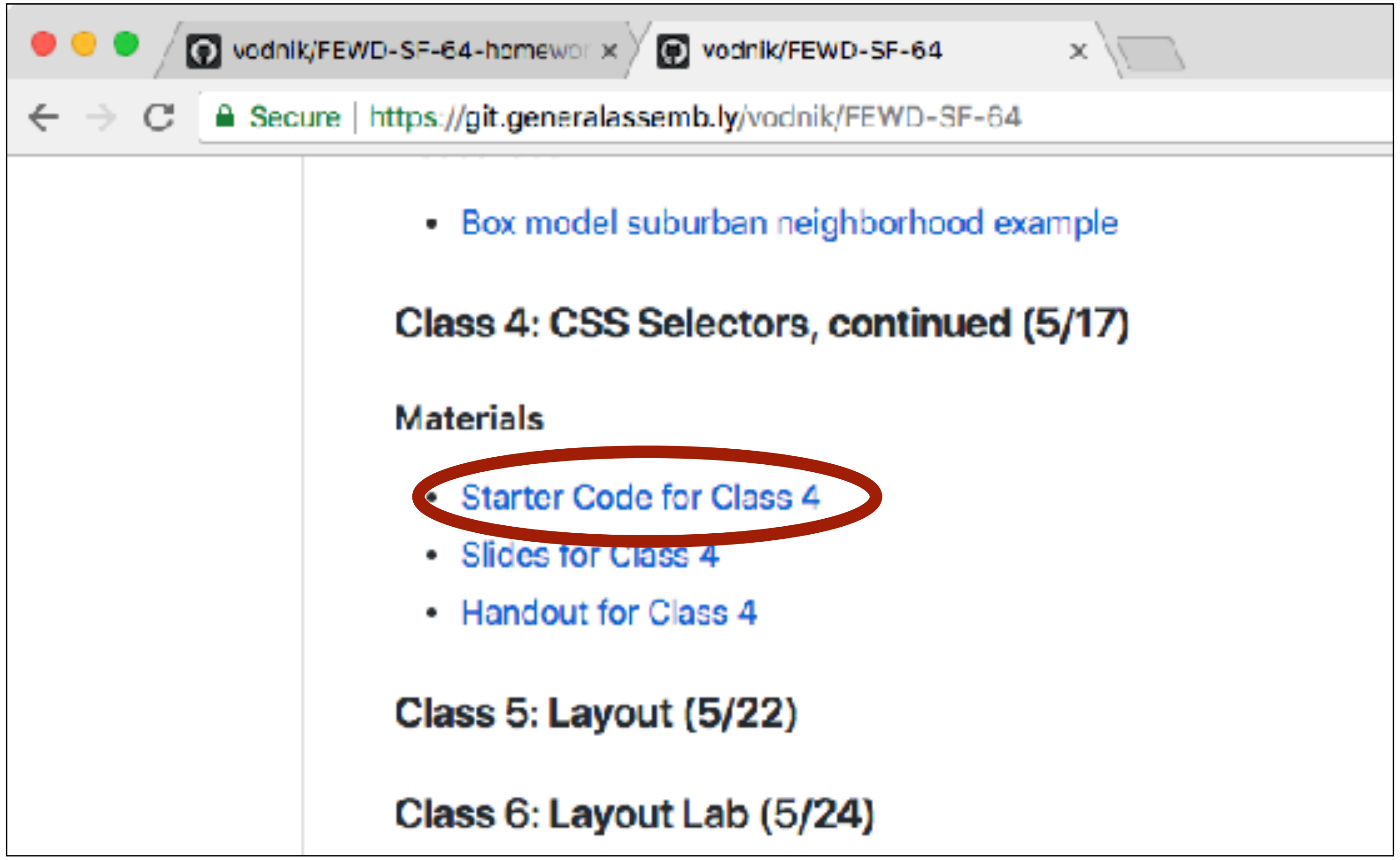
HELLO!

- Go to the class repo on GitHub Enterprise
- Click the **Starter Code for Class 5** link to download the files
- Move the downloaded file to your FEWD folder
- Unzip the files



- Box model suburban neighborhood example

Class 4: CSS Selectors, continued (5/17)

Materials

- Starter Code for Class 4
- Slides for Class 4
- Handout for Class 4

Class 5: Layout (5/22)

Class 6: Layout Lab (5/24)

LAYOUT

WEEKLY OVERVIEW

WEEK 3	Layout & Grid Systems / Layout Lab
WEEK 4	Responsive Design / CSS Positioning
WEEK 5	Forms / Final Project Lab

LEARNING OBJECTIVES

- Differentiate between block and inline elements
- Create multi-column layouts with flexbox
- Describe a grid system
- Use classes to set up modular, multi-column layouts

AGENDA

- Review
- CSS Specificity
- Display
- Flexbox & Multi-column Layouts
- Monument Lab

EXIT TICKET QUESTIONS

1. Besides anchor elements, what other children don't inherit their parent's properties?

HOMEWORK REVIEW

ACTIVITY

EXERCISE

KEY OBJECTIVE

- Review Monument blog and show off your work

TYPE OF EXERCISE

- Groups of 3-4

TIMING

6 min

1. Open Monument blog sites on laptops and display them proudly!
2. Give feedback to your peers: "I like" and "I wish/wonder”
3. Share a challenge you ran into in your project and discuss how other group members may have worked with it.

ACTIVITY

EXERCISE

KEY OBJECTIVE

- Review final project ideas

TIMING

1 min	1. Share your final project idea in the `#homework` channel on Slack

ACTIVITY — REVIEW

- When would you want to use a **`class`** and when would you want to use an **`id`**?
- What are three ways we could select these anchors in our CSS?

```
<header>
    <h1><a href="">Nested Selectors</a></h1>
    <nav class="main-nav">
        <a href="">Home</a>
        <a href="">About</a>
        <a href="">Contact</a>
        <a href="">Blog</a>
    </nav>
</header>
```

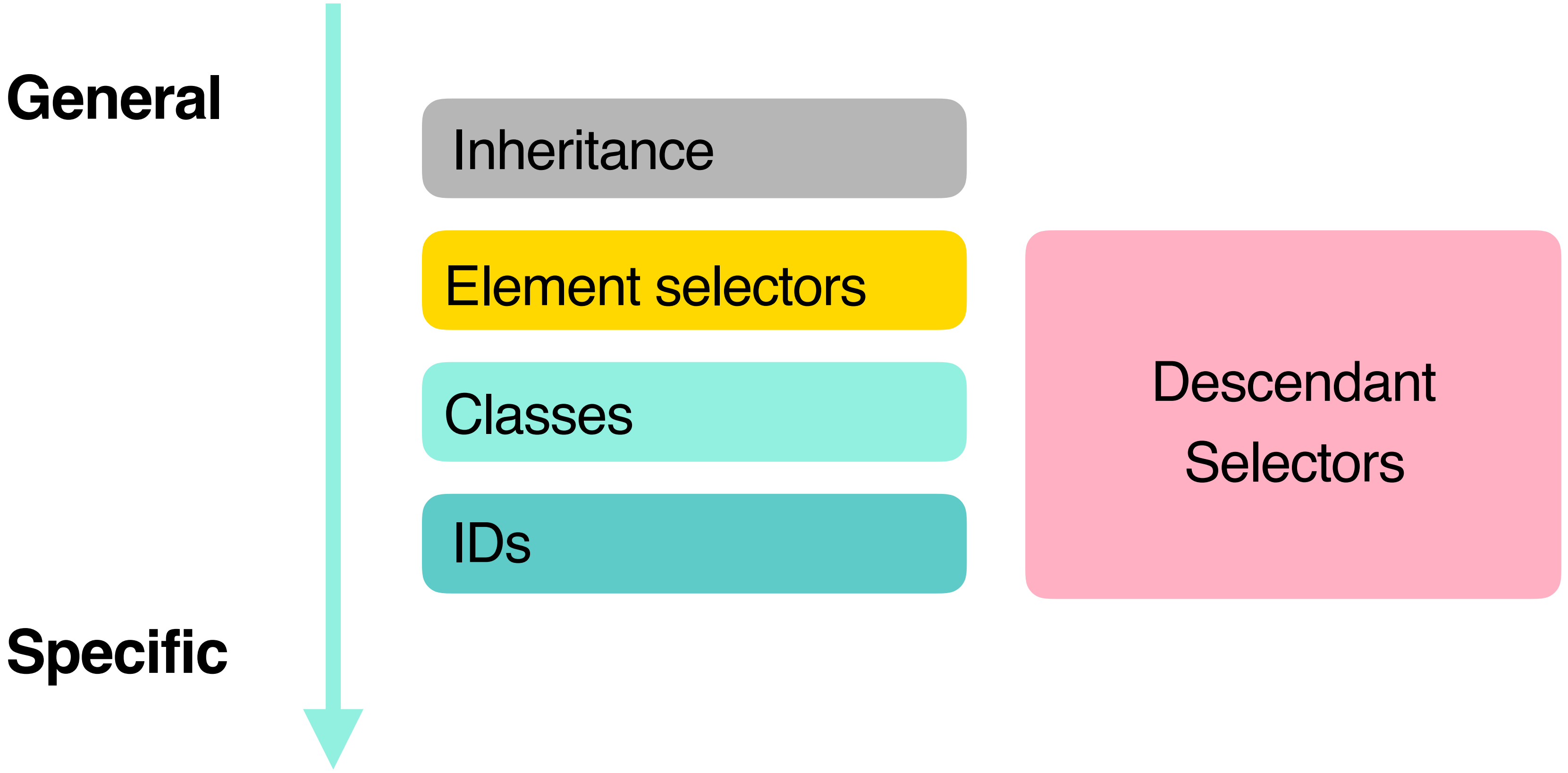
WHY IS CSS “CASCADING”?

SPECIFICITY GAME!!!

MORE ABOUT CASCADING — GENERAL TO MORE SPECIFIC

SPECIFICITY:

The *more specific rule* will take precedence over the more general rule



CSS CASCADING — THE SPECIFICITY GAME

- If multiple style rules are targeted at the same element, which style will be applied?
- We can calculate the specificity of the selectors to find out!
- The styles for the more specific selector are the styles that will be applied.

```css
p a {
    font-size: 50px;
}

a {
    font-size: 30px;
}
```

CSS CASCADING — THE SPECIFICITY GAME

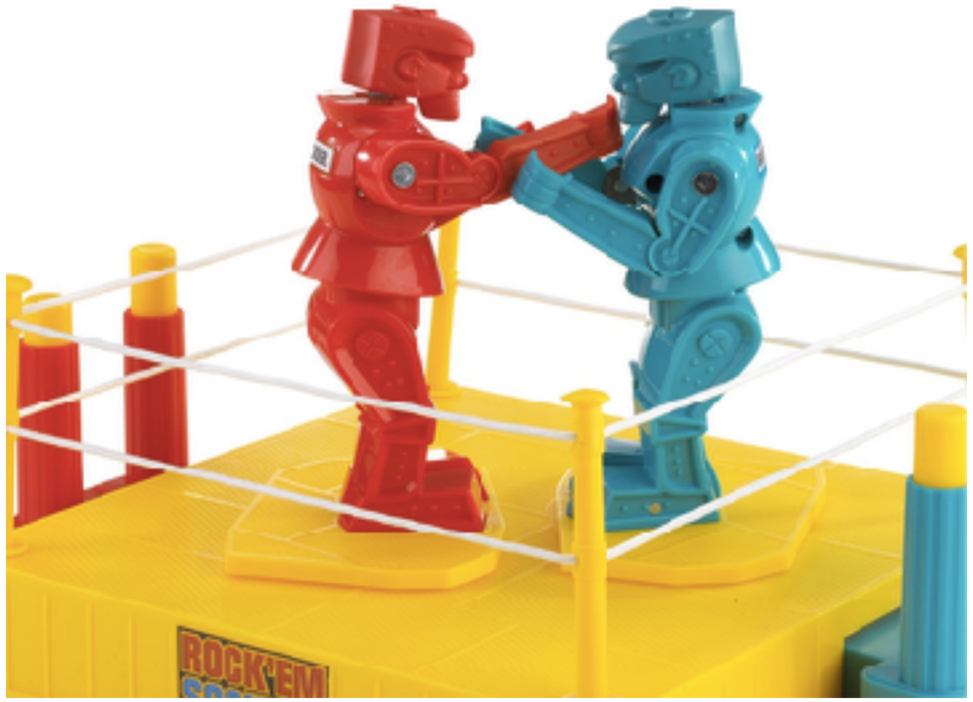
TO CALCULATE SPECIFICITY,
TAKE A LOOK AT THE ENTIRE "SELECTOR CHAIN"

```
p a {
    font-size: 50px;
}
```

CSS CASCADING — THE SPECIFICITY GAME

If you have conflicting styles under the “p a” selector and the “a” selector… who will win?

p a a



CSS CASCADING — THE SPECIFICITY GAME

Use a chart like this to calculate the specificity to find out!

IDs	Classes	Elements

CSS CASCADING — THE SPECIFICITY GAME

2 ELEMENTS

```
p a {
    font-size: 50px;
}
```

IDs	Classes	Elements
0	0	2

SCORE: 2

CSS CASCADING — THE SPECIFICITY GAME

1 CLASS

```
.about {
    font-size: 50px;
}
```

IDs	Classes	Elements
0	1	0

SCORE: 10

CSS CASCADING — THE SPECIFICITY GAME

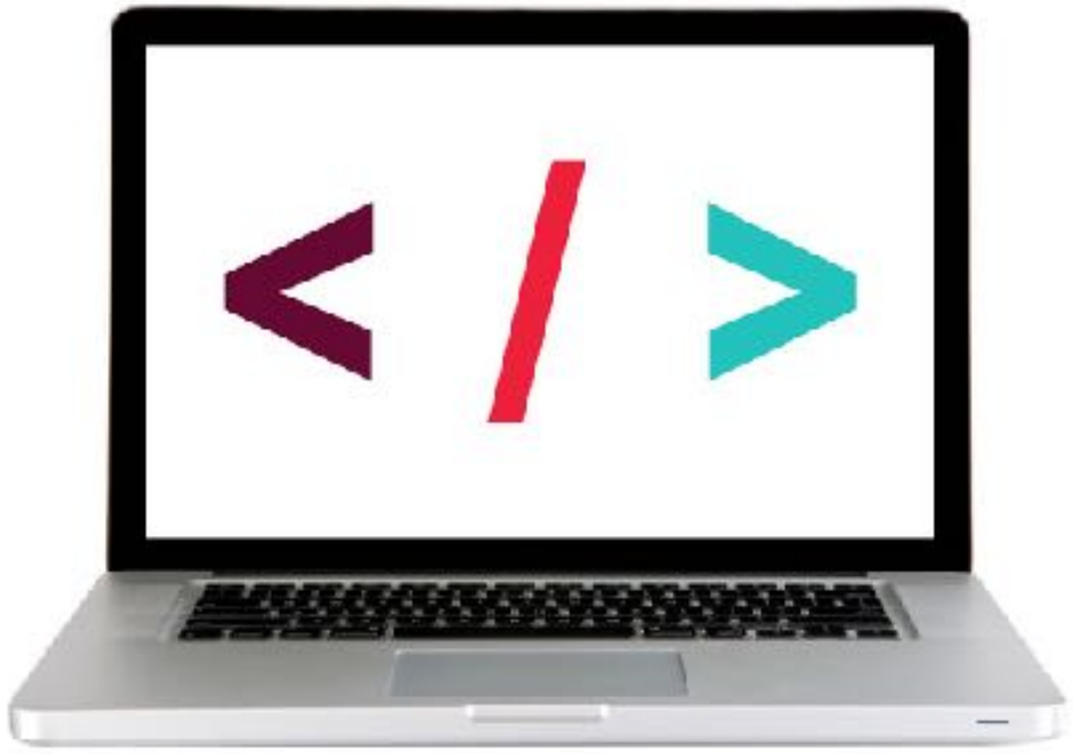
ID ELEMENT

```
#about a {
    font-size: 50px;
}
```

IDs	Classes	Elements
1	0	1

SCORE: 101

DEMO: HOW MANY CLASSES TO OVERRIDE AN ID?



https://codepen.io/svodnik/pen/PegwzR?editors=1100#0

CSS CASCADING — THE SPECIFICITY GAME

Write this down!

IDs	Classes	Elements

CSS CASCADING — THE SPECIFICITY GAME

POLL: *Will the font-size for the anchor be 50px or 30px?*

HTML:

```html
<p>Visit my <a href="#">Website</a></p>
```

CSS:

```css
p a {
    font-size: 50px;
}

a {
    font-size: 30px;
}
```

CSS CASCADING — THE SPECIFICITY GAME

p a

IDs	Classes	Elements
0	0	2

a

IDs	Classes	Elements
0	0	1

WINNER!

CSS CASCADING — THE SPECIFICITY GAME

POLL: *Will the font-size for the anchor be 50px or 30px?*

HTML:

```html
<p>Visit my <a href="#">Website</a></p>
```

CSS:

```css
p a { /* Score: 2 */
    font-size: 50px;
}

a { /* Score: 1 */
    font-size: 30px;
}
```

POLL: *Will the anchor with the class* home *be pink or blue?*

HTML:

```
<nav id="main-nav">
    <a href="#" class="home">Home</a>
    <a href="#">About</a>
    <a href="#">Resume</a>
</nav>
```

CSS:

```
#main-nav a {
    color: pink;
}

.home {
    color: blue;
}
```

CSS CASCADING

`#main-nav a`

IDs	Classes	Elements
1	0	1

WINNER!

