1) HAVE SLACK OPEN

2) IN CHROME, OPEN UP:

https://github.com/CHIFEWD7/your.name

SNACKS & DESIGN

TODAY!

TOM BUNTING & MICHAEL FISCHER

(GOOGLE SHEET IS PINNED IN SLACK)

Q & A

“How to name divs for use with nesting?”

“What are the most popular CSS frameworks?”

“How do I make multi-box layouts?”

“Can you recommend resources to help practice?”

“When creating a navigation, must you use <nav>?”

http://www.w3schools.com/html/html5_semantic_elements.asp

“How do you do borders in one line?”

https://developer.mozilla.org/en-US/docs/Web/CSS/Shorthand_properties

“When nesting selectors, can we add declarations that apply both to the parent and child, or do those only apply to the child?”

“Is the ‘DOM’ (Document Object Model) something you create for each site you make?”

“Why use terminal? Are there alternatives?”

“When do I use margin vs. padding?”

“It seems like you can put certain properties under different selectors. Is there a best practice on where you place the property if it has the same outcome?”

“What is the difference between border and padding?”

“How do you know whether to apply CSS rules to the highest level of organization or other?”

“e.g. when to apply to ‘nav ul’ and when to apply to ‘nav ul li’?”

COMING UP...

~~FEBRUARY 25: BOX MODEL~~

MARCH 1: LAYOUT (CSS)

MARCH 3: LAYOUT LAB (W/ ADVANCED CSS ~ RESETS)

MARCH 8: LAYOUT LAB CONT. (W/ ADVANCED CSS ~ POSITIONING)

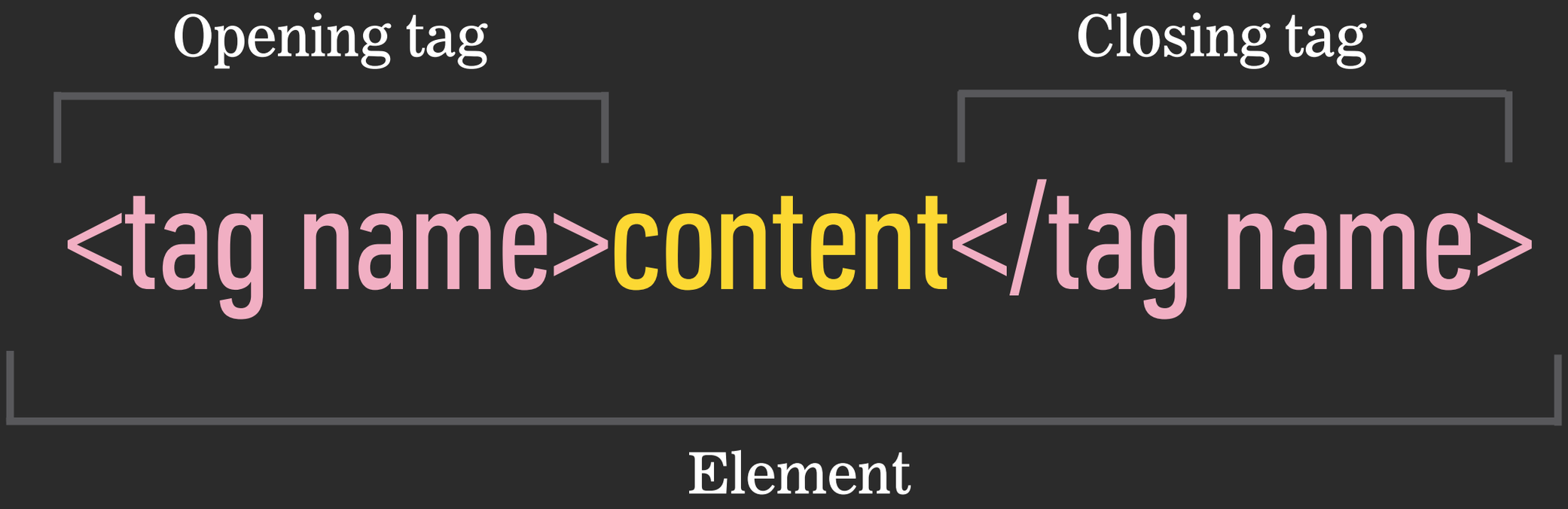
GENERAL ASSEMBLY

LAYOUT

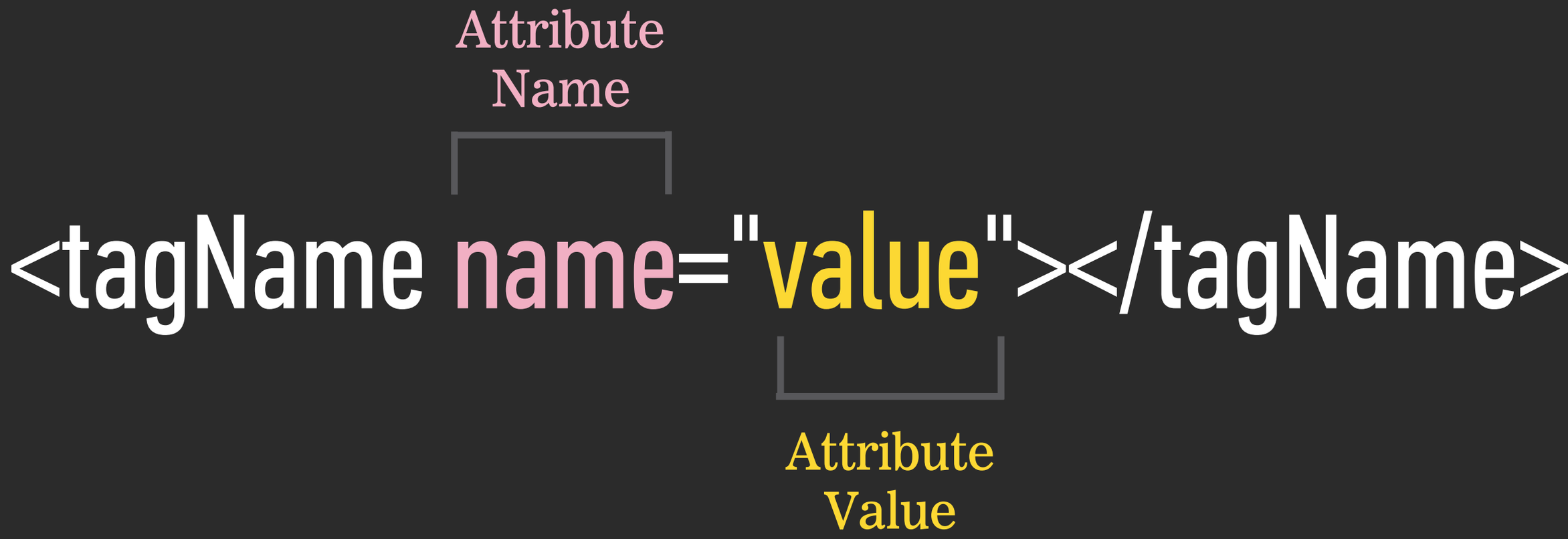
Eric Boyer

REVIEW

HTML SYNTAX — TAGS



HTML SYNTAX — ATTRIBUTES



CSS SYNTAX

CSS SYNTAX

```
h1, h2 {
        color: yellow;
        font-size: 16px;
}
```

Property

Value

NESTED SELECTORS

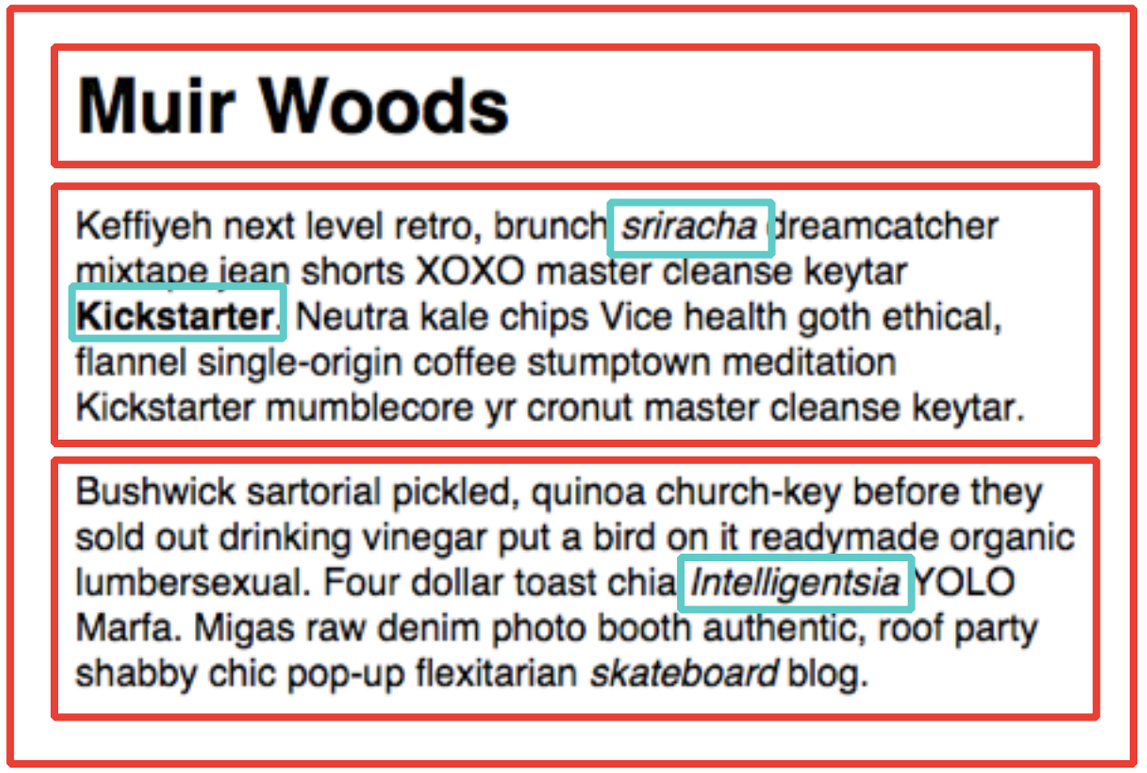
SELECTOR:	MEANING:	EXAMPLE:
UNIVERSAL	Applies to all elements in the document	* {}
TYPE	Matches element names	h1, h2, h3 {}
CHILD	Matches an element that is a direct child of another element	p>a {}
DESCENDANT	Matches an element that is a descendent (not just a child) of another element	p a {}
ADJACENT SIBLING	Matches the element that is directly after another element	p+a {}
GENERAL SIBLING	Matches the element that is a sibling of another	p~a {}

WHAT IS CSS?

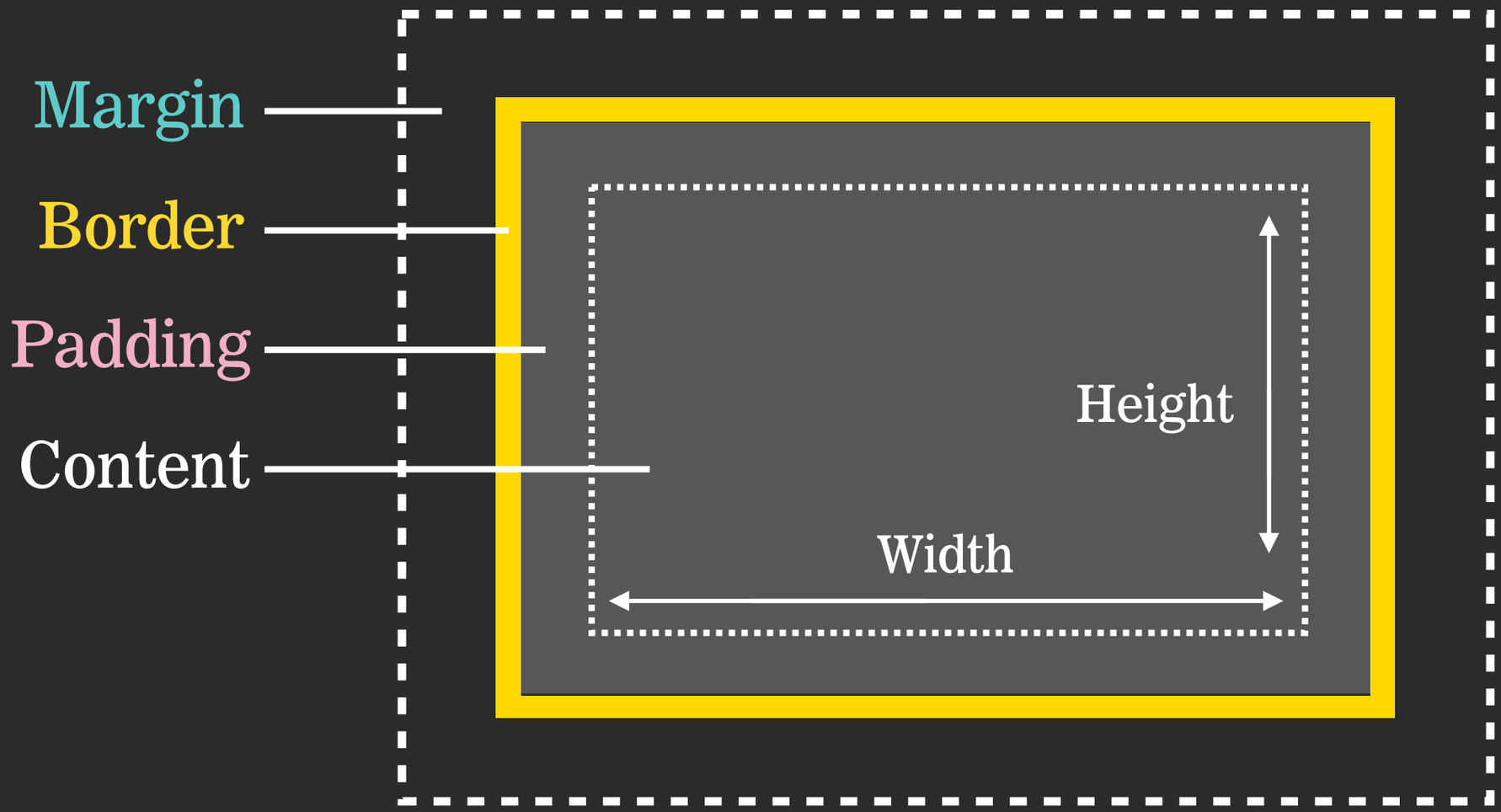
Muir Woods

Keffiyeh next level retro, brunch *sriracha* dreamcatcher mixtape jean shorts XOXO master cleanse keytar **Kickstarter**. Neutra kale chips Vice health goth ethical, flannel single-origin coffee stumptown meditation Kickstarter mumblecore yr cronut master cleanse keytar.

Bushwick sartorial pickled, quinoa church-key before they sold out drinking vinegar put a bird on it readymade organic lumbersexual. Four dollar toast chia *Intelligentsia* YOLO Marfa. Migas raw denim photo booth authentic, roof party shabby chic pop-up flexitarian *skateboard* blog.



