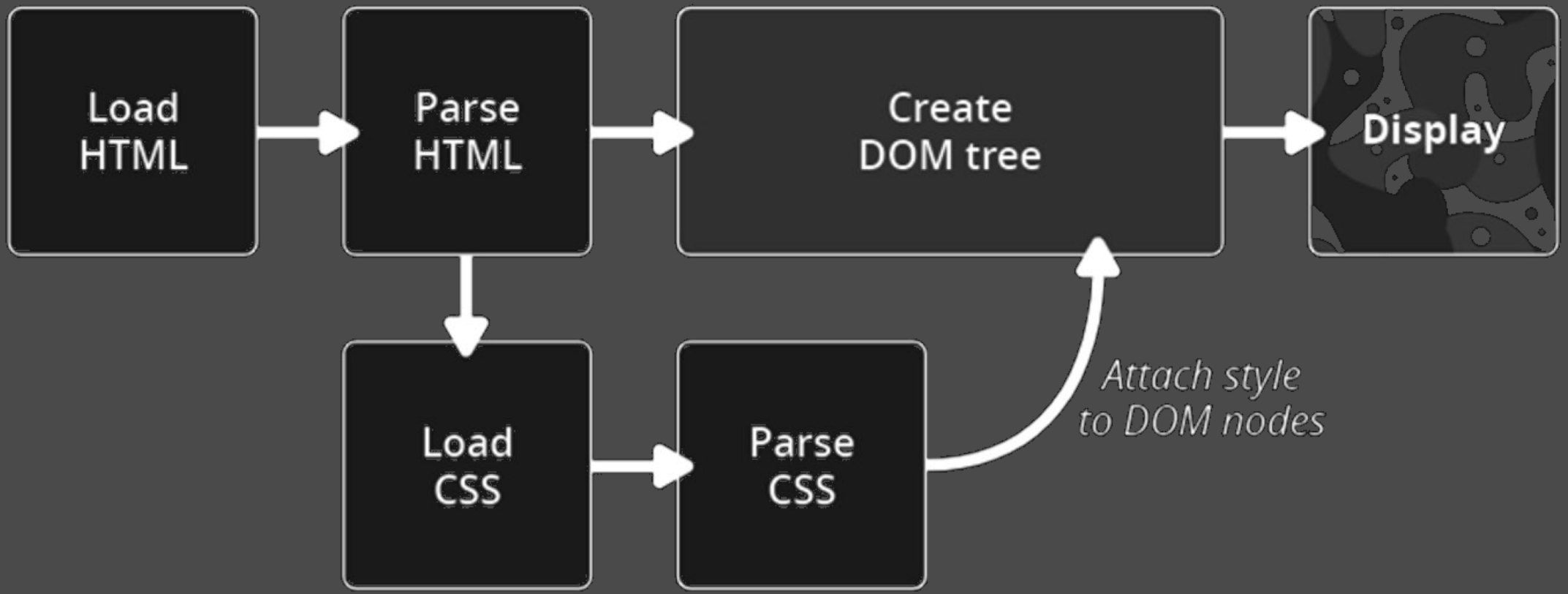
Jit Team & 3LO 2019-2020

{ CSS }

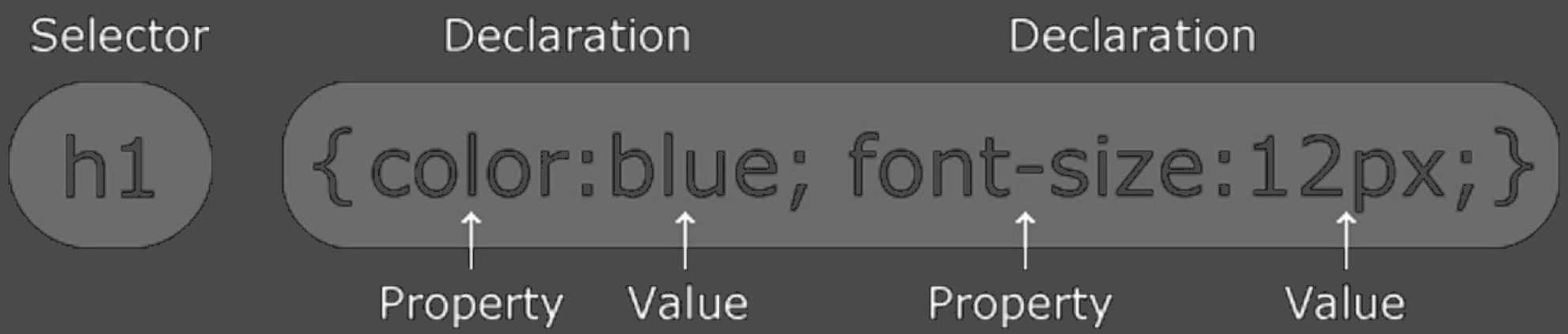
CSS cascading style sheets

Language describing how HTML elements should look like.

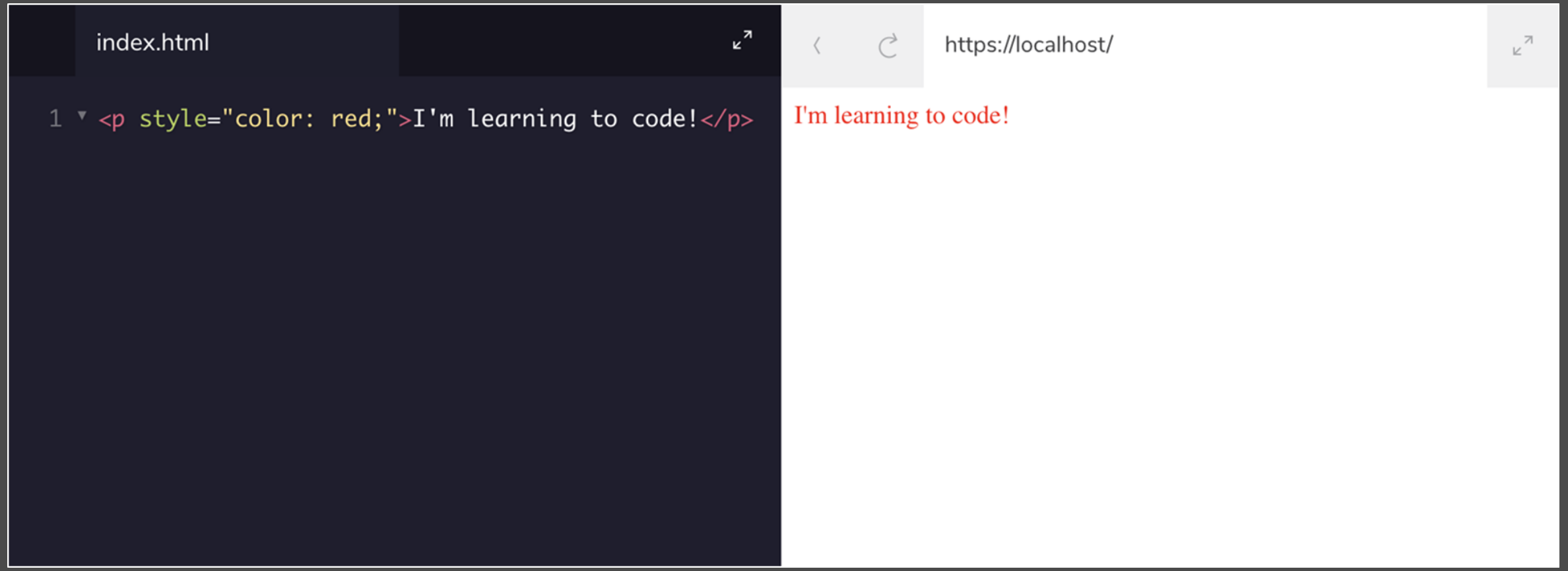
CSS behind the scenes



CSS syntax


Selector
Declaration
Declaration
h1
{color:blue; font-size:12px;}
Property
Value
Property
Value

CSS inline styles



CSS <style> tag

index.html style.css

```
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>
```

https://localhost/

I'm bigger and red too.