`.home`

IDs	Classes	Elements
0	1	0

CSS CASCADING — THE SPECIFICITY GAME

POLL: *Will the anchor with the class* home *be pink or blue?*

HTML:

```html
<nav id="main-nav">
    <a href="#" class="home">Home</a>
    <a href="#">About</a>
    <a href="#">Resume</a>
</nav>
```

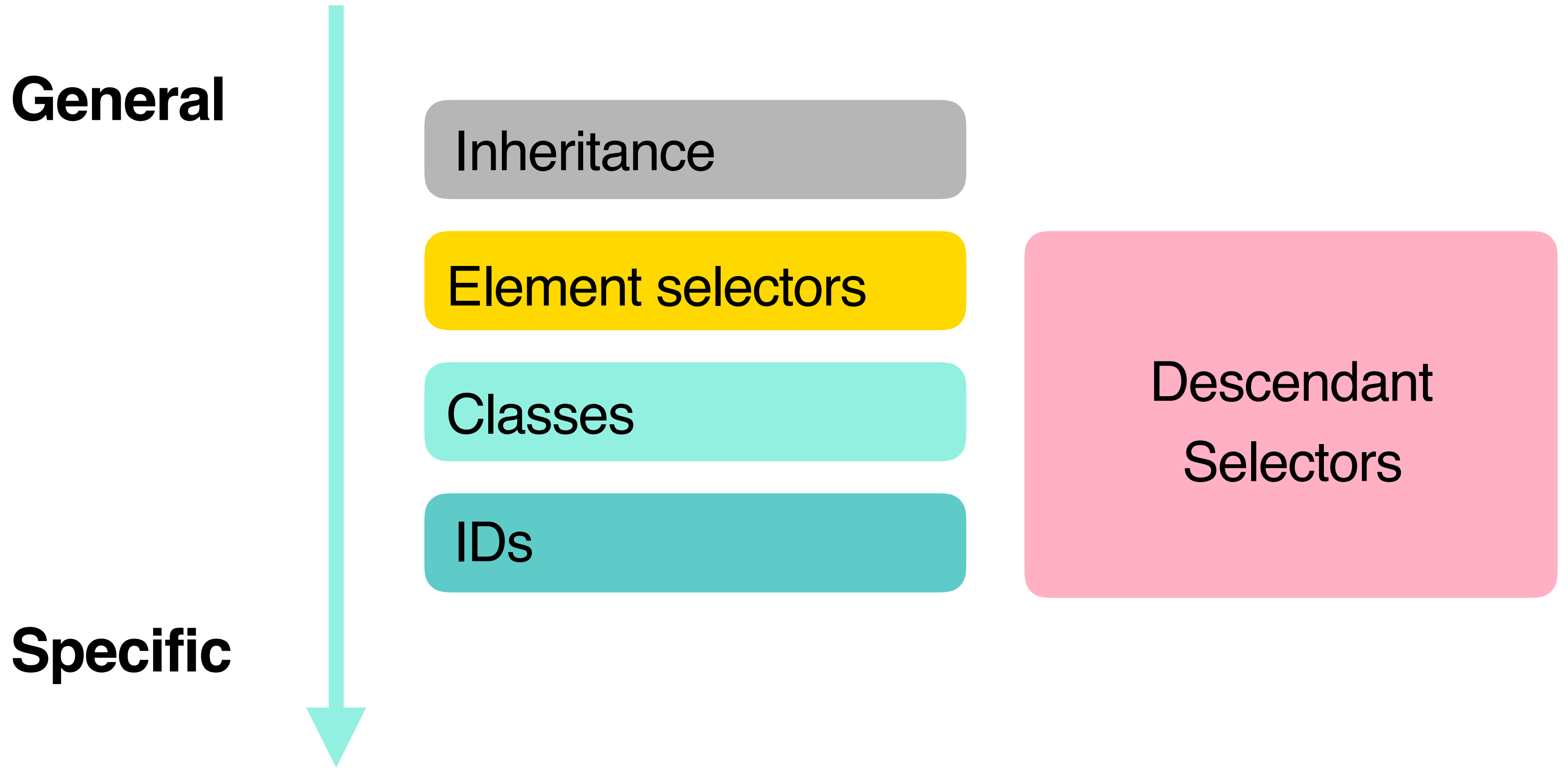
CSS:

```css
#main-nav a { /* Score: 101 */
    color: pink;
}

.home { /* Score: 10 */
    color: blue;
}
```

MORE ABOUT CASCADING — GENERAL TO MORE SPECIFIC

SPECIFICITY:

The *more specific rule* will take precedence over the more general rule



LAST RULE

CSS CASCADING — THE SPECIFICITY GAME

POLL: *Will the anchor have an underline or no underline?*

HTML:

```html
<li>
    <p>Visit this <a href="#">cool</a> site.</p>
</li>
```

CSS:

```css
li a { /* Score: 2 */
    text-decoration: underline;
}

p a { /* Score: 2 */
    text-decoration: none;
}
```

LAST RULE

`li a`

IDs	Classes	Elements
0	0	2

`p a`

IDs	Classes	Elements
0	0	2

TIE?!?

CSS CASCADING — THE SPECIFICITY GAME

POLL: *Will the anchor have an underline or no underline?*

HTML:

```html
<li>
    <p>Visit this <a href="#">cool</a> site.</p>
</li>
```

CSS:

```css
li a { /* Score: 2 */
    text-decoration: underline;
}

p a { /* Score: 2 */
    text-decoration: none;
}
```

TIEBREAKER? LAST RULE WINS!

CSS CASCADING — THE SPECIFICITY GAME

Let’s say we switched it…

HTML:

```html
<li>
    <p>Visit this <a href="#">cool</a> site.</p>
</li>
```

CSS:

```css
p a { /* Score: 2 */
    text-decoration: none;
}

li a { /* Score: 2 */
    text-decoration: underline;
}
```

TIEBREAKER? LAST RULE WINS!

ACTIVITY

EXERCISE

KEY OBJECTIVE

- When multiple style rules apply to the same element, how can we figure out which one will be applied?
- How can we apply this principle to allow us to write less CSS in the long run?

TYPE OF ACTIVITY

- Turn and Talk

TASKS

5 min	1. Talk with your groups. Have one person write out your responses in Slack.

THINGS YOU SHOULD USE IF YOU WANT TO BE

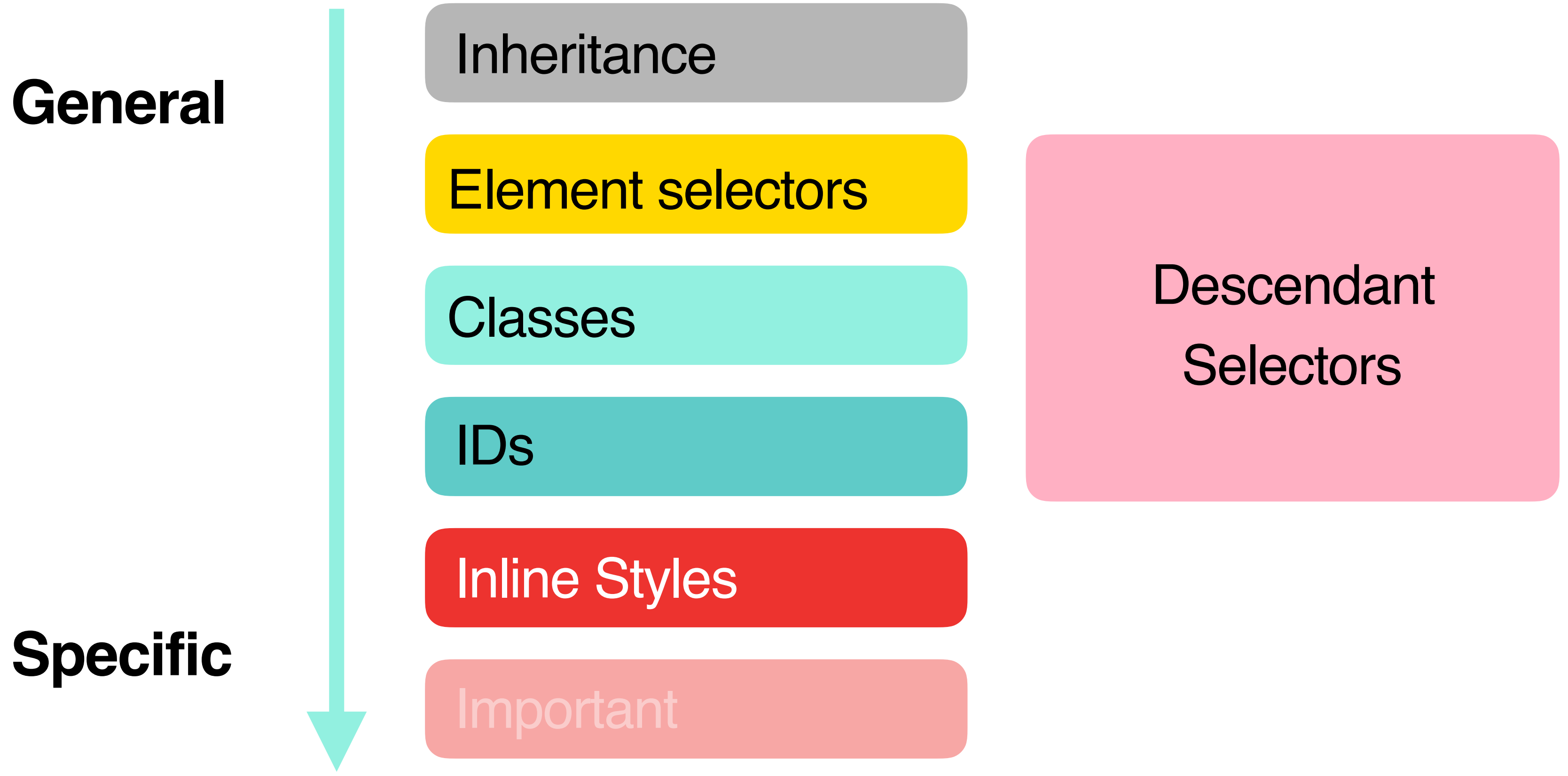
😭

INLINE STYLES

```html
<li style="color: red;">Content</li>
```

BAD!!!!

MORE ABOUT CASCADING



CSS IMPORTANCE

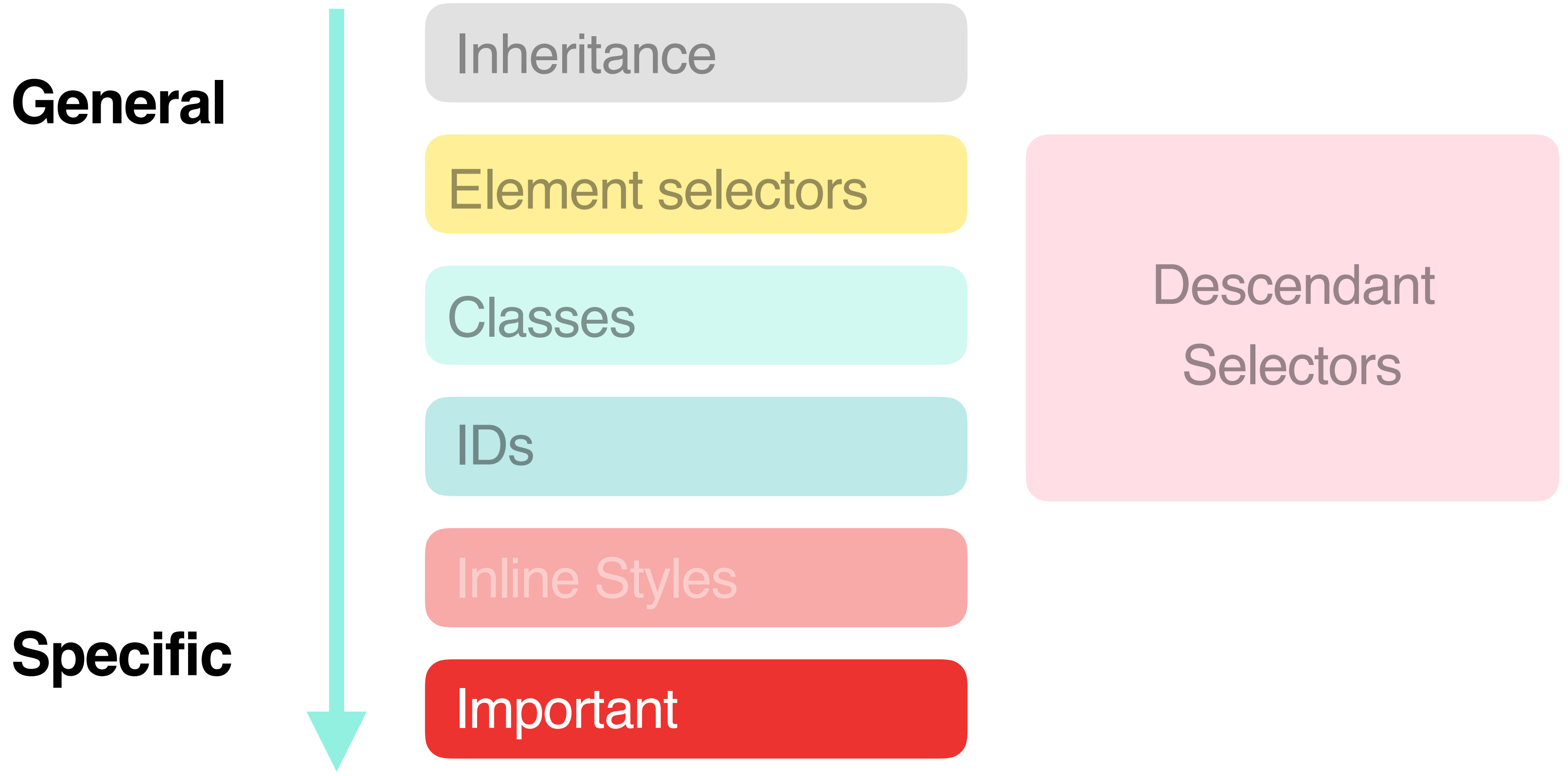
Adding **!important** after any property value indicates that it should be considered *more important than other rules that apply to the same element.*

```
li {
   font-size: 100px !important;
}
```

ONLY WHEN NECESSARY!!!

(Which is almost never)

MORE ABOUT CASCADING



HTML ELEMENTS FOR STYLING: DIVS/SPANS

GROUPING TEXT & ELEMENTS

THE <DIV> ELEMENT

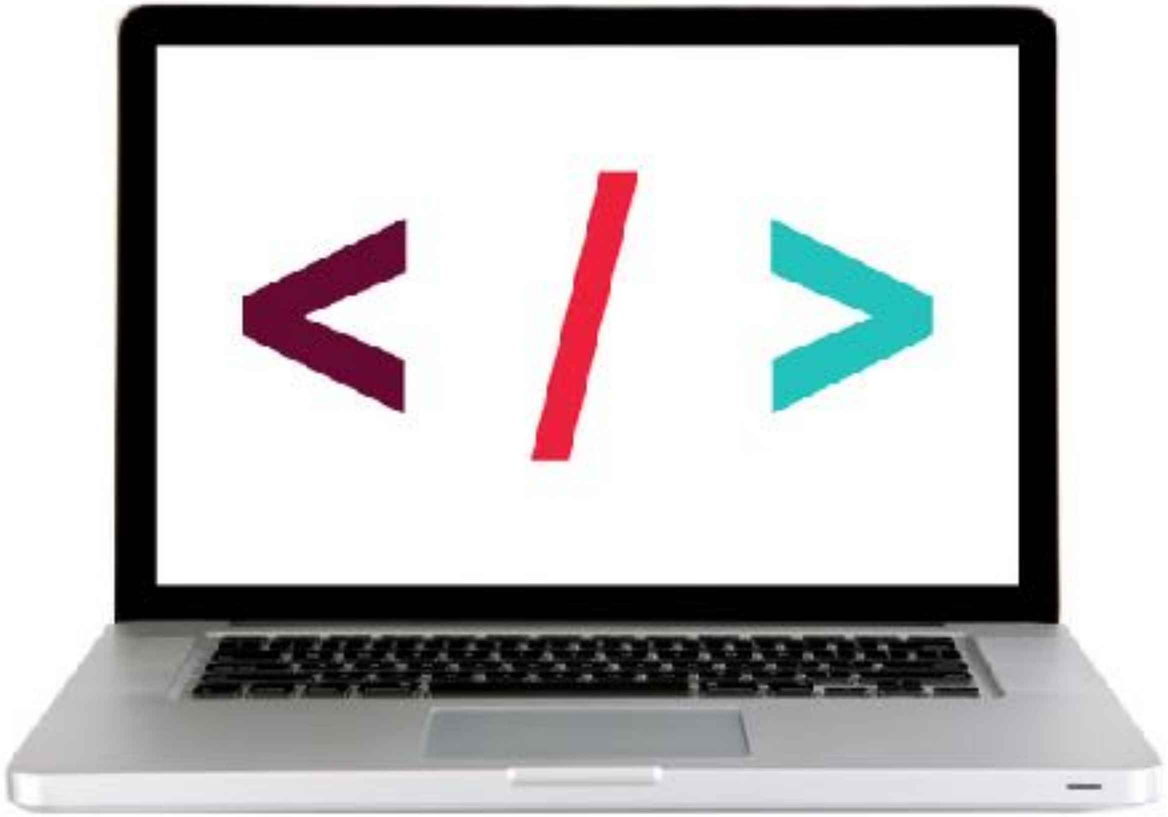
- Allows us to group a set of HTML elements together into a box
- We can then style that chunk of HTML

THE <SPAN> ELEMENT

The <span> element acts like the **inline equivalent** of the <div> element. It is used to either:

1. Style one little piece of text within a larger paragraph
2. Contain several inline elements

DIV & SPAN DEMO



SEMANTIC TAGS — STRONG, EM

ELEMENT	DESCRIPTION	EXAMPLE
strong	Strong Importance	`<p>Do <strong>not</strong> walk</p>`
em	Emphasis	`<p>I <em>think</em> Bill went.</p>`

- Default browser styles: **strong will be bold** and *em will be italic*
- <em> indicates emphasis that may *subtly change* the meaning of a sentence:

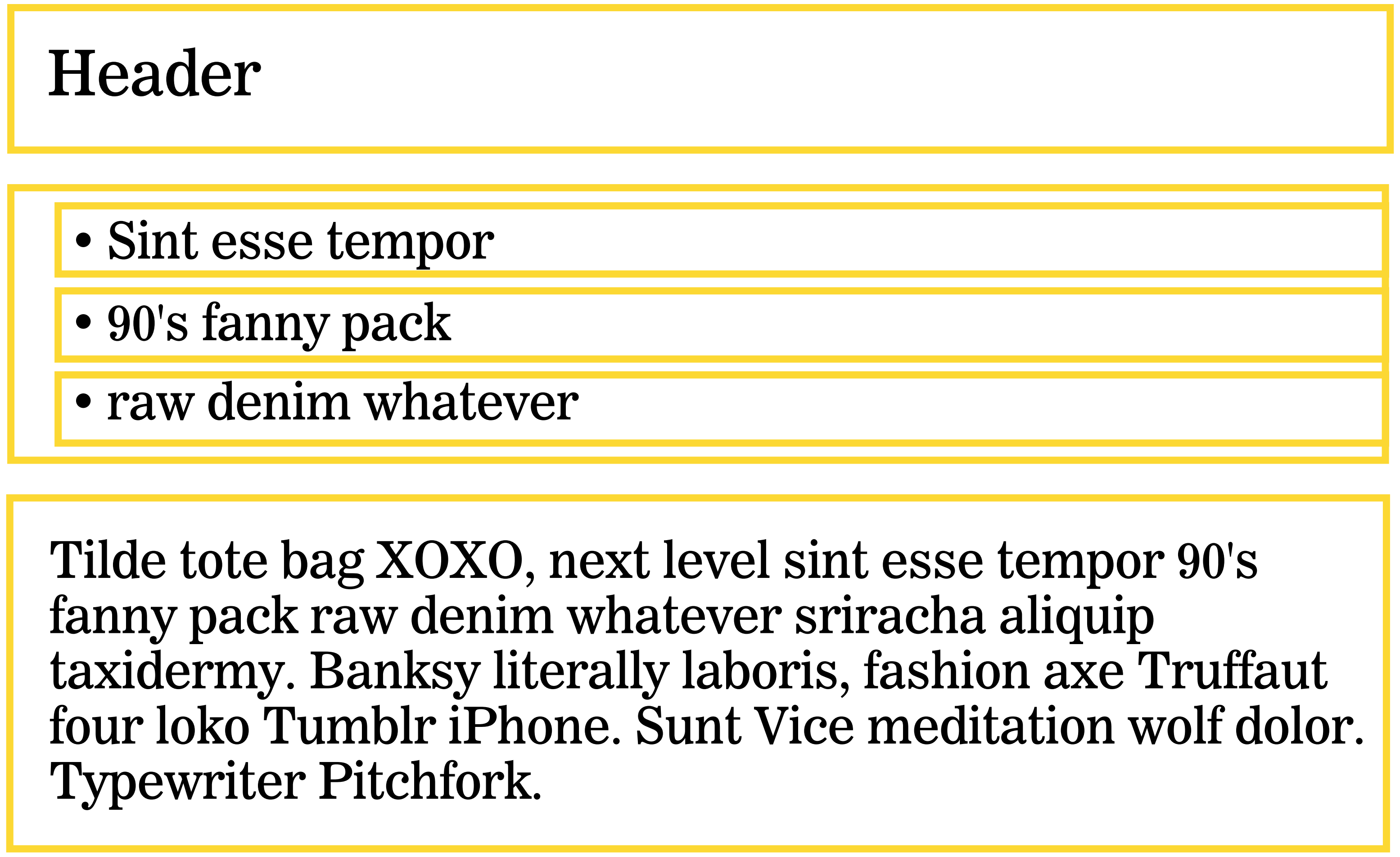
I *think* John was there

I think *John* was there

DISPLAY

BLOCK ELEMENTS

An **block element** will always start on a new line.