REFRESHER — BOX MODEL



CLASSES AND IDS

- Classes and ids allow us to assign ‘labels’ to elements so that we can target them in our stylesheets

IDS

- Ids are used to target *one specific element*
- **Important**: two elements on the same page cannot have the same id

```
<h3 id="about">Content</h3>
```

```
#about {
  color: #ff0000;
}
```

CLASSES

- Classes are used to group elements together

```
<li class="emphasis">Content</li>
```

```
.emphasis {
  color: #ff0000;
}
```

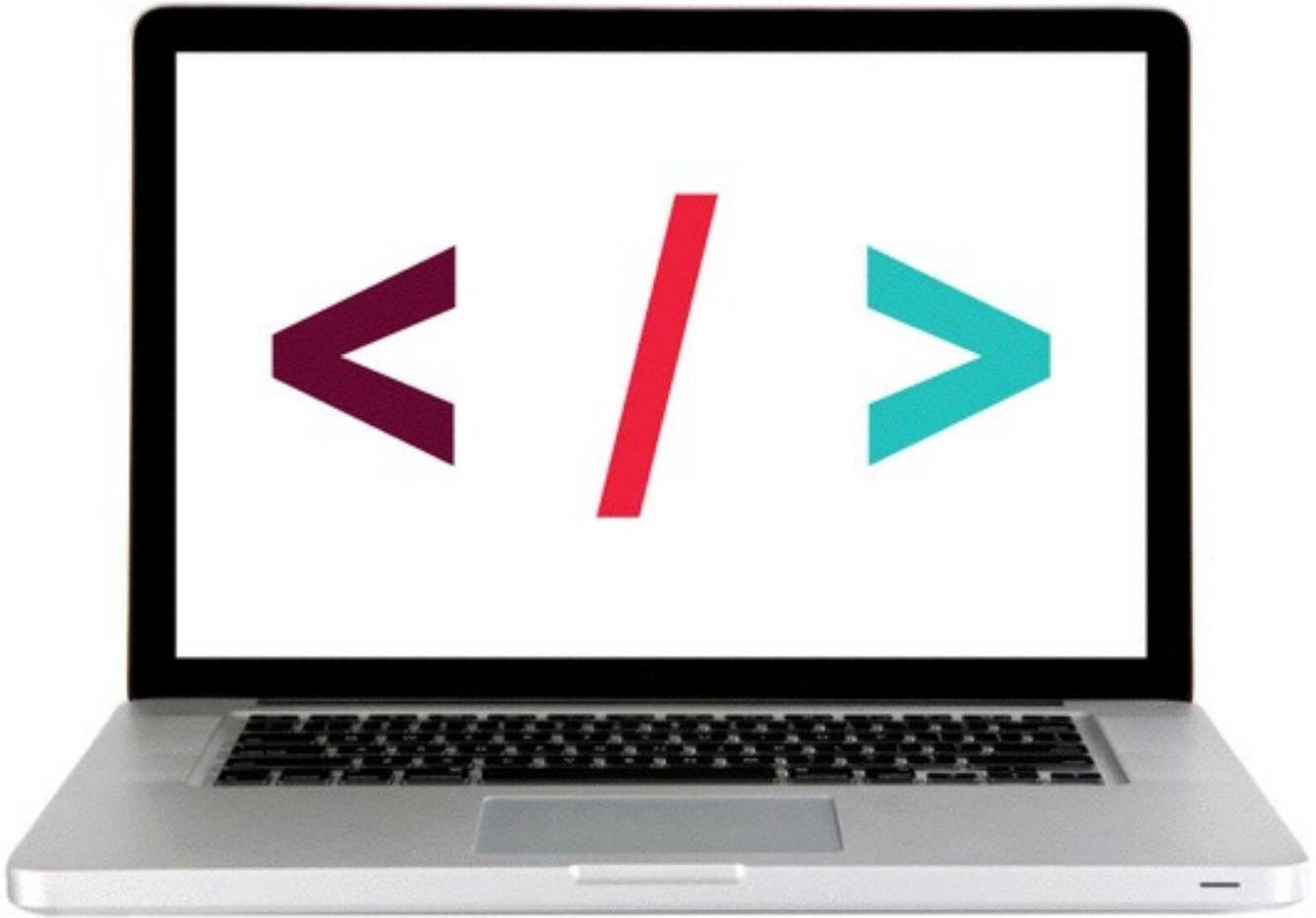
LEARNING OBJECTIVES

- Differentiate between block and inline elements
- Identify when HTML5 structural elements should be used
- Apply header, footer, sidebar, and multi-column layouts to build a web page.
- Experiment and predict effects of floats and clearing CSS positioning.

AGENDA

- Push homework to Github
- Display
- Classes & IDs
- Specificity
- Divs
- HTML5 Structural Elements
- Box-Sizing Part 2
- Floats
- Multi-column layouts
- Lab — Travel Blog pt. 2

LET'S TAKE A CLOSER LOOK

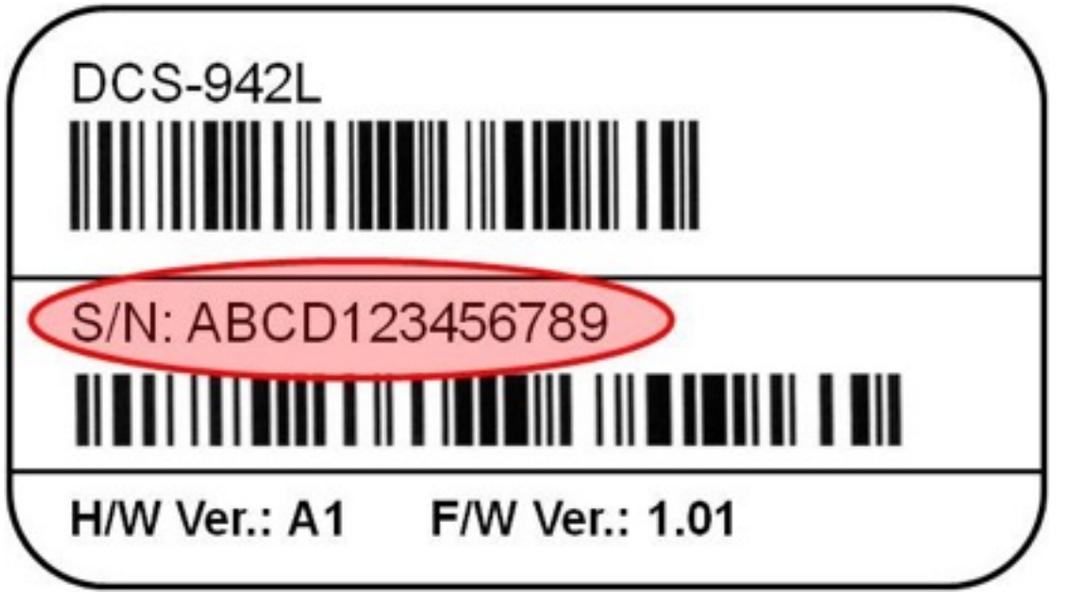


CLASSES AND IDS

TARGETING SPECIFIC ELEMENTS

- Classes and ids allow us to assign ‘labels’ to elements so that we can target them in our stylesheets

TARGETING SPECIFIC ELEMENTS

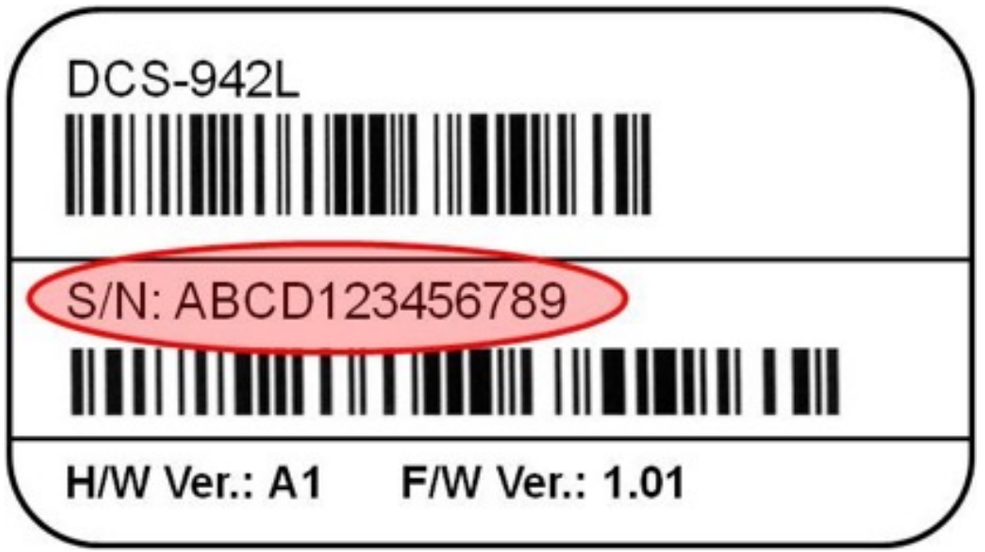


IDS

- Ids are used to target *one specific element*
- Each element can only have one id
- **Important**: two elements on the same page cannot have the same id

```
<h3 id="about">Content</h3>
```

```
#about {
  color: #ff0000;
}
```



CLASSES

- Classes are used to group elements together
- Elements can have multiple classes

```
<li class="emphasis">Content</li>
```

```
.emphasis {
  color: #ff0000;
}
```

LET'S CHAT MORE ABOUT SPECIFICITY & IMPORTANCE & INHERITANCE

INHERITANCE — SETTING BASE STYLES

- Certain properties are passed on from a parent element down to its children
- If you specify the font-family or color properties on the <body> element, they will apply to most child elements unless there is a more specific rule that applies. This is because the font-family property is **inherited** by child elements.

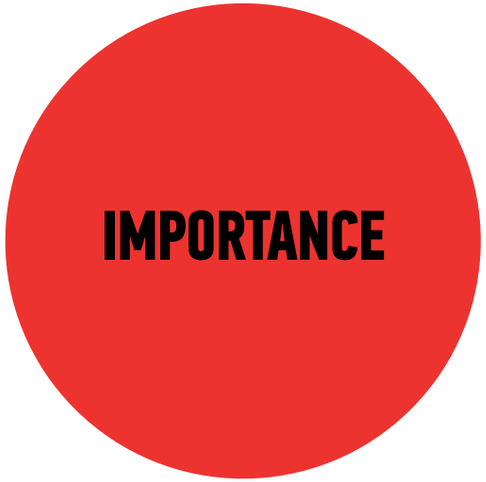
Inherited properties you'll use in this course:

- color
- font-family
- font-size
- font-style
- font-weight
- letter-spacing
- line-height
- list-style
- text-align
- text-indent
- text-transform
- word-spacing

MORE ABOUT CASCADING

- CSS rules are able to override one another and cancel each other out, depending on their order.
- CSS rules cascade downward until they are canceled out by another rule.

CONFLICT!! — WHEN TWO RULES APPLY TO THE SAME ELEMENT

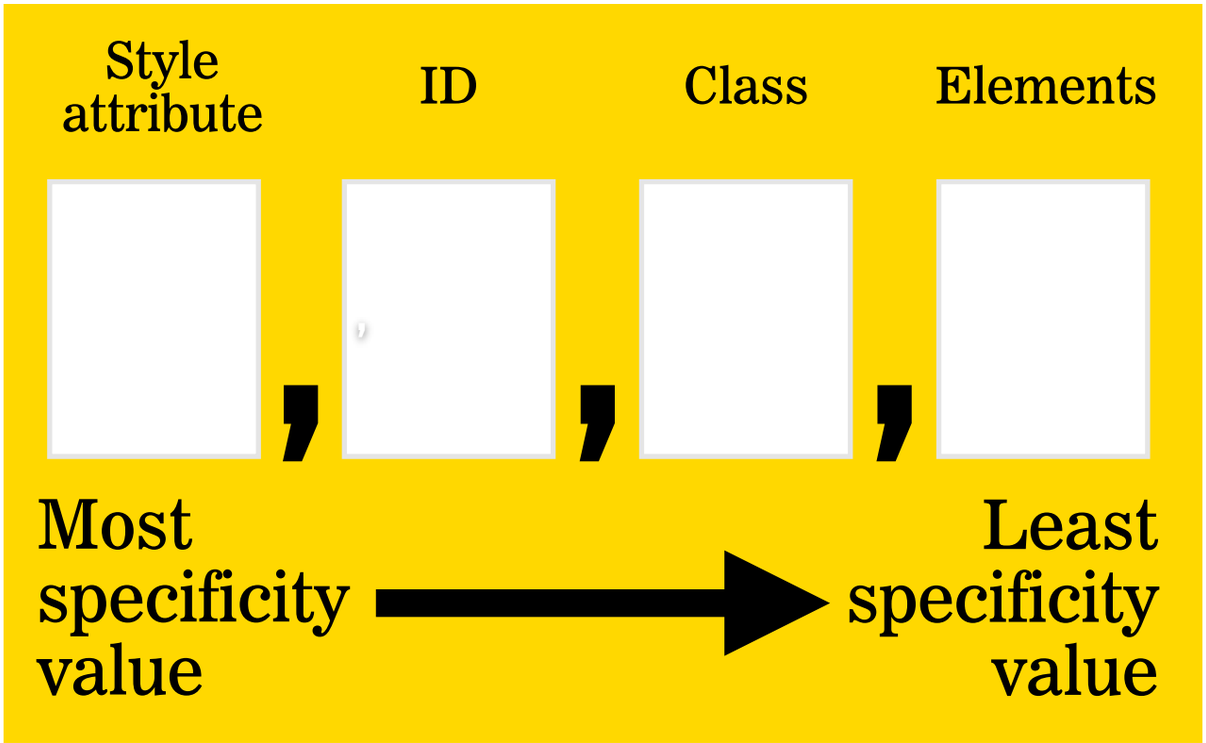


Adding **!important** after any property value indicates that it should be considered *more important than other rules that apply to the same element*.

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

If the two selectors have the same importance and specificity, the latter will take precedence

THE SPECIFICITY GAME!