CSS external styles

index.html

```
<!DOCTYPE html>
<html>
  <head>
    <title>Hello World</title>
    <link href="./style.css" type="text/css" rel="stylesheet">
  </head>
  <body>
    <p>I took style from external file.</p>
  </body>
</html>
```

https://localhost/

I took style from external file.

style.css

```
p {
  font-family: Arial;
  color: red;
}
```

CSS selectors: tag name

CSS selectors: class name

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

```css
.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

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

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

CSS nested selectors

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

```css
.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: "";
}
```

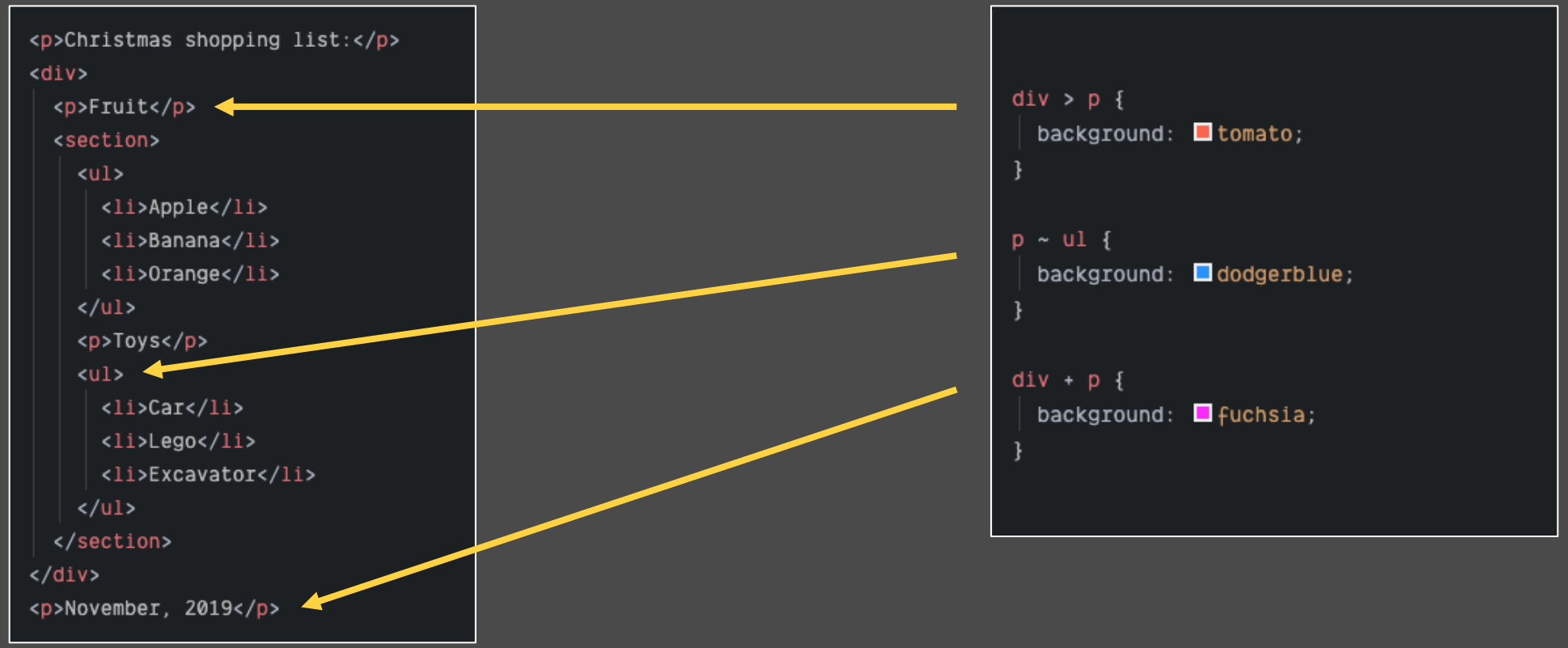
The **:before** and **:after** pseudo-elements allow you to insert content before or after any HTML element that isn't self closing (like <img> and <input>).

The content property is required but can be left blank.

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

CSS combinators

- 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 preceeded by a div element



CSS multiple selectors

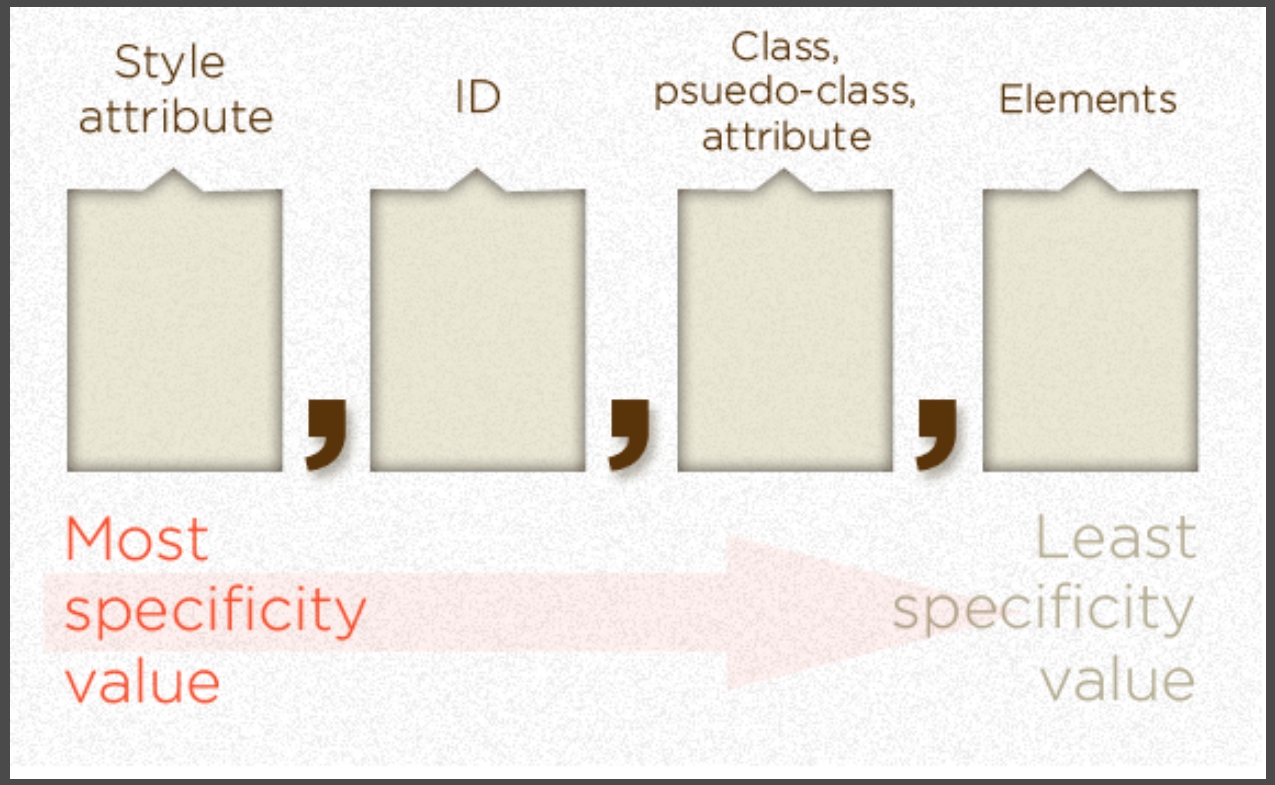
```css
h1 {
  font-family: Georgia;
}

