WEB DESIGN DECAL

LECTURE 3

The CSS Box Model

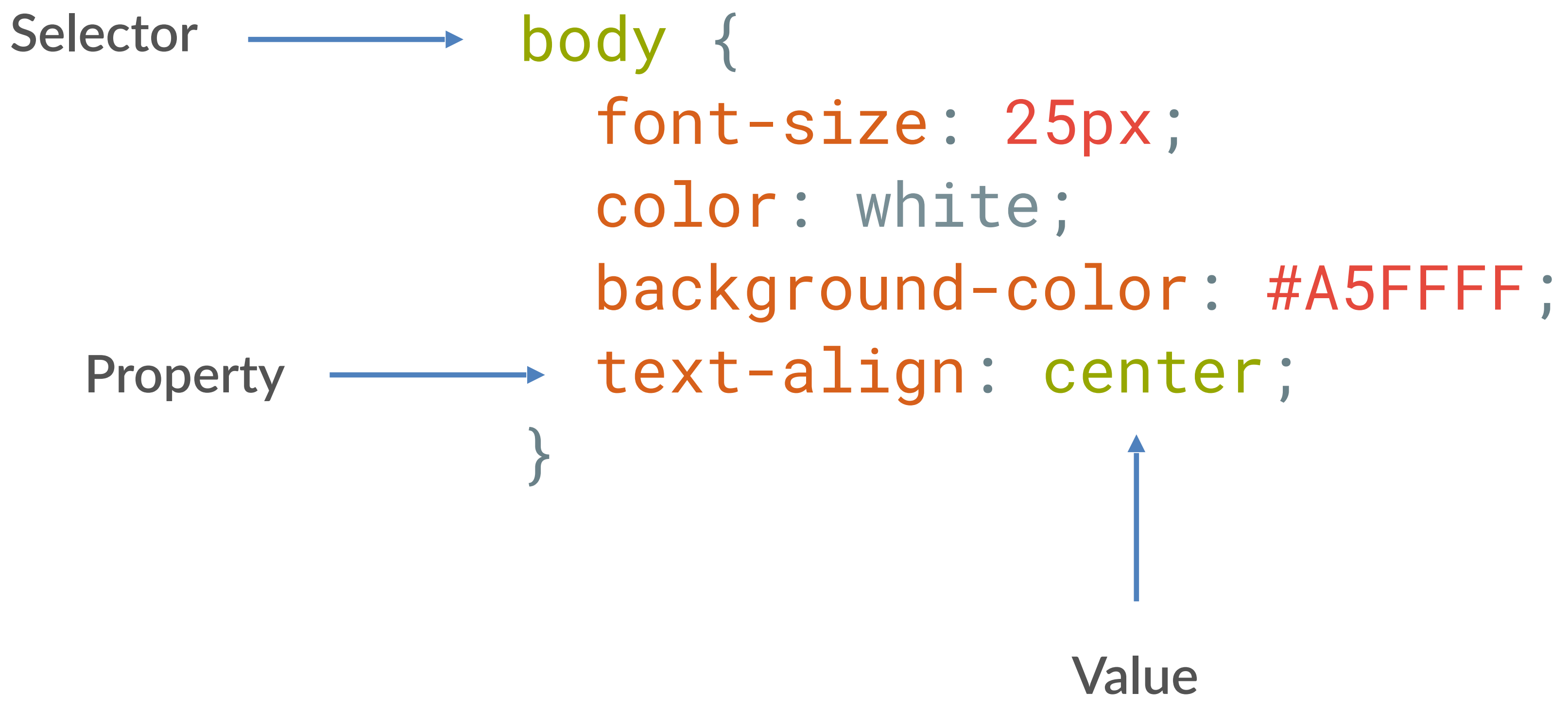
Basic webpage layouts and more complex CSS selectors

Eric Liang

CSS from first principles

We want to make a language that selects elements in HTML and applies rules with certain values.

Anatomy of a CSS rule



```
body {
   font-size: 25px;
   color: white;
   background-color: #A5FFFF;
   text-align: center;
}
```

CSS Selectors

1. Class selectors
2. ID selectors
3. Element selectors
4. Child selectors
5. Adjacent sibling selectors
6. Attribute selectors
7. Pseudo-class selectors
8. Pseudo-element selectors

Each one selects elements differently. Will only cover a few.

CSS Selectors

1. **Class selectors**
2. **ID selectors**
3. **Element selectors**
4. Child selectors
5. Adjacent sibling selectors
6. Attribute selectors
7. Pseudo-class selectors
8. Pseudo-element selectors

Most emphasized in this class. Best practice? Questionable

Class and ID Selectors

Used so we can be specific about what we want to style

ID’s & Classes

Use an id:

Single ID element per page

Use a class:

Multiple classes per page
Use when things are repeated
Think of classes like “templates”

Again: **ID's** use **#'s** in the CSS file!