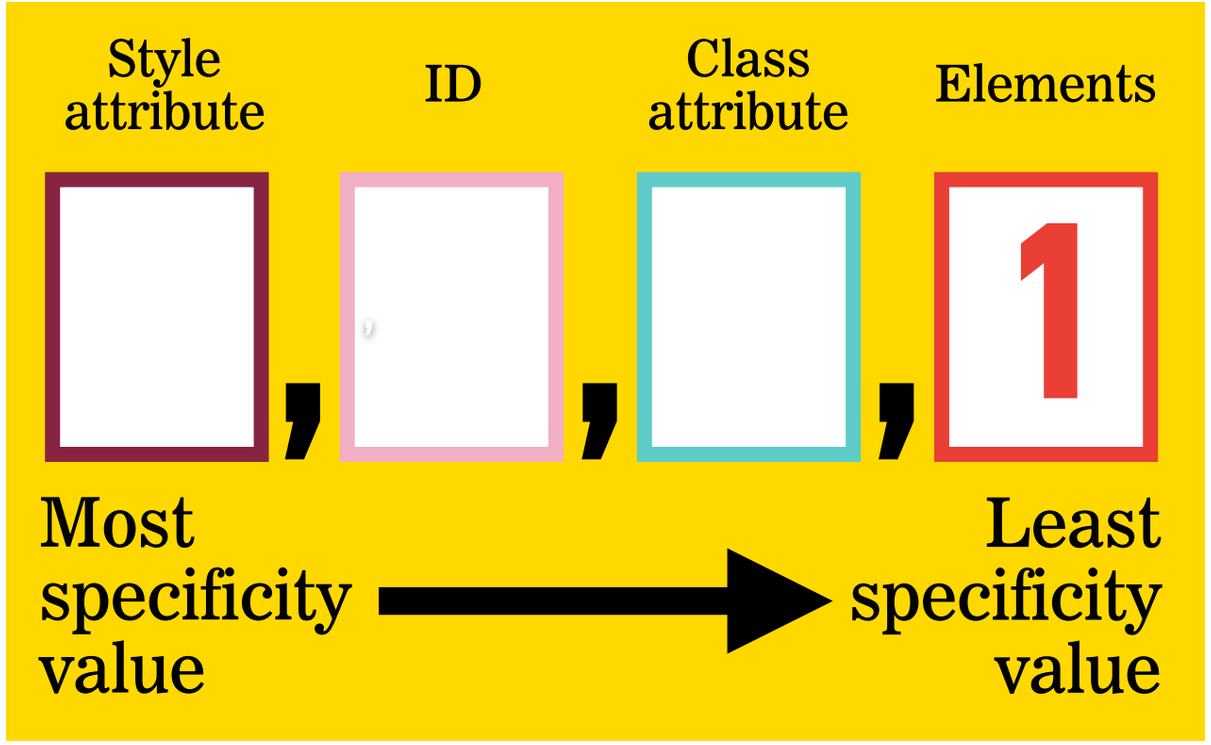


MORE ABOUT CASCADING

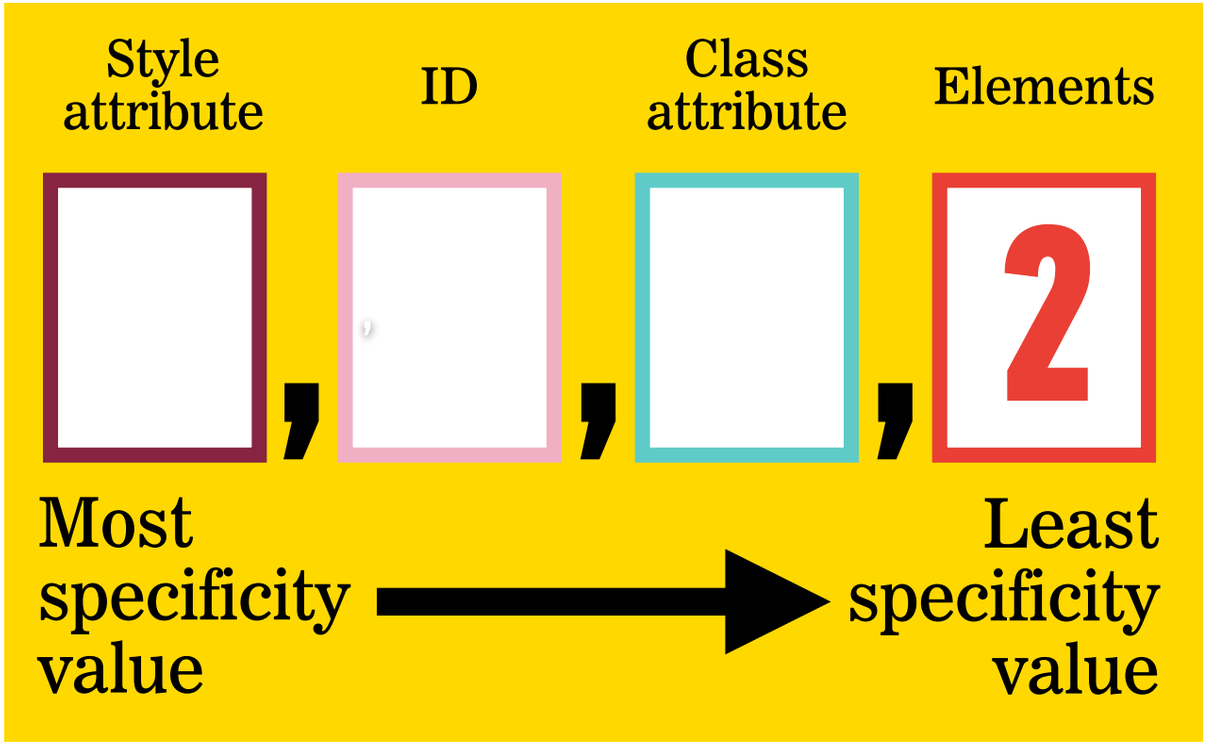
SPECIFICITY

WINNER!

a



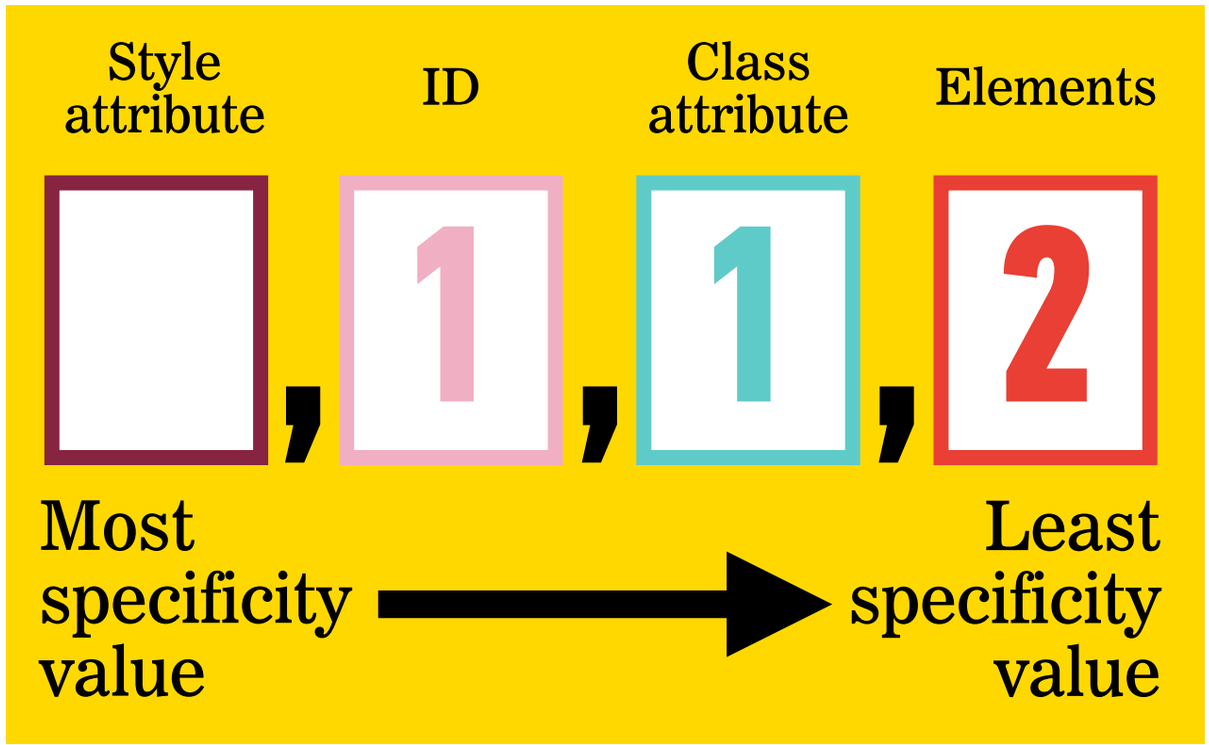
p a



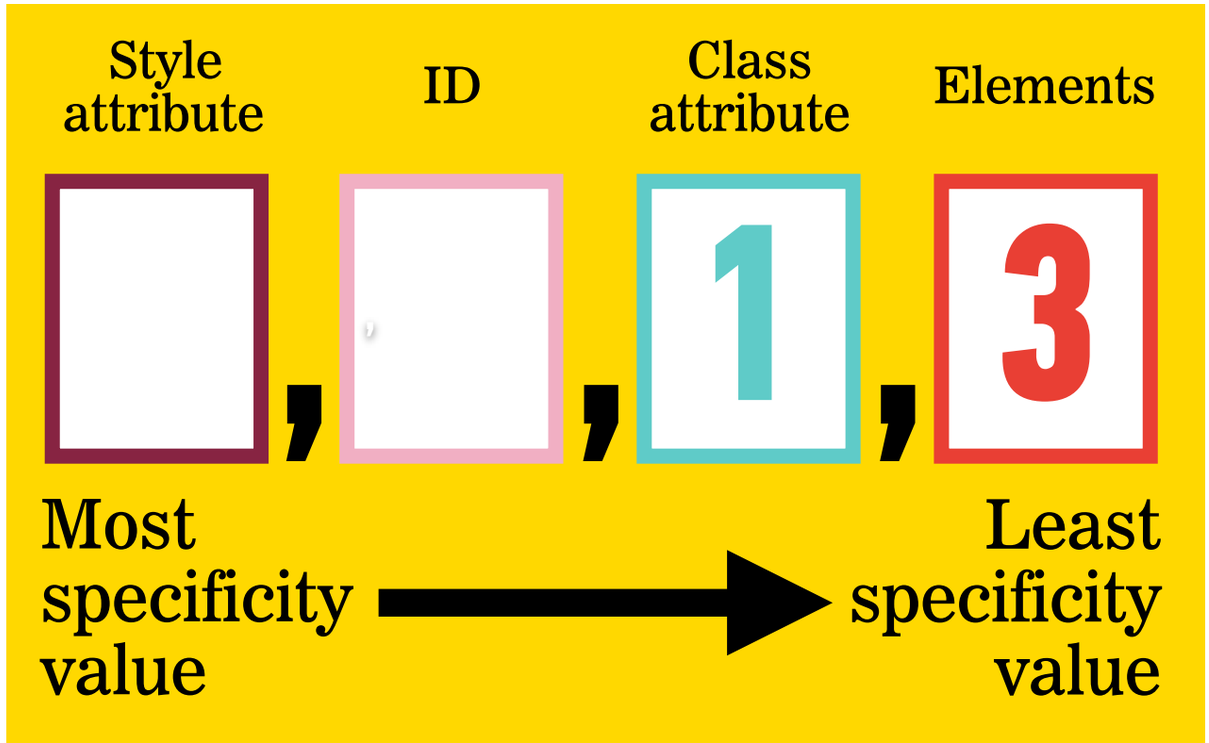
CSS CASCADING

WINNER!

#nav li.active p



.home li h2 + p

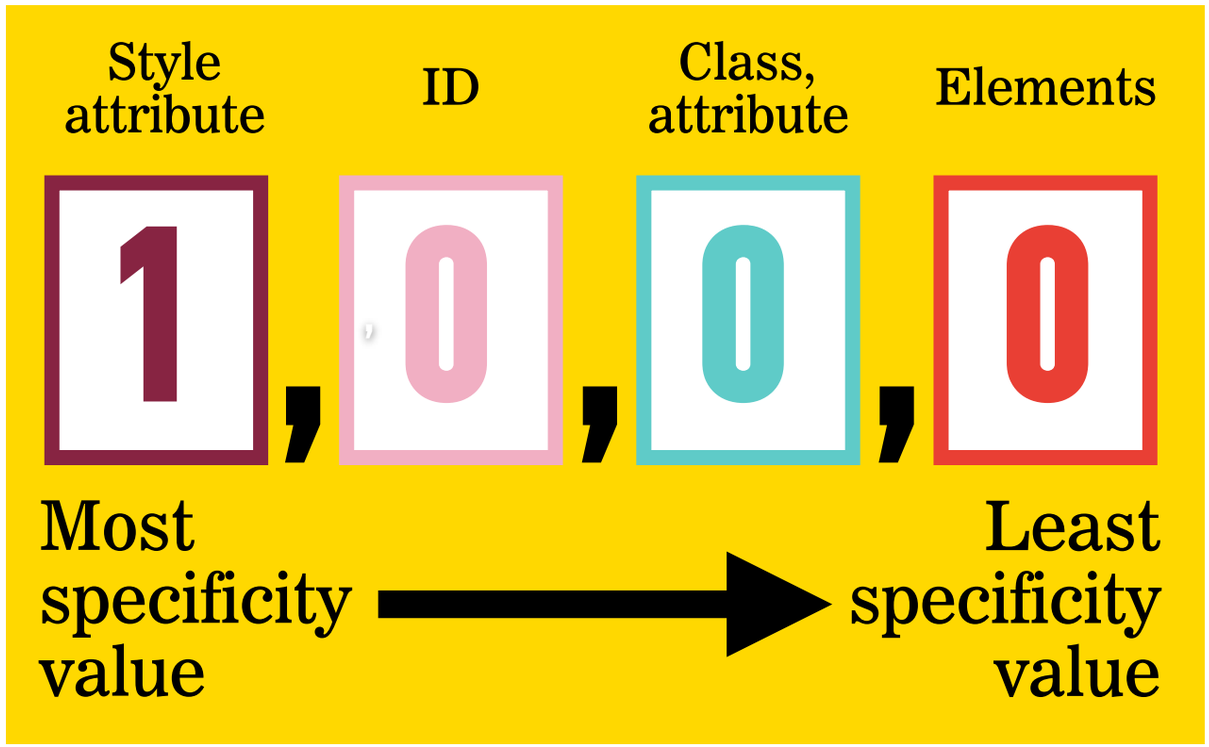


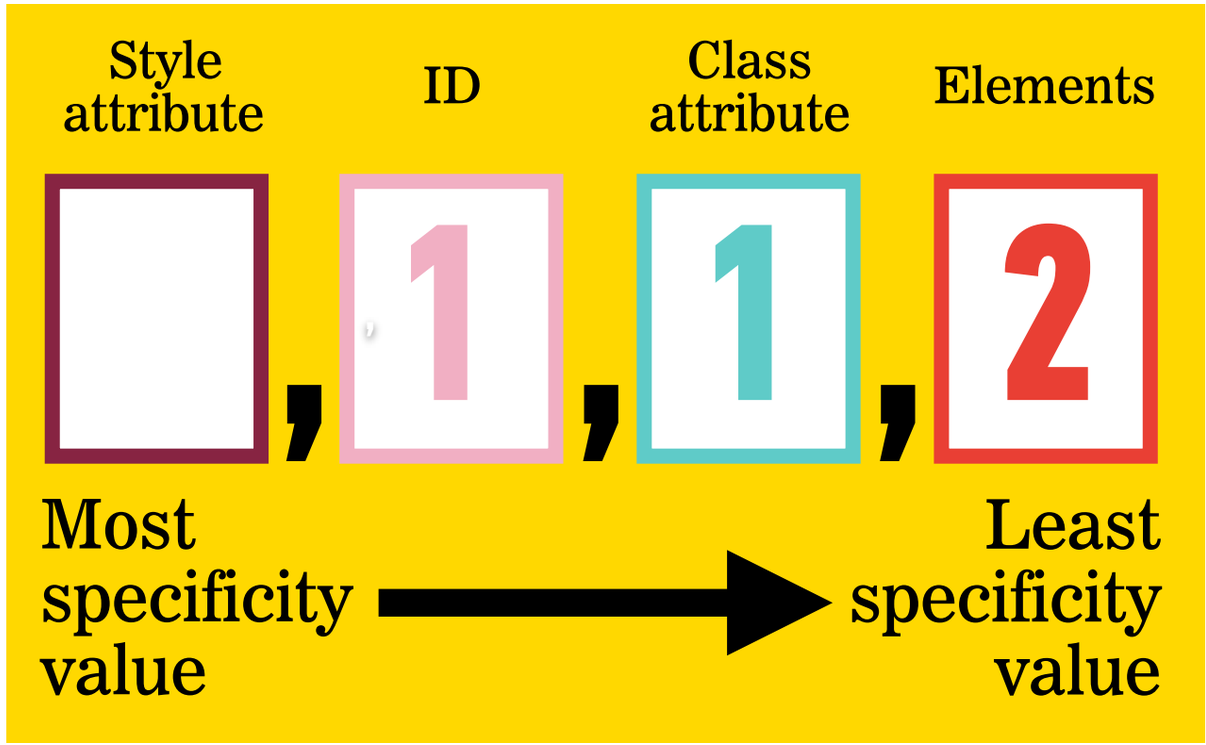
CSS CASCADING

WINNER!

