5.2.4 Implement a Bootstrap Grid Layout

IMPORTANT

> For more information on everything we'll learn in this step, see **Bootstrap's documentation on the grid (https://getbootstrap.com/docs/4.3/layout/grid/)**. Before you start, skim through the documentation and familiarize yourself with it.

Before we jump into Bootstrap's grid system, let's recall the most important concept regarding responsive web layouts—the parent-child relationship. When we created layouts in the past, we wrapped an HTML container element around all of the HTML child elements we wanted to control and position with CSS layout properties. With the content organized that way, we could dictate how that container's children displayed on the page by adding CSS properties to the parent.

PAUSE

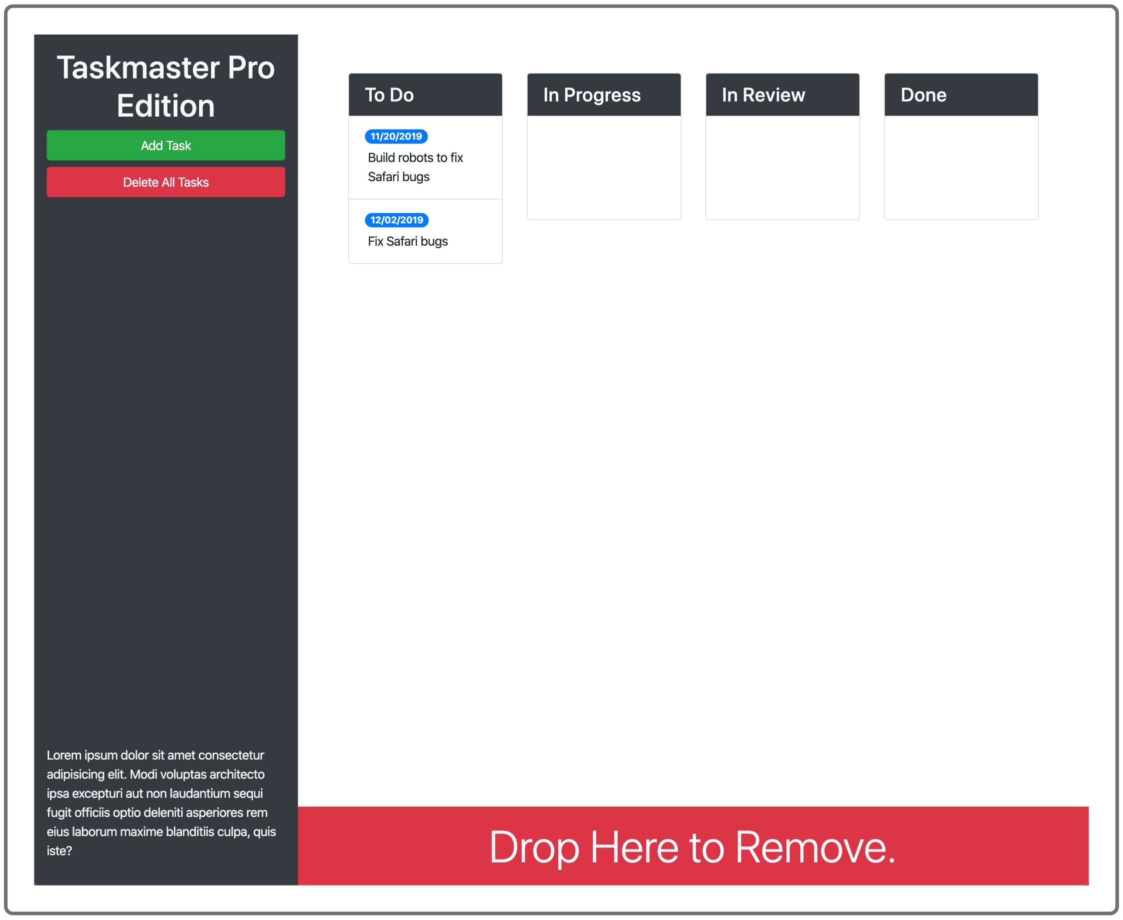
What CSS property and value have we used to create responsive layouts?