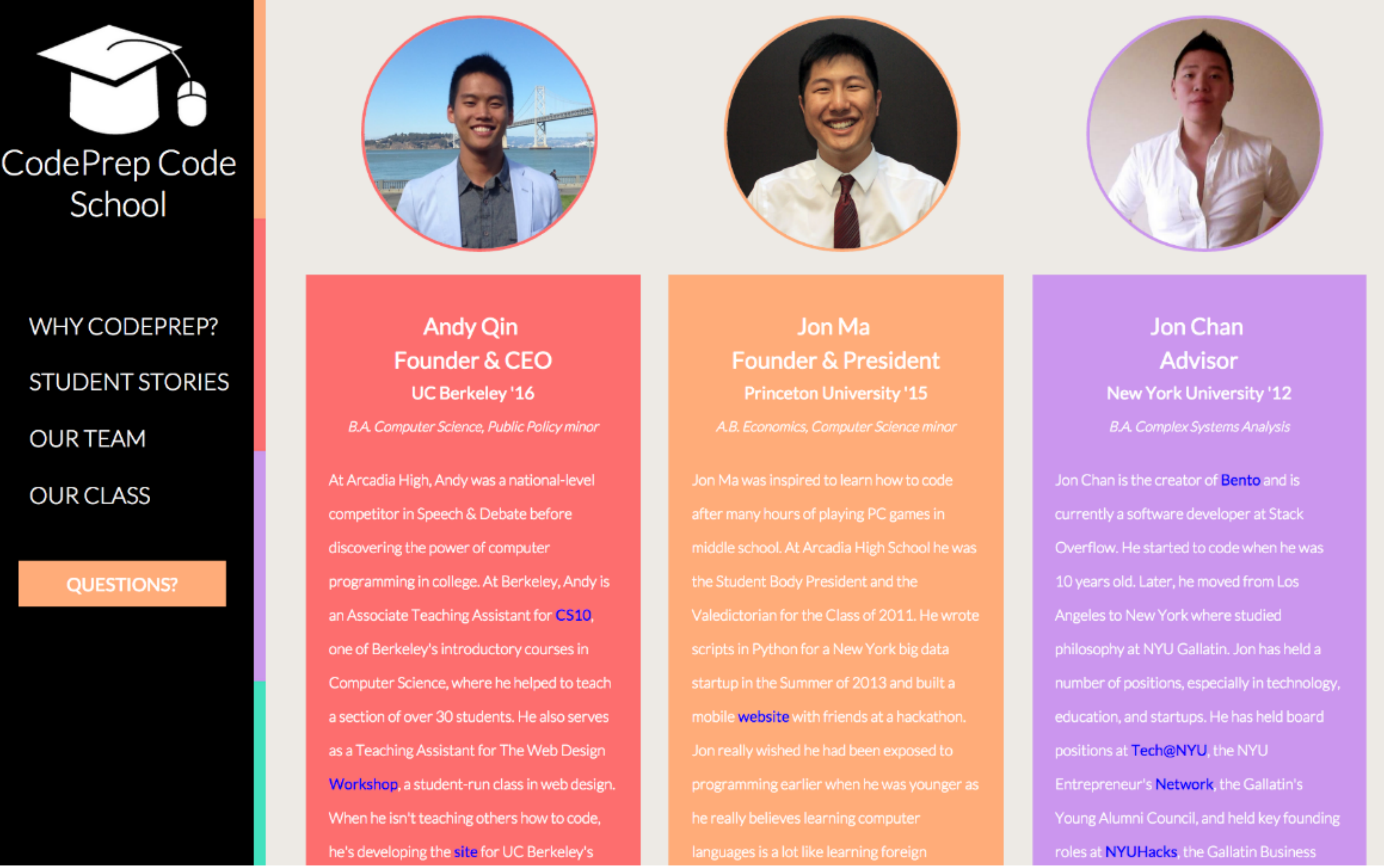
```
<div id=“example”></div>
```

```
#example {
  font-size: 25px;
  color: white;
}
```

Likewise, **Classes** use **.**’s in the CSS!

```
<div class="example"></div>
```

```
.example {
  font-size: 25px;
  color: white;
}
```



CodePrep Code School

WHY CODEPREP?

STUDENT STORIES

OUR TEAM

OUR CLASS

QUESTIONS?

Andy Qin
Founder & CEO
UC Berkeley '16
B.A. Computer Science, Public Policy minor

At Arcadia High, Andy was a national-level competitor in Speech & Debate before discovering the power of computer programming in college. At Berkeley, Andy is an Associate Teaching Assistant for CS10, one of Berkeley's introductory courses in Computer Science, where he helped to teach a section of over 30 students. He also serves as a Teaching Assistant for The Web Design Workshop, a student-run class in web design. When he isn't teaching others how to code, he's developing the site for UC Berkeley's

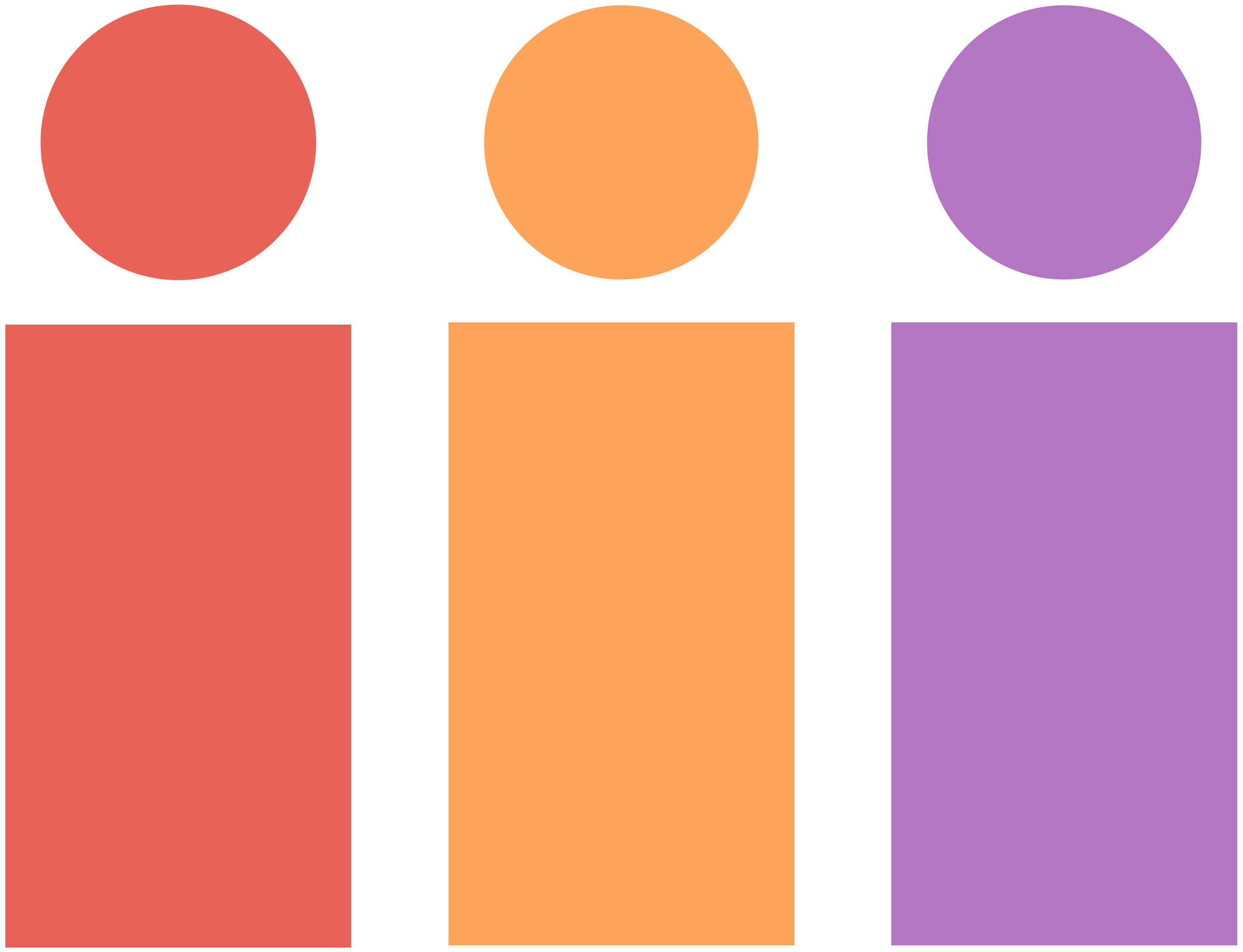
Jon Ma
Founder & President
Princeton University '15
A.B. Economics, Computer Science minor