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

=

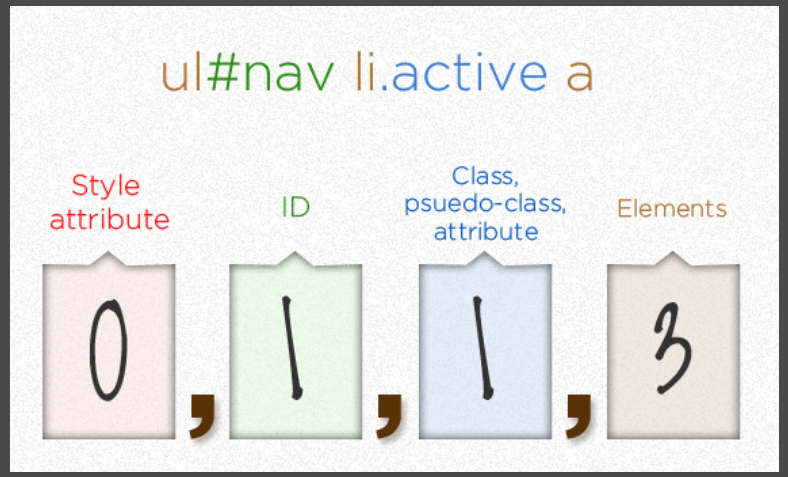
```css
h1,
.menu {
  font-family: Georgia;
}
```

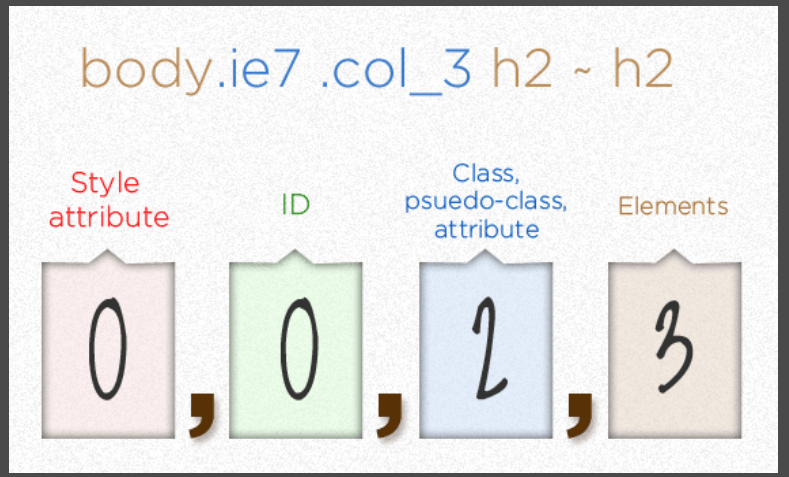
CSS specificity

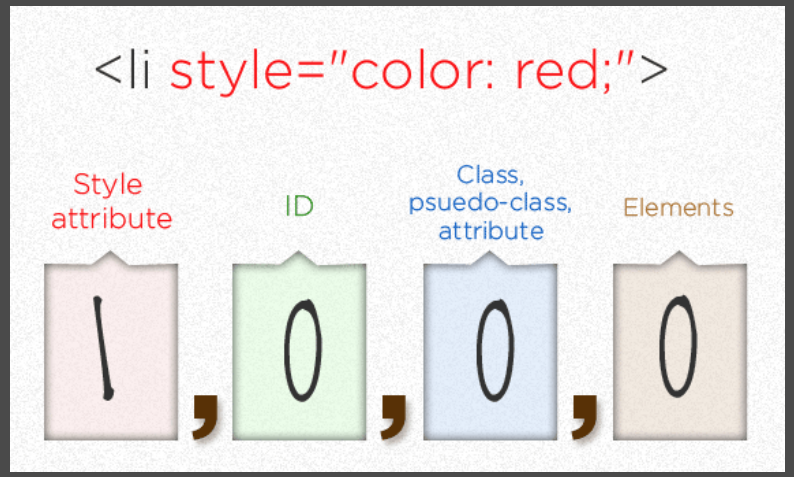
Specificity is the order by which the browser decides which CSS styles will be displayed.