We've used flexbox—`display: flex;`

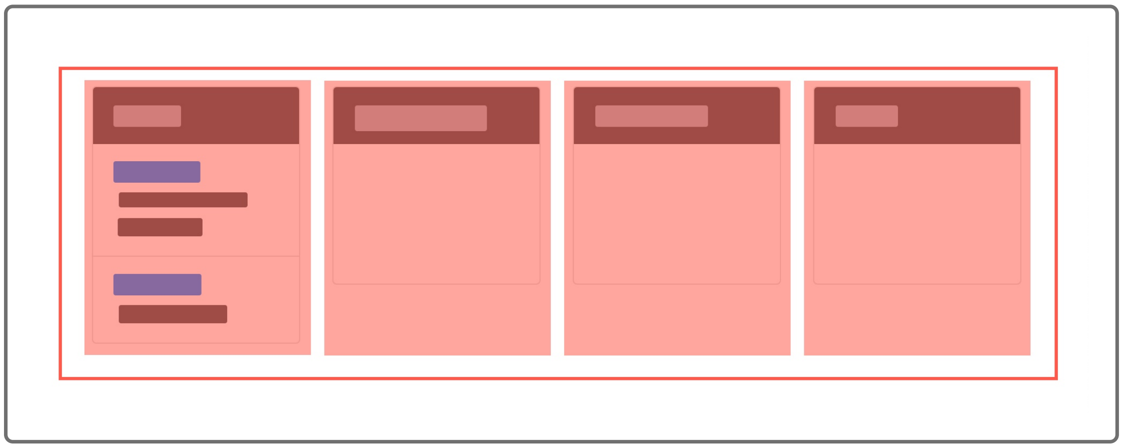
Hide Answer

Multiple parent-child flexbox layouts on a page may all have different spacing and sizing properties with no adverse effects on each other. Luckily, Bootstrap's grid system uses flexbox in the same way!

Let's look again at the following mock-up image—specifically the right side, where we'll add the task status lists—then we'll discuss how to use Bootstrap to get there:


Taskmaster Pro Edition
Add Task
Delete All Tasks
Lorem ipsum dolor sit amet consectetur adipisicing elit. Modi voluptas architecto ipsa excepturi aut non laudantium sequi fugit officiis optio deleniti asperiores rem eius laborum maxime blanditiis culpa, quis iste?
To Do
11/20/2019
Build robots to fix Safari bugs
12/02/2019
Fix Safari bugs
In Progress
In Review
Done
Drop Here to Remove.

Within this right column, we'll need four lists with equal widths. Now let's look at it again. This time we'll use our layout skills and view the column as boxes instead of content, as shown in the following image:



In terms of layout, we need to create a parent element to hold the four task lists, and we need to set them up to break onto the next horizontal line when viewed in a narrower window. In other words, we need to create one row with four columns.

IMPORTANT

> Bootstrap has a `container` class that's used for a different purpose. To avoid confusion, assume that any mention of the word "container" will refer to a flexbox row or column.
>
> For more information, see the **Bootstrap documentation on containers (https://getbootstrap.com/docs/4.3/layout/overview/)**.

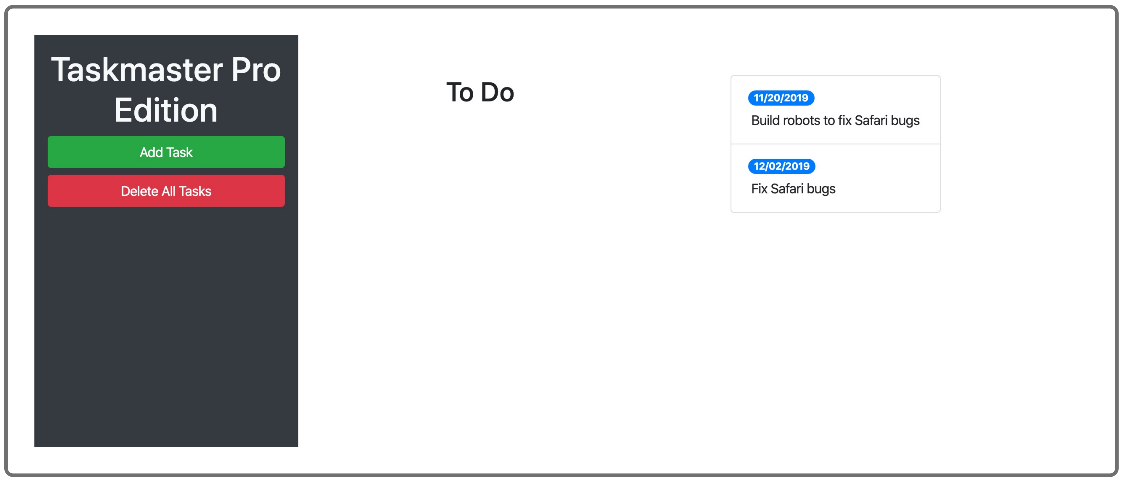
Set Up the Grid

For the time being, we can ignore almost everything in the `index.html` file and focus on what's going on between the opening and closing `<main>` tags. Because we're going to build a grid layout within these tags, nothing else on the page should be affected.

We have a `<div>` element that wraps an `<h2>` element and `<ul>`, but let's update it to look like this code:

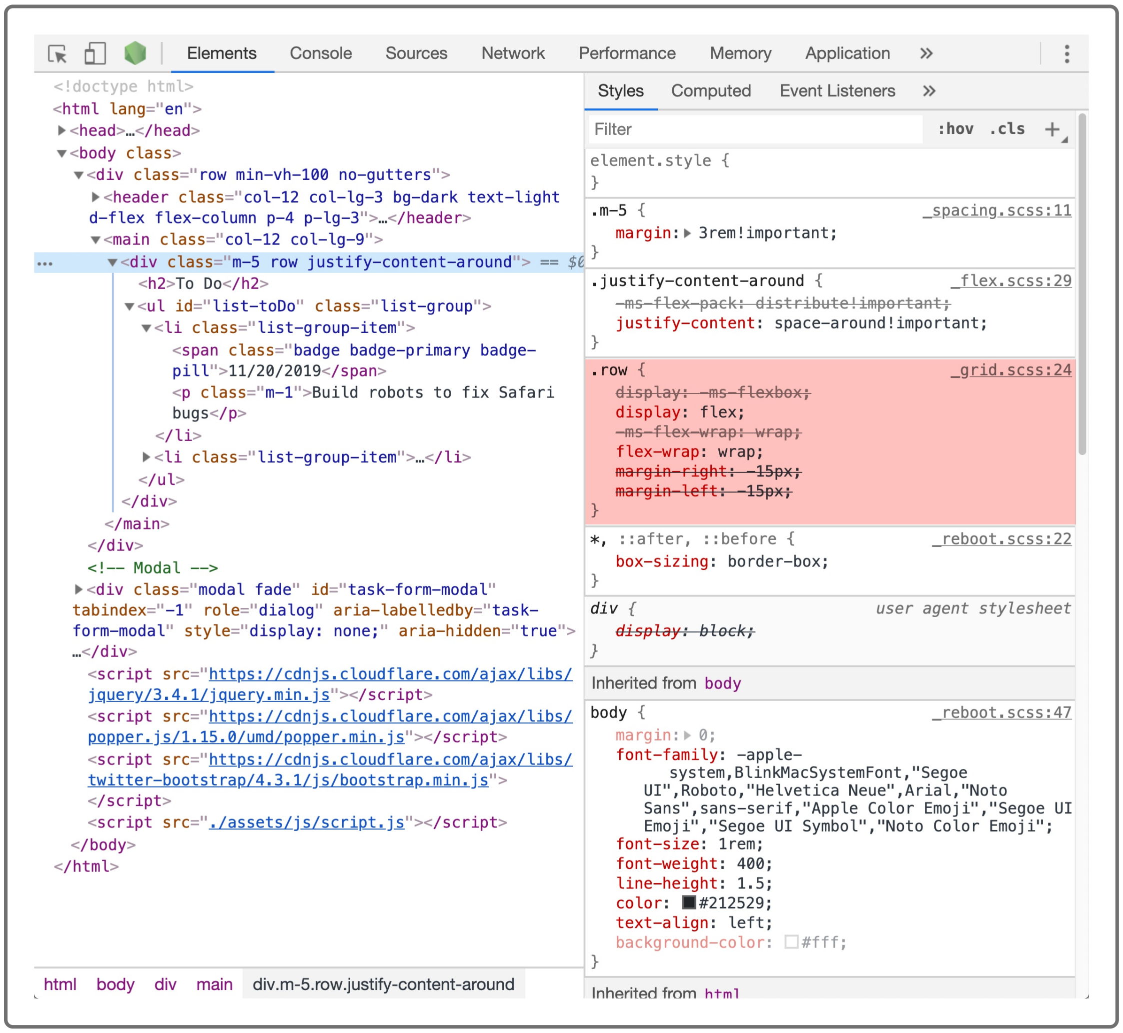
```
<div class="m-5 row justify-content-around">
  <h2>To Do</h2>
  <ul id="list-toDo" class="list-group"></ul>
</div>
```

When we save and refresh the page, we'll see something like this image:



Well, that's not what we want. Why did the addition of a class of `row` cause two elements to move side by side? What CSS `display` property value lays out child elements horizontally by default?

Let's see which properties are assigned to the `row` class in DevTools. We should see something like this image:



```
<!doctype html>
<html lang="en">
  ▶<head>…</head>
  ▼<body class>
    ▼<div class="row min-vh-100 no-gutters">
      ▶<header class="col-12 col-lg-3 bg-dark text-light
      d-flex flex-column p-4 p-lg-3">…</header>
      ▼<main class="col-12 col-lg-9">
...     ▼<div class="m-5 row justify-content-around"> == $0
          <h2>To Do</h2>
          ▼<ul id="list-toDo" class="list-group">
            ▼<li class="list-group-item">
                <span class="badge badge-primary badge-
                pill">11/20/2019</span>
                <p class="m-1">Build robots to fix Safari
                bugs</p>
              </li>
            ▶<li class="list-group-item">…</li>
            </ul>
          </div>
        </main>
      </div>
      <!-- Modal -->
    ▶<div class="modal fade" id="task-form-modal"
    tabindex="-1" role="dialog" aria-labelledby="task-
    form-modal" style="display: none;" aria-hidden="true">
    …</div>
      <script src="https://cdnjs.cloudflare.com/ajax/libs/
      jquery/3.4.1/jquery.min.js"></script>
      <script src="https://cdnjs.cloudflare.com/ajax/libs/
      popper.js/1.15.0/umd/popper.min.js"></script>
      <script src="https://cdnjs.cloudflare.com/ajax/libs/
      twitter-bootstrap/4.3.1/js/bootstrap.min.js">
      </script>
      <script src="./assets/js/script.js"></script>
    </body>
</html>
```