Header

- Sint esse tempor
- 90's fanny pack
- raw denim whatever

Tilde tote bag XOXO, next level sint esse tempor 90's fanny pack raw denim whatever sriracha aliquip taxidermy. Banksy literally laboris, fashion axe Truffaut four loko Tumblr iPhone. Sunt Vice meditation wolf dolor. Typewriter Pitchfork.

INLINE ELEMENTS

An **inline element** does not start on a new line and only takes up as much space as needed.

Tilde tote bag XOXO, next level sint esse tempor 90's fanny pack raw denim whatever sriracha aliquip taxidermy. Banksy **literally** laboris, fashion axe Truffaut four loko Tumblr iPhone. Sunt Vice meditation *wolf* dolor. Typewriter www.Pitchfork.com.

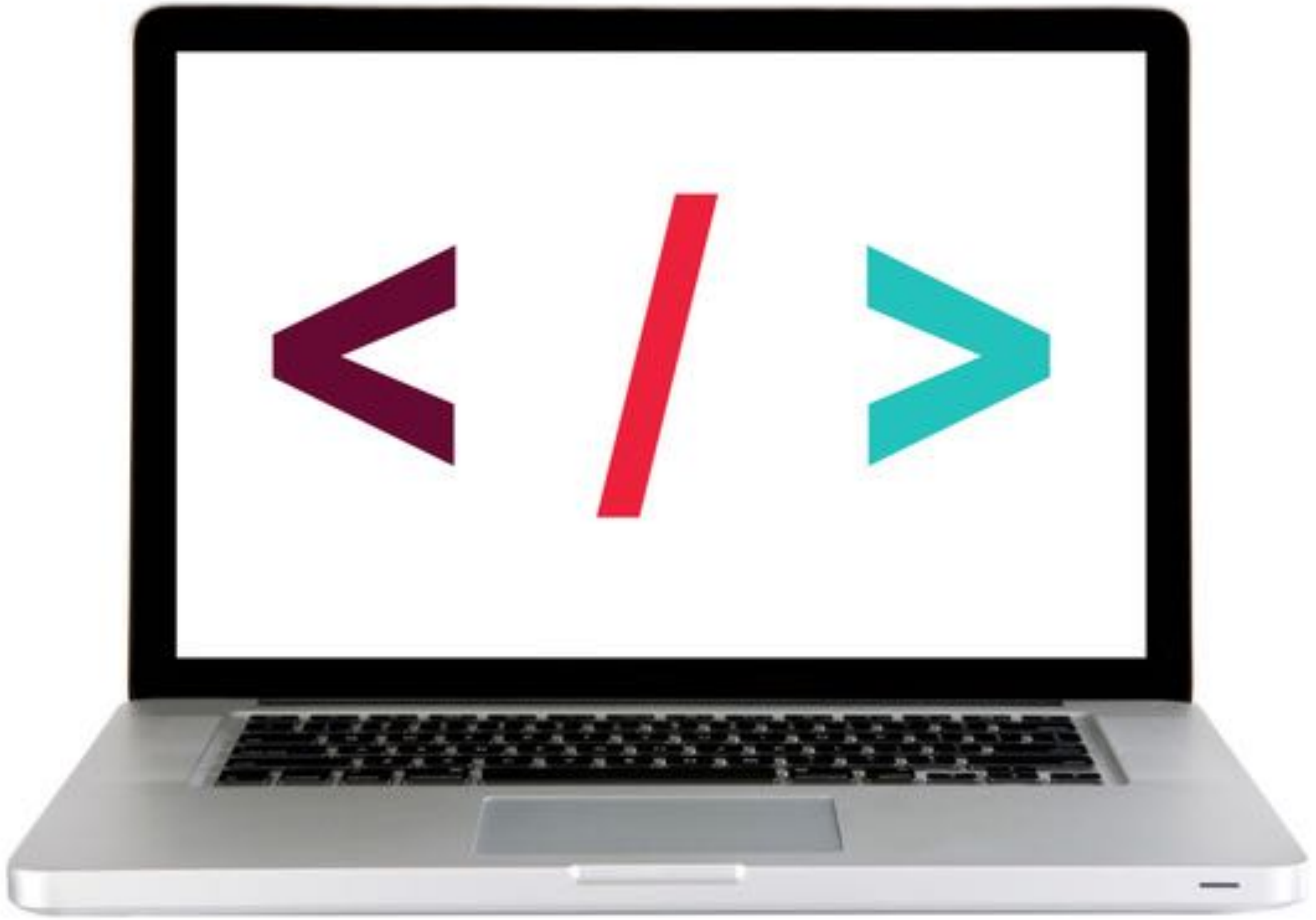
Banksy literally laboris, fashion axe Truffaut four loko Tumblr iPhone. Sunt Vice meditation wolf dolor. Typewriter www.Pitchfork.com.

DISPLAY

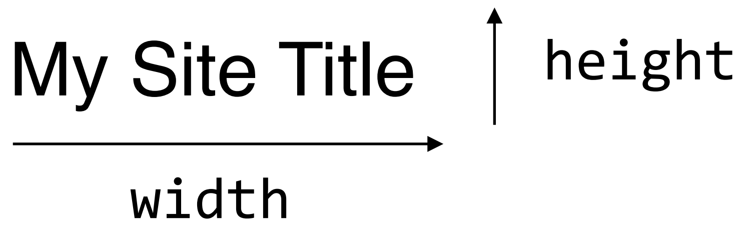
You can change whether elements are displayed as inline or block elements by using the **`display`** property.

```css
li {
  display: inline-block;
}
```

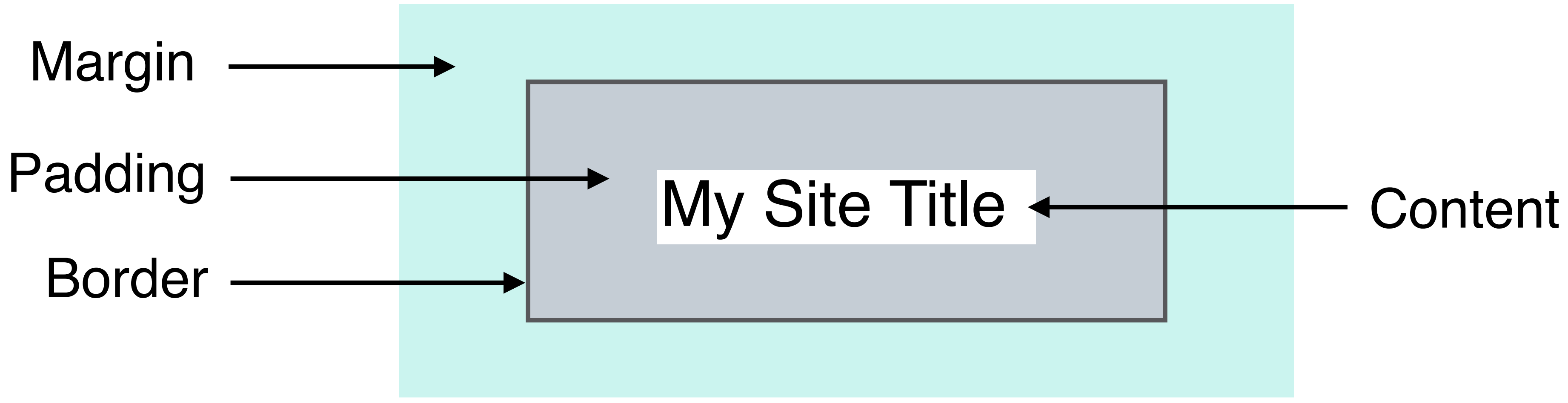
DISPLAY DEMO: PART 1 - INLINE VS BLOCK



CONTENT



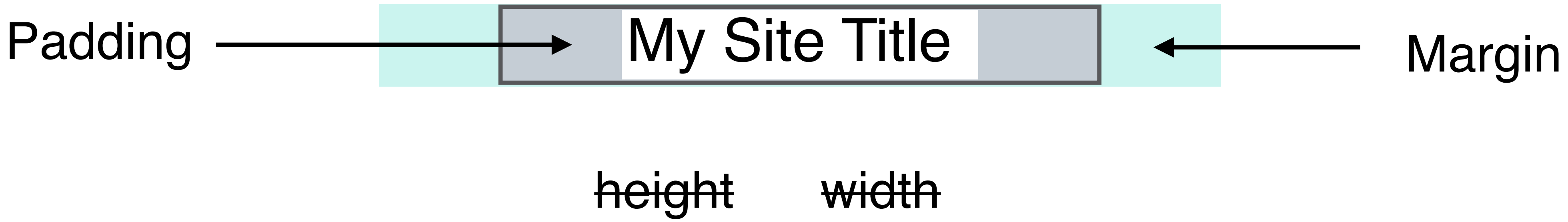
BOX MODEL — BLOCK ELEMENTS



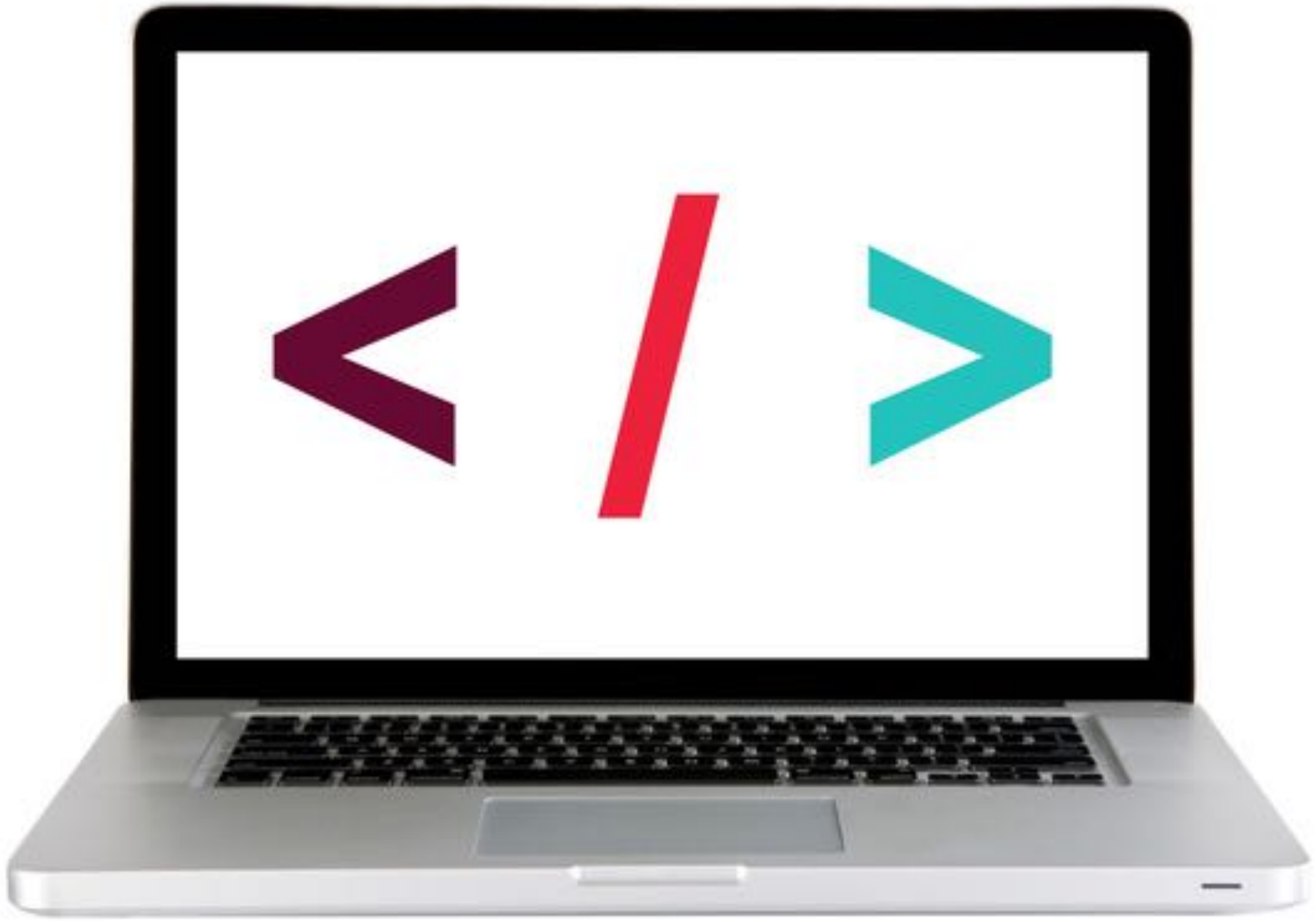
DIMENSION – A KEY DIFFERENCE BETWEEN INLINE AND BLOCK

If you try to add dimension to an inline element:

- `Padding` and `margin` will only apply to the *left and right*
- `Width` and `height` will have *no effect*



DISPLAY DEMO: PART 2 – DIMENSION IN INLINE VS BLOCK



DISPLAY — INLINE (RARE)

To make a block-level element act like an inline element:

BEFORE:

Monday

Tuesday

Wednesday

```css
li {
   display: inline;
}
```

Monday Tuesday Wednesday

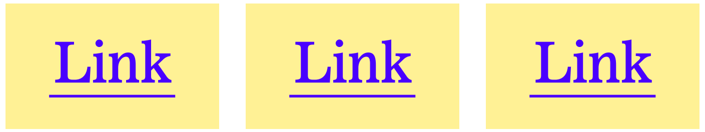
- Elements will sit next to each other
- **Still can't set a width, height, or margin and padding on top and bottom**

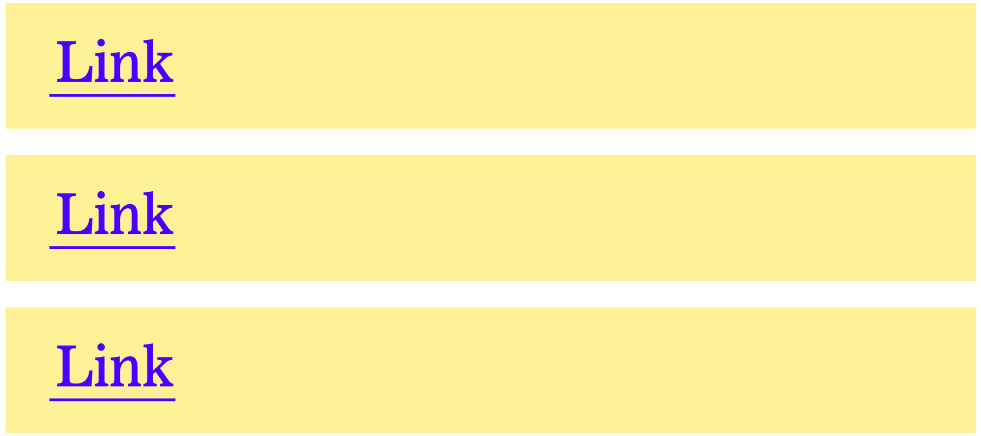
DISPLAY — BLOCK

Make an inline element act like a block-level element:

```
a {
  display: block;
}
```

BEFORE:



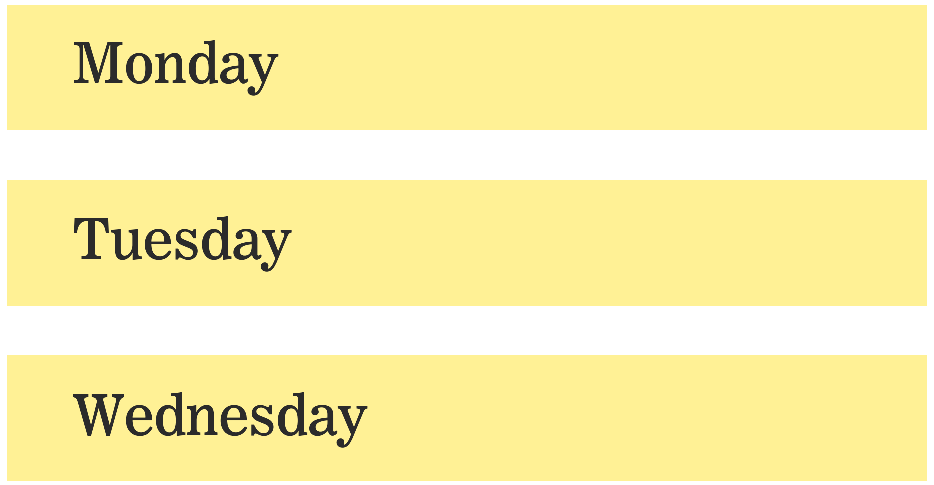


- Elements will stack on top of each other
- We can add all dimensions (width, height, padding, margin)

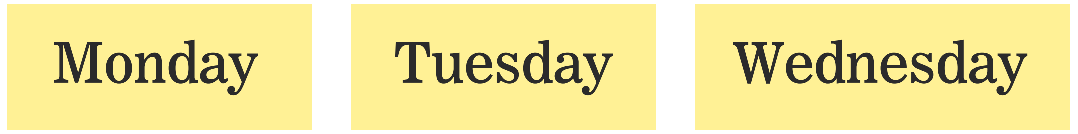
DISPLAY — INLINE-BLOCK

Make a **`block`** *or* **`inline`** element flow like an **`inline`** element, while allowing us to set a **`width`**, **`height`**, **`padding`**, and **`margin`**:

BEFORE:

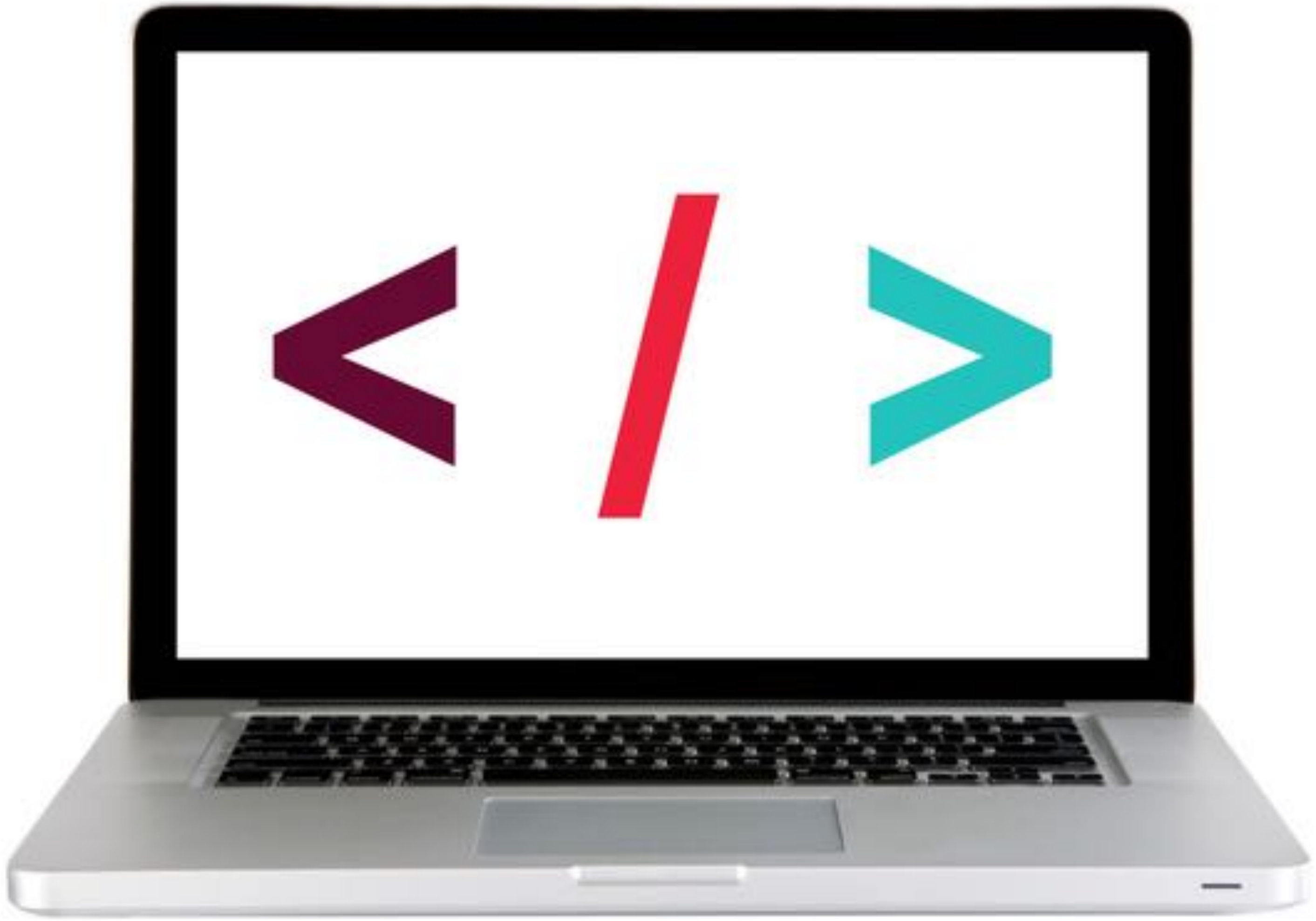


```css
li {
   display: inline-block;
}
```



- Elements will sit on a line next to each other
- **We can now set a width, height, and margin & padding on top and bottom!!**

DISPLAY DEMO: PART 3 - INLINE BLOCK



DISPLAY — NONE

Hide an element from the page:

BEFORE:

Monday

Tuesday

Wednesday

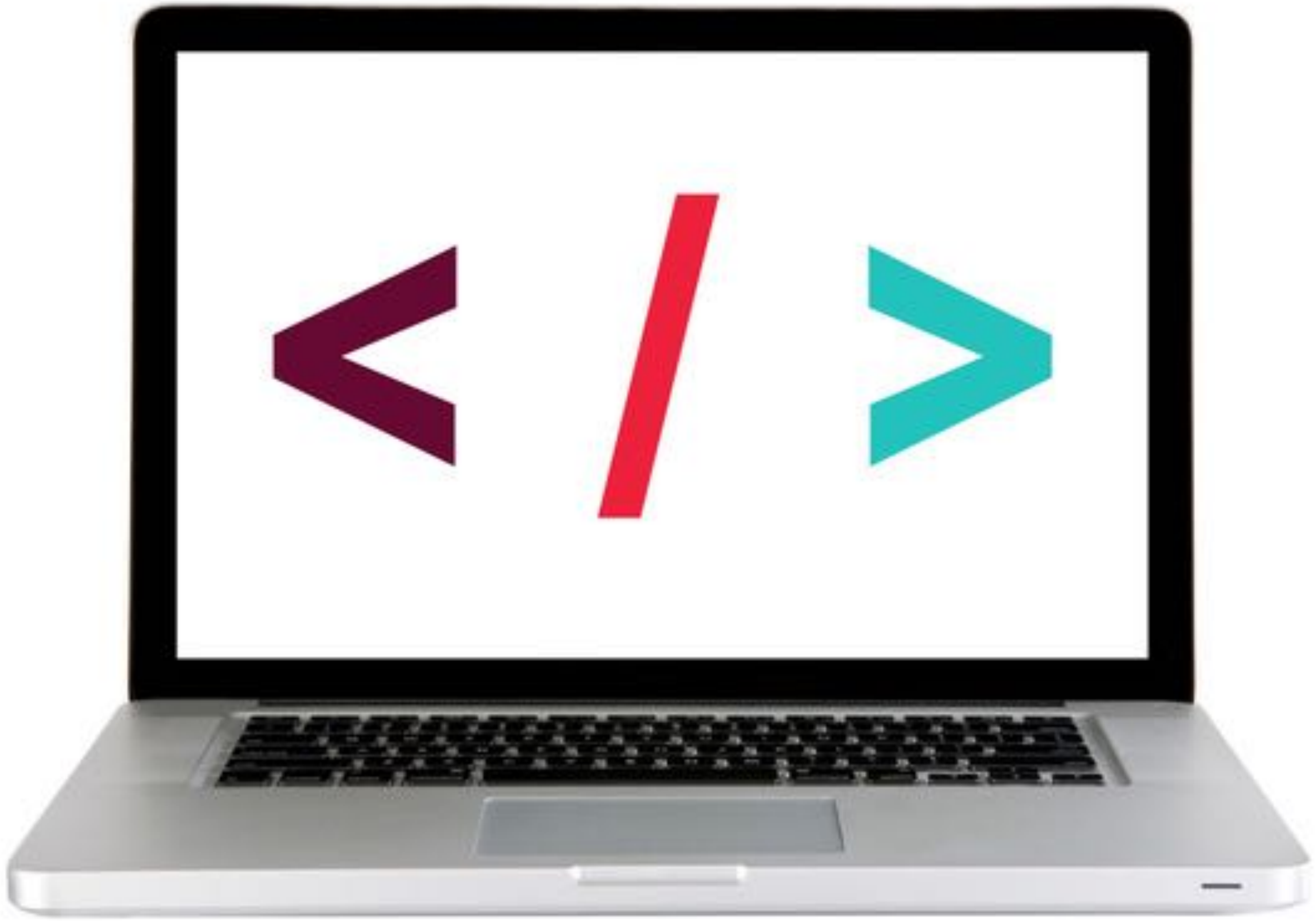
```css
li {
   display: none;
}
```

- Elements will be hidden from the page

TEXT-ALIGN

	TEXT-ALIGN
BLOCK	yes
INLINE / INLINE-BLOCK	no

DISPLAY DEMO: PART 4 - TEXT-ALIGN



ACTIVITY — DISPLAY LAB

EXERCISE

KEY OBJECTIVE

- Get practice using the `display` property

LOCATION OF FILES

- starter code > **display_lab** folder

TIMING

5 min 1. Follow the instructions in steps 1-4 in the CSS file

CENTER ALL THE THINGS!

CENTERING THINGS — TEXT

```
<p>
    <a href="">Home</a>
    <a href="">About</a>
    <a href="">Contact</a>
</p>
```

Set `text-align: center` on parent. This will be inherited by all children.

```
p {
    text-align: center;
}
```

CENTERING THINGS — IMAGES AND CONTAINERS

Centering an image:

```
img {
    display: block;
    margin: 0 auto;
}
```

Centering a block-level element:

```
p {
    width: 400px;
    margin: 0 auto;
}
```

FLEXBOX

WHAT IS FLEXBOX?

- Another display property!
- Used to build layouts on your website

FLEXBOX — HTML STRUCTURE

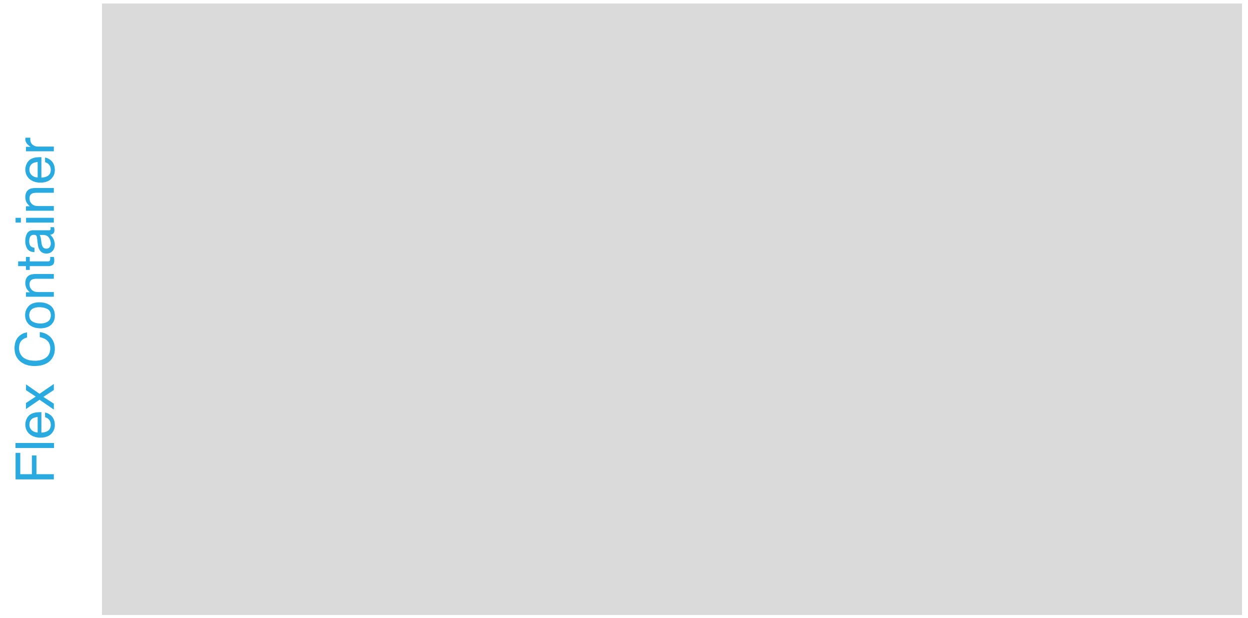
WHAT IS FLEXBOX?

```
<section>
        <article>
            <h1>Article title</h1>
            <p>Lorem ipsum dolor sit amet.</p>
        </article>
        <article>
            <h1>Article title</h1>
            <p>Lorem ipsum dolor sit amet.</p>
        </article>
        <article>
            <h1>Article title</h1>
            <p>Lorem ipsum dolor sit amet.</p>
        </article>
</section>
```

WHAT IS FLEXBOX?

parent wrapper:
flex container

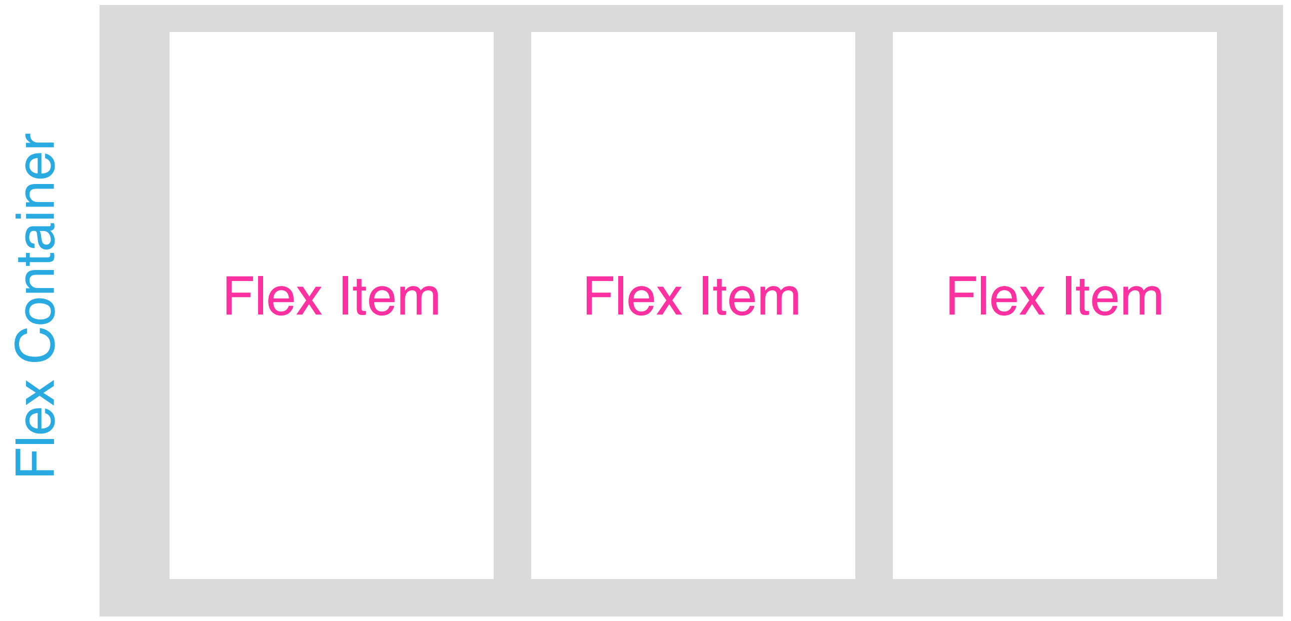
```
<section>
        <article>
            <h1>Article title</h1>
            <p>Lorem ipsum dolor sit amet.</p>
        </article>

        <article>
            <h1>Article title</h1>
            <p>Lorem ipsum dolor sit amet.</p>
        </article>

        <article>
            <h1>Article title</h1>
            <p>Lorem ipsum dolor sit amet.</p>
        </article>
</section>
```



WHAT IS FLEXBOX?

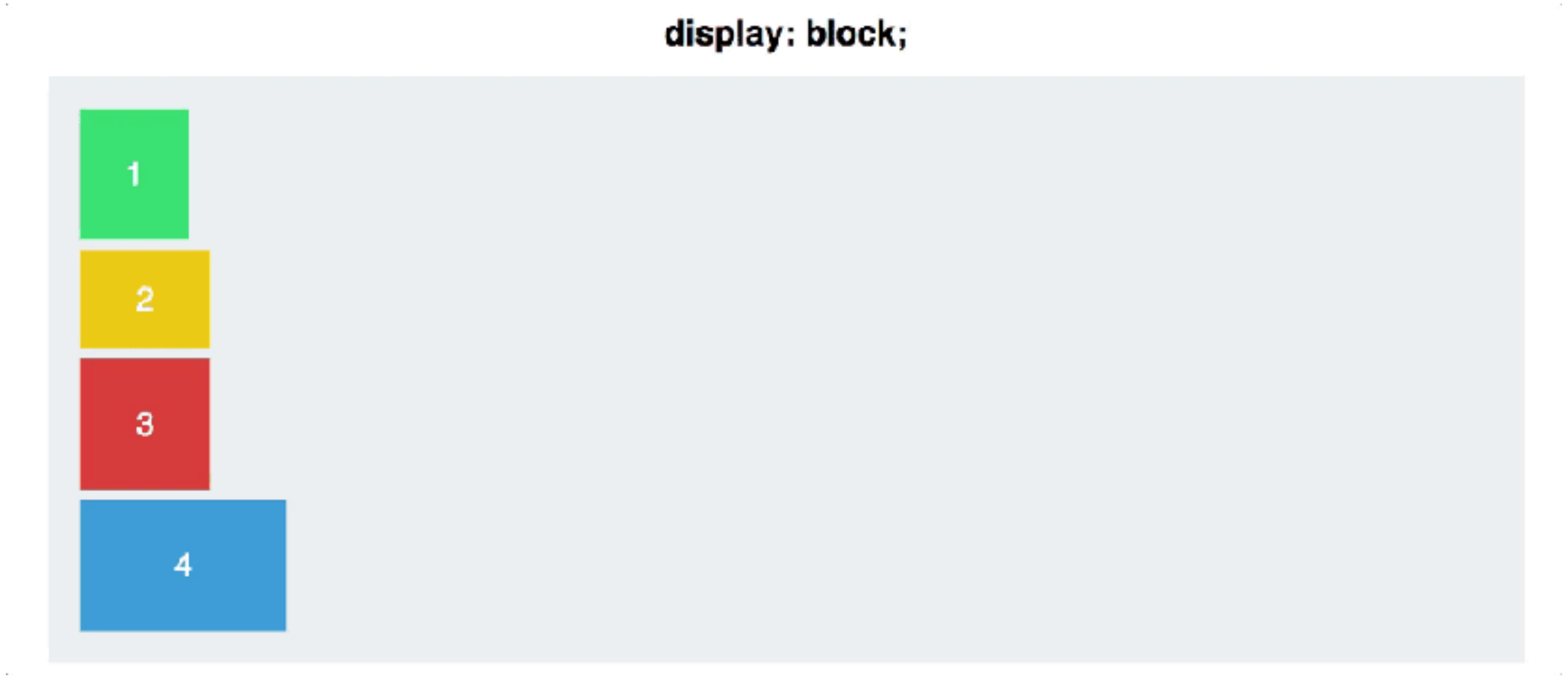
direct children:
flex items

```
<section>
    <article>
        <h1>Article title</h1>
        <p>Lorem ipsum dolor sit amet.</p>
    </article>

    <article>
        <h1>Article title</h1>
        <p>Lorem ipsum dolor sit amet.</p>
    </article>

    <article>
        <h1>Article title</h1>
        <p>Lorem ipsum dolor sit amet.</p>
    </article>
</section>
```



MULTI-COLUMN LAYOUTS

HOW DOES FLEXBOX WORK – ANIMATION



PARENT WRAPPER (FLEX CONTAINER)

In order to get started with Flexbox, you need to make your container into a flex container. This is as easy as:

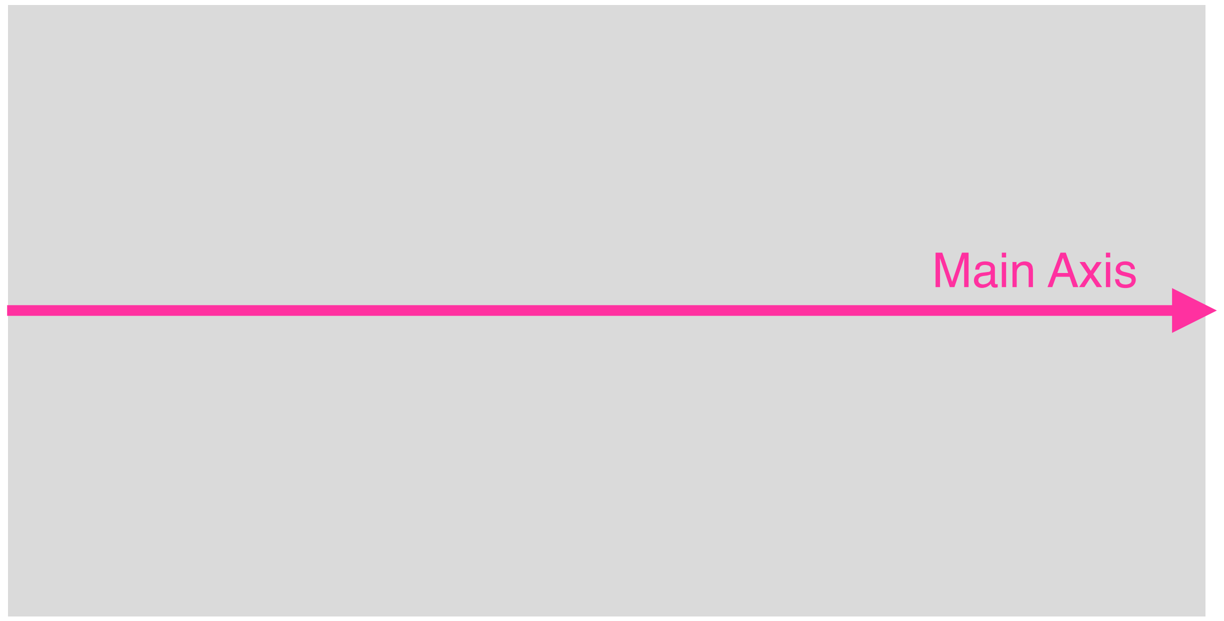
parent wrapper:
flex container

```
<section>
        <article>
            …
        </article>
        <article>
            …
        </article>
        <article>
            …
        </article>
</section>
```

```
section {
  display: flex;
}
```

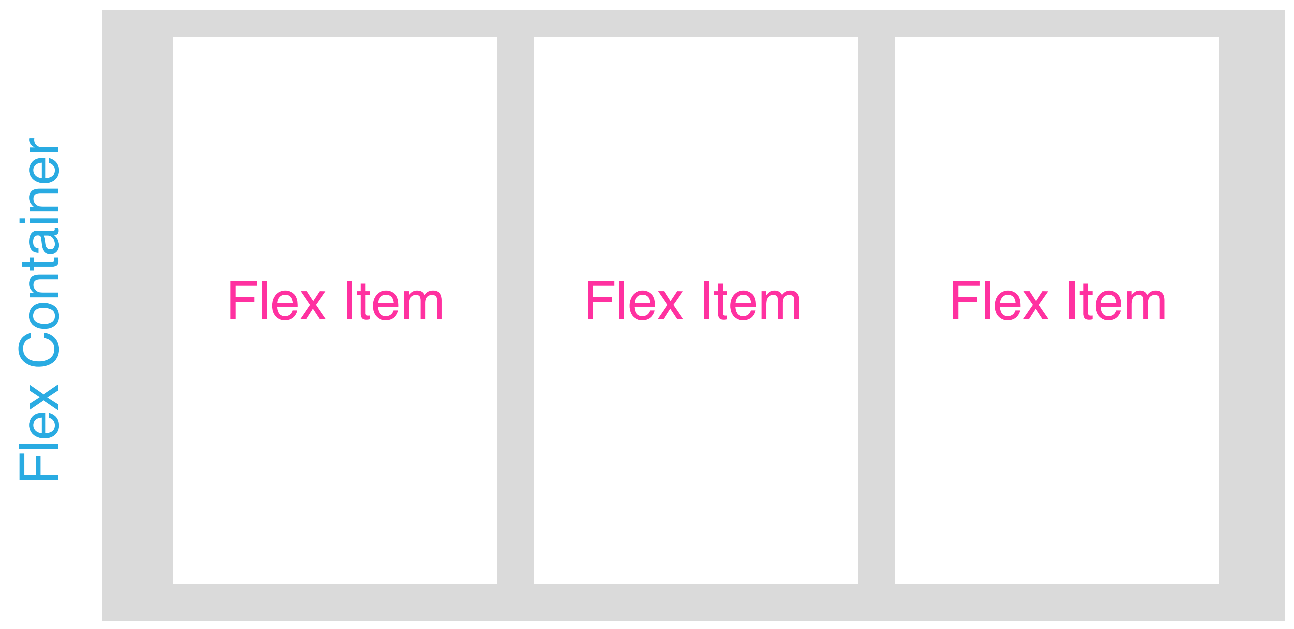
HOW DOES FLEXBOX WORK?

By default, items are arranged along the main axis, from left to right. This is why your squares defaulted to a horizontal line once you applied **`display: flex`**.



WHAT IS FLEXBOX?

★ `display: flex;` ★



STEPS TO ACHIEVE A MULTI-COLUMN LAYOUT — CSS

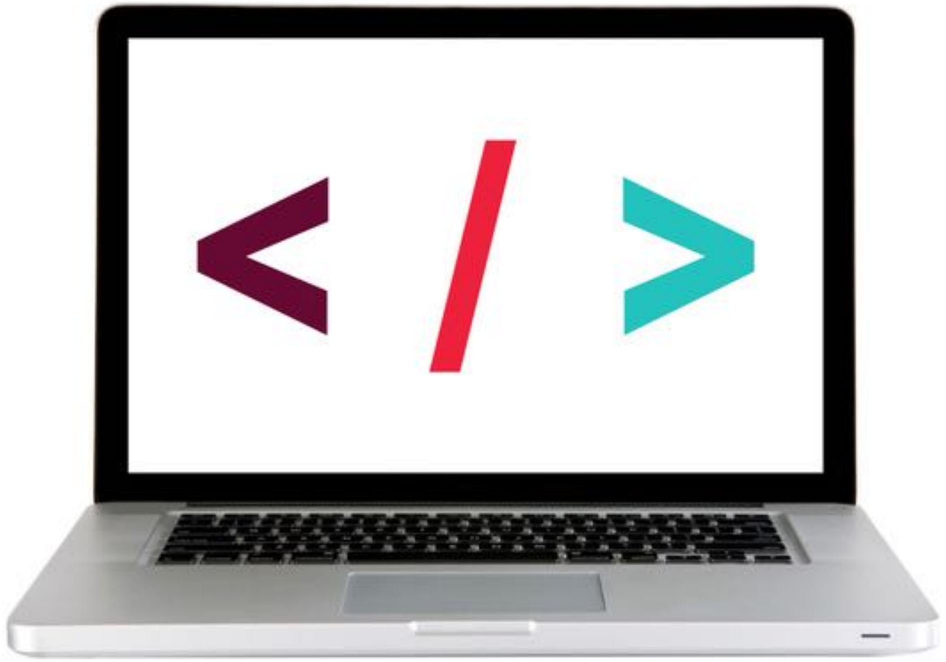
FLEX CONTAINER:

- Add `display: flex`

FLEX ITEMS:

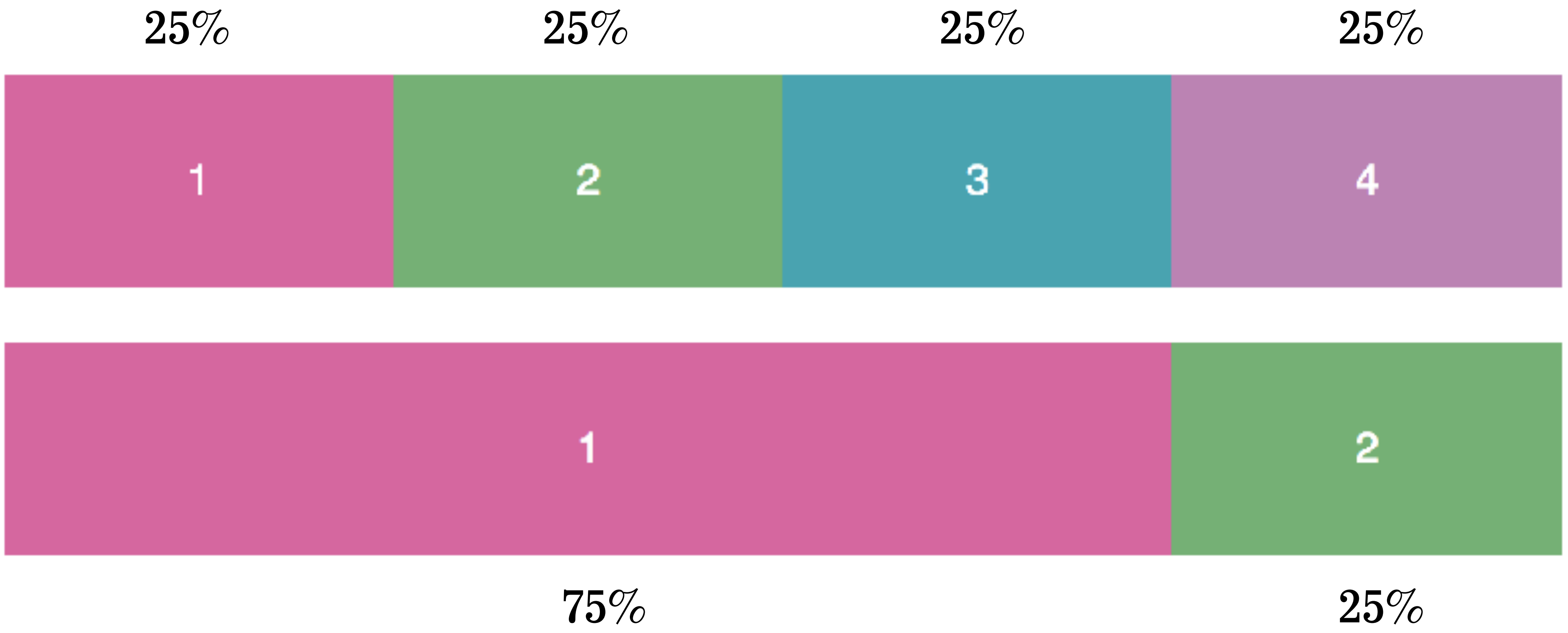
- Set a `width` in percentages

CODEALONG – FLEXBOX LAYOUT SYSTEM



https://codepen.io/svodnik/pen/WJWbZP?editors=1100#0

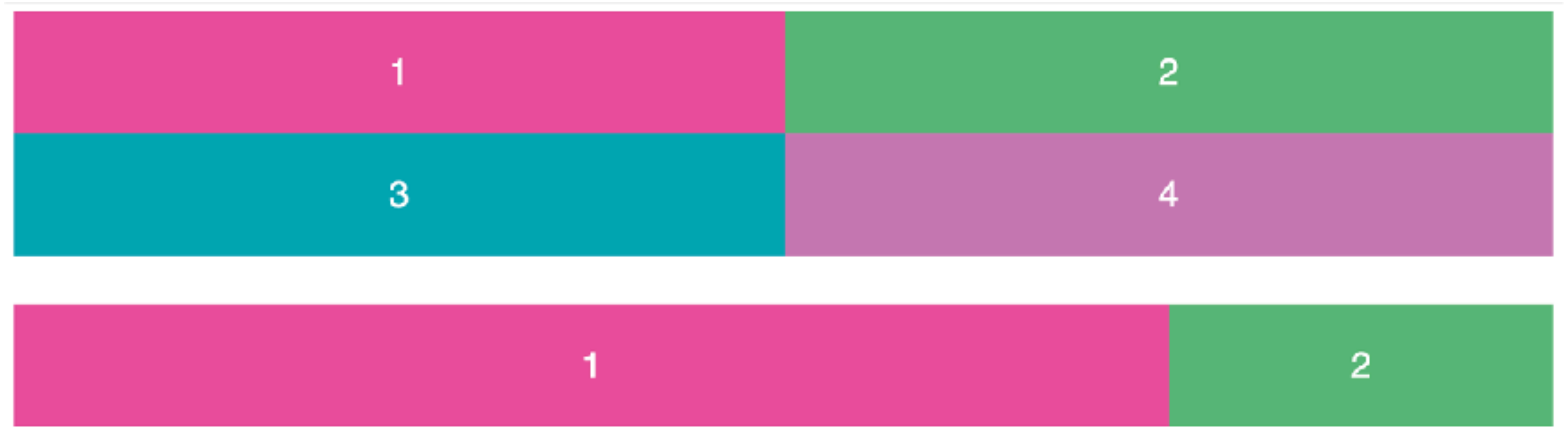
CODEALONG: PART 1



CODEALONG: PART 2

50%

50%

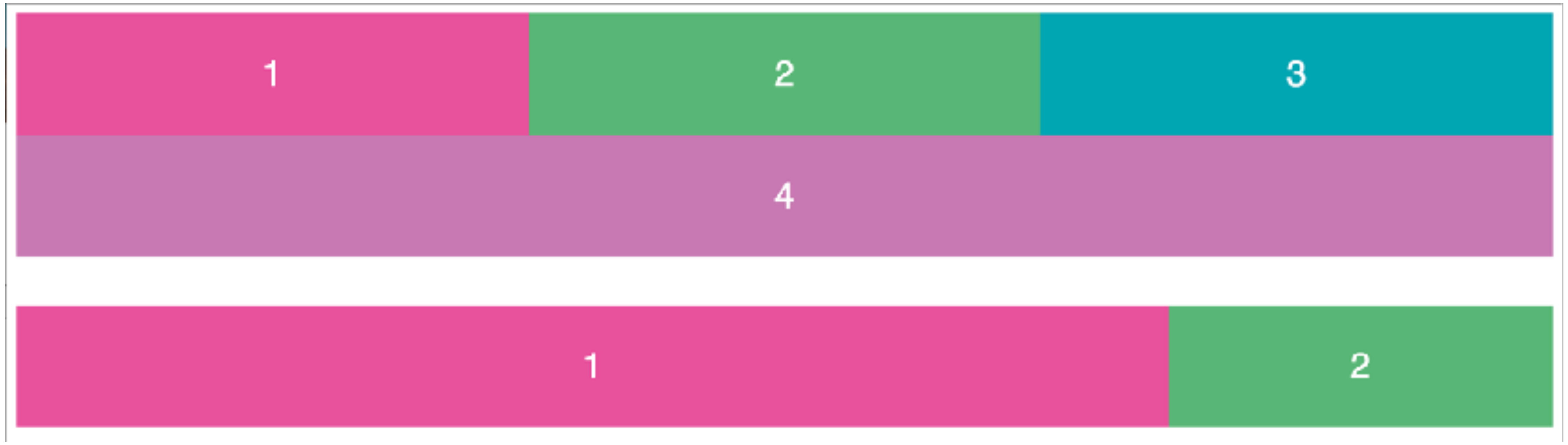


75%

25%

CODEALONG: PART 3

33.333333% 33.333333% 33.333333%



75% 25%

ACTIVITY: MULTI COLUMN LAB

EXERCISE

KEY OBJECTIVE

- Work through steps to create a multi-column layout

LOCATION OF FILES

- starter code >multi_column_lab folder

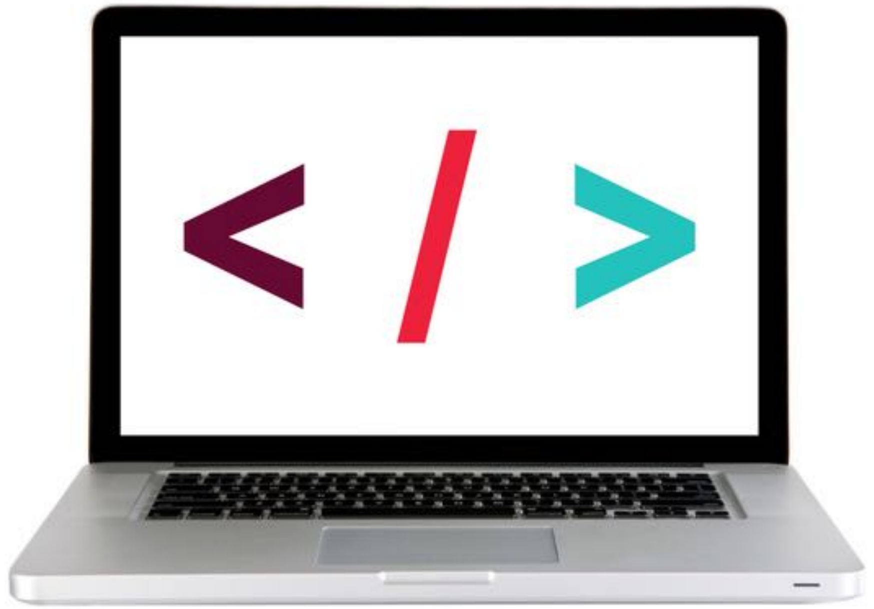
TIMING

5 min

1. Follow steps in HTML and CSS
2. Make it look like mocks.png

OTHER FLEXBOX PROPERTIES

CODEALONG — OTHER FLEXBOX PROPERTIES



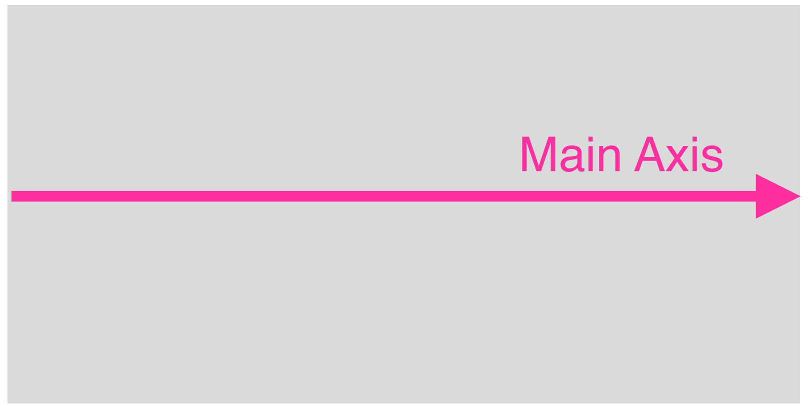
https://codepen.io/svodnik/pen/yjrypB?editors=1100

JUSTIFY-CONTENT

`justify-content` controls how you align items on the **main axis.**

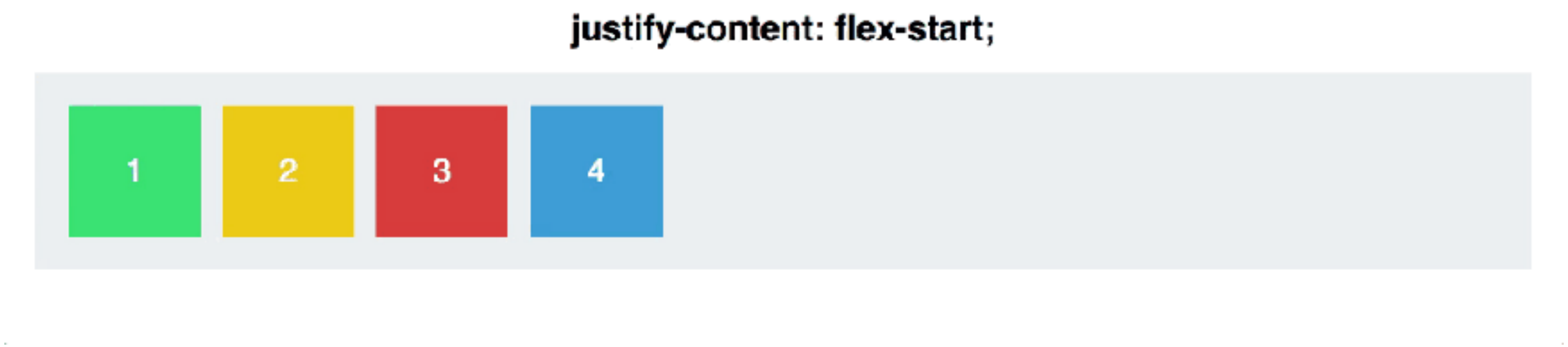
1. flex-start (default)
2. flex-end
3. center
4. space-between
5. space-around

```
section {
   display: flex;
   justify-content: flex-start;
}
```



JUSTIFY-CONTENT

`justify-content` controls how you align items on the **main axis.**



ACTIVITY: FLEXBOX PROPERTIES: JUSTIFY CONTENT

EXERCISE

KEY OBJECTIVE

- Learn the justify-content property and values

LOCATION

- CodePen/Whiteboard

TIMING

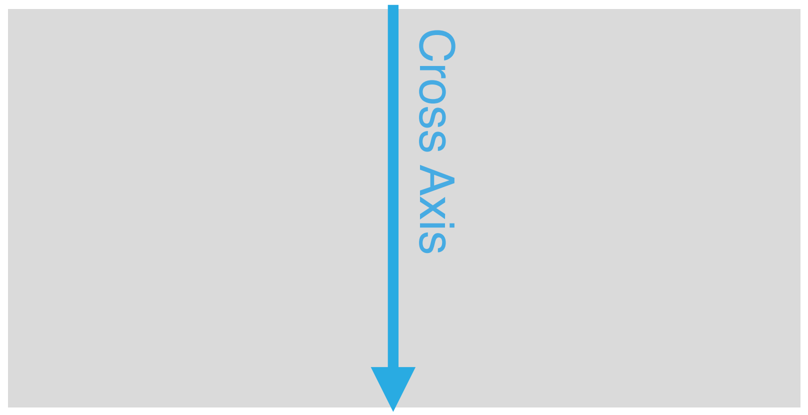
1. Play with the different values of justify-content: flex-start, flex-end, center, space-around, space-between
2. Draw a diagram for what happens to the elements when you change the value of justify-content

ALIGN-ITEMS

`align-items` controls how you align items on the **cross axis**.