Jon Ma was inspired to learn how to code after many hours of playing PC games in middle school. At Arcadia High School he was the Student Body President and the Valedictorian for the Class of 2011. He wrote scripts in Python for a New York big data startup in the Summer of 2013 and built a mobile website with friends at a hackathon. Jon really wished he had been exposed to programming earlier when he was younger as he really believes learning computer languages is a lot like learning foreign

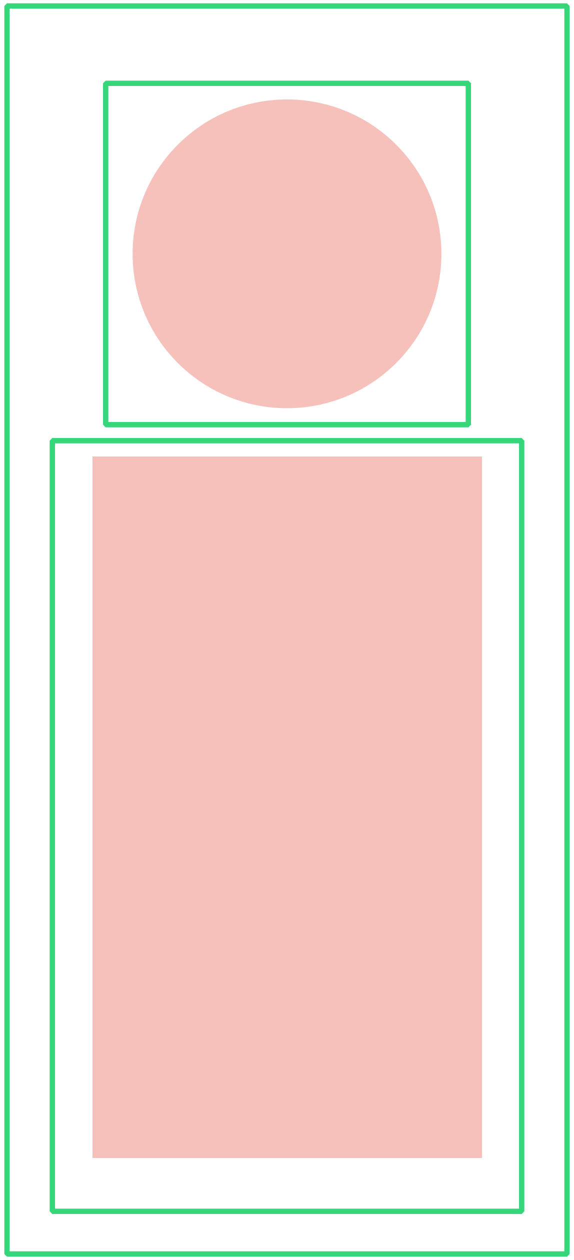
Jon Chan
Advisor
New York University '12
B.A. Complex Systems Analysis

Jon Chan is the creator of Bento and is currently a software developer at Stack Overflow. He started to code when he was 10 years old. Later, he moved from Los Angeles to New York where studied philosophy at NYU Gallatin. Jon has held a number of positions, especially in technology, education, and startups. He has held board positions at Tech@NYU, the NYU Entrepreneur's Network, the Gallatin's Young Alumni Council, and held key founding roles at NYUHacks, the Gallatin Business

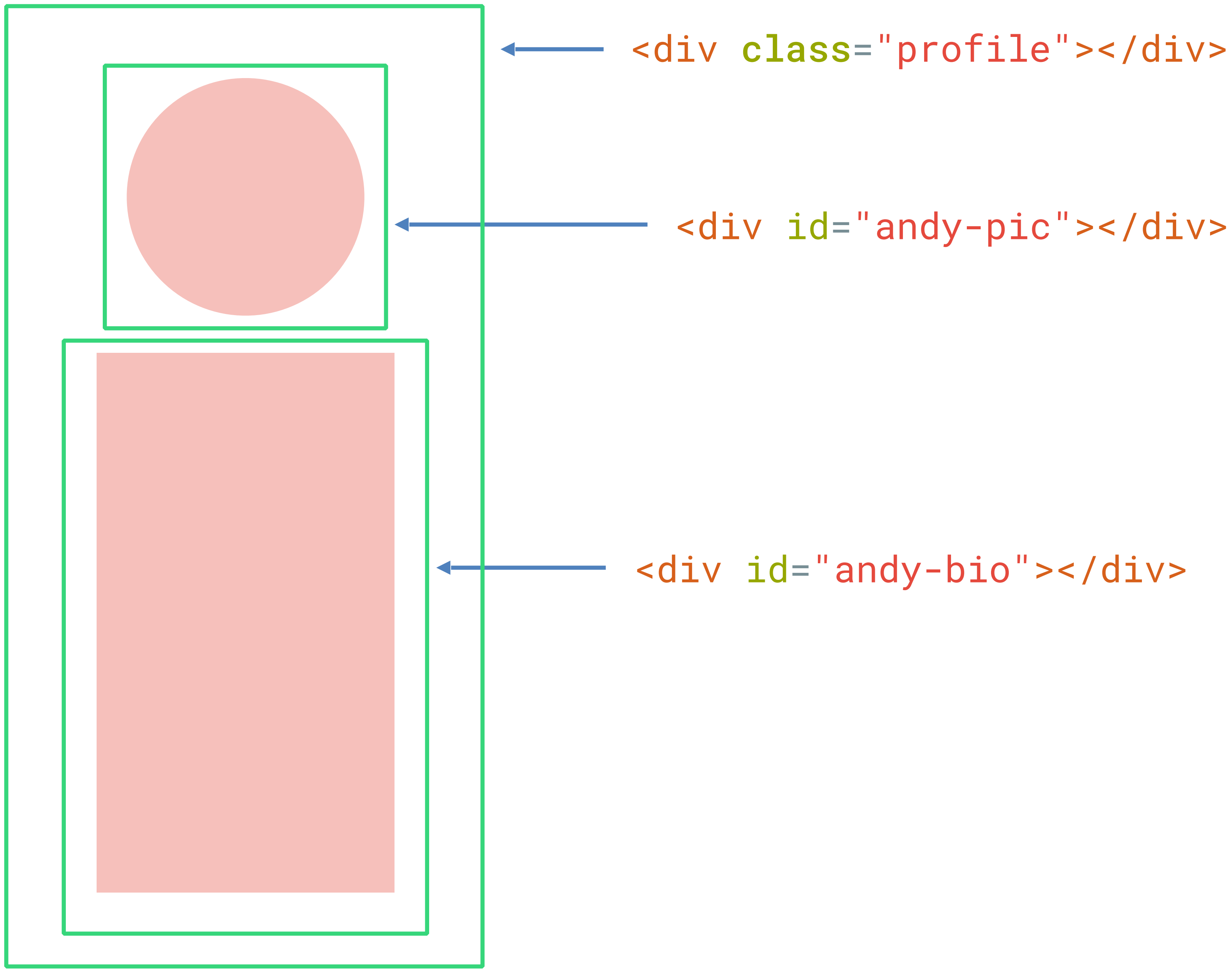
How would I "stack" these profiles?