html body div main div.m-5.row.justify-content-around

```
element.style {
}
.m-5 {                                    _spacing.scss:11
   margin:▶ 3rem!important;
}
.justify-content-around {                    _flex.scss:29
   -ms-flex-pack: distribute!important;
   justify-content: space-around!important;
}
.row {                                       _grid.scss:24
   display: -ms-flexbox;
   display: flex;
   -ms-flex-wrap: wrap;
   flex-wrap: wrap;
   margin-right: -15px;
   margin-left: -15px;
}
*, ::after, ::before {                     _reboot.scss:22
   box-sizing: border-box;
}
div {                                user agent stylesheet
   display: block;
}
Inherited from body
body {                                     _reboot.scss:47
   margin:▶ 0;
   font-family: -apple-
       system,BlinkMacSystemFont,"Segoe
     UI",Roboto,"Helvetica Neue",Arial,"Noto
     Sans",sans-serif,"Apple Color Emoji","Segoe UI
     Emoji","Segoe UI Symbol","Noto Color Emoji";
   font-size: 1rem;
   font-weight: 400;
   line-height: 1.5;
   color: ■#212529;
   text-align: left;
   background-color: □#fff;
}
Inherited from html
```

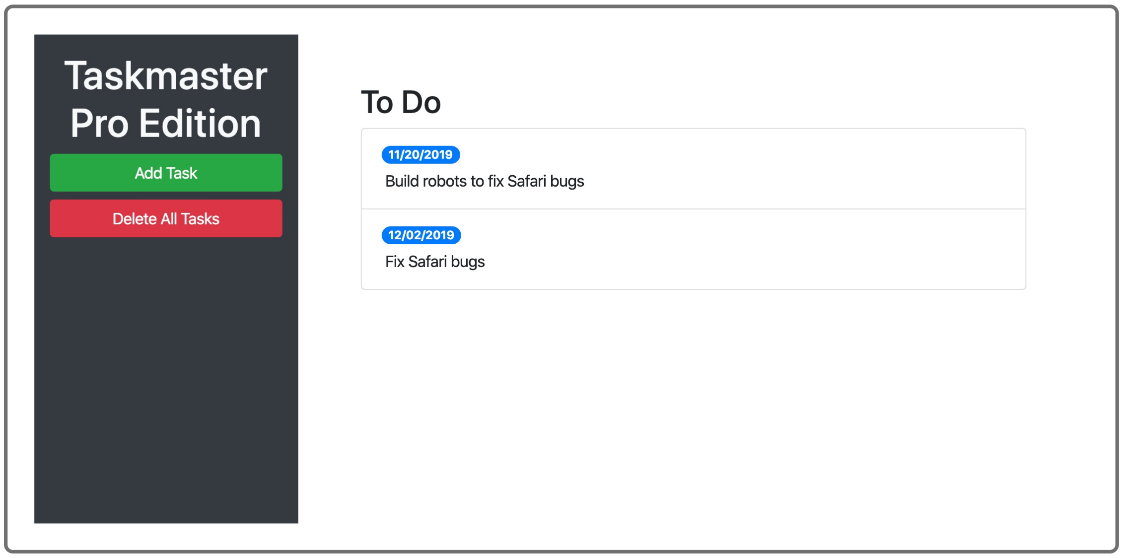
The `row` class creates a flexbox that wraps overflowing content! That's also why the inclusion of the `justify-content-around` class put space around the two elements, as that's a flexbox-specific setting for how the flex parent should space out its child elements.

So if rows in Bootstrap are actually flexbox parents or containers, how do we manage the child elements? Remember that we can give each item in a flexbox a set width using CSS. This approach works, but it involves a fair amount of math to manage a lot of elements. Bootstrap solves the problem by limiting the number of flex items that can fit on a row before it wraps the content onto the next line.

Let's change the code to look like this:

```
<div class="m-5 row justify-content-around">
  <div class="col">
    <h2>To Do</h2>
    <ul id="list-toDo" class="list-group"></ul>
  </div>
</div>
```