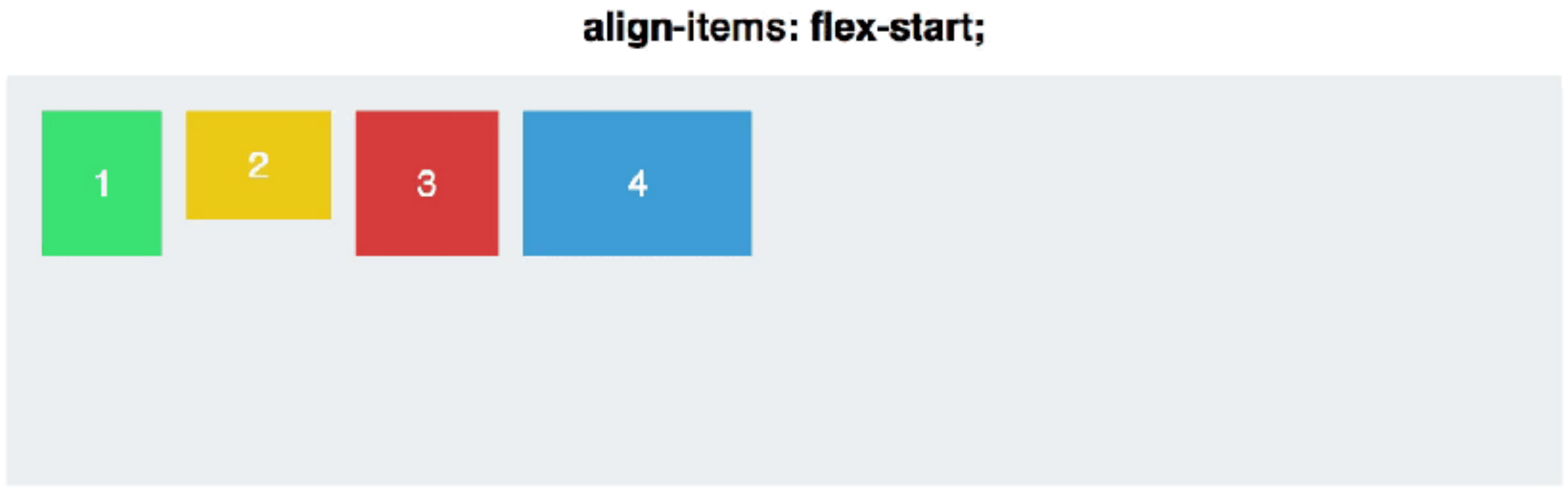
1. `flex-start`
2. `flex-end`
3. `center`
4. `stretch`

```
section {
   display: flex;
   align-items: center;
}
```



ALIGN-ITEMS

`align-items` controls how you align items on the **cross axis**.

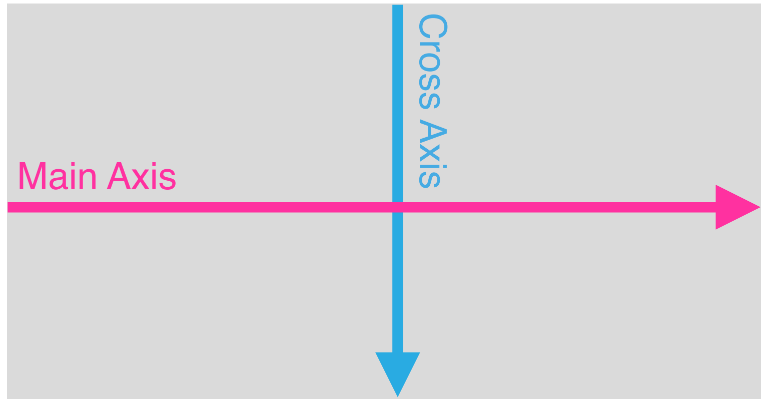


FLEX DIRECTION

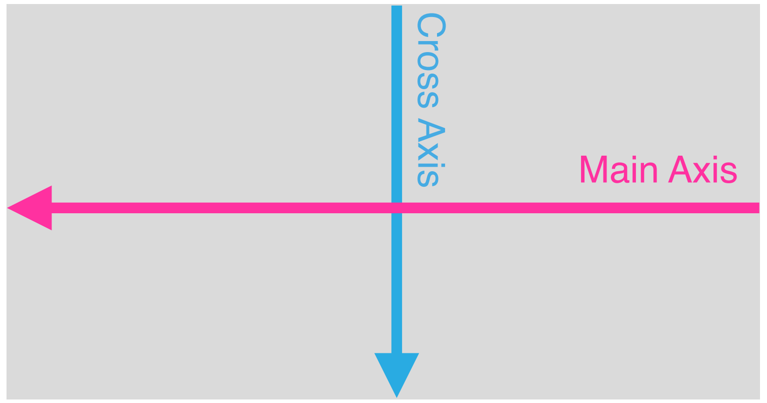
`flex-direction` lets you rotate the **main axis**.

```
section {
   display: flex;
   flex-direction: row;
}
```

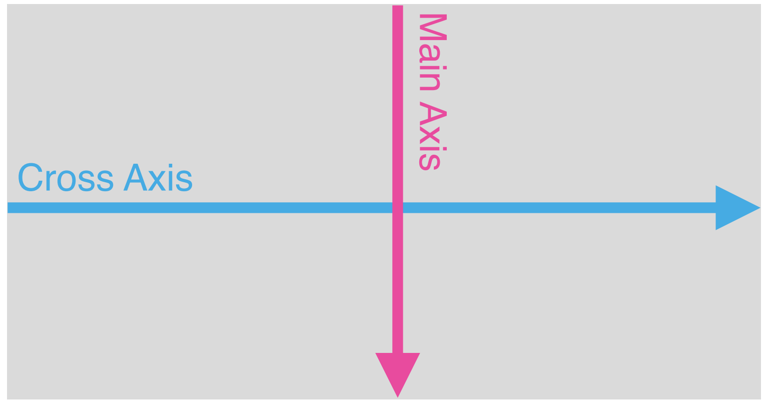
`flex-direction: row;`



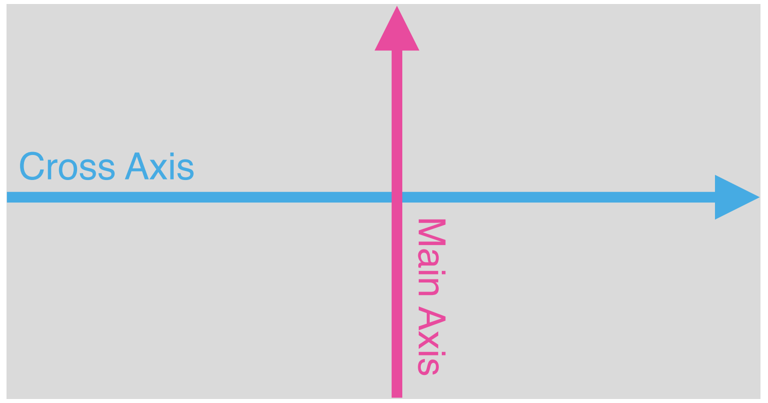
`flex-direction: row-reverse;`



`flex-direction: column;`

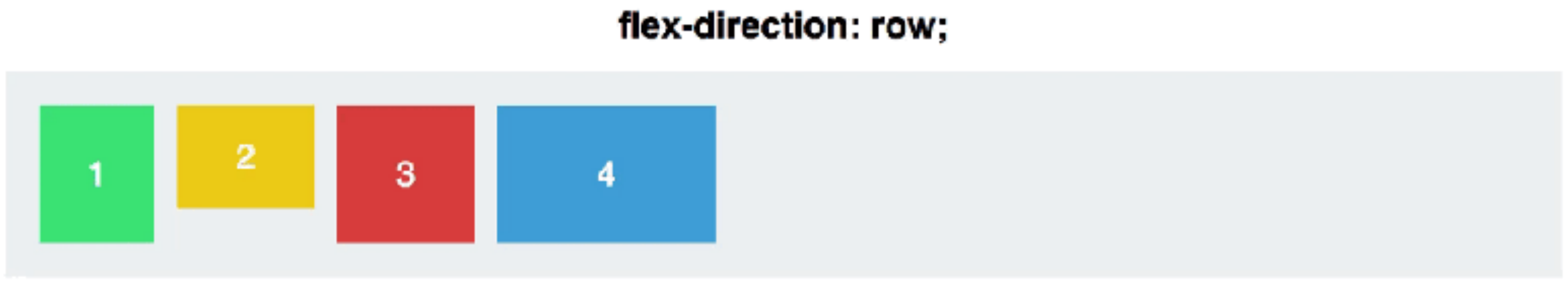


`flex-direction: column-reverse;`



FLEX-DIRECTION

`flex-direction` lets you rotate the **main axis**.



ACTIVITY: FLEXBOX PROPERTIES: FLEX-DIRECTION

EXERCISE

KEY OBJECTIVE

- Learn the flex-direction property and values

LOCATION

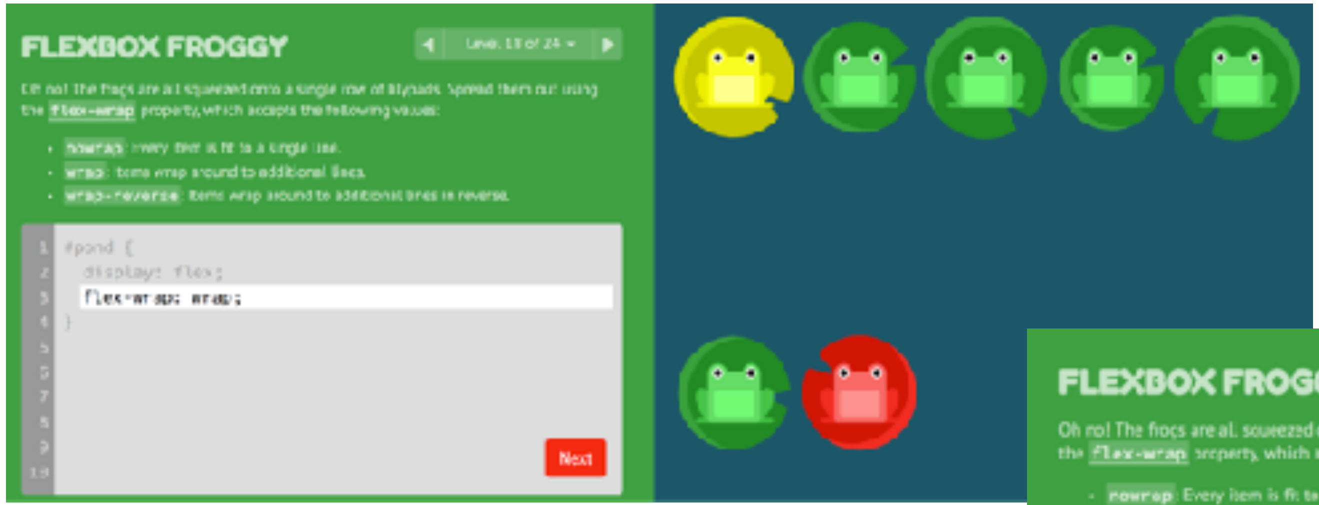
- CodePen/Whiteboard

TIMING

1. Play with the different values of flex-direction: row, row-reverse, column, column-reverse
2. Draw a diagram for what happens to the elements when you change the value of flex-direction

FLEX-WRAP

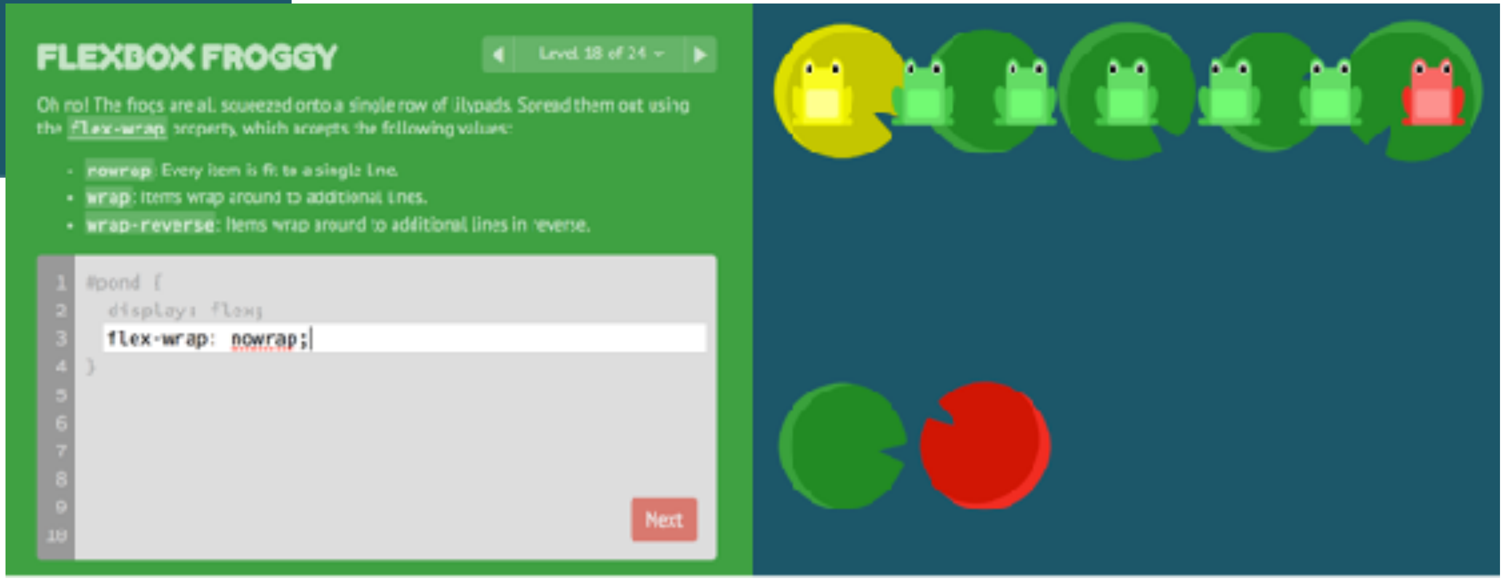
Flex items are on the same row by default. flex-wrap allows them to wrap around.



Oh no! The frogs are all squeezed onto a single row of lilypads. Spread them out using the `flex-wrap` property, which accepts the following values:

- `nowrap`: Every item is fit to a single line.
- `wrap`: Items wrap around to additional lines.
- `wrap-reverse`: Items wrap around to additional lines in reverse.

```
#pond {
  display: flex;
  flex-wrap: wrap;
}
```



Oh no! The frogs are all squeezed onto a single row of lilypads. Spread them out using the `flex-wrap` property, which accepts the following values:

- `nowrap`: Every item is fit to a single line.
- `wrap`: Items wrap around to additional lines.
- `wrap-reverse`: Items wrap around to additional lines in reverse.

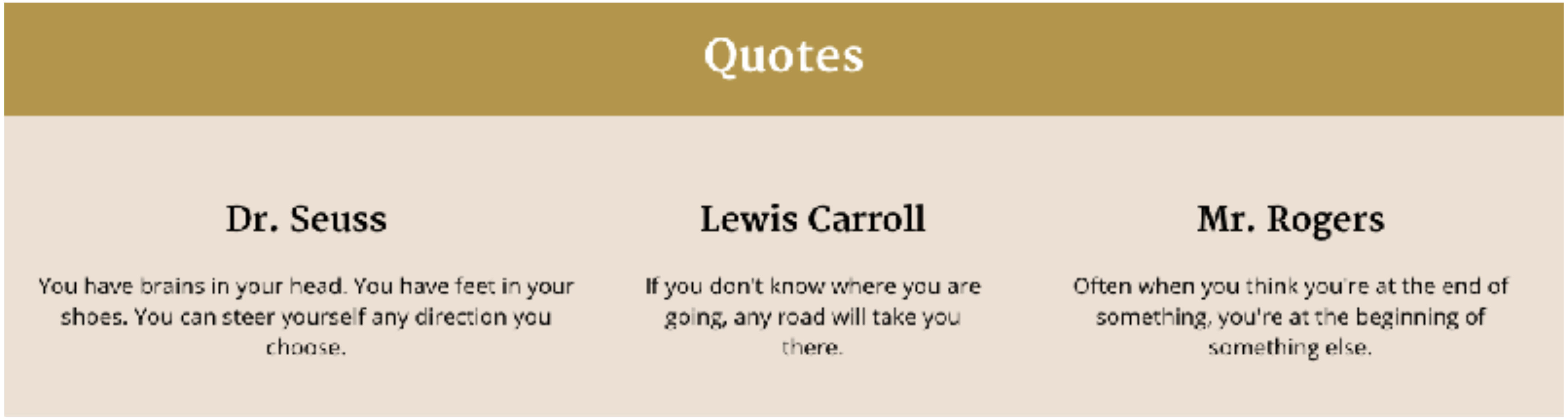
```
#pond {
  display: flex;
  flex-wrap: nowrap;
}
```

http://flexboxfroggy.com/

FLEXBOX PRACTICE

ACTIVITY: FLEXBOX PRACTICE

Recreate this layout in this CodePen



You have brains in your head. You have feet in your shoes. You can steer yourself any direction you choose.

If you don't know where you are going, any road will take you there.