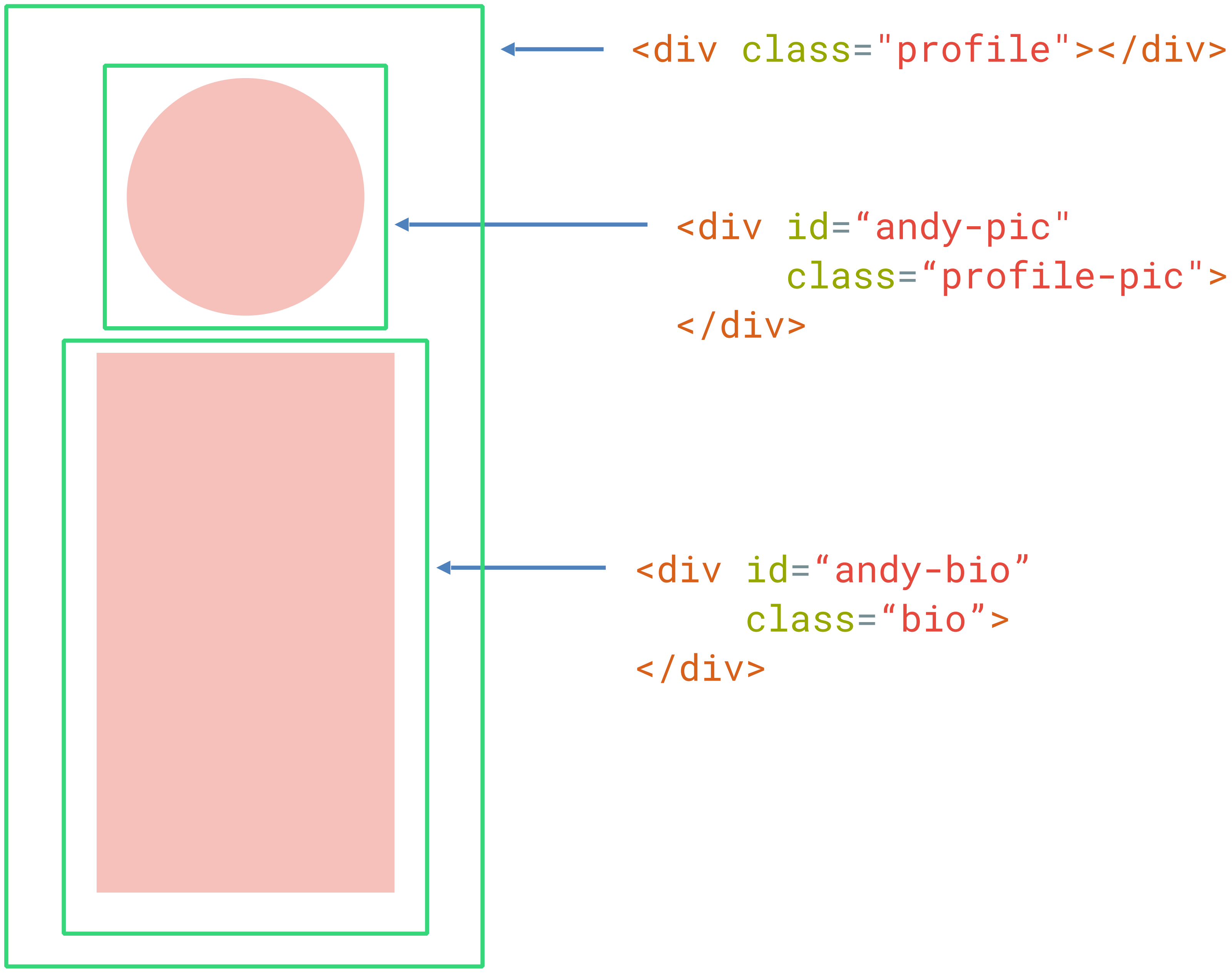
i



Which should be id’s and which should be classes?



```html
<div class="profile"></div>
<div id="andy-pic"></div>
<div id="andy-bio"></div>
```



```
<div class="profile"></div>
```

```
<div id=“andy-pic"
     class=“profile-pic">
</div>
```

```
<div id=“andy-bio”
     class=“bio”>
</div>
```

ID’s & Classes: Profile Page

Use an id:

Differentiate unique elements

- The color of the profile

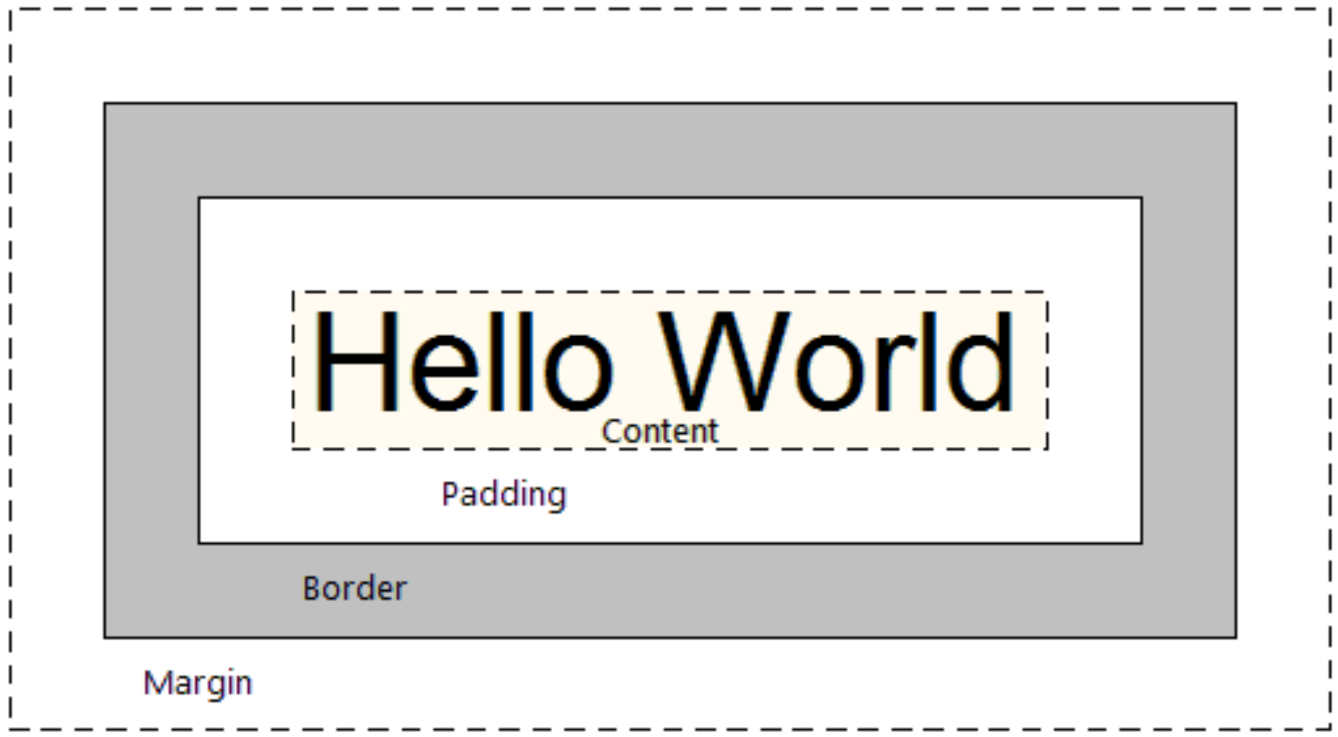
Use a class:

Reuse the code!

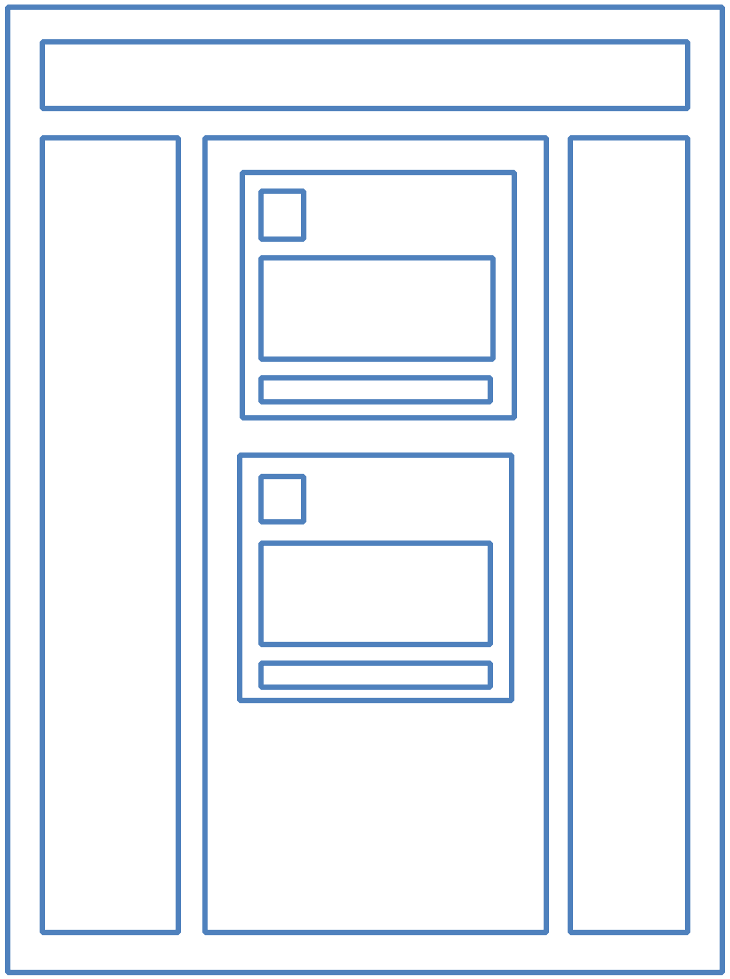
Essentially everything else:

- Circularity of images
- Formatting of the biography
- The margin between image & bio

This way of structuring your elements is the CSS Box Model.