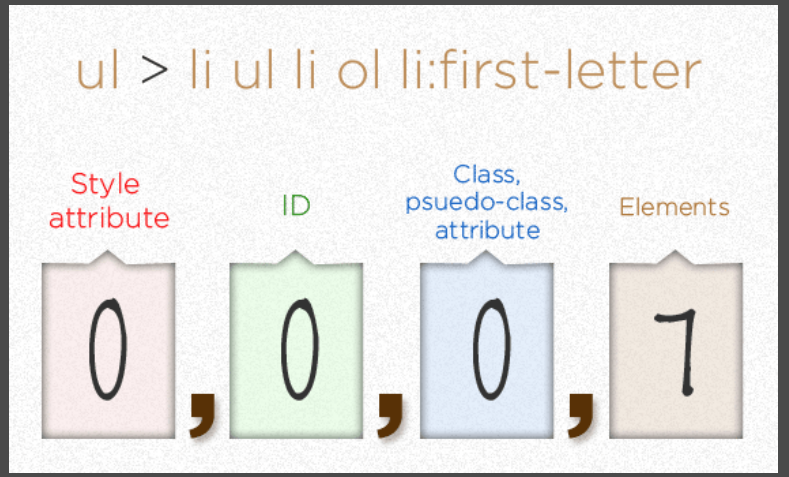


CSS specificity





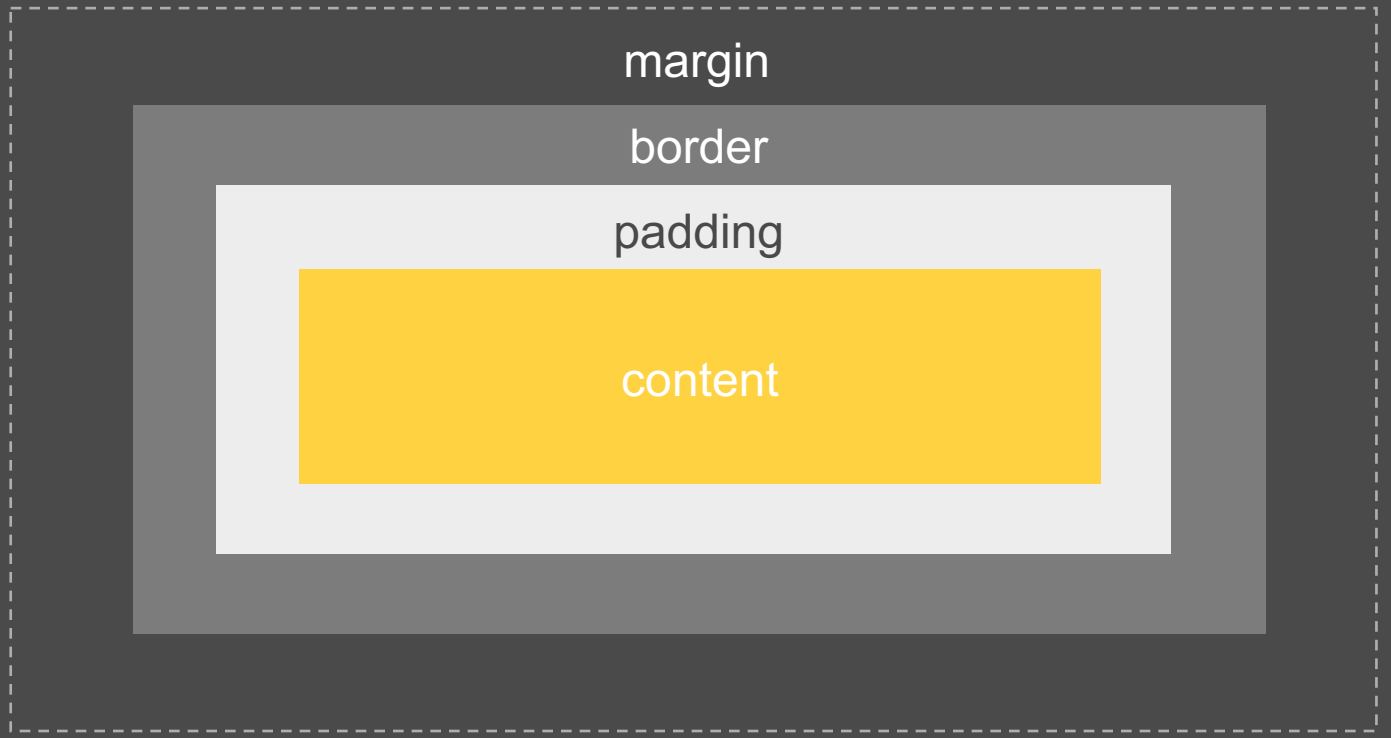




CSS basic properties

- 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



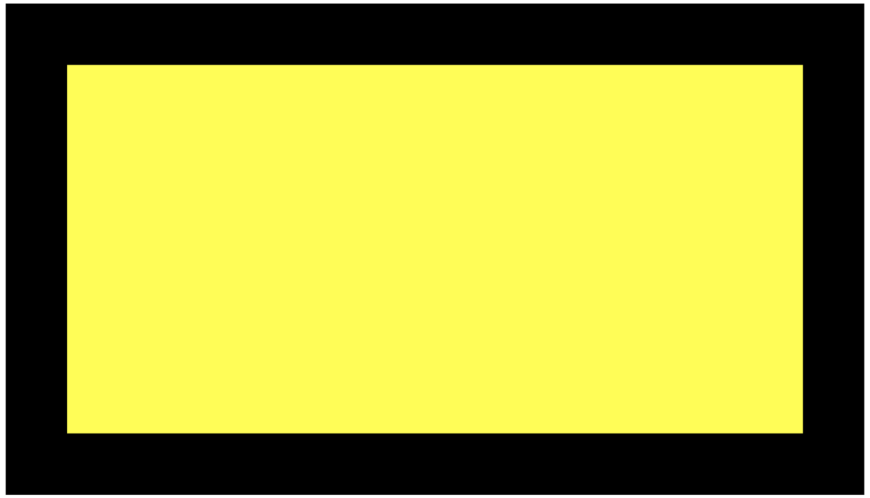
CSS box model

style.css

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

index.html

https://localhost/



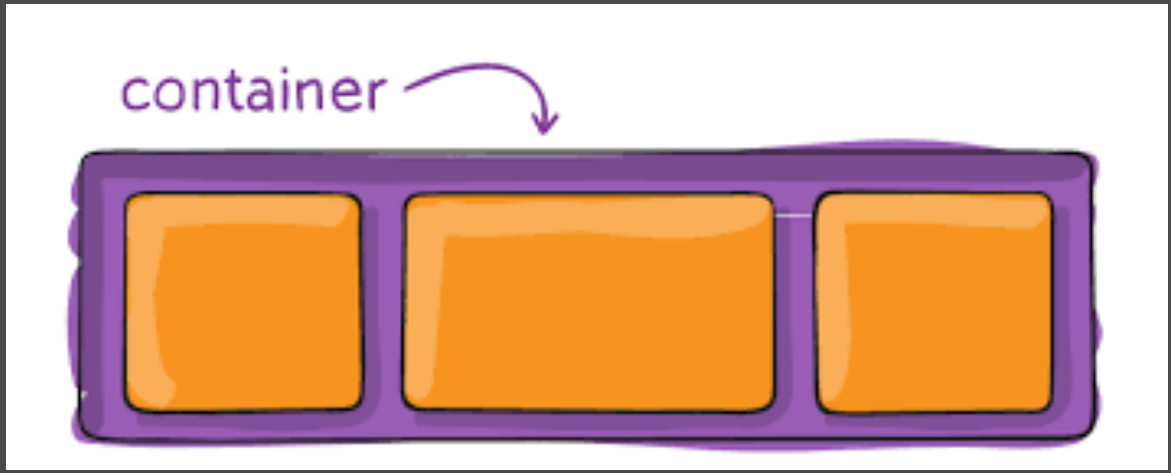
CSS media queries

style.css

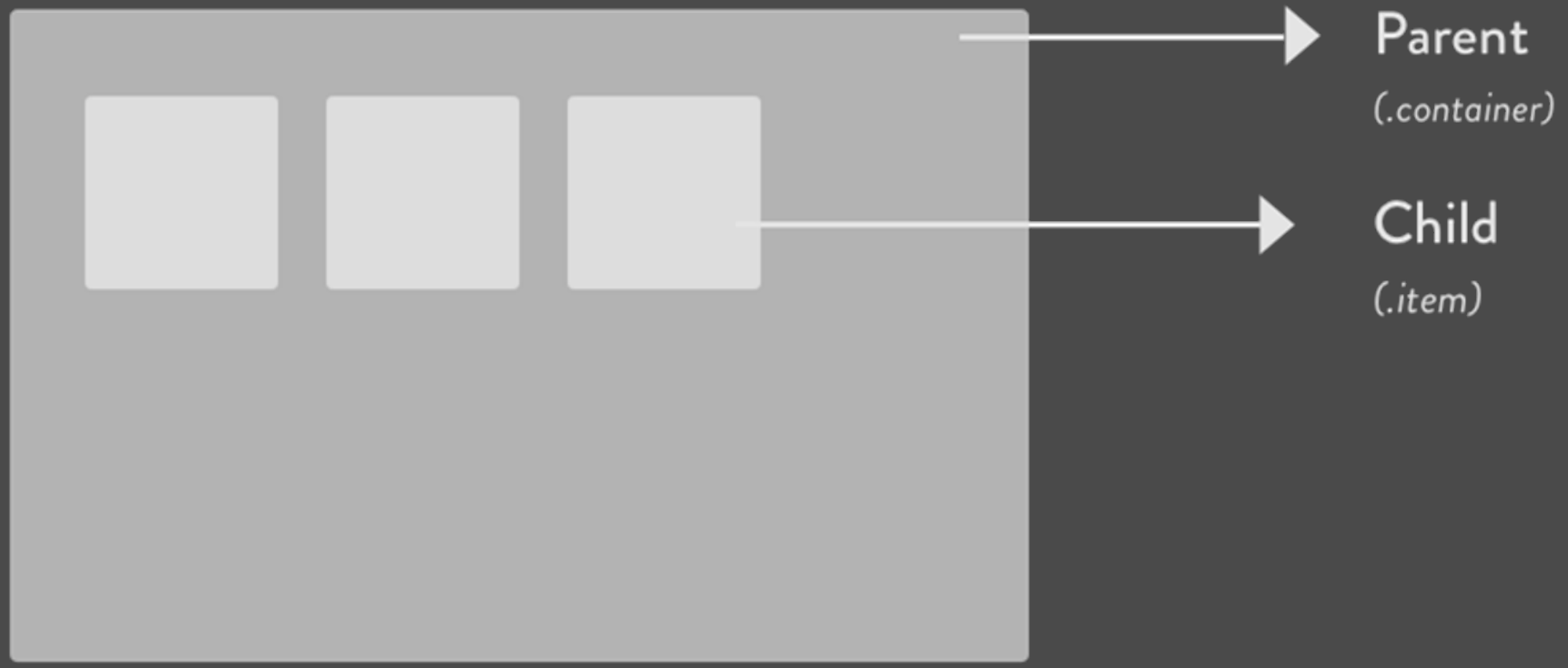
```
/* Large desktop */
@media (min-width: 1200px) {
  /* CSS Classes */
}
/* Portrait tablet to landscape and desktop */
@media (min-width: 768px) and (max-width: 979px)
{
  /* CSS Classes */
}
/* Landscape phone to portrait tablet */
@media (max-width: 767px) {
  /* CSS Classes */
}
/* Landscape phones and down */
@media (max-width: 480px) {
  /* CSS Classes */
}
```