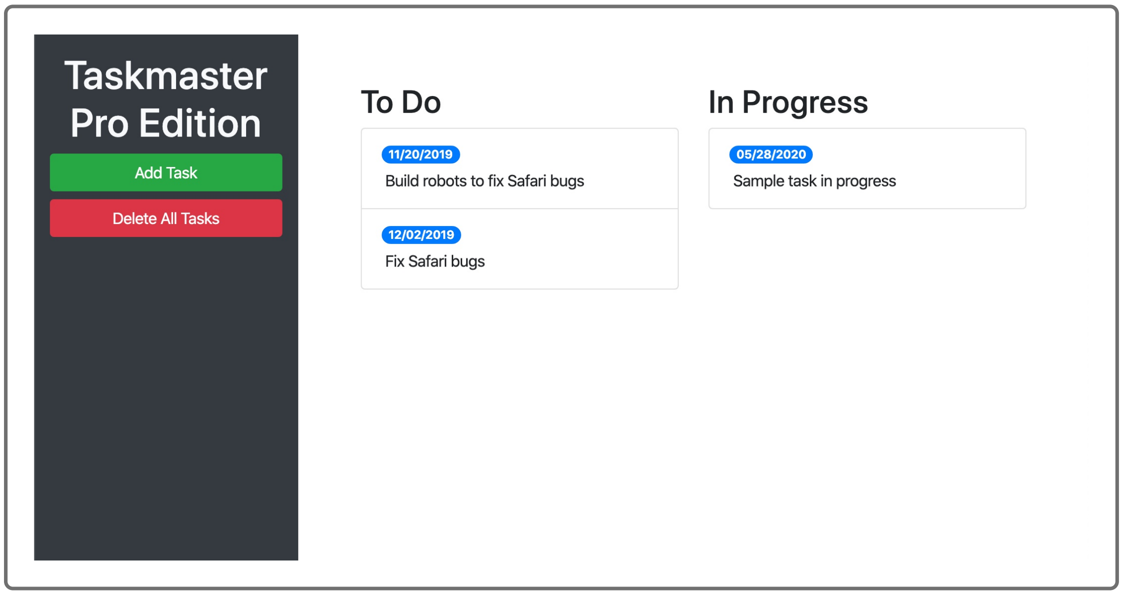
If we save and refresh, the layout should look close to what we had earlier, something like this image:



This approach may seem like more code for the same layout, but watch what happens when we add more to it. Let's create another task list for In Progress tasks, with a dummy task item inside it so we can see how it looks. Update the code as follows:

```
<div class="m-5 row justify-content-around">
  <!-- todo start -->
  <div class="col">
    <h2>To Do</h2>
    <ul id="list-toDo" class="list-group"></ul>
  </div>
  <!-- todo end -->
  <!-- in progress start -->
  <div class="col">
    <h2>In Progress</h2>
    <ul id="list-inProgress" class="list-group">
      <li class="list-group-item">
        <span class="badge badge-primary badge-pill">05/28/2020</span>
        <p class="m-1">Sample task in progress</p>
      </li>
    </ul>
  </div>
  <!-- in progress end -->
</div>
```

Save and refresh. Now two lists share that row evenly, like this image shows:



Notice how we organized the HTML so that directly inside the `<div>` with a class `row` we have two `<div>` elements with a class of `col`? Bootstrap considers all children in a row with a class that matches `col-*` syntax to be columns, so now we have a row shared by two columns. If we resize the browser, they grow and shrink at the same rate.

DEEP DIVE

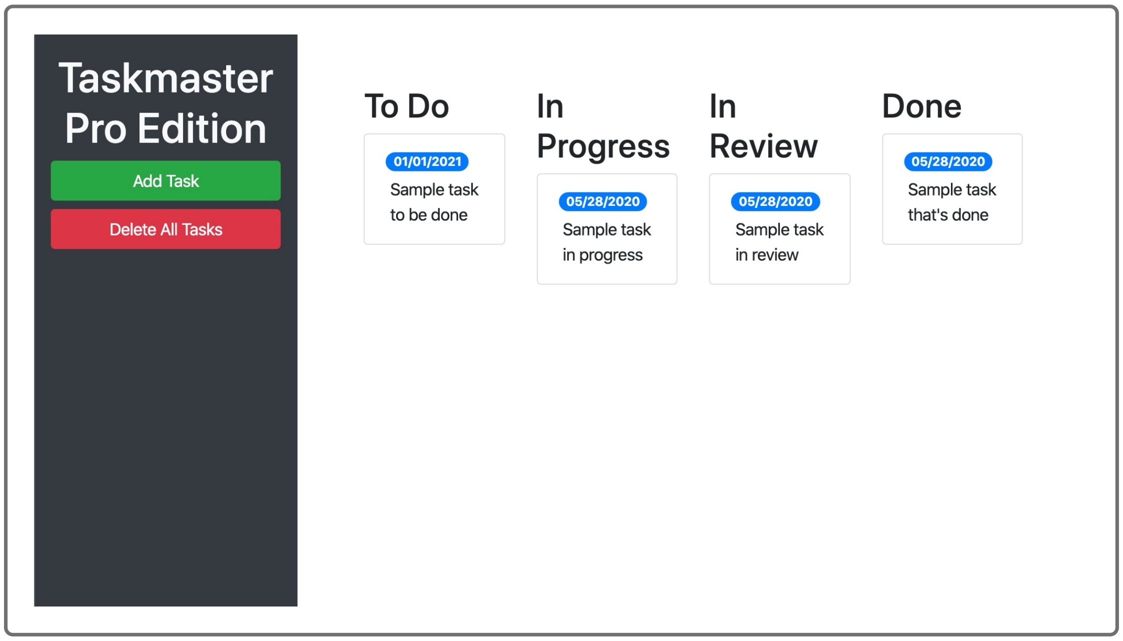
DEEP DIVE

To learn more about equal-width columns, read **Bootstrap's documentation on auto-layout (https://getbootstrap.com/docs/4.3/layout/grid/#auto-layout-columns)** .