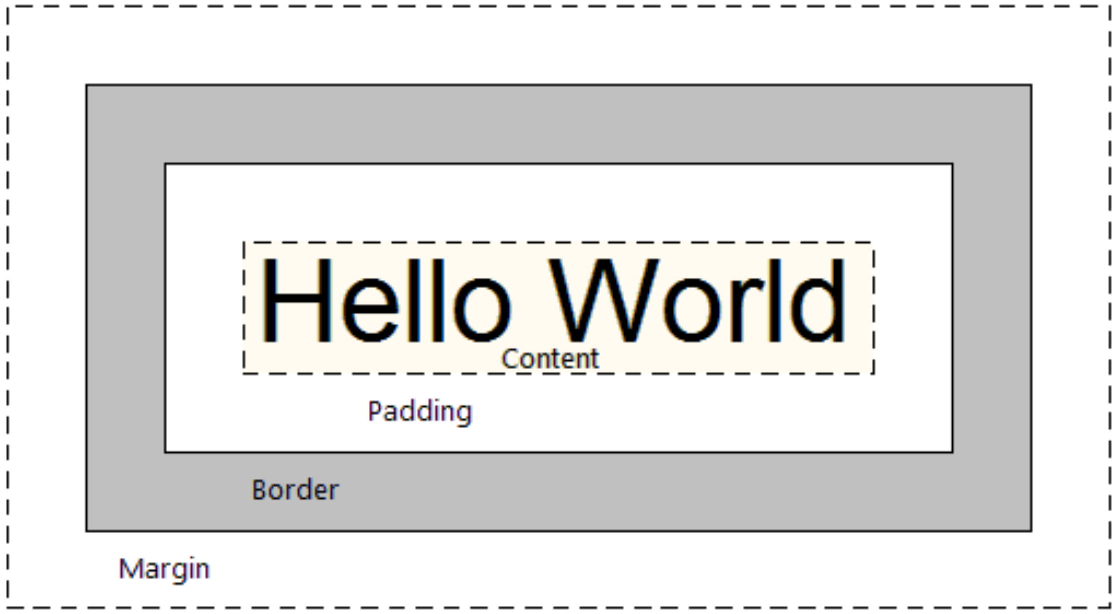
For example: Facebook's News Feed



The CSS Box Model

4 Key Components

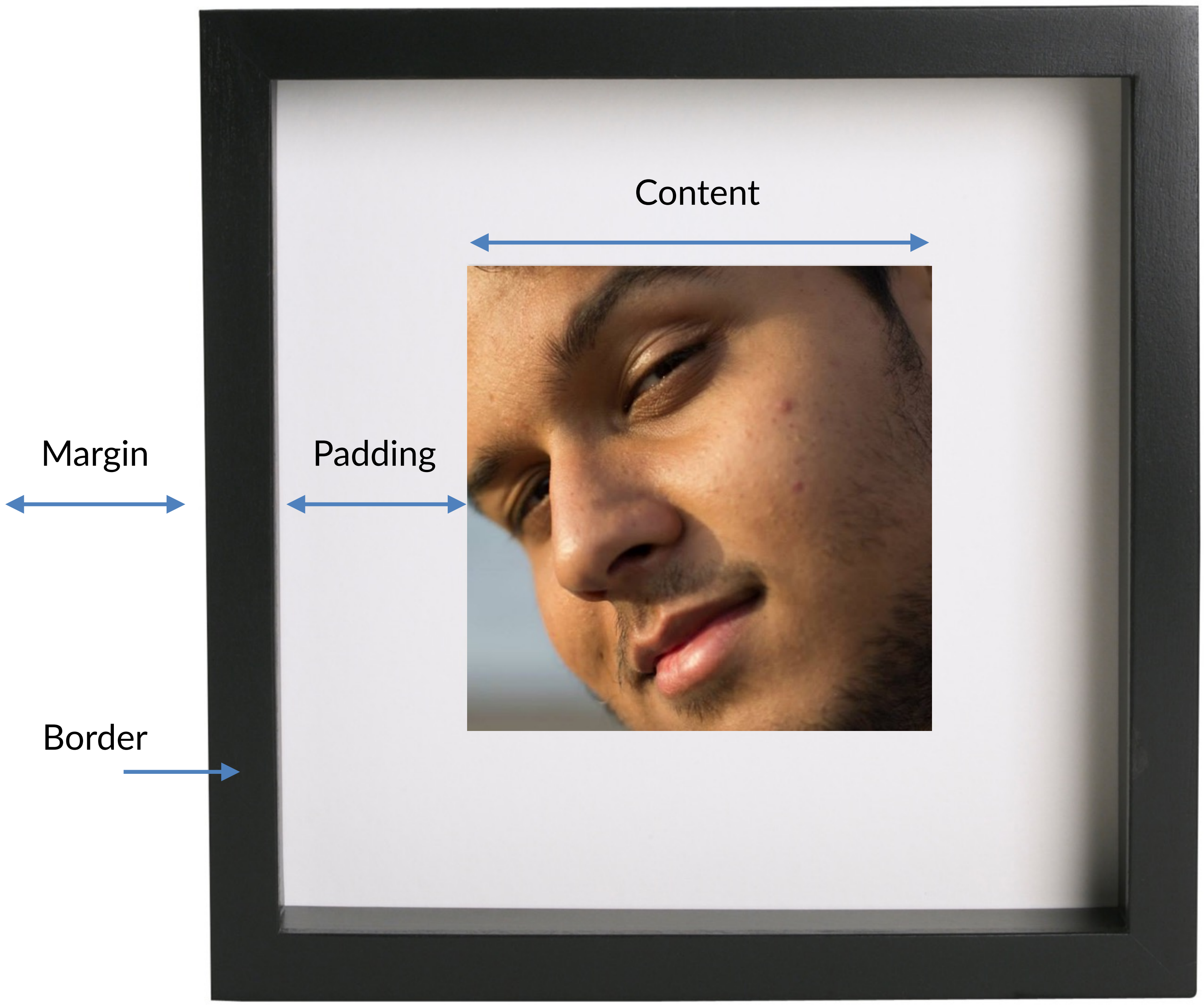
- Content
- Padding
- Border
- Margin



The CSS Box Model

4 Key Components

- **Content** - inside the element
- **Padding** - between the content and border
- **Border** - surrounds the content
- **Margin** - space between border and nearby elements


Content
Margin
Padding
Border

The CSS Box Model - Border

Border

Takes in 3 values: width, style, color

```
  border: 1px solid #AAA;
```

Can be declared separately

```
   border-width: 1px;
   border-style: solid;
   border-color: #AAA;
```

The CSS Box Model - Padding & Margin

Padding and Margin

```css
padding: 10px;
```

10px padding to all four sides

```css
padding: 10px 5px;
```

10px of padding above and below
5px of padding left and right

The CSS Box Model - Padding & Margin

Padding and Margin

```
padding: 10px;
```

10px padding to all four sides

```
padding: 10px 5px;
```

10px of padding above and below
5px of padding left and right

```
padding: 10px 5px 15px;
```

apply in the order:
top, left & right, bottom

```
padding: 10px 5px 15px 20px;
```

apply clockwise:
top, right, bottom, left

CSS Box Model

Currently our divs only stack vertically. Why?