`<li style="color: red"></li>`

`#about .first h2 + li`

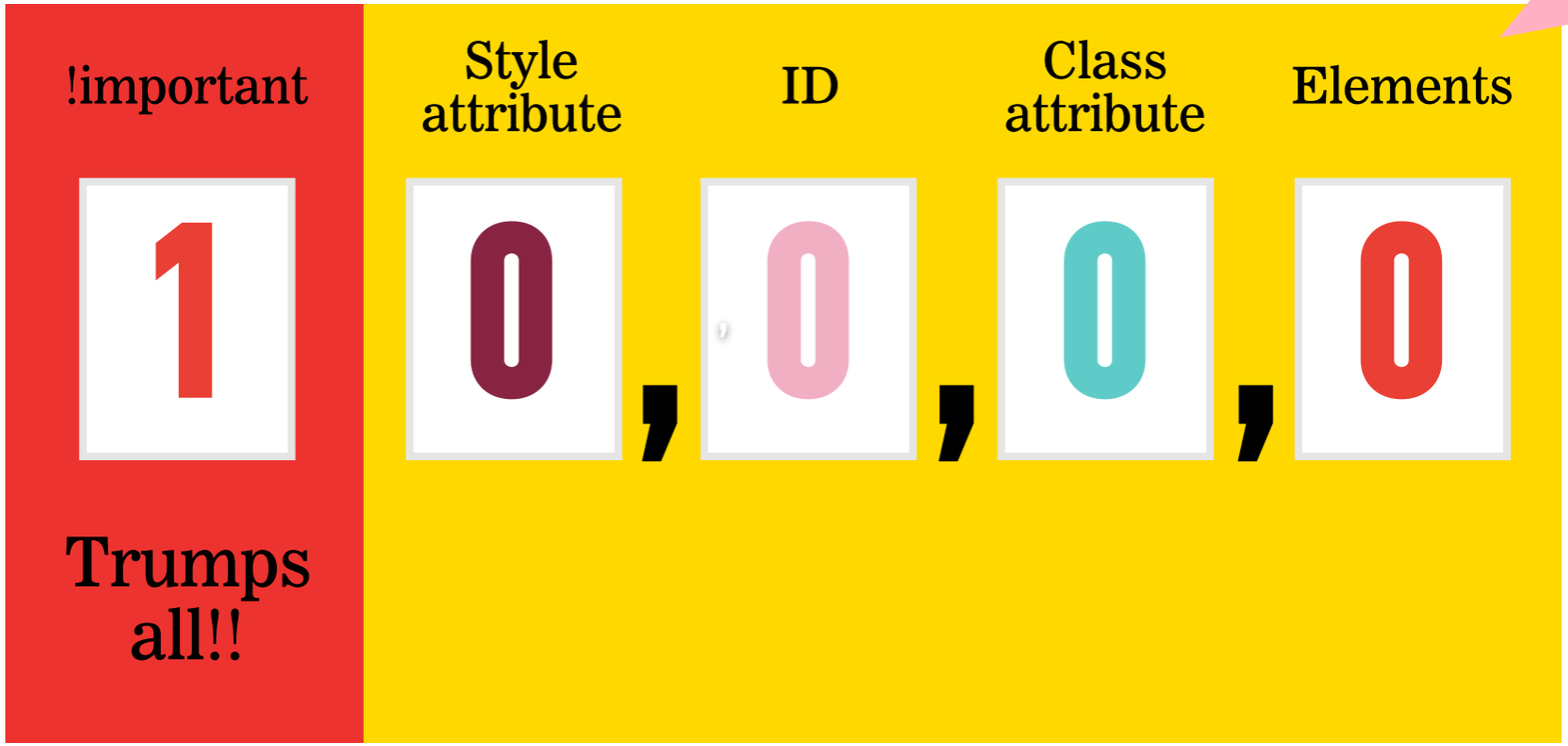




CSS CASCADING

!IMPORTANT

li {color: green *!important*;}



WINNER!

!important can only be overridden by another !important

ACTIVITY

KEY OBJECTIVE

- Summarize CSS “cascade” including: importance, specificity and inheritance.

TYPE OF ACTIVITY

- Turn and Talk

TASKS

2 min	1. Turn and talk to a partner
2 min	2. A few pairs will share with the class

DISPLAY

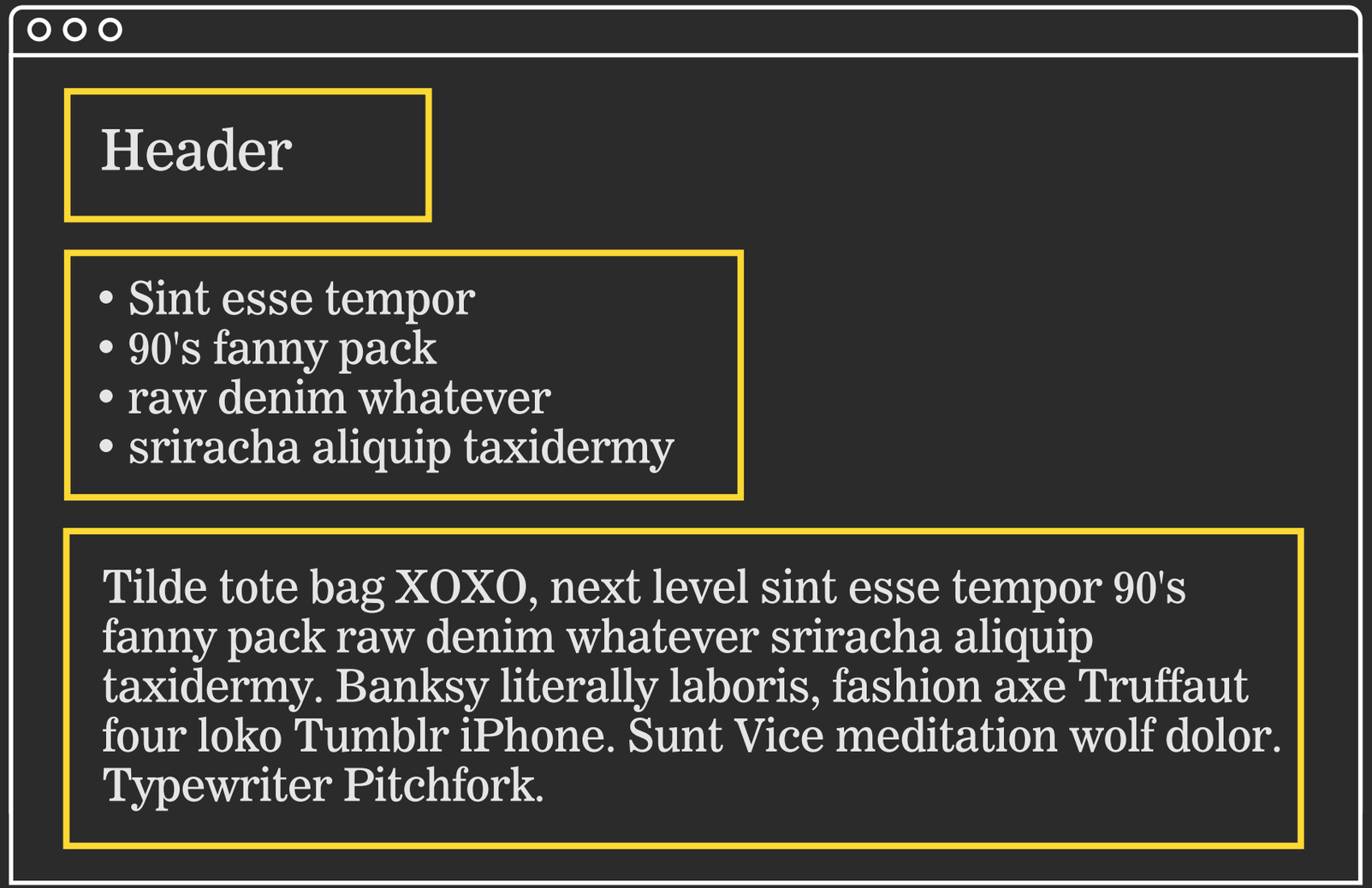
BUILDING BLOCKS

BLOCK-LEVEL ELEMENTS

- Will always start on a new line

Examples:

- <h1>-<h6>
-
-
- <p>
-
-



INLINE ELEMENTS

Will always appear to continue on the same line as their neighboring elements

Examples:

- <a>
- <img>
-
-
- <q>
-

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.

[Placeholder Images]

https://placekitten.com/

https://www.placecage.com/

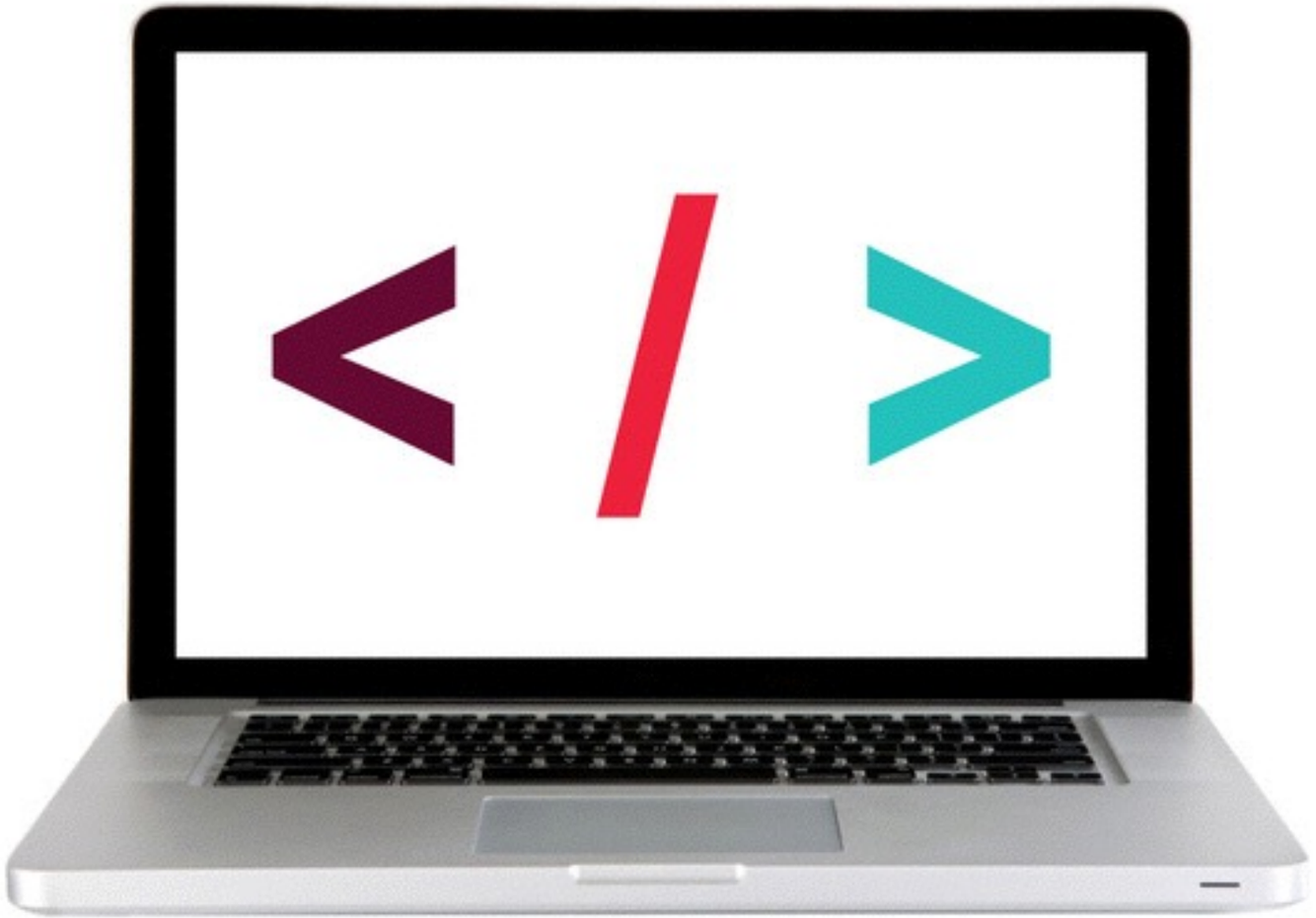
http://placehold.it/

[Lorem Ipsum]

http://html-ipsum.com/

http://hipsum.co/

LET'S TAKE A CLOSER LOOK – PART 1



DIMENSION – A KEY DIFFERENCE BETWEEN INLINE AND BLOCK ELEMENTS

If you try to add dimension to an inline element:

- Some properties will be applied
- Some properties will be *partially* applied
- Others will *not* be applied at all

The most noticeable properties are width, height, margin and padding.

DIMENSION – A KEY DIFFERENCE BETWEEN INLINE AND BLOCK ELEMENTS

SUMMARY — WHICH DIMENSIONS CAN BE CHANGED?

	WIDTH & HEIGHT	PADDING & MARGIN
BLOCK	yes	can apply to all sides
INLINE	no	will only affect left and right sides

DISPLAY

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

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

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

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

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

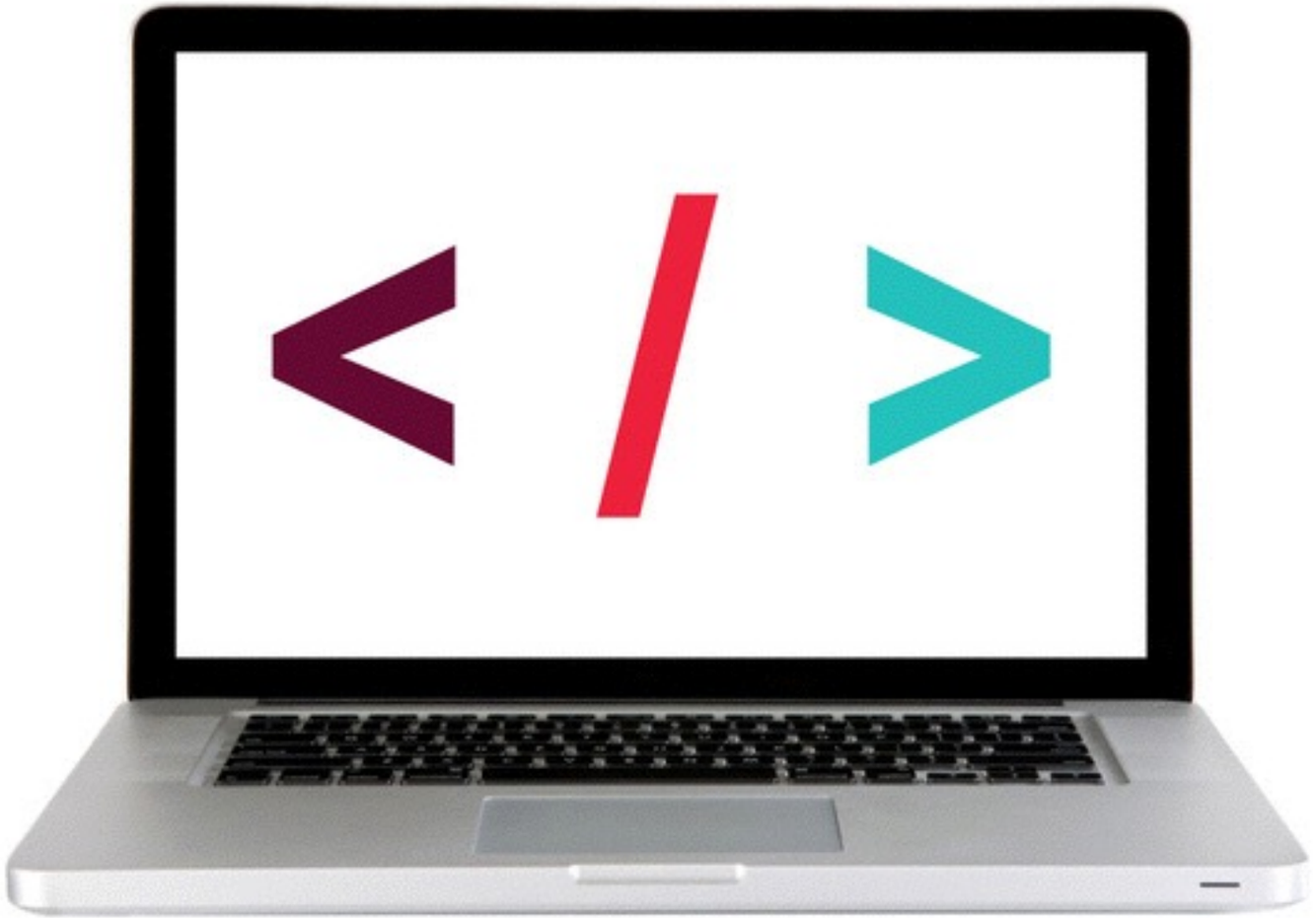
3. Make a block-level element flow like an inline element, while retaining width, height, padding, and margin:

```
h2 {
  display: inline-block;
}
```

4. Hide an element from a page:

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

LET'S TAKE A CLOSER LOOK



ACTIVITY — DISPLAY LAB

EXERCISE

KEY OBJECTIVE

- Get practice using the 'display' property

LOCATION OF FILES

- starter_code_lesson_4 > **display_lab** folder

TIMING

5 min 1. Follow the instructions in steps 1-3

DIVS

GROUPING TEXT & ELEMENTS

THE <DIV> ELEMENT

- Defines a section or division in an HTML document
- Allows us to group a set of elements together into a block-level box

