

Activity: Building Bootstrap

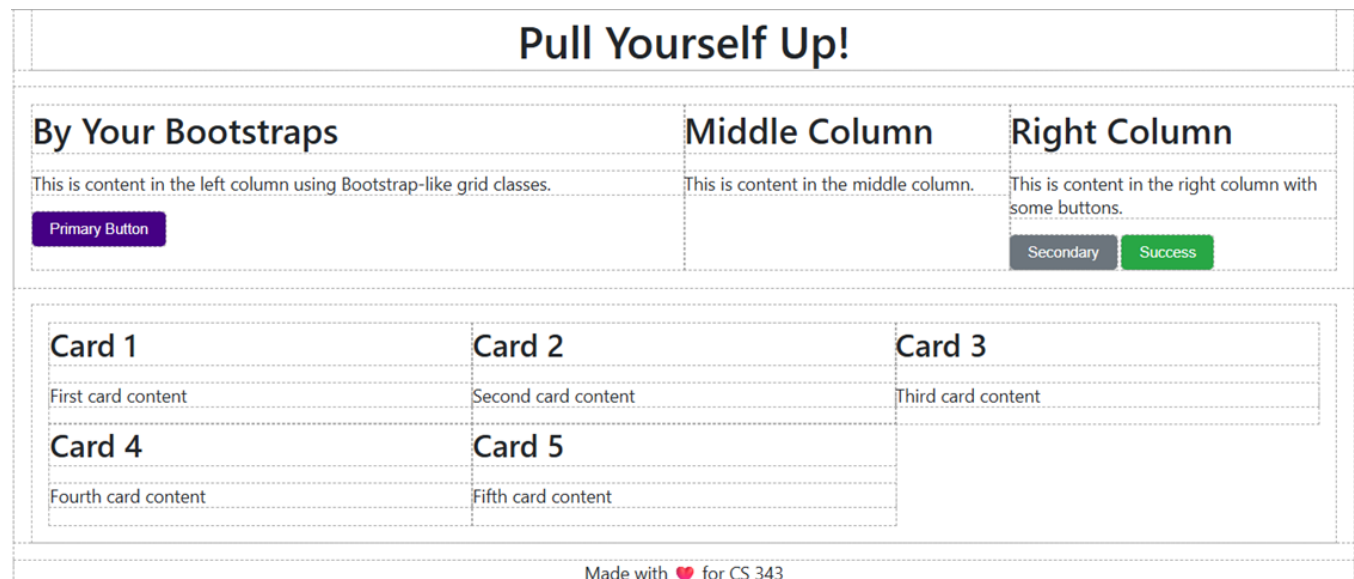
Name:

Date:

Bootstrap is a big stylesheet of reusable classes, like `col-md-6` and `btn-primary`. In this activity, you will figure out what some of those classes mean, then write a tiny version of Bootstrap yourself, and finally make it respond to screen size. Work through this worksheet in order, and write your answers in the spaces provided. Predict first, then check your prediction!

What you need: the starter files `index.html` and `shoestring.css` (from the Building Bootstrap page on the course website), saved in the same folder; and VS Code with a live preview of `index.html`. You will only edit `shoestring.css`. The dashed gray outlines on the page are on purpose: they show the edges of each box.

Here is the finished page. Refer to it throughout:



Part 1: Read the Classes

Open `index.html` (don't change it!). Everything here can be answered by reading the HTML and the image above.

Q1. Find every element with the `row` class in `index.html`. For each one, write the tag, where it is on the page, and how many `col-md-*` columns are directly inside it. Then **draw a box** around each row in the image above.

Tag	Where on the page?	# of columns inside

Q2. Look at the main row (the one with "By Your Bootstraps"). Write the class on each of its three columns, then add up the numbers at the end of them. What do you notice?

Q3. In the image, the left column is about **half** as wide as its row, and the middle and right columns are each about **a quarter** as wide. Compare that to the classes in Q2. What does the **number** at the end of `col-md-N` mean? Explain in your own words.

Q4. Use your answer to Q3 to complete the table. Give the fraction of the row (out of 12) and the percentage, rounded to 2 decimal places.

Class	Fraction of the row	Percentage
<code>col-md-6</code>	6/12	50%
<code>col-md-3</code>		
<code>col-md-4</code>		
<code>col-md-1</code>		
<code>col-md-12</code>		

Note: the percentage is always of the column's parent row, not of the whole page.

Q5. The Cards row has **five** columns, each `col-md-4`. A row has room for 12 slices per line. How many cards fit on one line? Which cards end up on the second line? Does the image agree with your prediction?

Part 2: Build Bootstrap

Now write the CSS that makes those classes work. Open `shoestring.css` and read the top of the file, including how the **CSS variables** (like `--primary-color`) are defined and used. `.container` and `.text-center` are already written for you. Fill in the empty rules **in order**, and check the page after each step.