Let's now add the last two lists so that the whole row's code looks like this:

```
<div class="m-5 row justify-content-around">
  <!-- todo start -->
  <div class="col">
    <h2>To Do</h2>
    <ul id="list-toDo" class="list-group"></ul>
  </div>
  <!-- todo end -->
  <!-- in progress start -->
  <div class="col">
    <h2>In Progress</h2>
    <ul id="list-inProgress" class="list-group">
      <li class="list-group-item">
        <span class="badge badge-primary badge-pill">05/28/2020</span>
        <p class="m-1">Sample task in progress</p>
      </li>
    </ul>
  </div>
  <!-- in progress end -->
  <!-- in review start -->
  <div class="col">
    <h2>In Review</h2>
    <ul id="list-inReview" class="list-group">
      <li class="list-group-item">
        <span class="badge badge-primary badge-pill">05/28/2020</span>
        <p class="m-1">Sample task in review</p>
      </li>
    </ul>
  </div>
  <!-- in review end -->
  <!-- done start -->
  <div class="col">
    <h2>Done</h2>
    <ul id="list-done" class="list-group">
      <li class="list-group-item">
        <span class="badge badge-primary badge-pill">05/28/2020</span>
        <p class="m-1">Sample task that's done</p>
      </li>
    </ul>
  </div>
  <!-- done end -->
</div>
```

Depending on the width of the browser window, the result of adding two more columns may vary. With the browser window at its smallest, we'll see two rows of two columns. At its widest, we'll see all four columns side by side. Aside from those two extremes, though, at a more standard browser width the columns look disjointed, like this image shows:



The client neglected to tell us that they provided the mock-up from their office computer, which has a 27" monitor, so they probably keep their browser screen around 1200px or wider. This size is popular, but we can't assume every user will have a screen that size. We'll have to adjust this layout for different screen sizes.

Different Screen Widths, Different Column Widths

Bootstrap dictates that no more than twelve columns can fit onto one horizontal line before they break onto the next line. This means up to twelve HTML elements with a class of `col` can fit onto one horizontal line in an element with a class of `row`, granted the content of that HTML element can become that small.

Bootstrap also knows that not every column in a row will likely have equal widths, so they also provide more classes to help divvy up those twelve columns.

Consider the following examples:

```
<!-- A row with two mixed-size elements -->
<div class="row">
  <!-- Span 3 of 12 columns, or 1/4 of the row -->
  <div class="col-3"></div>

  <!-- Span 9 of 12 columns, or 3/4 of the row -->
  <div class="col-9"></div>
</div>

<!-- A row with three mixed-size elements -->
<div class="row">
  <!-- Span 3 of 12 columns, or 1/4 of the row -->
  <div class="col-3"></div>

  <!-- Span 6 of 12 columns, or 1/2 of the row -->
  <div class="col-6"></div>

  <!-- Span 3 of 12 columns, or 1/4 of the row -->
  <div class="col-3"></div>
</div>
```

```
<!-- A row with twelve equally sized elements -->
<div class="row">
  <!-- Each span 1 of 12 columns, or 1/12 of the row -->
  <div class="col-1"></div>
  <div class="col-1"></div>
  <div class="col-1"></div>
  <div class="col-1"></div>
  <div class="col-1"></div>
  <div class="col-1"></div>
  <div class="col-1"></div>
  <div class="col-1"></div>
  <div class="col-1"></div>
  <div class="col-1"></div>
  <div class="col-1"></div>
  <div class="col-1"></div>
</div>
```

The numeric value in the syntax `col-<number of columns>` denotes how many columns that element will span in the twelve-column row. The syntax `col-6` means it will take up six out of the twelve columns, or one half. And `col-12` means it will take up twelve out of the twelve columns, or the entire row.

HIDE PRO TIP

To see how these width values are calculated, look at each column class in Chrome DevTools!

What happens if a row's columns don't add up to twelve? If the columns add up to a value under twelve, like an element with `col-4` and an element with `col-5`, there will be unused space to the right of those two elements. If the columns add up to more than twelve, the element that puts it past twelve will overflow onto the next line.

If we have four elements for our task lists, how many columns does each one span? Well, if the grid allows up to twelve, we can assume that each list will span three columns. Each one will span three columns until the screen gets too narrow and breaks the last column onto a new line. Having each list span three columns works on the aforementioned 27" monitor the client uses, but that won't work well for other screen sizes, such as phones or tablets.

PAUSE

In CSS, how can we tell an element to have a style at one screen size, but then a different style at another screen size?

With media queries.