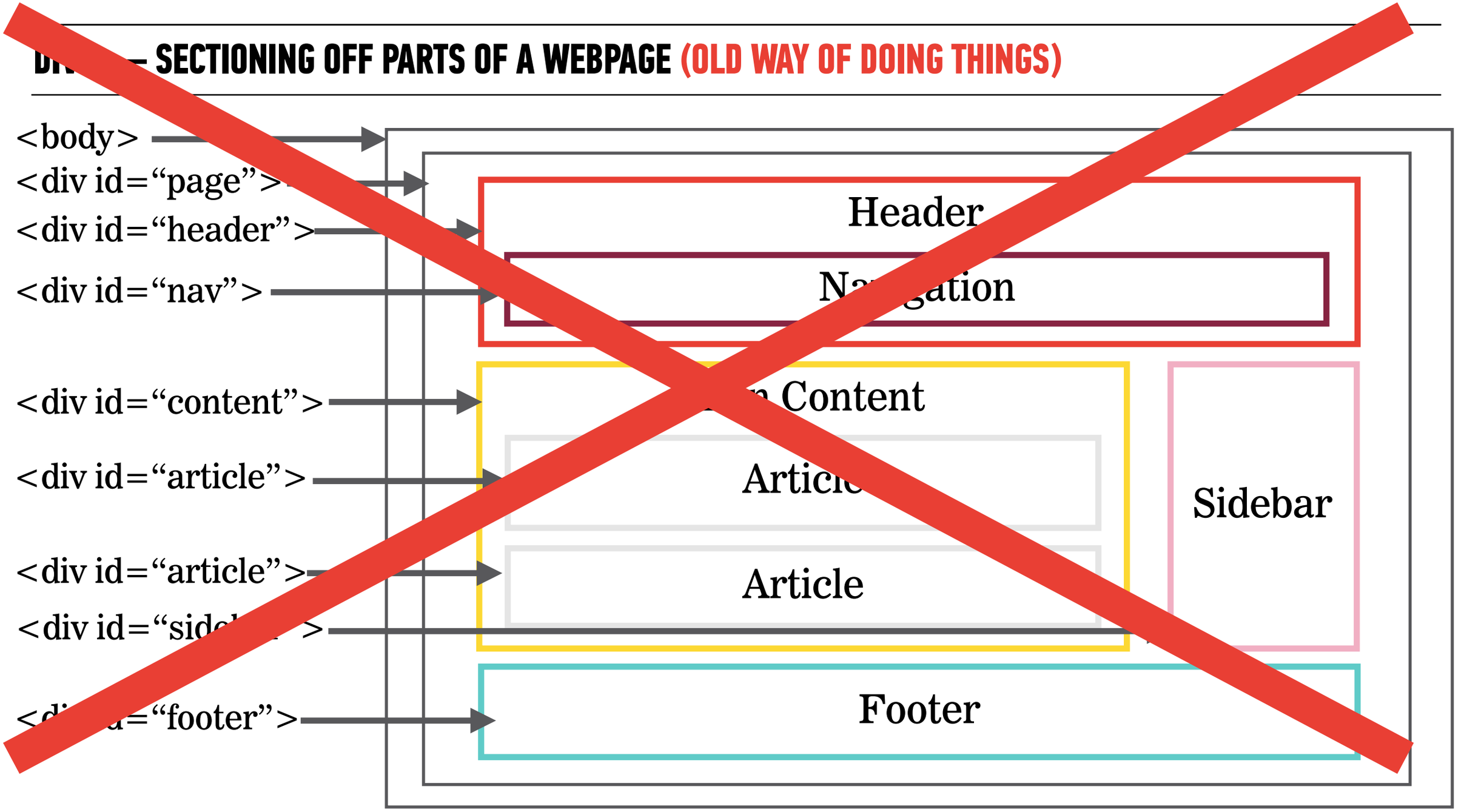
** Divs allow developers to section off parts of a page.

THE <SPAN> ELEMENT

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

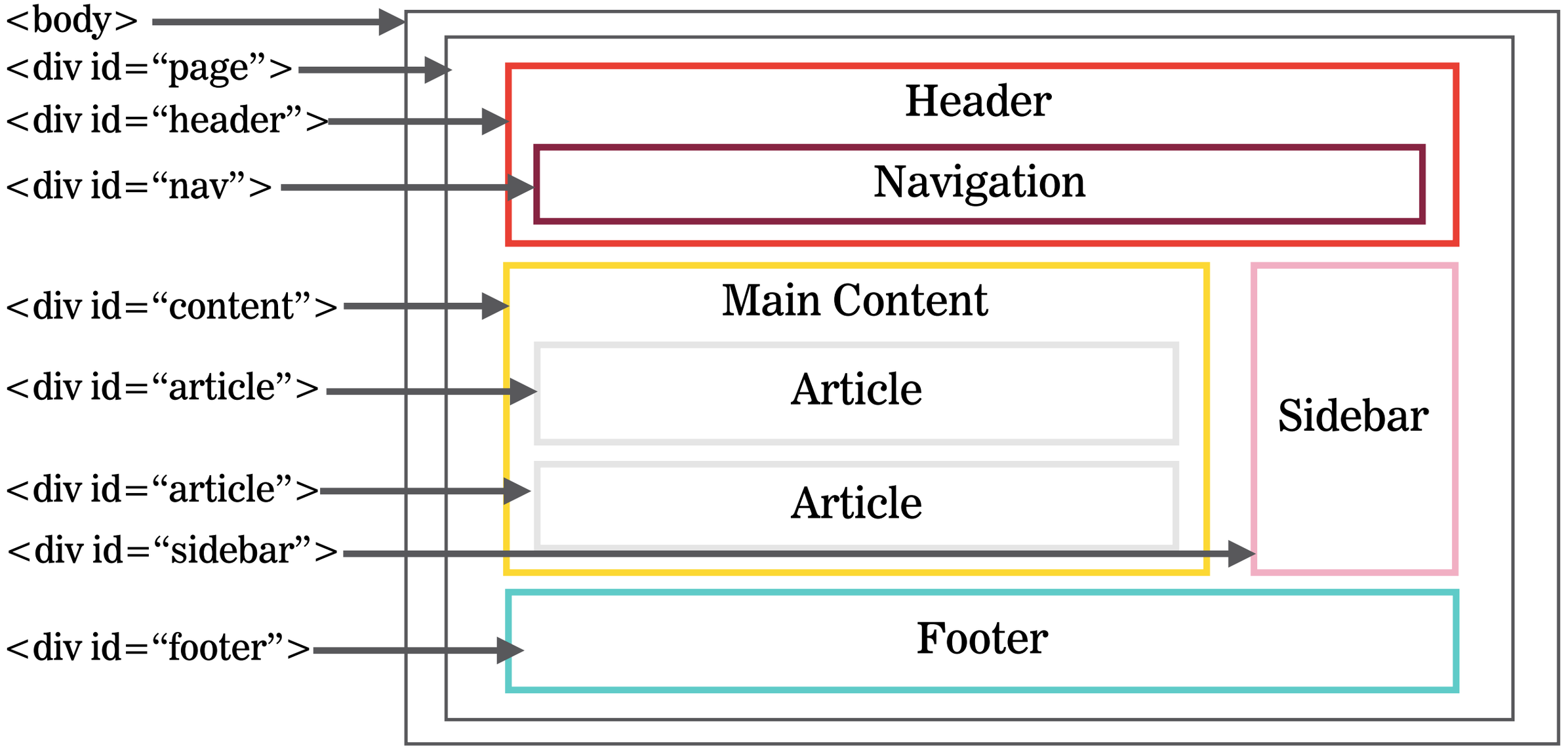
1. Style one little piece of text within a larger paragraph
2. Contain several inline elements

DI[illegible] – SECTIONING OFF PARTS OF A WEBPAGE (OLD WAY OF DOING THINGS)

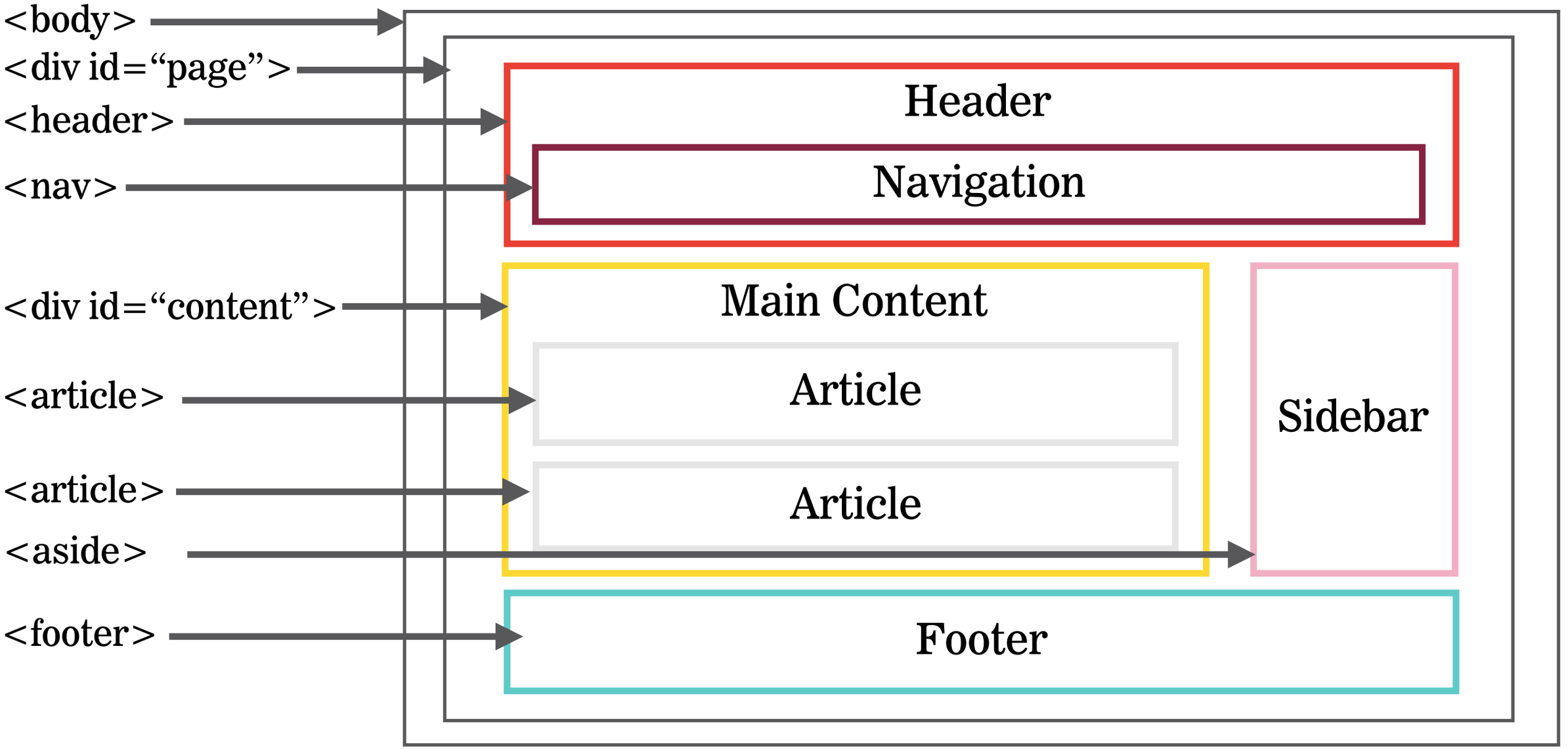


HTML5 STRUCTURAL ELEMENTS

TRADITIONAL HTML LAYOUTS



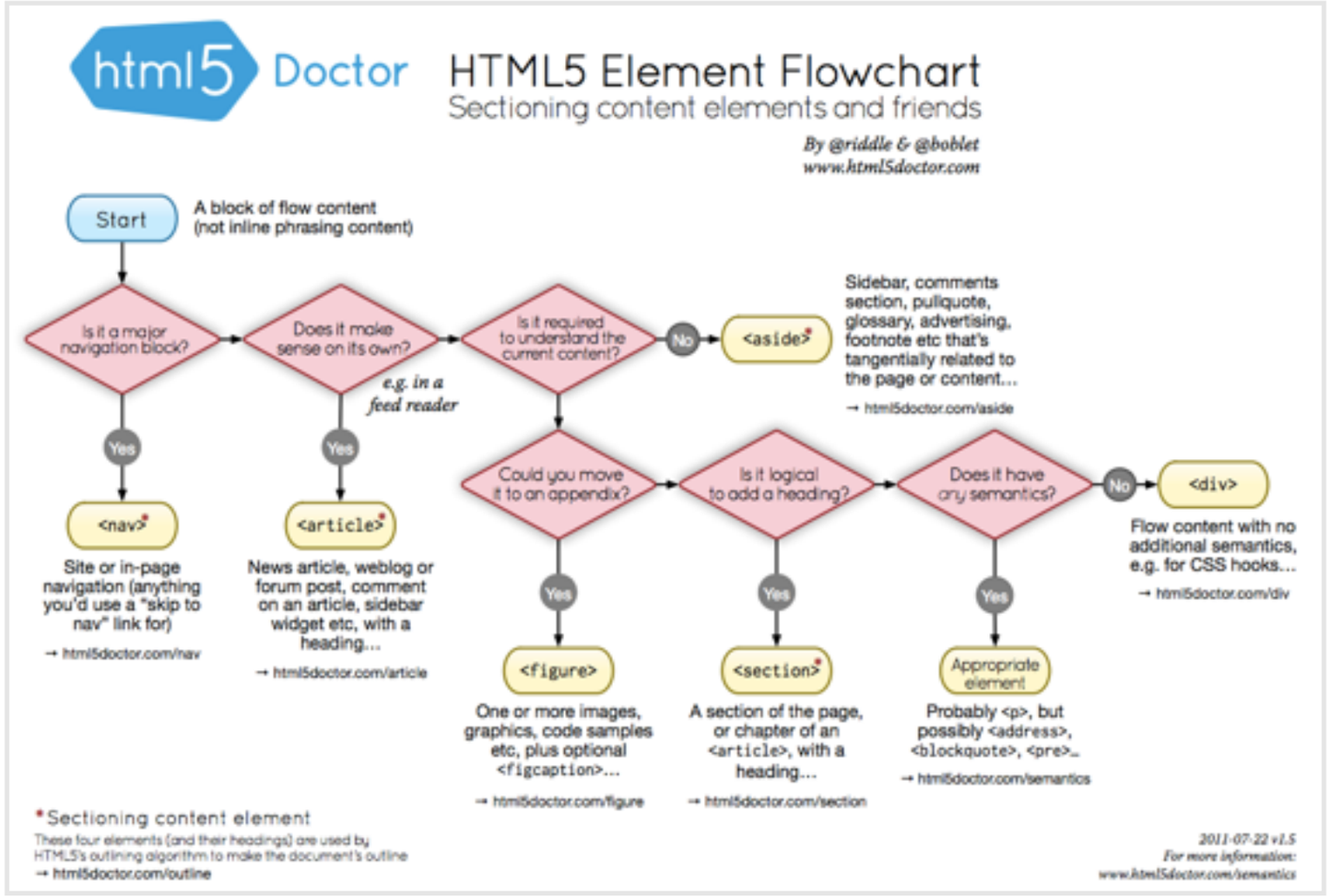
STRUCTURAL ELEMENTS



SO…IS THERE STILL A PLACE FOR DIVS IN AN HTML5 WORLD?

- Yes! The still has a place in the HTML5 world
- You should use when there is **no other more semantically appropriate element that suits your purpose**
- Its most common use will likely be for stylistic purposes — i.e., wrapping some semantically marked-up content in a CSS-styled container.