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

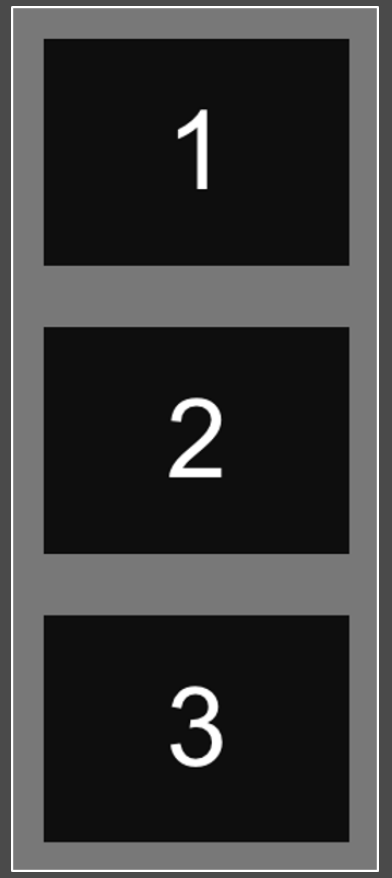
Flex item properties

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

CSS flexbox properties

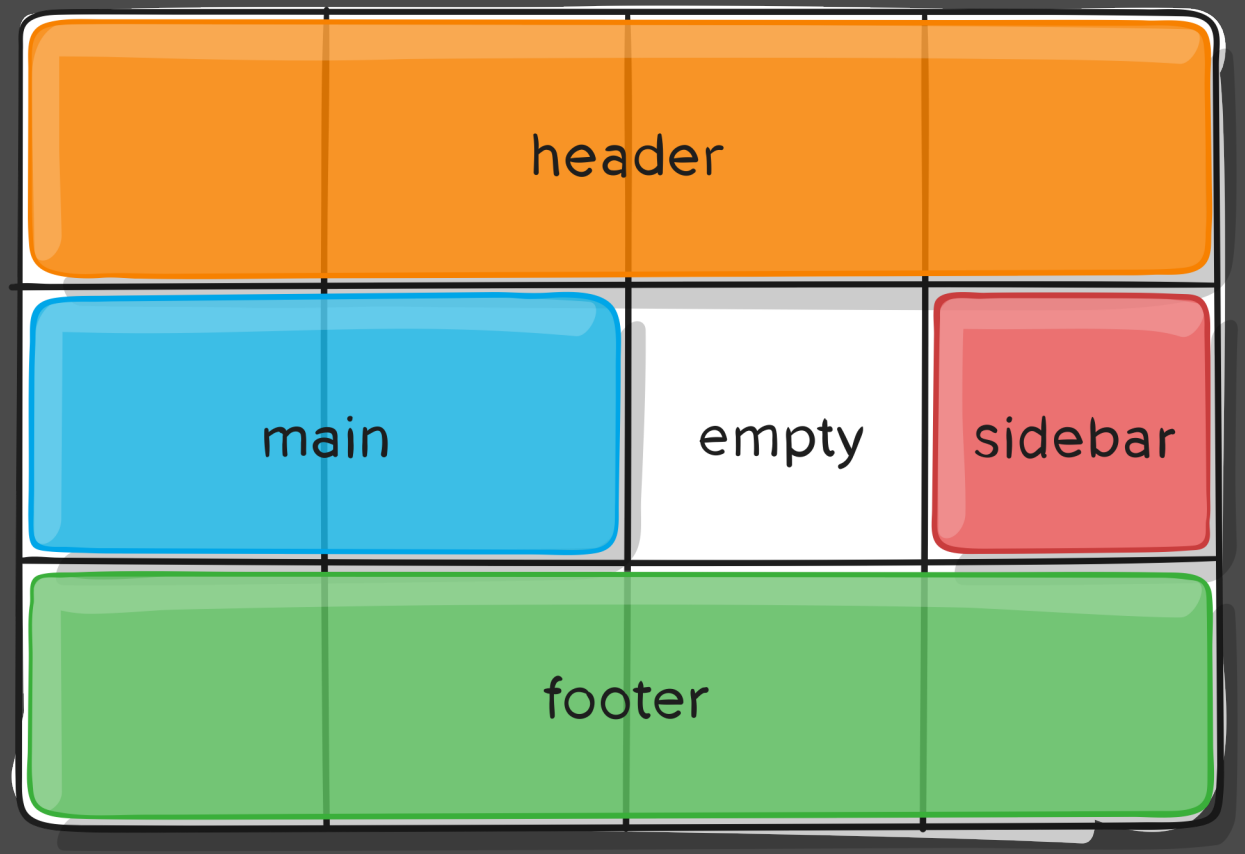
```html
<div class="flex-container">
  <div>1</div>
  <div>2</div>
  <div>3</div>
</div>
```

```css
.flex-container {
  display: flex;
  flex-direction: column;
}
```