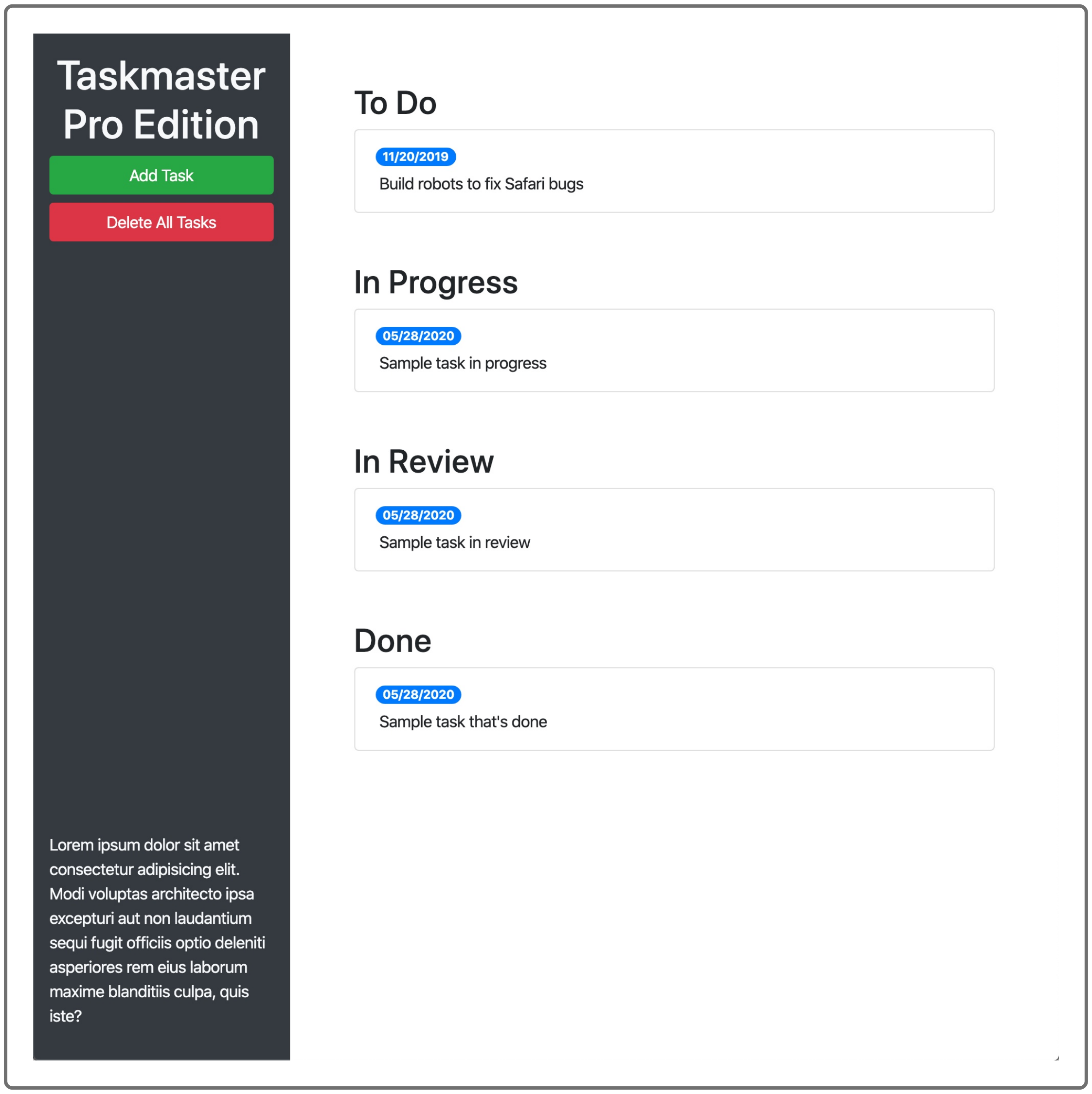
Hide Answer

Bootstrap has taken a mobile-first approach to layouts. This means that the standard `col` class will apply to the smallest screen size and up, and if we want to have a different layout at different screen sizes, we'll have to use multiple classes.

In the row of task lists, let's change all the `col` classes to `col-12` so that they look like this:

```
<div class="col-12">
  <h2>Done</h2>
  <ul id="list-done" class="list-group">
    <li class="list-group-item">
      <span class="badge badge-primary badge-pill">05/28/2020</span>
      <p class="m-1">Sample task that's done</p>
    </li>
  </ul>
</div>
```

Save the file and refresh the page. We should see something like this image:



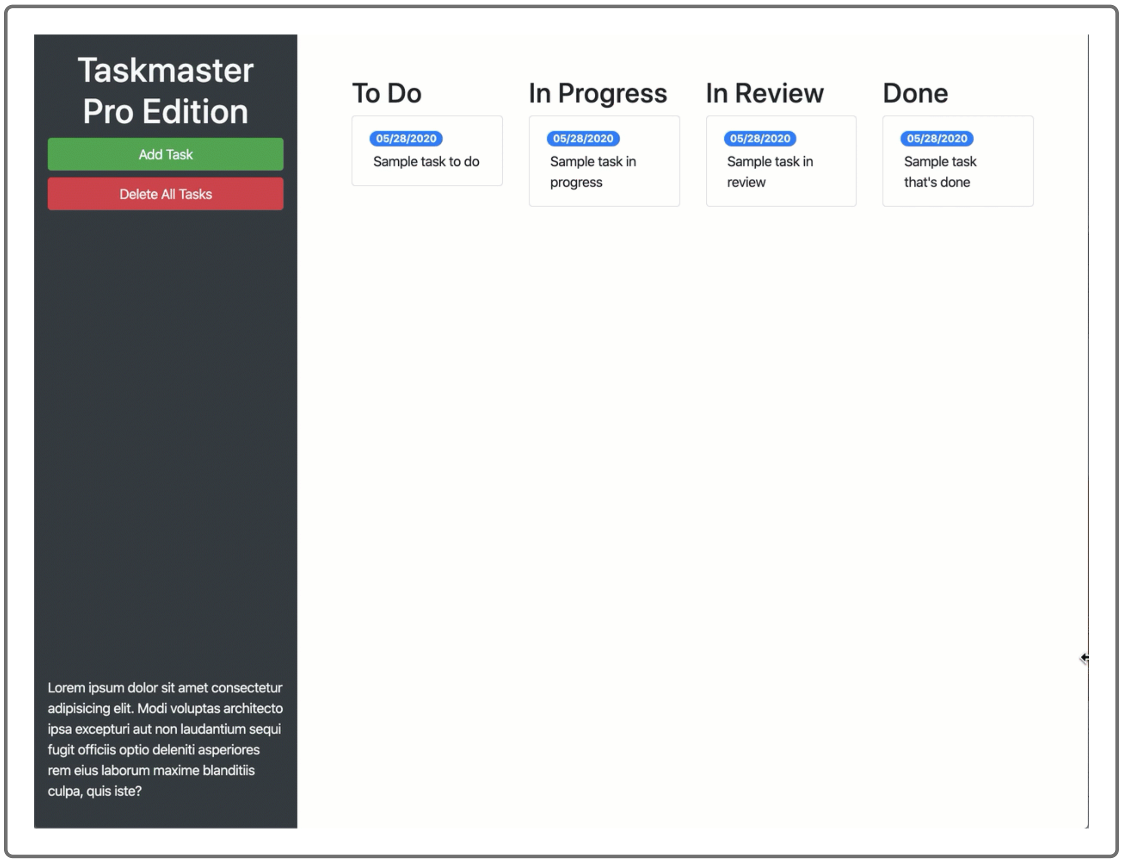
Lorem ipsum dolor sit amet consectetur adipisicing elit. Modi voluptas architecto ipsa excepturi aut non laudantium sequi fugit officiis optio deleniti asperiores rem eius laborum maxime blanditiis culpa, quis iste?

All the lists now occupy the full width of the row! By using `col-12` as the class, we've instructed each `<div>` element to span the entire row, pushing the next `<div>` onto the next line. Now, this works for mobile devices, as those narrow screens make it tricky to put anything side by side in an orderly way. But how do we get back to the four-column layout the client wants to see?

Let's add more classes to the `<div>` elements so that all four look like this (the `.mb-3` class, which we haven't discussed yet, is specifying margin-bottom spacing):

```
<div class="col-12 col-md-6 col-xl-3 mb-3">
  <h2>Done</h2>
  <ul id="list-done" class="list-group">
    <li class="list-group-item">
      <span class="badge badge-primary badge-pill">05/28/2020</span>
      <p class="m-1">Sample task that's done</p>
    </li>
  </ul>
</div>
```

Now let's save and refresh the page. Then let's resize the browser window to see how the `<div>` elements change at different screen widths. It should look something like this GIF:



Lorem ipsum dolor sit amet consectetur adipisicing elit. Modi voluptas architecto ipsa excepturi aut non laudantium sequi fugit officiis optio deleniti asperiores rem eius laborum maxime blanditiis culpa, quis iste?

The lists look good at every screen size! On the client's 27" monitor, they'll get the four-across layout they were hoping for, and those who use the app on their 13" laptops or their mobile devices can still see all the content clearly. So what made this happen? What's with the `md` and `xl` between the column class names?

To learn more about what all this means, read the **documentation on Bootstrap's grid (https://getbootstrap.com/docs/4.3/layout/grid/#grid-options)**. We don't have to worry about using custom CSS to create media queries and breakpoints, as Bootstrap has already done that work for us. We just need to add a class with how many columns an element should span at what breakpoint.

For the task list `<div>` elements, we use `col-12` to make each one run full width on small devices. Then when the screen is 768px wide, the `col-md-6` class takes over and tells each list to span six columns instead of twelve. When we reach a much larger size—1200px to be exact—the `col-xl-3` class takes over, and each list spans only three columns.

HIDE HINT

Don't forget that we can see the CSS properties being overridden in DevTools!

The Bootstrap grid is an amazing system for developers who don't need to create a unique layout. It allows us to use flexbox properties such as alignment, justification, and ordering so that we don't have to write custom CSS. We only have to create a parent element with a class of `row`, give all of its child elements a mix of `col-` classes, and we have the responsive grid!

LEGACY LORE

Earlier iterations of Bootstrap used CSS floats to create their grid system. It wasn't until Bootstrap 4 that flexbox was used instead.

You also may have noticed by now that the entire page is wrapped in a Bootstrap `row`, and that's why the left and right columns stack on top of each other when the screen shrinks. So you can use it not only for smaller parts of the layout but also pretty much anywhere!

To see another, simpler example of a Bootstrap grid, watch the following video:

Now that we can create responsive layouts using Bootstrap, let's dive into their UI components.