2a: Rows and Columns

- ☐ Write the `.row` rule so that its children sit **side by side**, but **wrap** onto the next line when they don't fit.
- ☐ Write the five `.col-md-*` rules. Each sets the element's **width** to the percentage you found in Q4. Type all the decimals (like `8.333333%`) so the columns add up to exactly 100%.

Q6. Which single CSS property made Cards 4 and 5 move to a new line instead of squeezing in beside the other cards? **Test it:** comment that property out. What happens to the cards? Put it back when you are done.

2b: Spacing

Bootstrap's spacing classes are a letter (m or p) followed by a size.

- ☐ Write the `.m-0` and `.p-0` rules (set margin or padding to 0), and the `.m-3` and `.p-3` rules (set it to 1rem).

Q7. The header row is `class="row m-3"` and the main row is `class="row p-3"`. Compare the dashed outlines of each row and its contents. What is the difference between what m-3 did and what p-3 did? What do the letters **m** and **p** stand for?

2c: Buttons

- ☐ Write the `.btn-primary`, `.btn-secondary`, and `.btn-success` rules. Each sets its background-color to the matching theme variable: `--primary-color`, `--secondary-color`, or `--success-color` (use `var(...)`).

- ☐ Compare your page with the image on page 1. They should match.

Q8. Why is it better to write `var(--primary-color)` than to type the color's hex code in every rule? If you wanted the whole page to use a different primary color, what would you change?

Part 3: Responsive Design

Q9. Before we make the page responsive, think about the classes you just built. What are the **advantages** and **disadvantages** of writing reusable classes like `col-md-6` instead of writing rules for specific elements? What would the CSS look like if every column had its own `id` and its own width rule?

The `md` in `col-md-6` stands for a “medium” **breakpoint**, at about 768px wide. Above that width, the page looks like the image on page 1.

Q10. Predict: what will happen to the three columns in the main row when the window is very narrow (like on a phone)? Then test it by dragging the window narrower, or using your browser's device mode (DevTools). What did you see? Is it a good phone layout?

Mini-Tutorial: Media Queries

So far, every rule you’ve written applies **all the time**. A **media query** makes a group of rules apply only when a condition is true, such as “the viewport (visible part of the page) is narrow.”

```
@media (max-width: 768px) {  
  h1 {  
    color: red;  
  }  
}
```

It has three parts: **(1)** `@media`, which starts a media query; **(2)** the **condition** in parentheses — `max-width: 768px` means “when the viewport is *at most* 768px wide” (`min-width` means “at least”); and **(3)** ordinary CSS rules inside the curly braces. Remember to close **both** the rule’s brace and the query’s brace!

- ☐ Type the example above at the bottom of `shoestring.css` (or copy it from the course website) and save. Slowly drag your browser window narrower — or use DevTools device mode, which shows the viewport’s width in pixels — and watch the page title as you cross 768px.

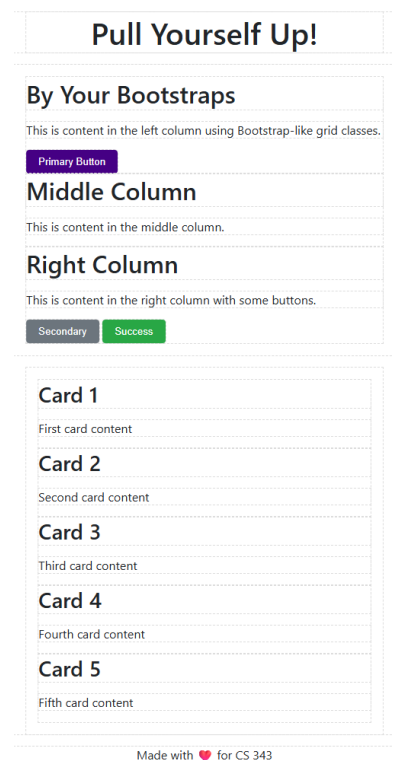
Q11. At about what width did the title change color? What would you change in the code to make it change at 500px instead? What do you think would happen if you swapped `max-width` for `min-width`? (Try it!)

Stacking the Columns

We want every column to stack vertically and take up the full width of its row, like in the image on the right.

- ☐ Edit your example media query into the real thing: replace the `h1` rule with **one rule** that sets the width of all five `.col-md-*` classes to `100%`. (A comma between selectors lets them share one rule.)
- ☐ Narrow your browser window below 768px to check that the columns stack, like in the image on the right.

Q12. You didn’t change `.row` in the media query — only the column widths. Why do the columns now stack on top of each other instead of overflowing off to the right? (*Hint: look back at Q6.*)



Submit `index.html` and `shoestring.css` to Gradescope, following the instructions on the course website. You do not need to submit this worksheet.