Divs are default “blocks”:

- Take up the entire row
- By default you can’t stack ‘em side-by-side:

CSS Box Model - Display

To stack ‘em side-by-side, alter the `display` property

Display has 3 important values: `block`, `inline`, `inline-block`

`block`: respects margins, paddings, but has auto-line break (after every block element, there is an automatic new line)

CSS Box Model - Display

To stack ‘em side-by-side, alter the `display` property

Display has 3 important values: `block`, `inline`, `inline-block`

`block`: respects margins, paddings, but has auto-line break (after every block element, there is an automatic new line)

CSS Box Model - Display

To stack ‘em side-by-side, alter the `display` property

Display has 3 important values: `block`, `inline`, `inline-block`

`block`: respects margins, paddings, but has auto-line break

`inline-block`: similar to block, but allows you to stack horizontally

CSS Box Model - Display

To stack ‘em side-by-side, alter the `display` property

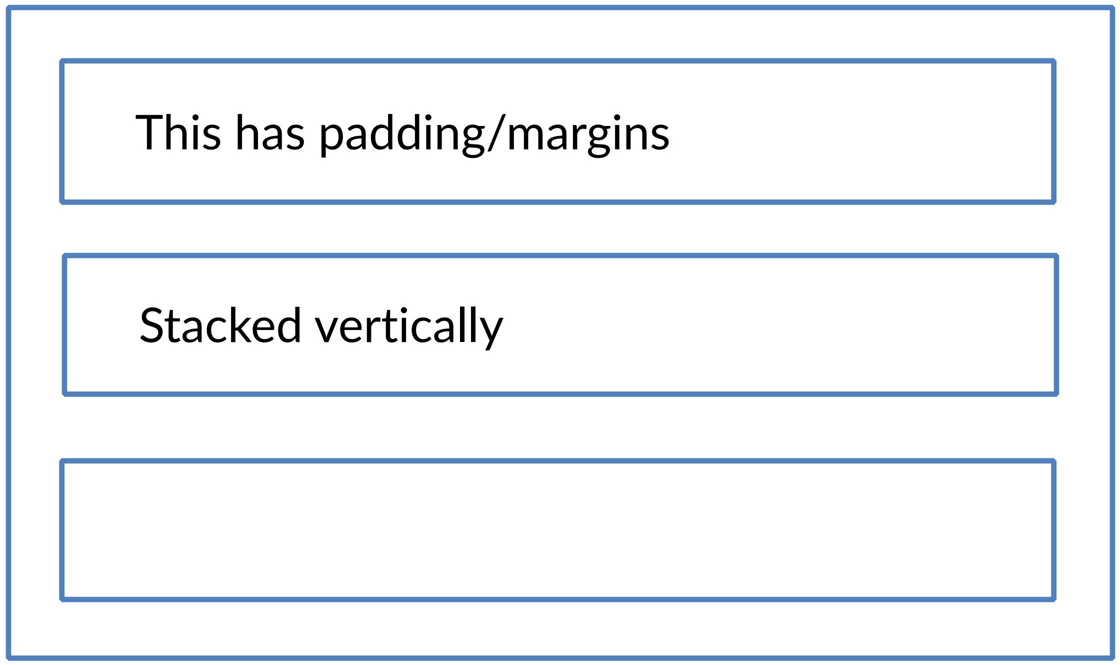
Display has 3 important values: `block`, `inline`, `inline-block`

`block`: respects margins, paddings, but has auto-line break

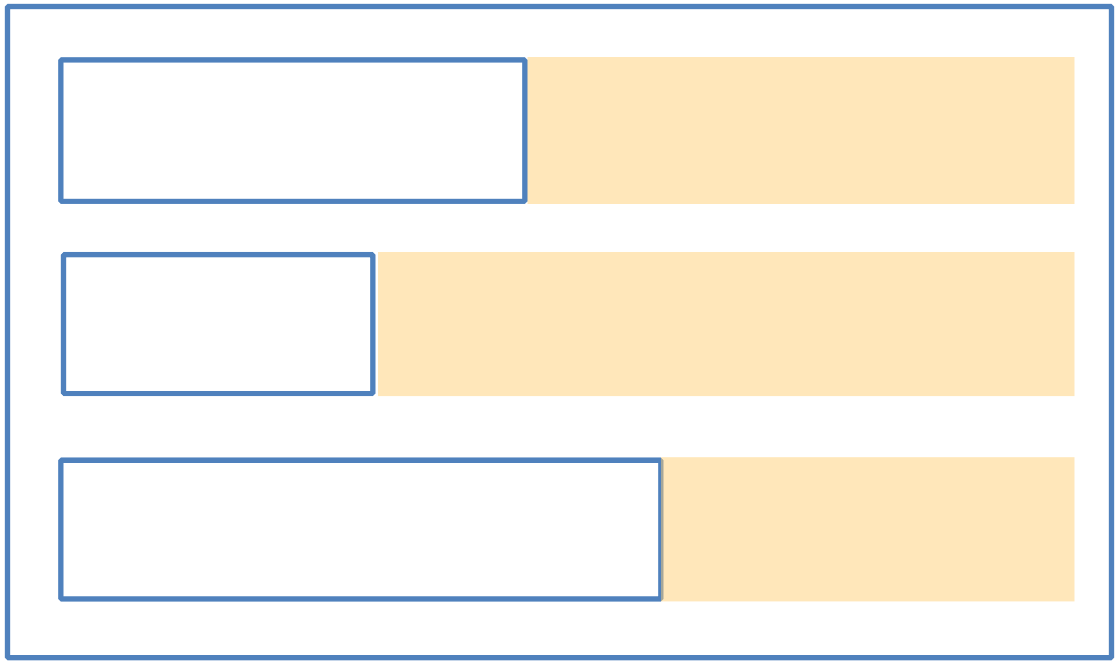
`inline-block`: similar to block, but allows you to stack horizontally

`inline`: allows you to stack horizontally, but you can’t control the height or margin