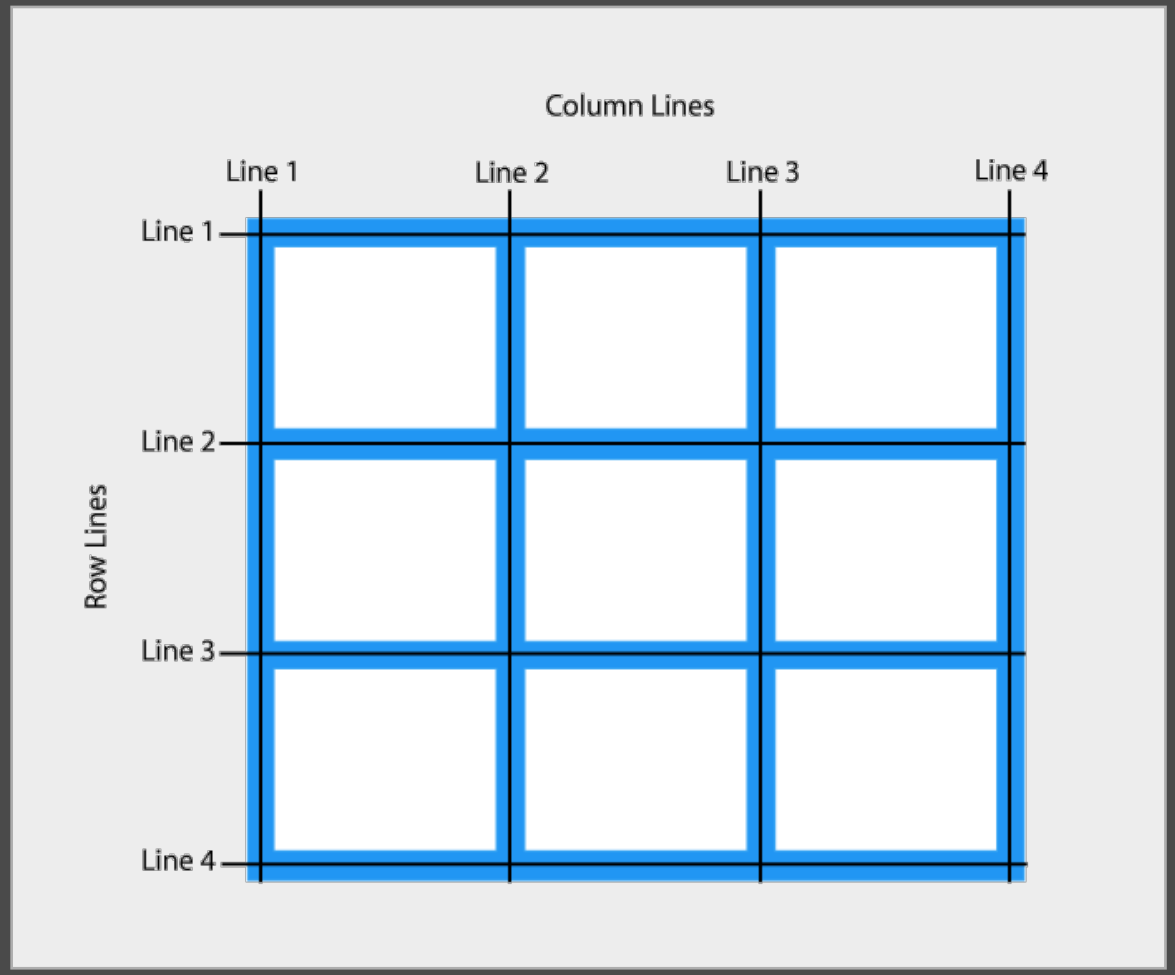


Grid Layout Module

Makes creating layouts easier



Grid anatomy



Grid properties

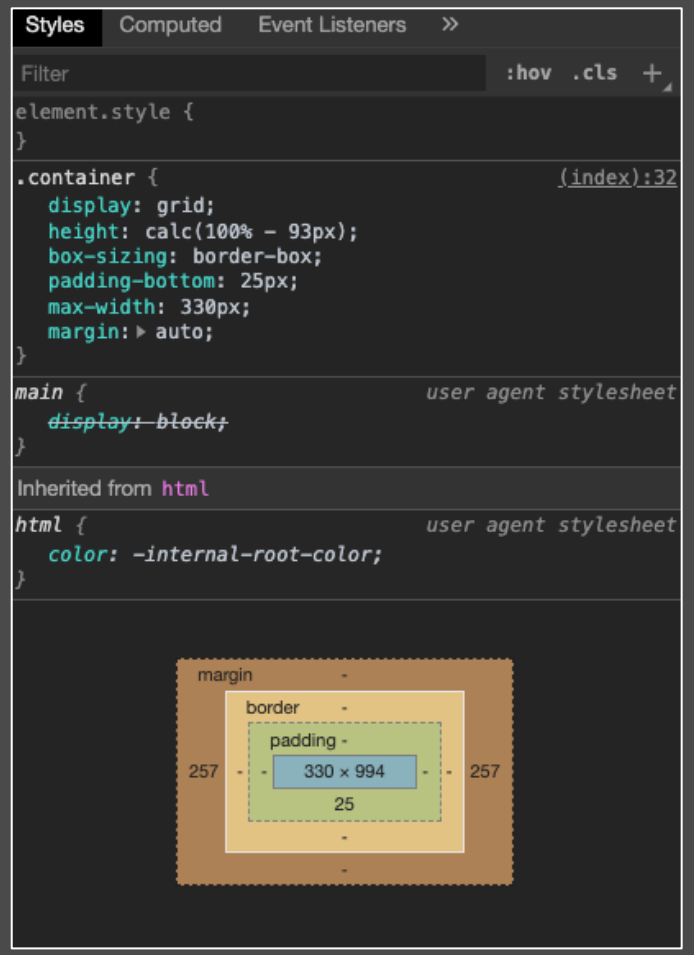
Grid container properties:

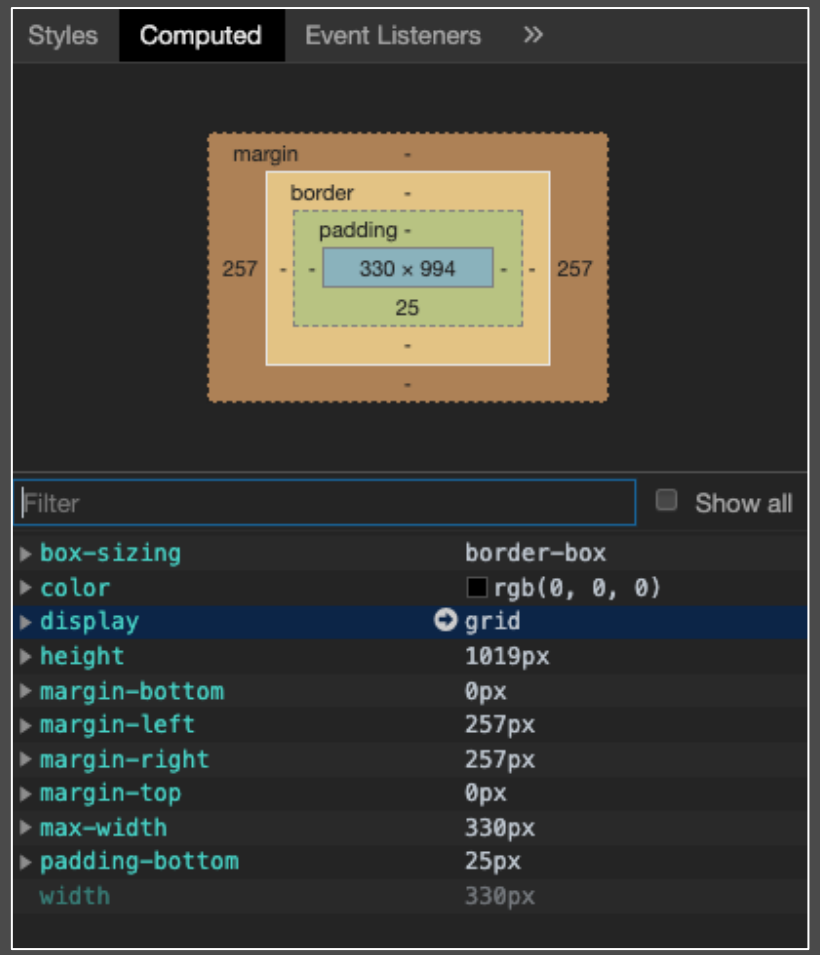
- Grid-template-columns
- Grid-template-rows
- Grid-template-areas
- Grid-gap
- Justify-items (horizontally)
- Justify-content
- Align-items (vertically)
- Align-content
- Grid-auto-columns
- Grid-auto-rows
- Grid-auto-flow

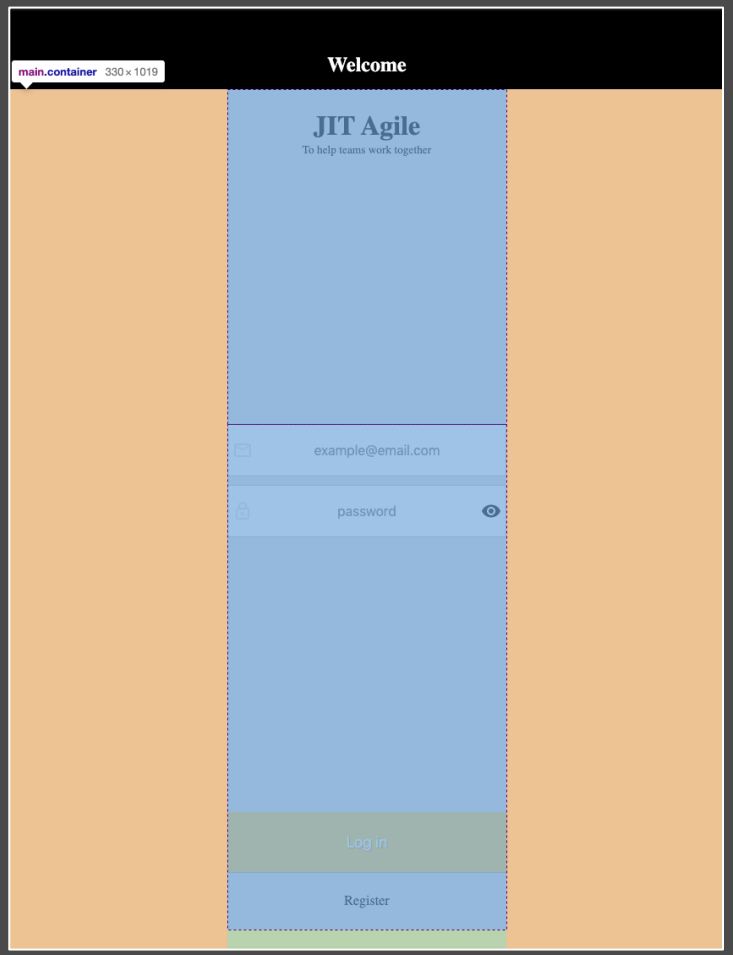
Grid item properties:

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

CSS debugging



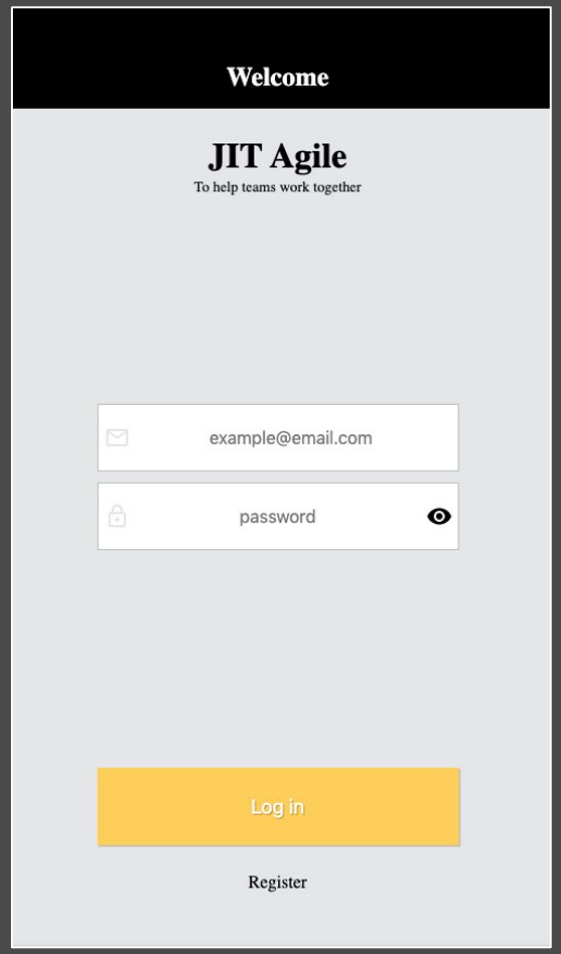




CSS where to go from here?

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

Exercise #1 login page



Exercise #1 primary header

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

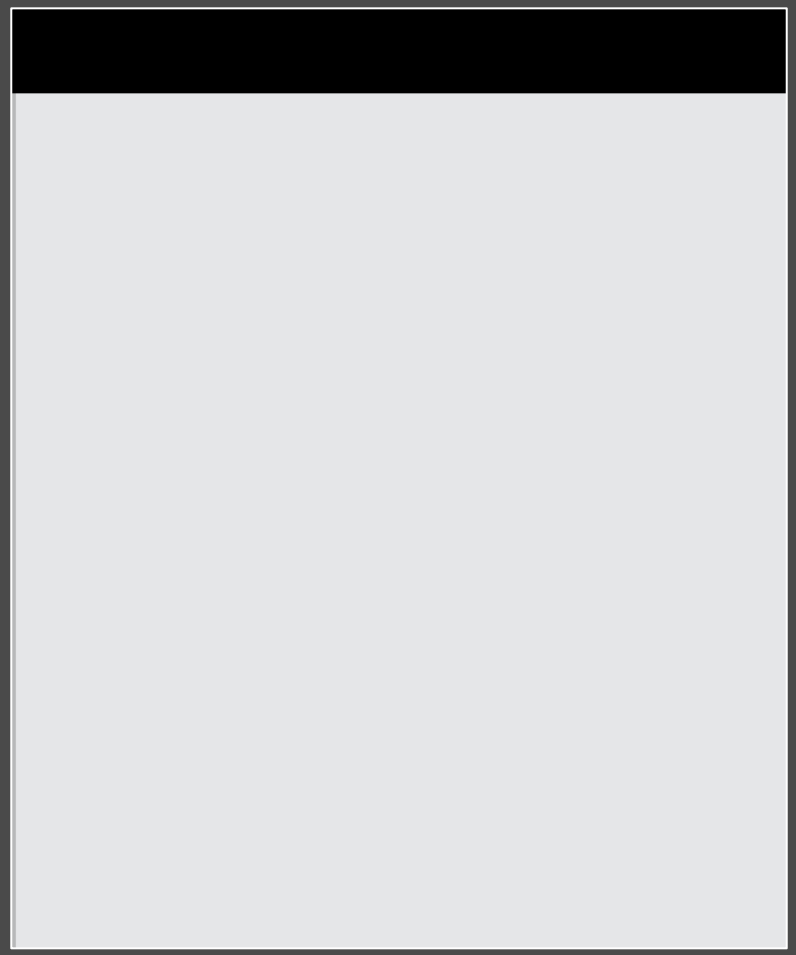
CSS properties you'll need for this task: background, padding, margin, color, text-align

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:

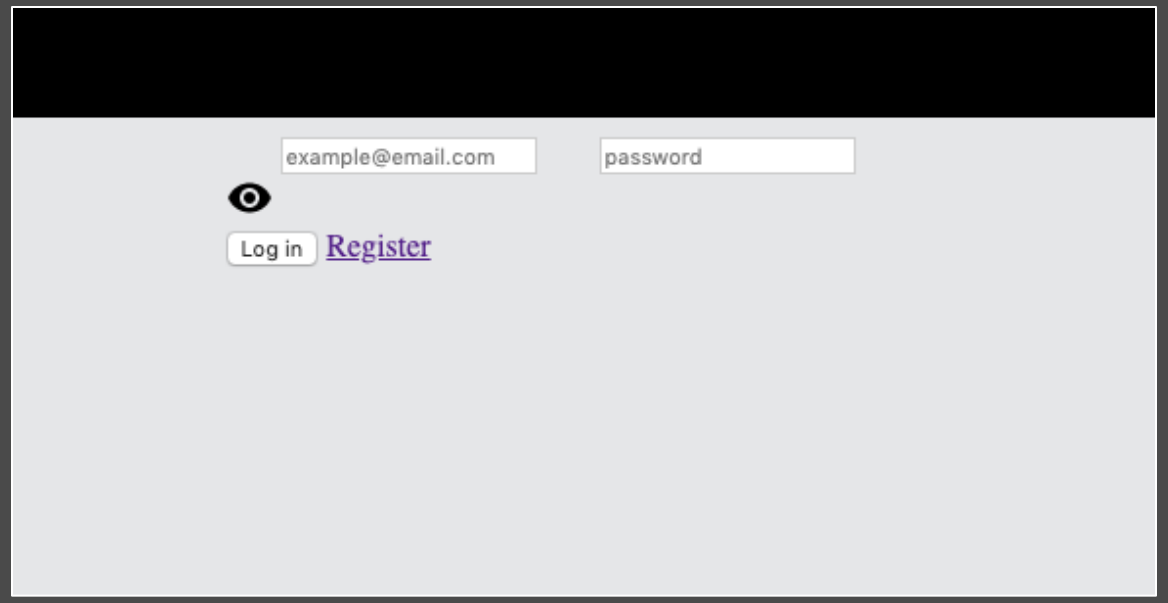
- be aligned horizontally on the page
- Have a 25 px space from primary header