HTML5 ELEMENT FLOWCHART



ACTIVITY — 'DIV' UP THE CONTENT

KEY OBJECTIVE

- Identify content sections

TYPE OF EXERCISE

- Partner

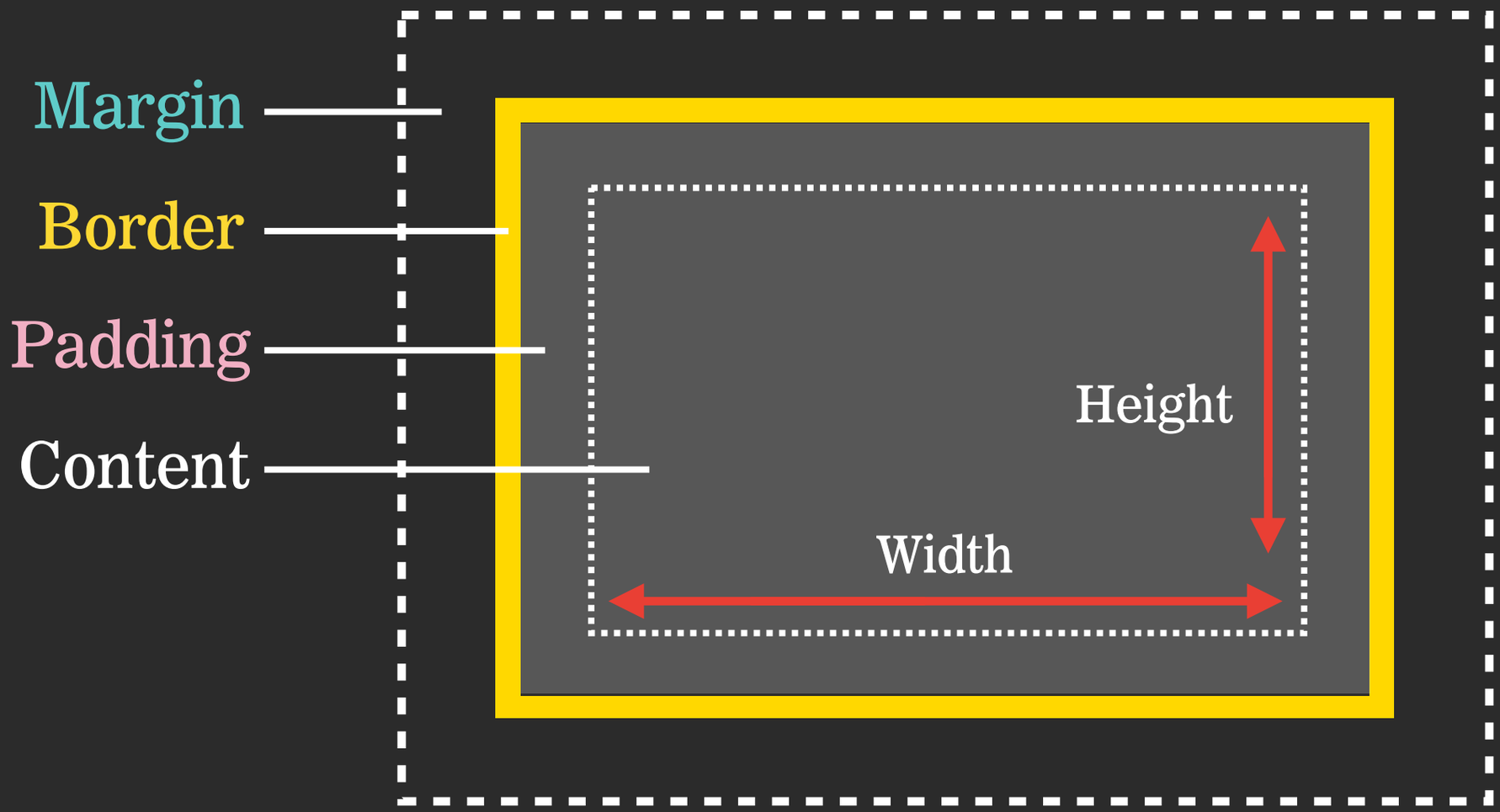
TIMING

8 min

1. First draw boxes around the content you think should live inside a sectioning element — a div, header, footer, etc.
2. Then determine which boxes/divs should have a class or id. Look for similarities to determine what should be a class.

BOX-SIZING FTW!

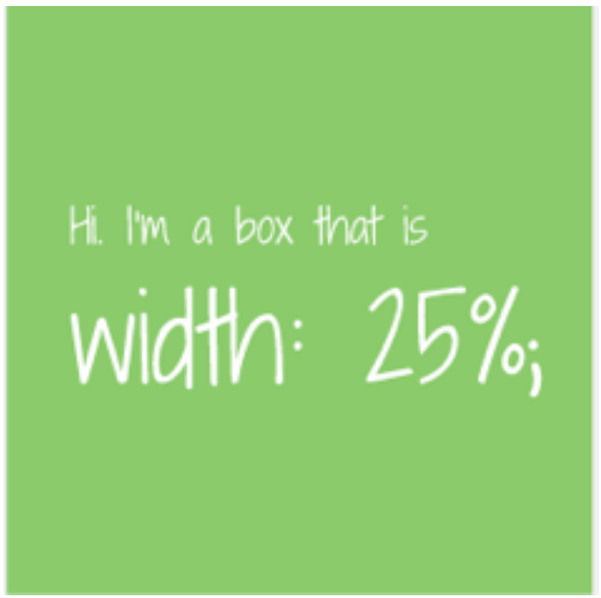
REFRESHER — BOX MODEL



THE DEFAULT WAY — ANNOYING!!

- **Default box-sizing** (box-sizing: content-box): As soon as an element has either padding or border applied, the actual rendered width is wider than the width you set in your CSS.

  Actual width = width + border-left + border-right + padding-left + padding-right



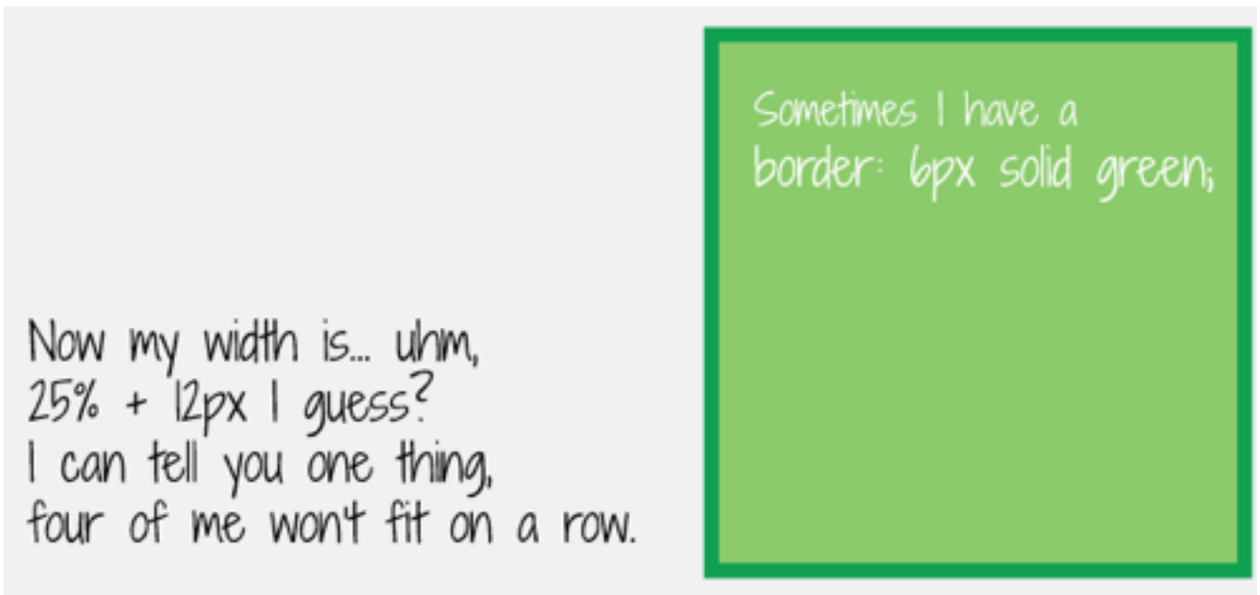
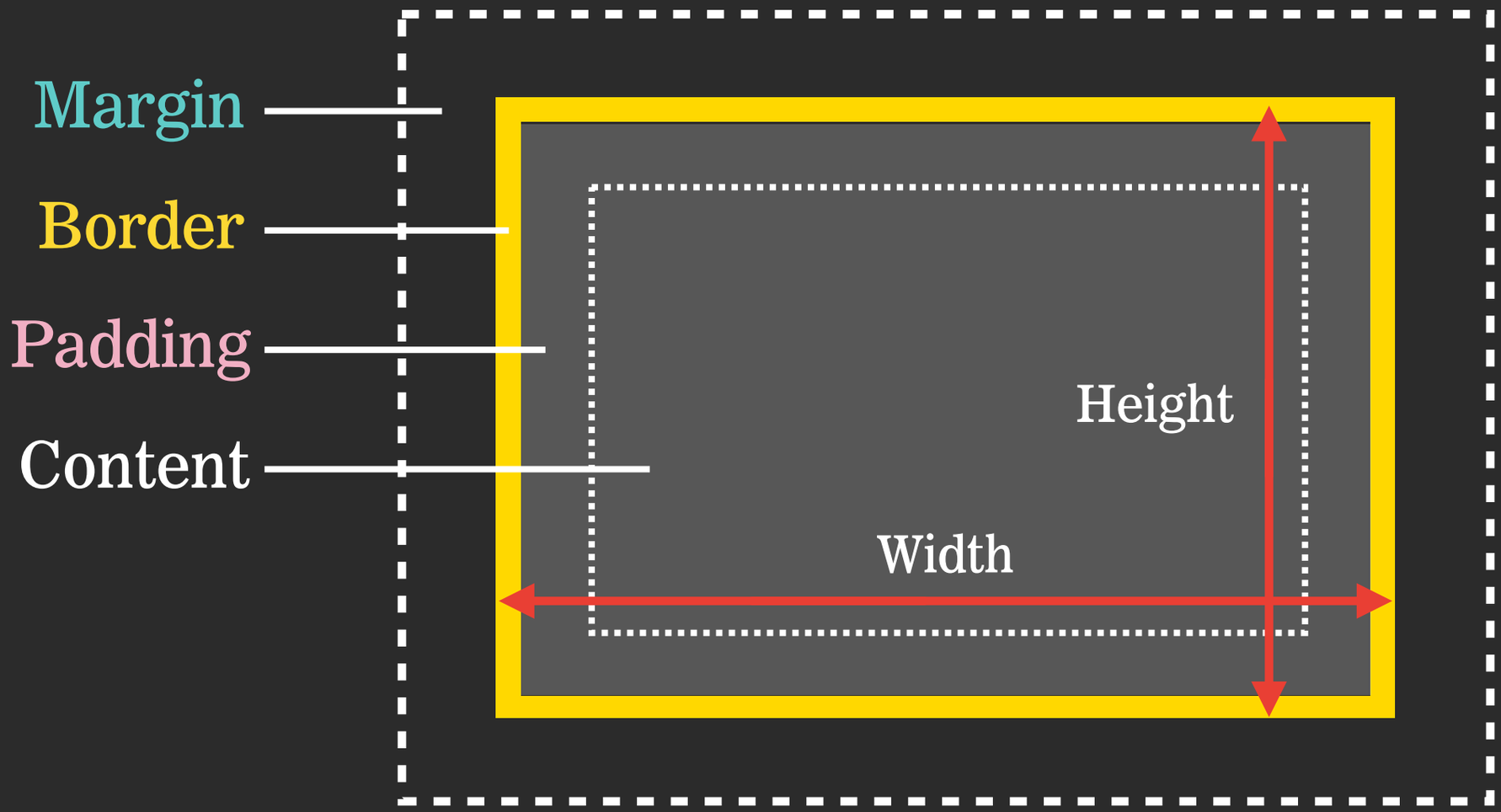


Image credit: Chris Coyier's International Box Sizing Awareness Day

BOX-SIZING: BORDER-BOX



HERE'S THE SYNTAX

```
* {
  box-sizing: border-box;
}
```

WHY IS THIS SO AWESOME?

- With **box-sizing: border-box** — the padding and border press their way inside the box instead of expanding the box.

Actual width = Width set in CSS 

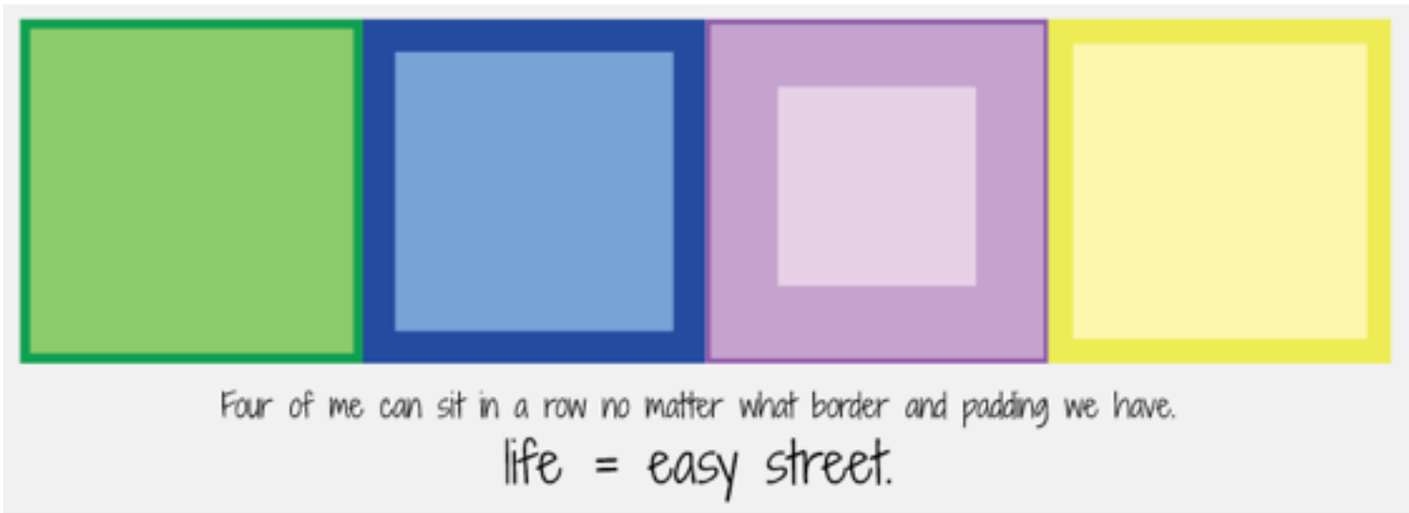
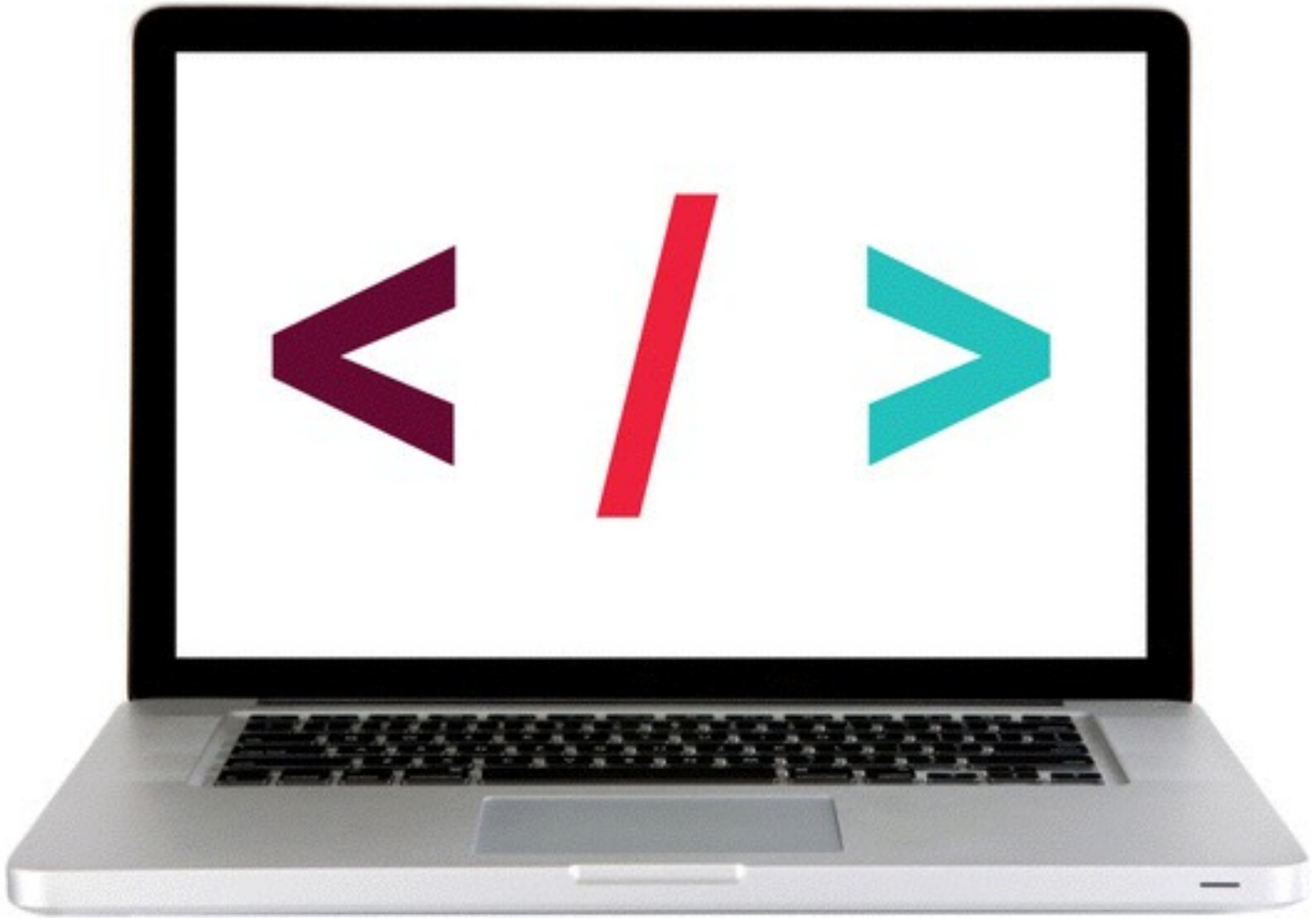