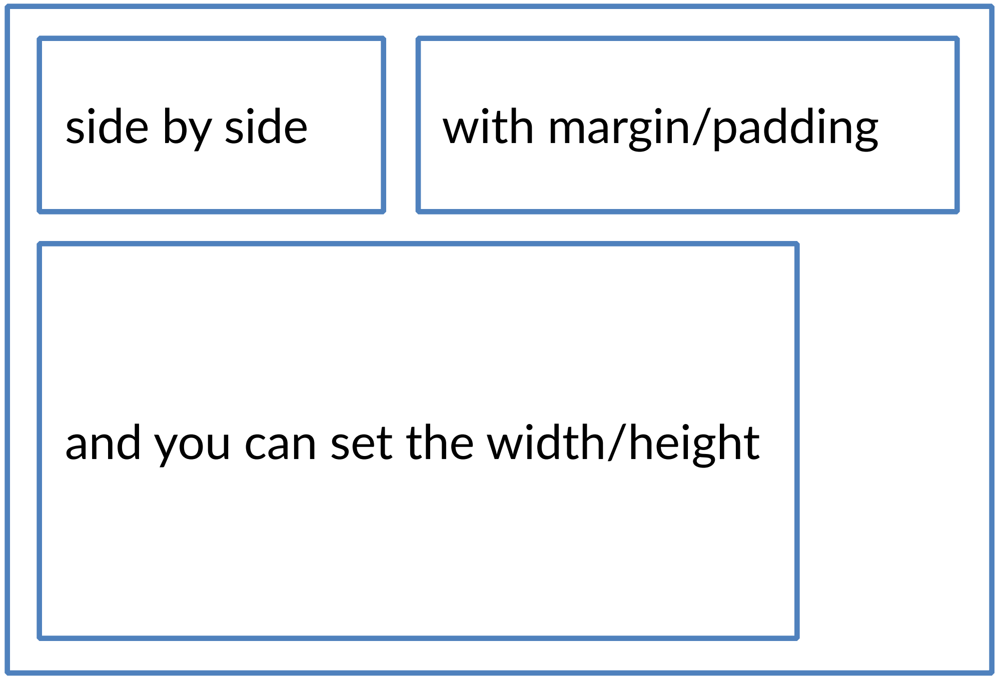
CSS Box Model - Block Example



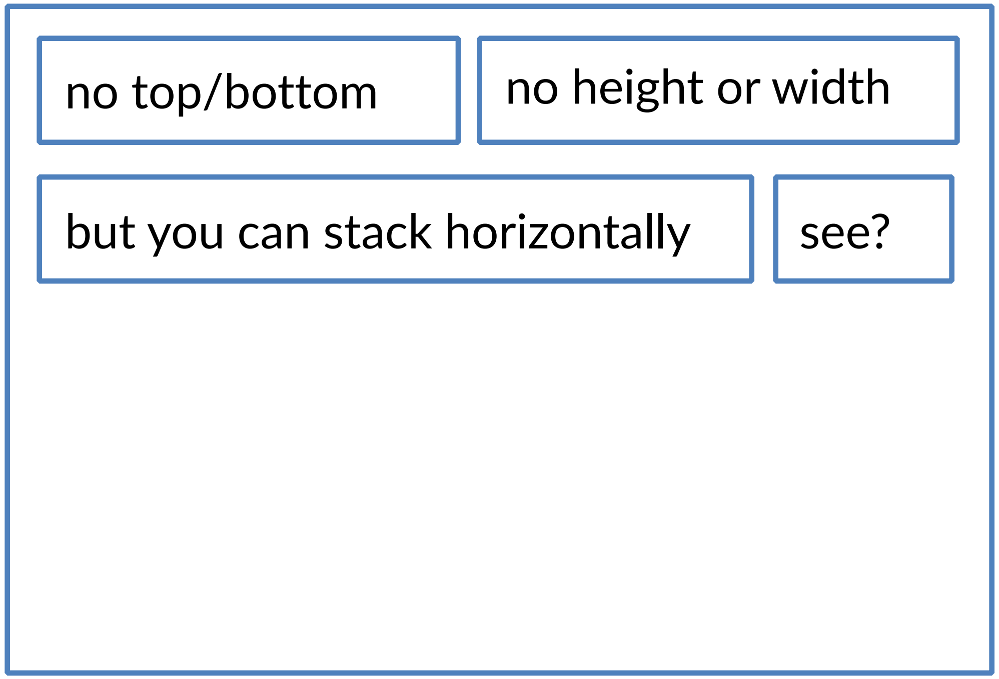
CSS Box Model - Block Example



CSS Box Model - Inline-Block Example



CSS Box Model - Inline Example



Essentially a `<span>`!

Switching gears now to talk about CSS selectors. Any questions?

Different types of CSS selectors

In CSS, there's way more than `.class` and `#id`, for instance:

```
element1 element2 {
    /* Selects all element2 elements in all element 1's */
}

div p {
    /* Selects all <p> elements in all <div> elements*/
}
```

Different types of CSS selectors

A space makes the difference!
Choosing elements with both classes vs. something else

```
.class1 .class2 {
   /* Selects elements of class2 whose parents are class1 */
}

.class1.class2 {
   /* Selects elements of BOTH class1 and class2. */
}
```

Different types of CSS selectors

Pseudo-classes react to various page events.
For example, hovering over a certain element:

```
#test:hover {
  /* Styles elements with id="test" when user hovers */
}

#popup:hover {
  /* Styles elements with id="popup" when user hovers */
}
```

Different types of CSS selectors

Finally, nth-child, which is a little tricker.

```
element1:nth-child(n) {
   /* Selects every element1 that is the nth child of its
parent */
}

p:nth-child(3) {
   /* Selects every <p> that is the 3rd one of its parent */
}
```

Different types of CSS selectors

```
p:nth-child(3) {
   /* Selects every <p> that is the 3rd one of its parent */
}

<div class="parent">
  <p></p>
  <p></p>
  <p></p>
  <p></p>
</div>
```

Different types of CSS selectors

```
p:nth-child(3) {
   /* Selects every <p> that is the 3rd one of its parent */
}
```

```
<div class="parent">
  <p></p>
  <p></p>
  <p></p>  ←——————  n = 3
  <p></p>
</div>
```

Hands on!