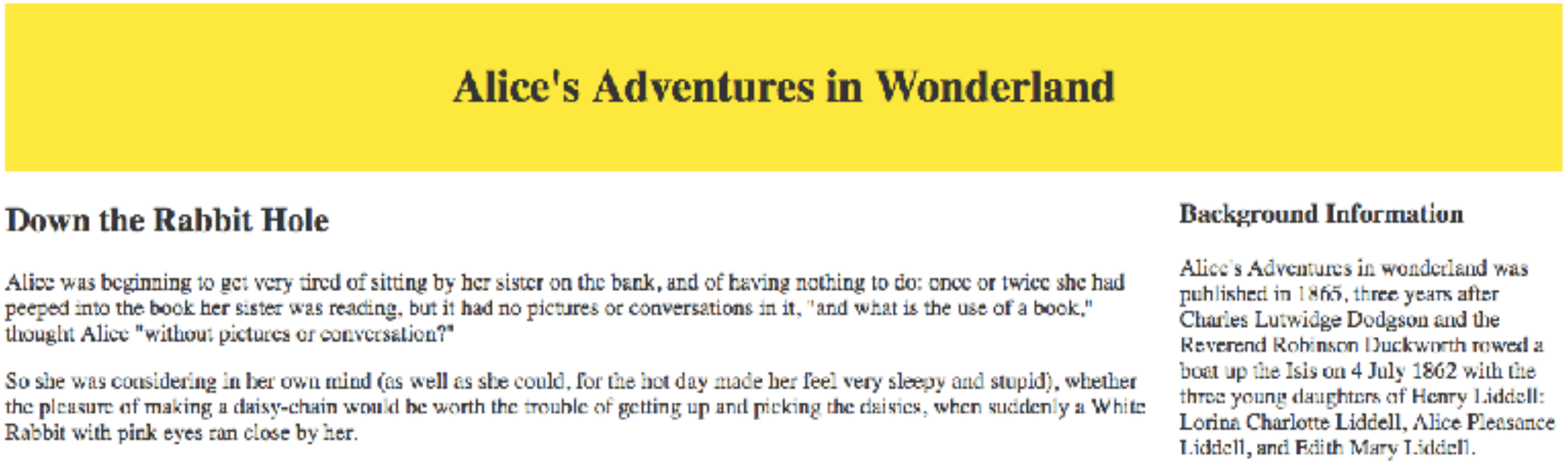
Often when you think you're at the end of something, you're at the beginning of something else.

ACTIVITY: FLEXBOX PRACTICE

Recreate this layout in CodePen!

Mocks

Lorem Ipsum Generators:
* Lorem Ipsum
* Cat Ipsum
* Office Ipsum
* Hipster Ipsum



Alice's Adventures in Wonderland

Down the Rabbit Hole

Alice was beginning to get very tired of sitting by her sister on the bank, and of having nothing to do: once or twice she had peeped into the book her sister was reading, but it had no pictures or conversations in it, "and what is the use of a book," thought Alice "without pictures or conversation?"

So she was considering in her own mind (as well as she could, for the hot day made her feel very sleepy and stupid), whether the pleasure of making a daisy-chain would be worth the trouble of getting up and picking the daisies, when suddenly a White Rabbit with pink eyes ran close by her.

Background Information

Alice's Adventures in wonderland was published in 1865, three years after Charles Lutwidge Dodgson and the Reverend Robinson Duckworth rowed a boat up the Isis on 4 July 1862 with the three young daughters of Henry Liddell: Lorina Charlotte Liddell, Alice Pleasance Liddell, and Edith Mary Liddell.

LAB: MONUMENT BLOG

ACTIVITY: MONUMENT BLOG PART 2

EXERCISE

KEY OBJECTIVE

- Create a multi-column layout with flexbox

TASKS

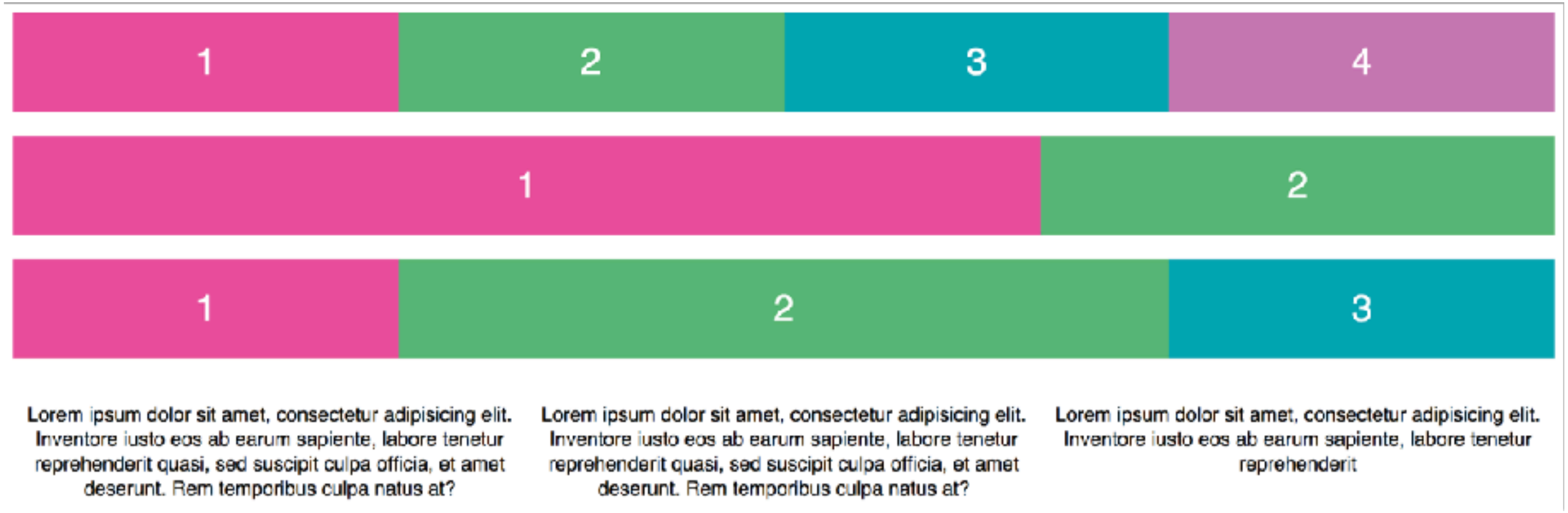
1. Update the Monument blog layout to a 2 column layout, as shown in mocks_part2.png
2. **Bonus 1:** Add the background-image in the header
3. **Bonus 2:** Add the FontAwesome icons in the footer
4. **Bonus 3:** Work from bonus_hover.gif to add a hover effect to links

REVIEW QUESTIONS

- What should our HTML structure look like to set up a Flexbox?
- How do we initiate Flexbox? Name the property. Which element would we set this property on (the flex container or flex items)?
- Which property can we use to determine the space between flex items? Which element would we set this property on (the flex container or flex items)?
- How can we determine how much width each column should take up?

GRID SYSTEMS PRACTICE

CODEALONG



Lorem ipsum dolor sit amet, consectetur adipisicing elit. Inventore iusto eos ab earum sapiente, labore tenetur reprehenderit quasi, sed suscipit culpa officia, et amet deserunt. Rem temporibus culpa natus at?

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Inventore iusto eos ab earum sapiente, labore tenetur reprehenderit quasi, sed suscipit culpa officia, et amet deserunt. Rem temporibus culpa natus at?

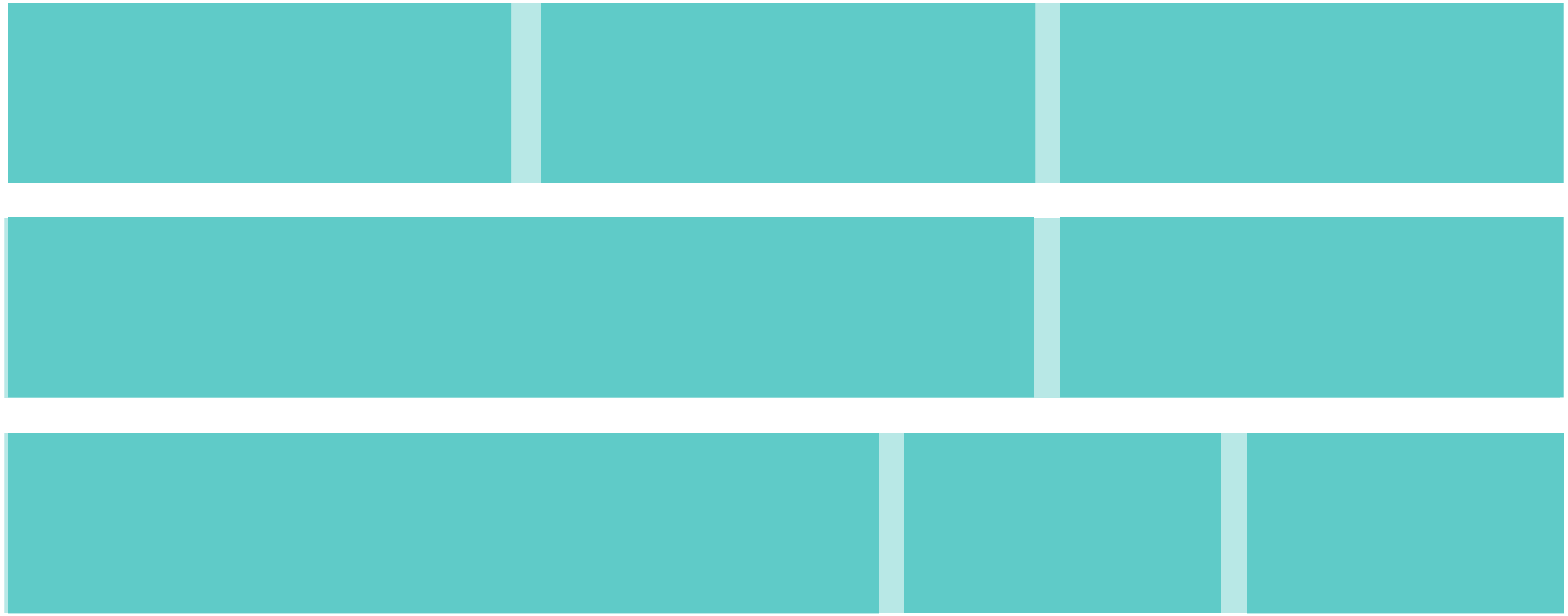
Lorem ipsum dolor sit amet, consectetur adipisicing elit. Inventore iusto eos ab earum sapiente, labore tenetur reprehenderit

ROWS

Each row will be a flex container!

COLUMNS

Each column will be a flex item (inside each row)!



CLASSES

We can create classes to reuse styles across multiple elements!

ROWS

Since all our rows will be set to **`display: flex;`** we can store this property in a class and apply it to any element to turn it into a row.

```css
.row {
  display: flex;
  margin-bottom: 20px;
}
```

COLUMNS

We can create classes for the different options for column widths. We can then apply these classes to any flex item to set its width.

4 columns total

```css
.col-1-4 {
  width: 25%;
}
```

1 column wide

100% divided by 4 = 25%

```css
.col-1-3 {
  width: 33.333333%;
}

.col-3-4 {
  width: 75%;
}
```

COLUMNS

We can create classes for the different options for column widths. We can then apply these classes to any flex item to set its width.

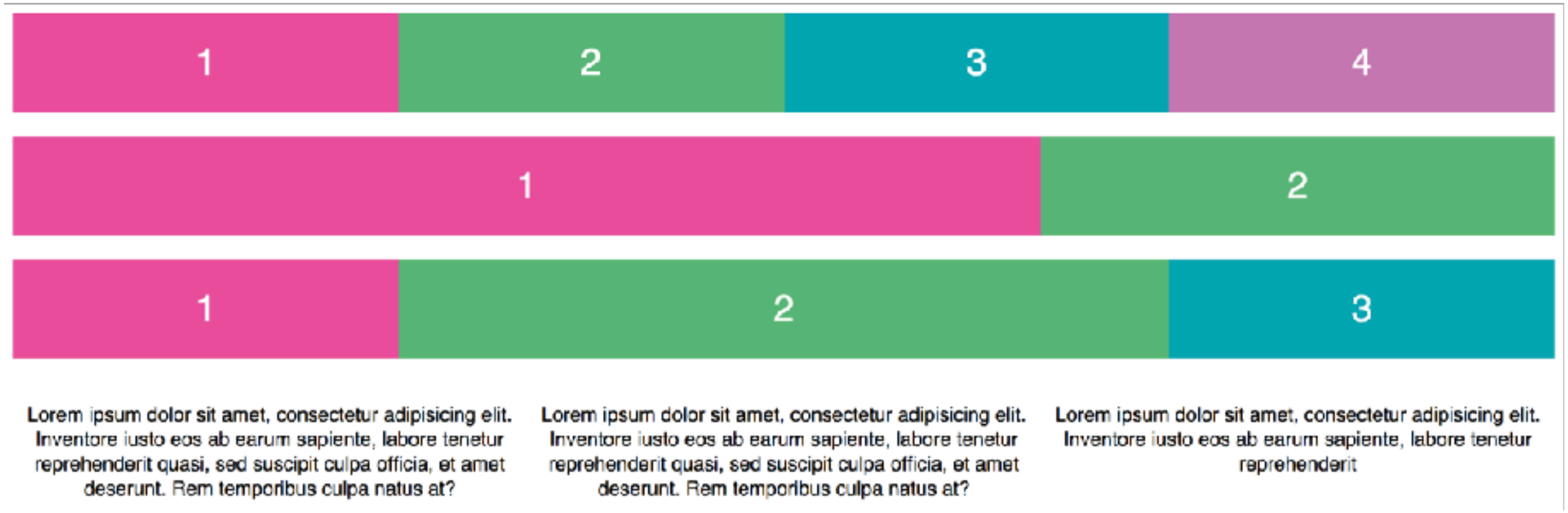
```css
.col-1-4 {
  width: 25%;
}

.col-1-3 {
  width: 33.333333%;
}

.col-3-4 {
  width: 75%;
}
```

4 columns total

3 columns wide

3*(100% divided by 4) = 75%

CODEALONG



Lorem ipsum dolor sit amet, consectetur adipisicing elit. Inventore iusto eos ab earum sapiente, labore tenetur reprehenderit quasi, sed suscipit culpa officia, et amet deserunt. Rem temporibus culpa natus at?

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Inventore iusto eos ab earum sapiente, labore tenetur reprehenderit quasi, sed suscipit culpa officia, et amet deserunt. Rem temporibus culpa natus at?

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Inventore iusto eos ab earum sapiente, labore tenetur reprehenderit

USING CHROME DEV TOOLS

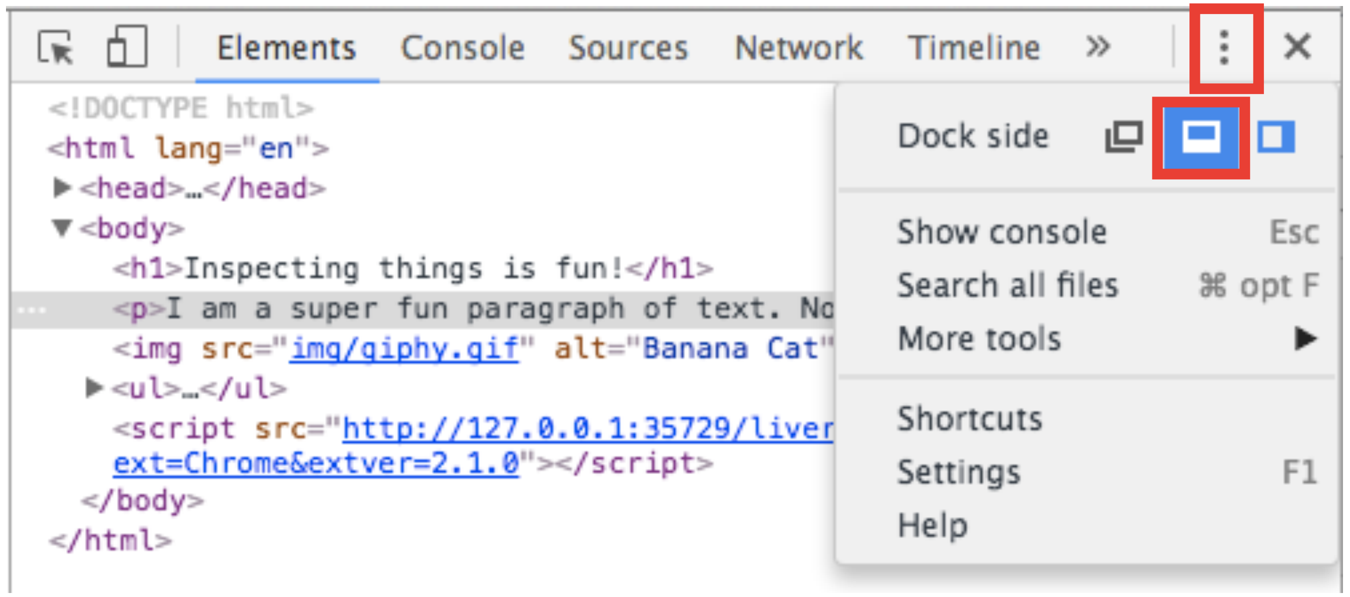
LET'S USE THE INSPECTOR!