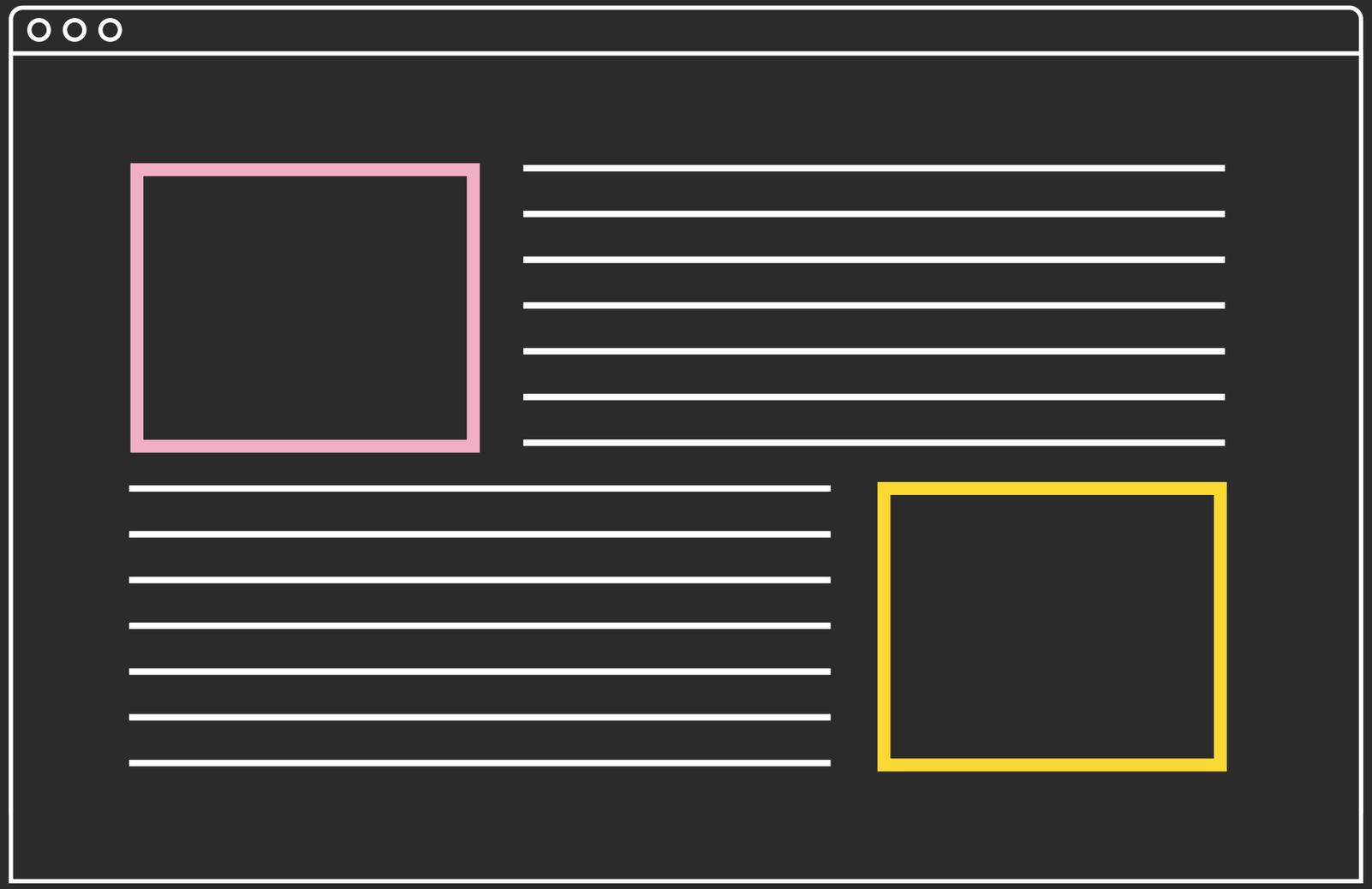
Image credit: Chris Coyier's [International Box Sizing Awareness Day]

LET'S TAKE A CLOSER LOOK



FLOATS

CSS — FLOATS



FLOATS

There are four valid values for the float property:

- **Left** and **Right** float elements those directions respectively
- **None** (the default) ensures the element will not float
- **Inherit** will assume the float value from that elements parent element

CLEARING FLOATS

- The **clear** property specifies which side(s) of an element other floating elements are not allowed

LEFT

- No floating elements allowed on the left side

RIGHT

- No floating elements allowed on the right side

```
.clear {
  clear: both;
}
```

BOTH

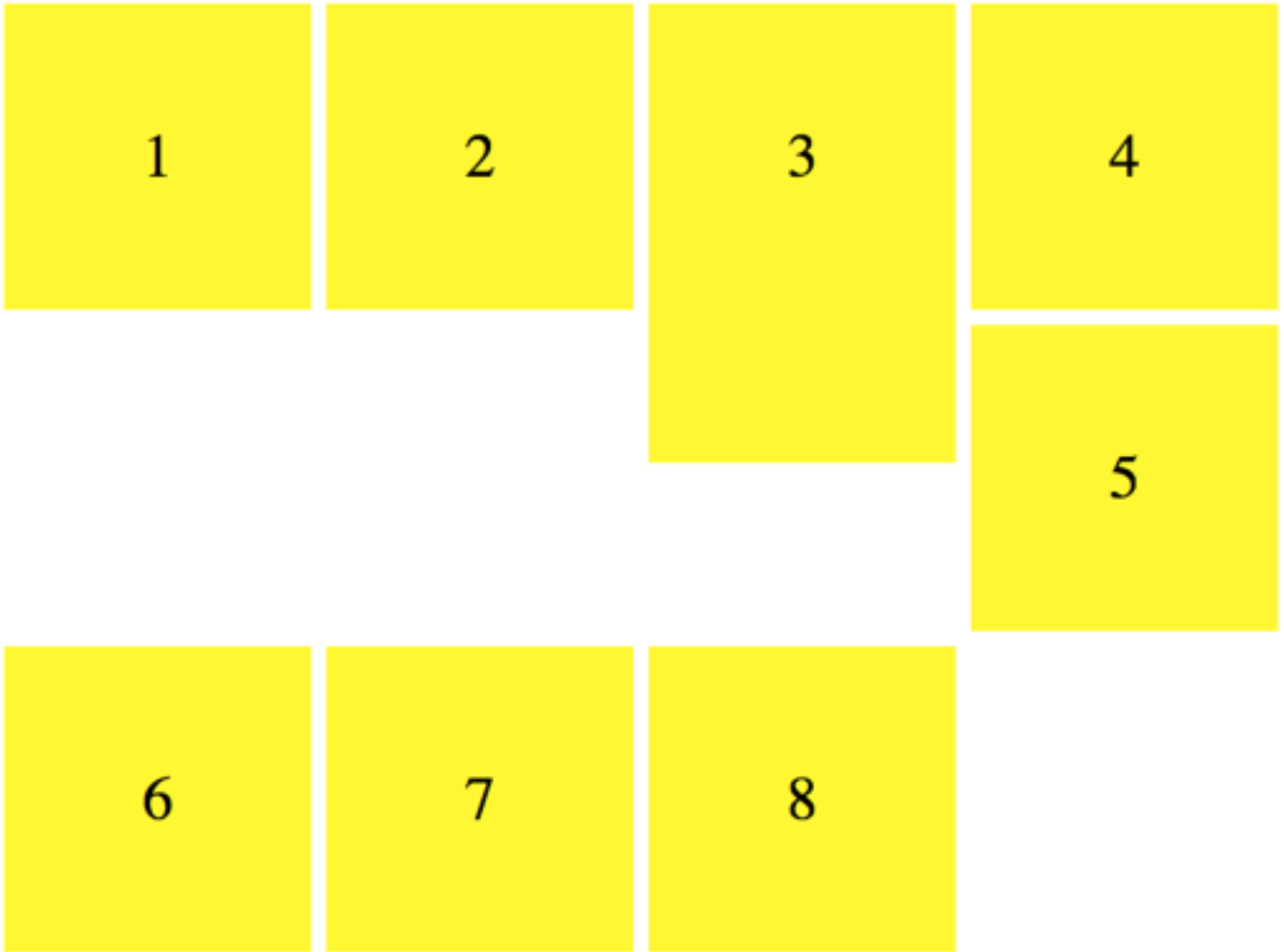
- No floating elements allowed on either the left or right side

NONE

- Allows floating elements on both sides

LET'S TAKE A LOOK

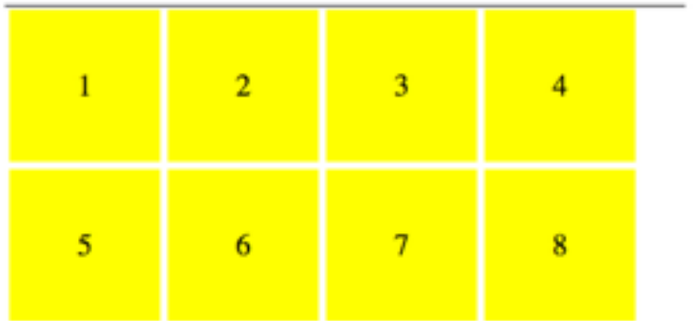
- I've added the example to Codepen so you can refer to it later if needed



PARENTS OF FLOATED ELEMENTS

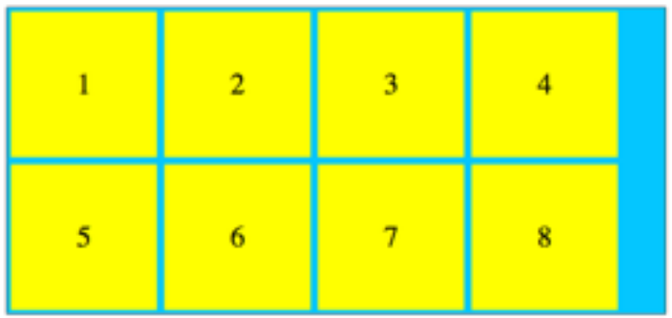
- If a containing element **only contains floated elements**, some browsers will treat it as if it is zero pixels tall.

PROBLEM:



Collapsed parent!

SOLUTION:



PT. 1 — ADD CSS CLASS:

```
.clearfix:after {
  content: "";
  display: table;
  clear: both;
}
```

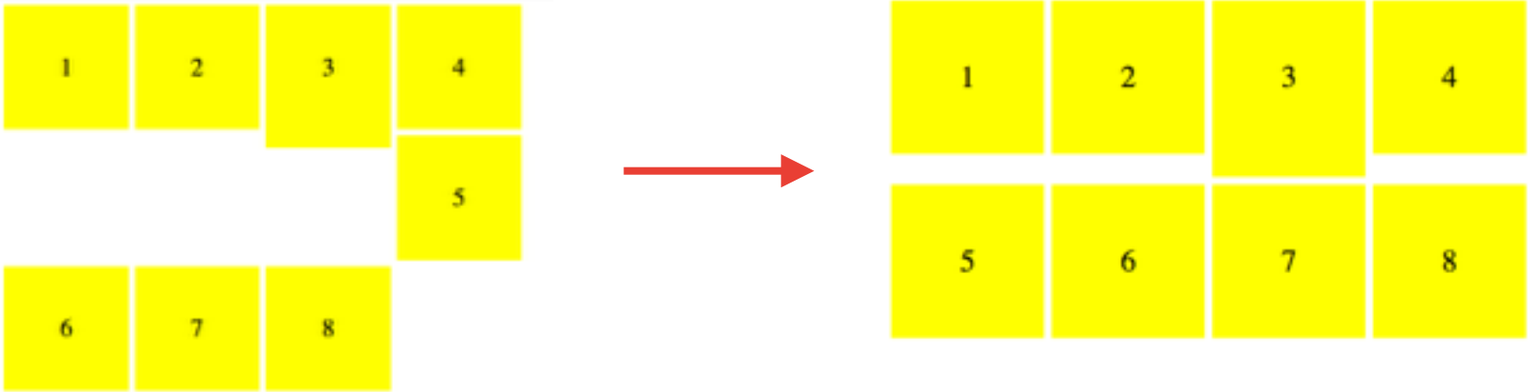
PT. 2 — ADD CLASS TO HTML:

```
<div class="clearfix">
    <p>1</p> <!-- float: left -->
    <p>2</p> <!-- float: left -->
    <p>3</p> <!-- float: left -->
</div>
```

CONFUSING NAMES — KEEPING THINGS STRAIGHT

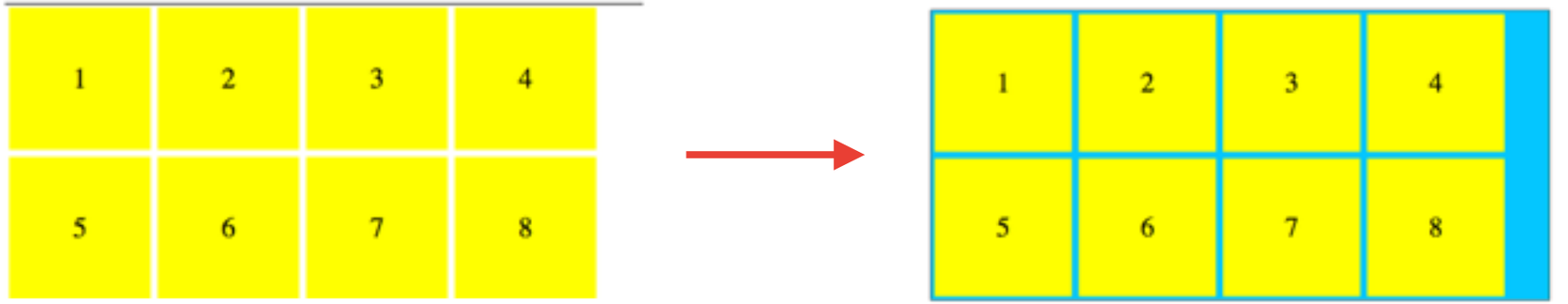
CLEAR: BOTH;