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

Exercise #1 form

It should:

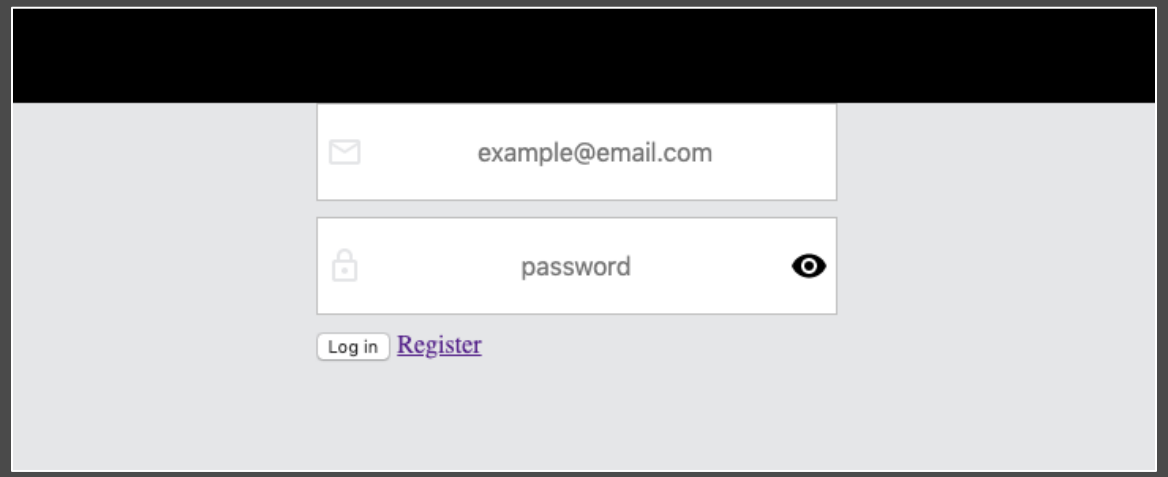
- be centered on the page
- Span 100% of the viewport width up to 330 pixels



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

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

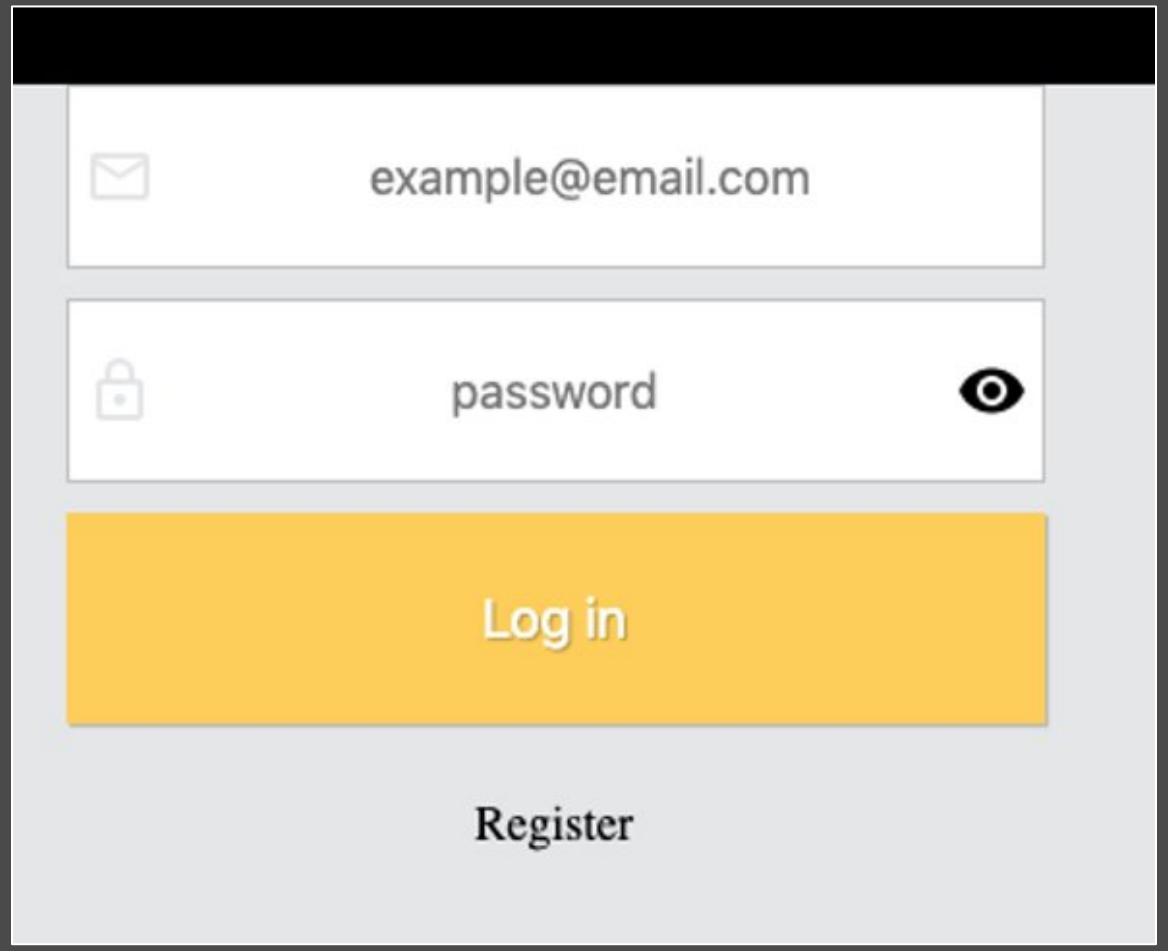


CSS properties you'll need for this task: display, border, margin, padding, background, flex, text-align, font-size

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

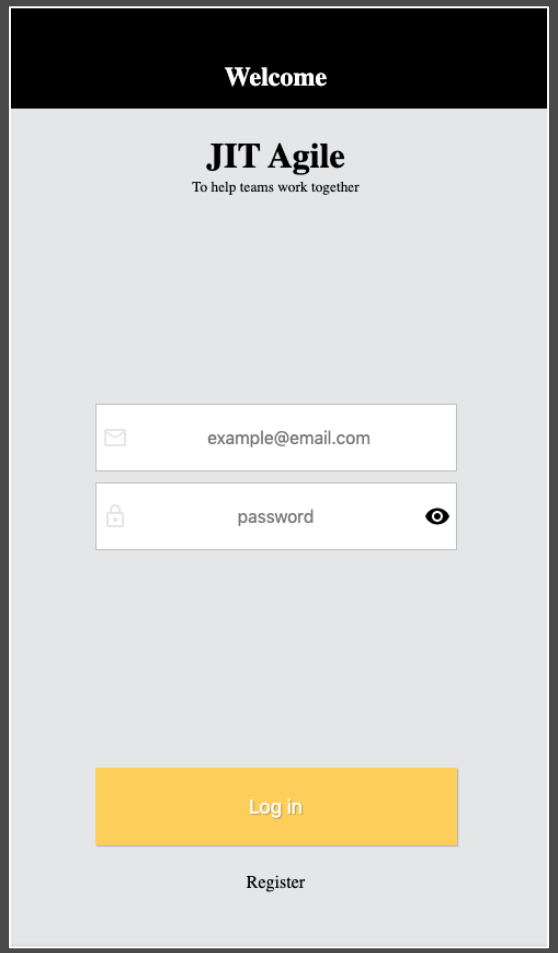


Exercise #1 layout

- 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