There are several ways to open Chrome dev tools:

- Right click on an element and click "inspect"
- In Chrome, go to view > developer > Developer Tools
- Keyboard shortcut: Mac: *Cmd + Opt + I*   Windows: *F12, Ctrl + Shift + I*

DOCK LOCATION

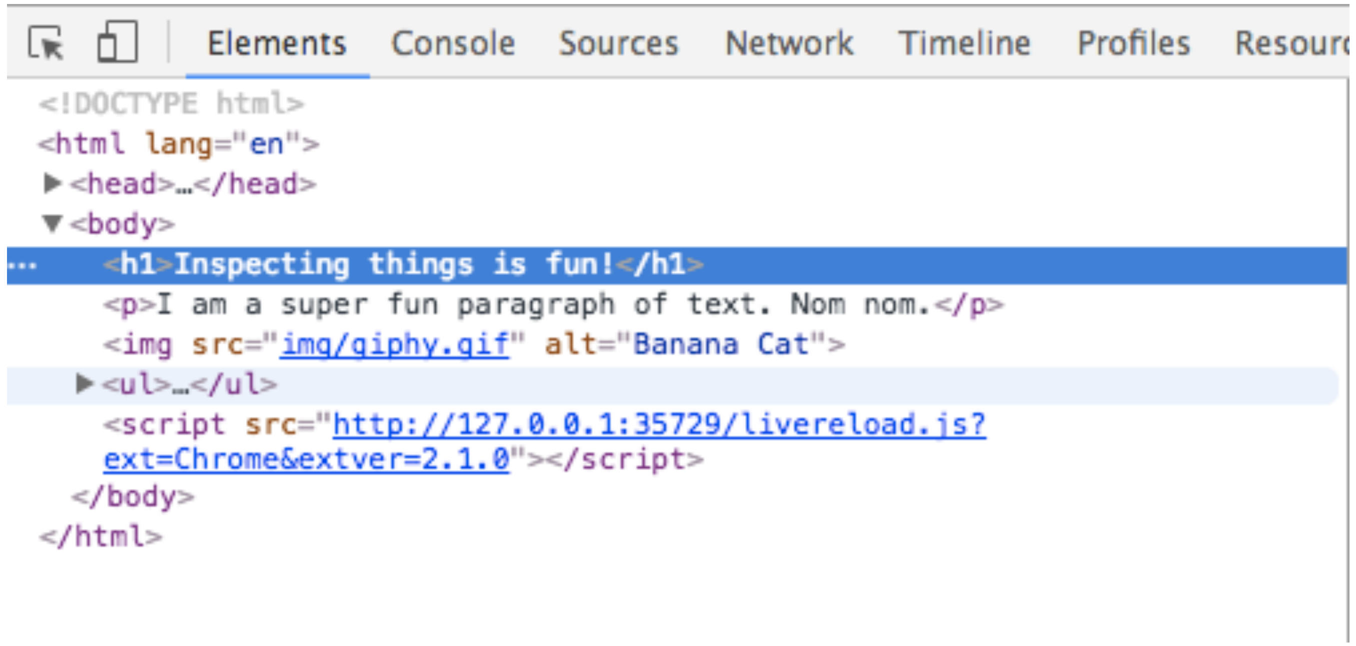
I find it easiest to move the dock to the bottom of the window.



```
<!DOCTYPE html>
<html lang="en">
▶<head>…</head>
▼<body>
    <h1>Inspecting things is fun!</h1>
    <p>I am a super fun paragraph of text. No
    <img src="img/giphy.gif" alt="Banana Cat"
  ▶<ul>…</ul>
    <script src="http://127.0.0.1:35729/liver
    ext=Chrome&extver=2.1.0"></script>
  </body>
</html>
```

PICKING AN ELEMENT

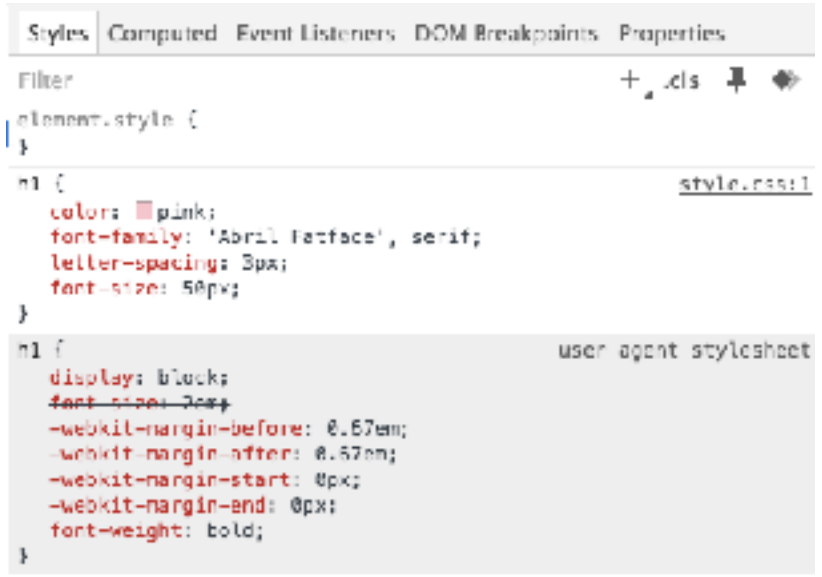
Choose an element to inspect by clicking on the element in the "Elements" tab



```
<!DOCTYPE html>
<html lang="en">
▶<head>…</head>
▼<body>
    <h1>Inspecting things is fun!</h1>
    <p>I am a super fun paragraph of text. Nom nom.</p>
    <img src="img/giphy.gif" alt="Banana Cat">
  ▶<ul>…</ul>
    <script src="http://127.0.0.1:35729/livereload.js?
    ext=Chrome&extver=2.1.0"></script>
  </body>
</html>
```

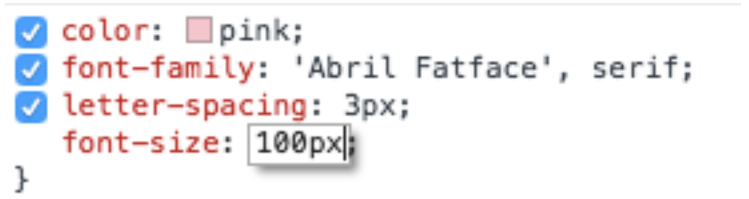
SEEING STYLES AND EXPERIMENTING

You can see what styles you've added (and default styles added by the browser) to the element in the "styles" panel.



```
element.style {
}
h1 {                                        style.css:1
  color: pink;
  font-family: 'Abril Fatface', serif;
  letter-spacing: 3px;
  font-size: 50px;
}
h1 {                                user agent stylesheet
  display: block;
  ~~font-size: 2em;~~
  -webkit-margin-before: 0.67em;
  -webkit-margin-after: 0.67em;
  -webkit-margin-start: 0px;
  -webkit-margin-end: 0px;
  font-weight: bold;
}
```

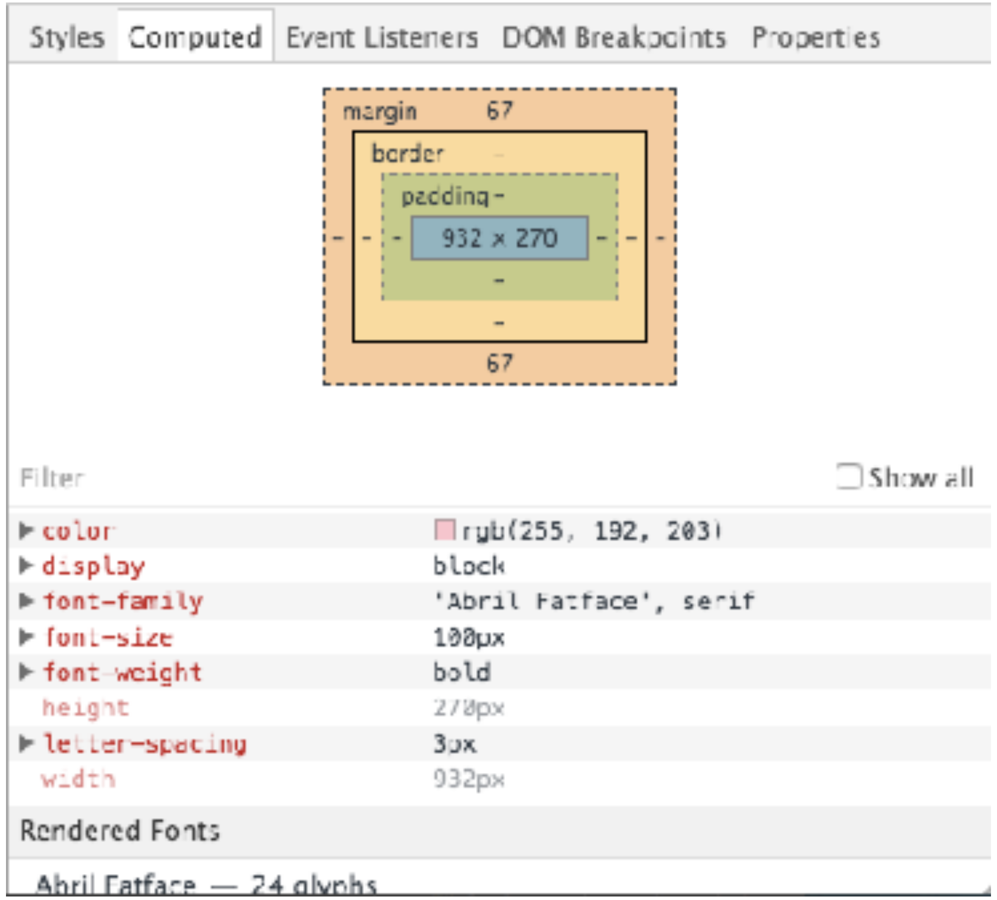
You can also experiment here!



```
✓ color: pink;
✓ font-family: 'Abril Fatface', serif;
✓ letter-spacing: 3px;
  font-size: 100px;
}
```

FINAL RENDERED STYLES AND DIMENSION

In the "computed" tab you can see all the styles that are being rendered on screen for an element. You can also see dimensions for an element including width, height, padding, margin and border.



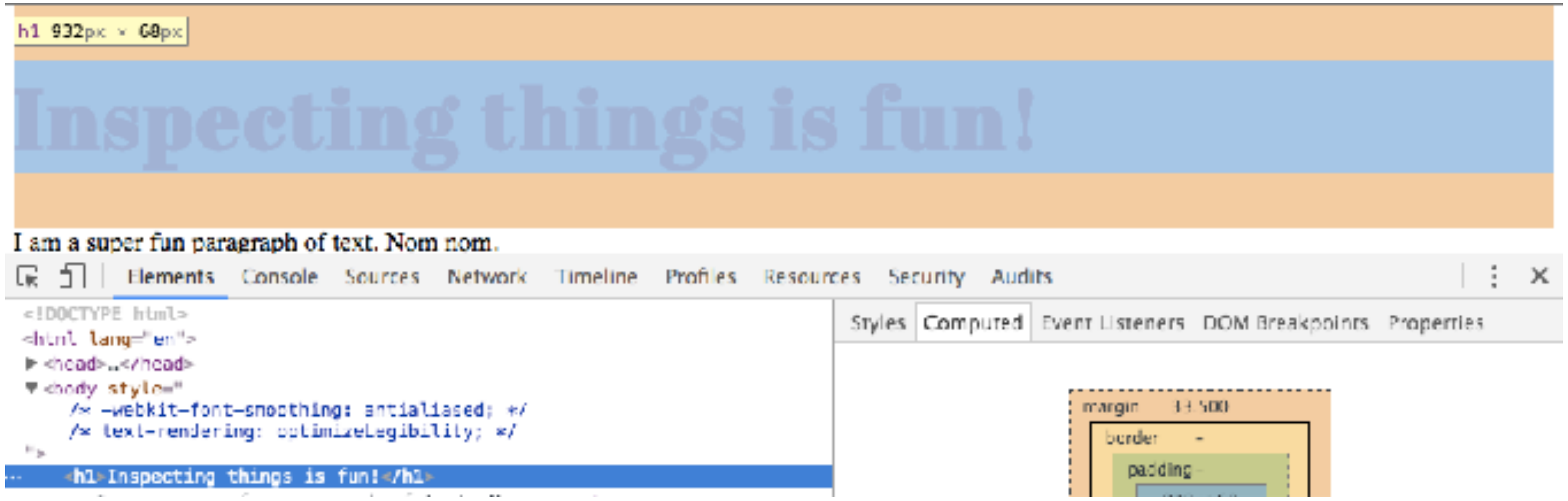
color	rgb(255, 192, 203)
display	block
font-family	'Abril Fatface', serif
font-size	100px
font-weight	bold
height	270px
letter-spacing	3px
width	932px

Rendered Fonts

Abril Fatface — 24 glyphs

VISUALIZING DIMENSIONS

One of the things I find most helpful when working through layout issues is to hover over different elements and see where everything is.



h1 932px × 60px

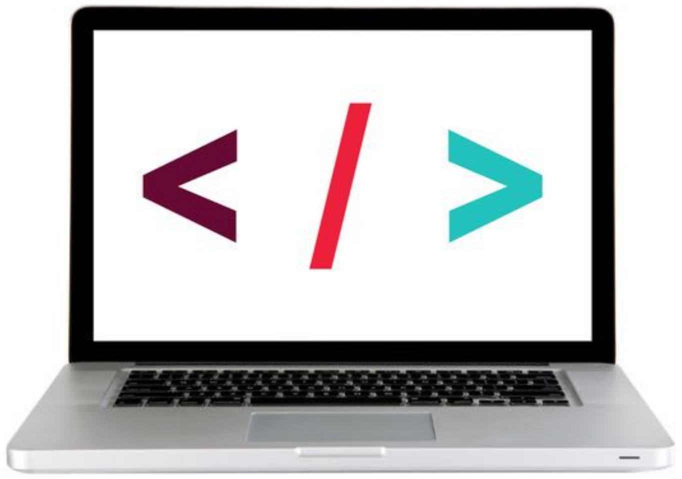
Inspecting things is fun!

I am a super fun paragraph of text. Nom nom.

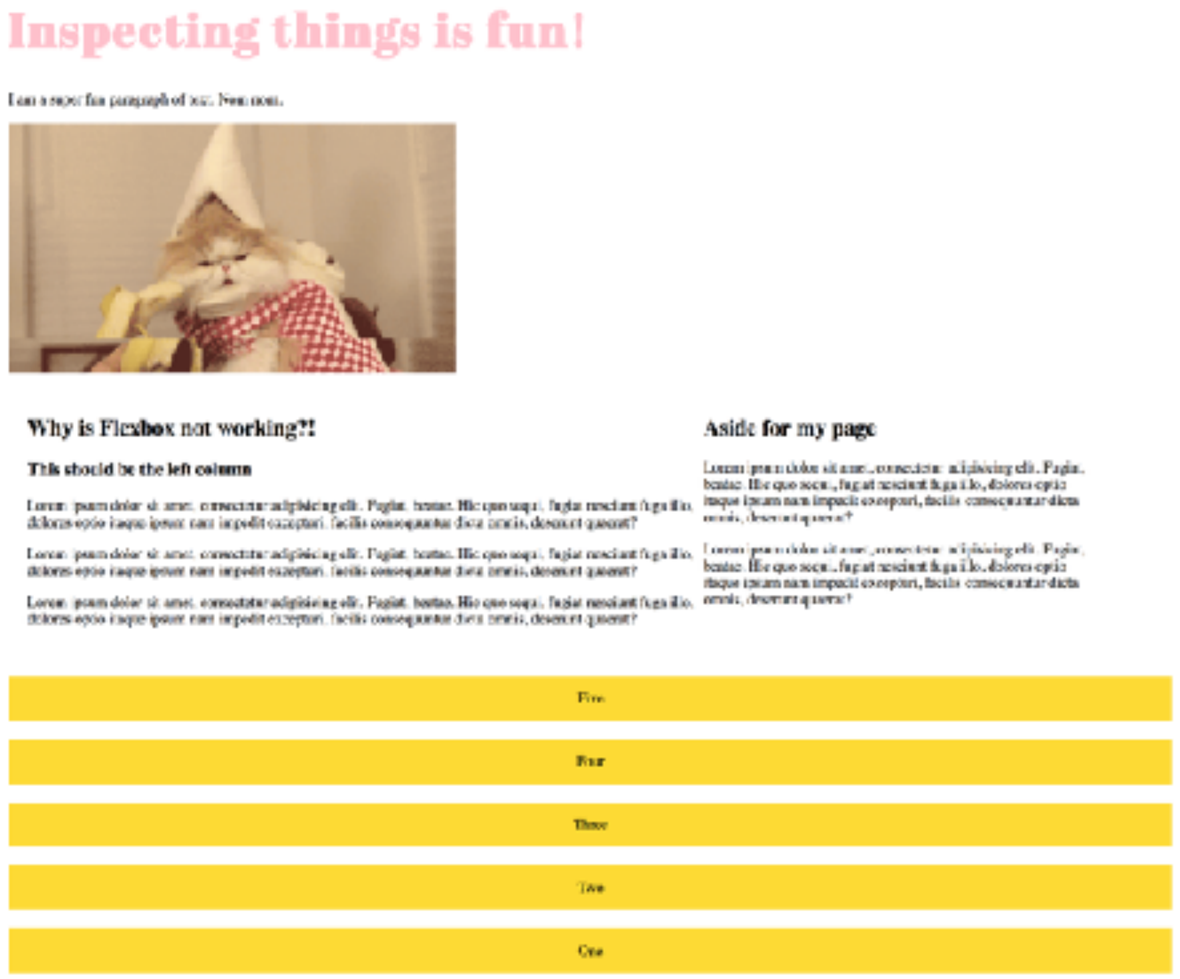
```
<!DOCTYPE html>
<html lang="en">
▶ <head>…</head>
▼ <body style="
    /* -webkit-font-smoothing: antialiased; */
    /* text-rendering: optimizeLegibility; */
  ">
    <h1>Inspecting things is fun!</h1>
```

Orange: margin
Green: padding
Also notice width and height in a little yellow box by the element

INSPECT ELEMENT LAB



Starter Code > inspect_element



Inspecting things is fun!

I am a super fun paragraph of text. Nom nom.

Why is Flexbox not working?!

This should be the left column

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Fugiat, beatae. Hic quo sequi, fugiat nesciunt fuga illo, dolores optio itaque ipsum nam impedit excepturi, facilis consequuntur dicta omnis, deserunt quaerat?

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Fugiat, beatae. Hic quo sequi, fugiat nesciunt fuga illo, dolores optio itaque ipsum nam impedit excepturi, facilis consequuntur dicta omnis, deserunt quaerat?

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Fugiat, beatae. Hic quo sequi, fugiat nesciunt fuga illo, dolores optio itaque ipsum nam impedit excepturi, facilis consequuntur dicta omnis, deserunt quaerat?

Aside for my page

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Fugiat, beatae. Hic quo sequi, fugiat nesciunt fuga illo, dolores optio itaque ipsum nam impedit excepturi, facilis consequuntur dicta omnis, deserunt quaerat?

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Fugiat, beatae. Hic quo sequi, fugiat nesciunt fuga illo, dolores optio itaque ipsum nam impedit excepturi, facilis consequuntur dicta omnis, deserunt quaerat?

Five

Four

Three

Two

One

LET'S USE THE INSPECTOR!

EXERCISE

LOCATION

- Starter Code > inspect_element

LAB SESSION

10 min

1. What color is the h1? Can you find the rgb value?
2. What is the margin-top of the h1 element? Margin-bottom? Are these styles added in our stylesheet or default styles given by the browser (in the user agent stylesheet).
3. Hover over the h1 to visualize these dimensions.
4. If we didn't want margin on top, what could we do? Try to update the h1 so there is no margin on top — in your inspector.
5. Our two-column layout isn't working! Using your inspector can you figure out why? Can you fix this — in your inspector.
6. What is the background-color for each li? How much padding and margin does each li have? What is the width and height?
7. Hover over an li to visualize these dimensions.
8. Bonus - can you figure out why the image isn't loading? Hint: look in the "Console" tab and compare with what you see in the folder in Finder. What should the image path be?

PACIFIC LAB

ACTIVITY — PLANNING

EXERCISE

KEY OBJECTIVE

- Get practice using a grid system

LOCATION OF FILES

- starter code > **pacific** folder

TIMING

1. Look at the pacific_burger.png file
2. Map out your grid system - how many rows will you have? How many types columns will you need to account for?
3. Discuss with a partner!

ACTIVITY — GRID SYSTEM LAB

EXERCISE

KEY OBJECTIVE

- Get practice using a grid system

LOCATION OF FILES

- starter code > **pacific** folder

TIMING

1. Map out the rest of your styles - We've done a bit of the HTML already, but which CSS properties will I need to use?
2. Build it out!

LEARNING OBJECTIVES

- Differentiate between block and inline elements
- Create multi-column layouts with flexbox
- Describe a grid system
- Use classes to set up modular, multi-column layouts

WEEKLY OVERVIEW

WEEK 3	Layout & Grid Systems / Layout Lab
WEEK 4	Responsive Design / CSS Positioning
WEEK 5	Forms / Final Project Lab

EXIT TICKETS