Make sure an element starts on a new line

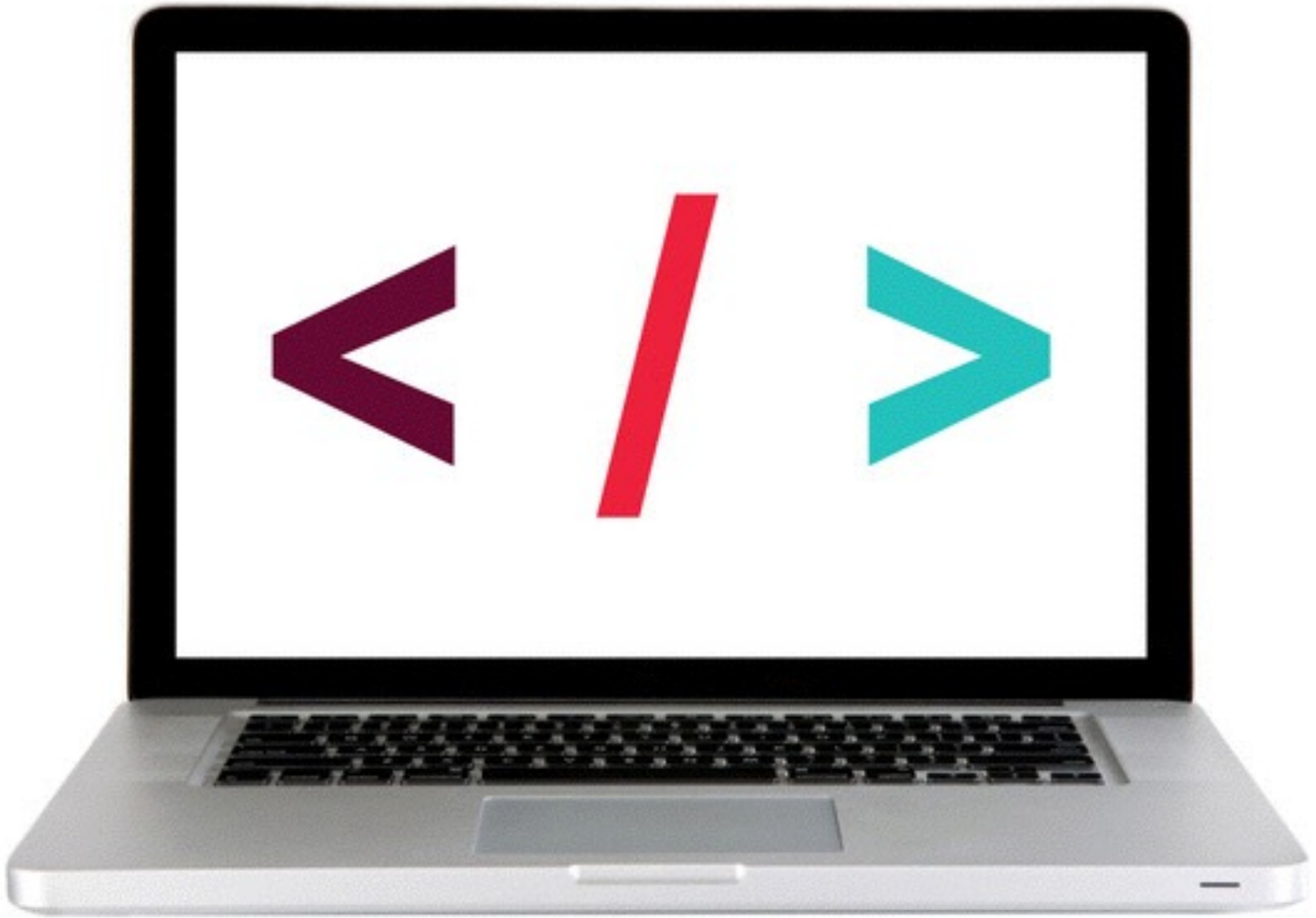


CLEARFIX:

Fixes collapsed parent

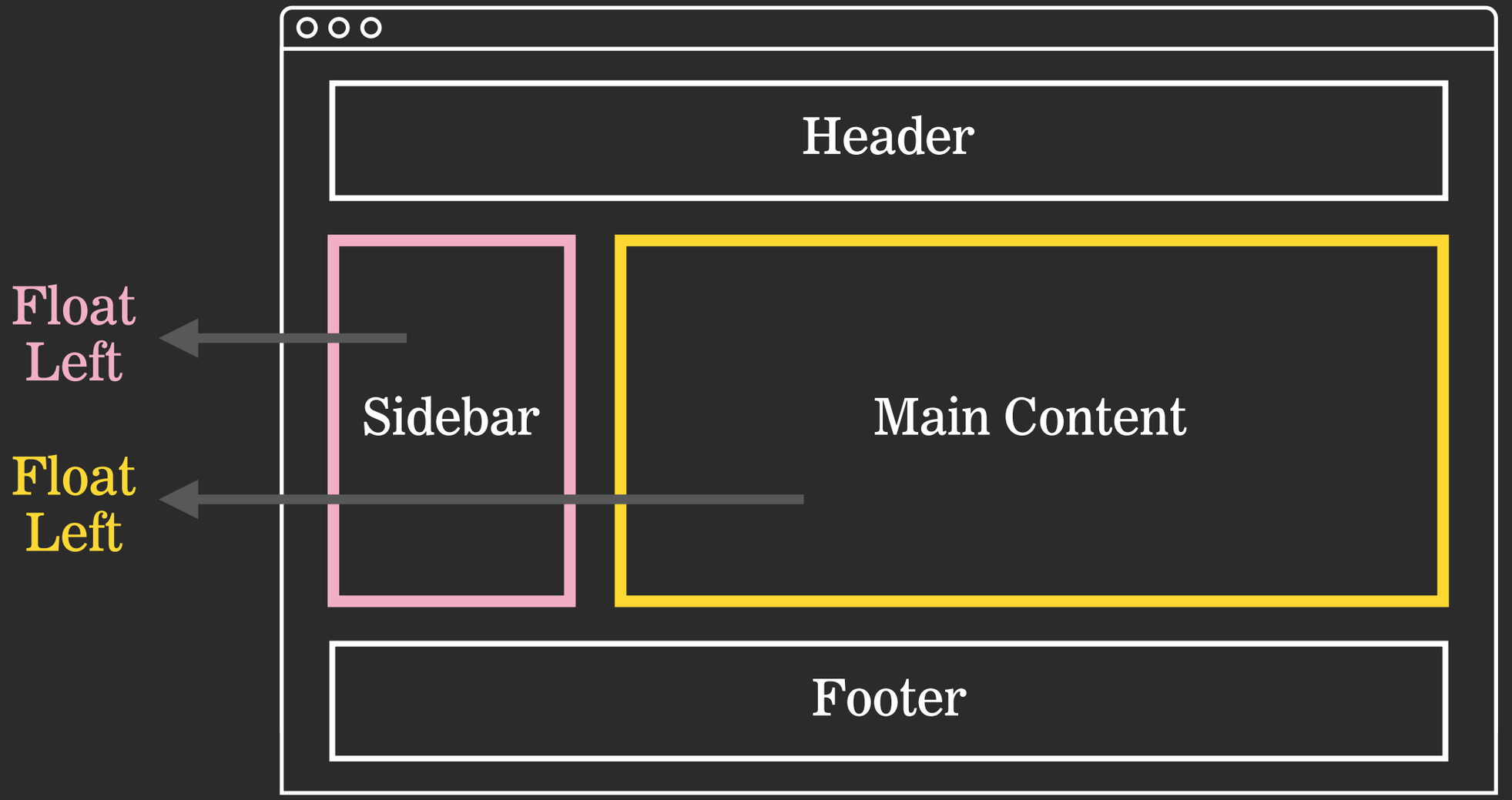


LET'S TAKE A CLOSER LOOK



MULTI-COLUMN LAYOUT

CSS — MULTI-COLUMN LAYOUT

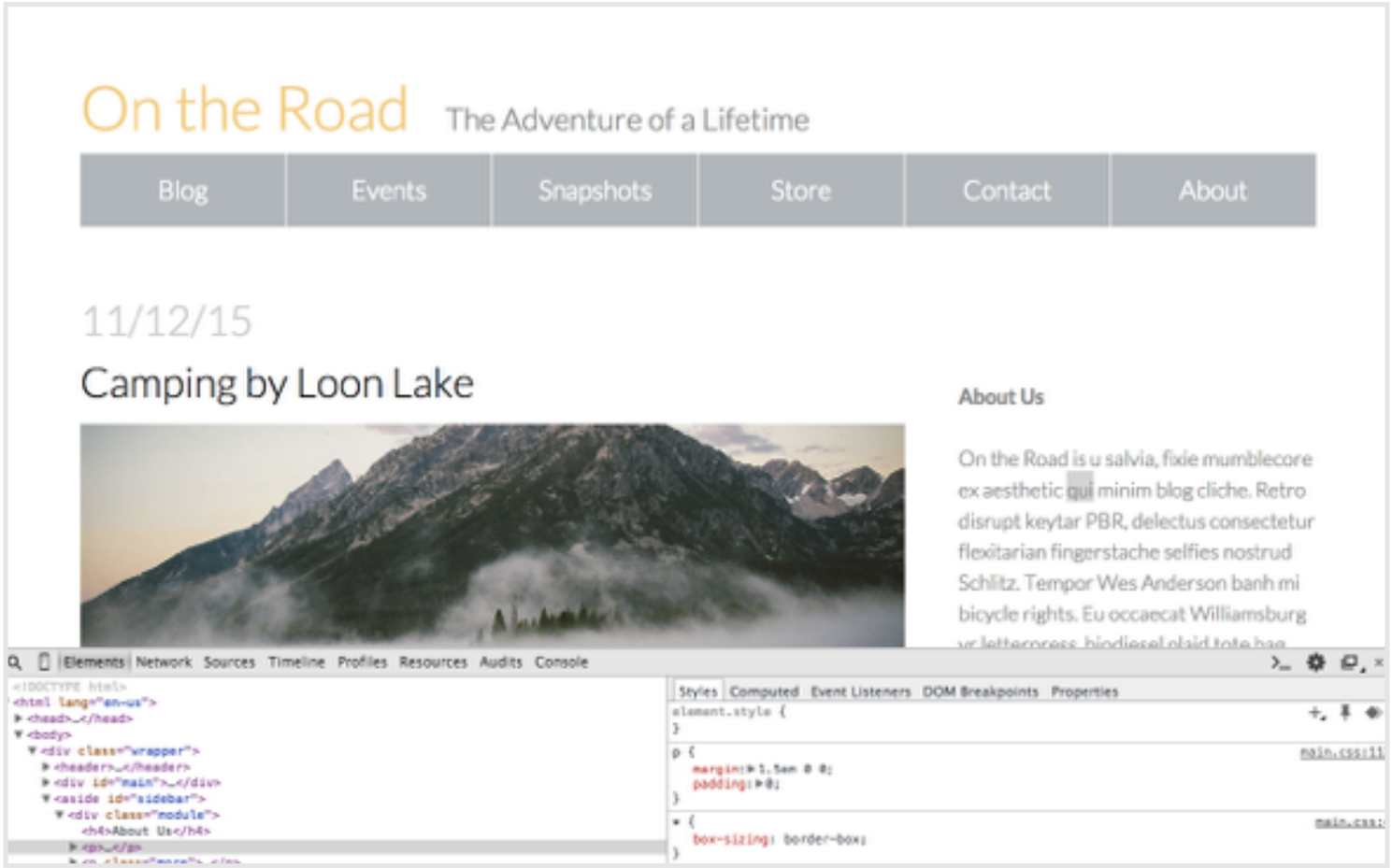


STEPS TO ACHIEVE A MULTI-COLUMN LAYOUT

1. Make sure each column has a wrapper around it in your HTML
2. Give a width to each column (preferably in %)
3. Float each column to left
4. Use padding to add space between columns
5. Add box-sizing: border-box; to everything (use the * CSS selector)
6. Clear anything underneath your columns i.e. a footer using the CSS clear property (clear: both;)

TRAVEL BLOG

LAB — WORKFLOW



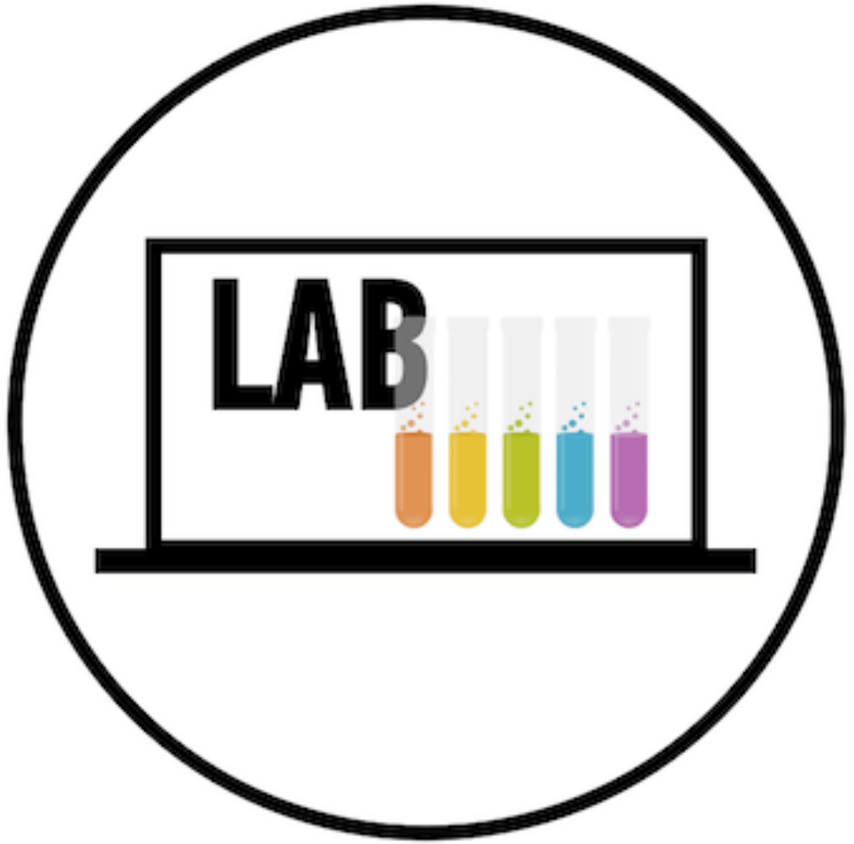
Right click > Inspect Element

LAB — WORKFLOW

Temporarily add a border to everything on the page using the * selector so that you can easily see how all the elements on the page line up.

```
* {
  box-sizing: border-box;
  border: 1px solid black;
}
```

LAB — TRAVEL BLOG PT. 2



ACTIVITY — TRAVEL BLOG

EXERCISE

KEY OBJECTIVE

- Demonstrate the ability to plan and build a website

TYPE OF EXERCISE

- Partner || on your own

TIMING

40 min

1. Recreate the Travel Blog site, using Travel_Blog.png as a reference (in starter_code folder)
2. Use HTML structural tags such as <header>, <aside>, <article> and <footer>

LEARNING OBJECTIVES

- Differentiate between block and inline elements
- Identify when HTML5 structural elements should be used
- Apply header, footer, sidebar, and multi-column layouts to build a web page.
- Experiment and predict effects of floats and clearing CSS positioning.

SNACKS & DESIGN

THURSDAY MAR 3RD

JIM HOWES & JON ILER

(GOOGLE SHEET IS PINNED IN SLACK)

EXIT TICKETS