1. Lightning Design System for Developers-
2. Understand the Grid System-

Understand the Grid System-

Learning Objectives

After completing this unit, you'll be able to:

- Describe how the **Grid System** component works.
- Layout an object home / list view page using the Grid System.

What Is a Grid System?

The foundation for all but the simplest pages and components is a layout grid of some kind. The Design System provides a dedicated component for this purpose imaginatively called the Grid System. If you have used other CSS frameworks such as Bootstrap, you will be familiar with the concept of a grid. If not, in very brief terms, a grid allows you to divide your page into rows and columns. You can then arrange your markup so that it is rendered in a particular row/column. Grids can be nested allowing for complex layouts.

The Design System grid is based on CSS Flexbox and provides a flexible, mobile-first, device-agnostic scaffolding system. The Design System also includes helper classes that you can use to alter the look and behavior of your grid, such as alignment, order, flow, and padding.

The Grid System allows you to create responsive pages by defining specific layout variations for small, medium and large screens. These are defined with breakpoints of 480px, 768px and 1024px respectively. The specifics of tailoring responsive pages is beyond this tutorial, but if you'd like to know more, please refer to the sizing utilities page documentation.

How to Use the Grid System

The grid system is based on two key building blocks: the grid wrapper (specified with the `slds-grid` class) and the columns within it (specified with the `slds-col` class). Here's an example:

To start, add the `slds-grid` class to an outer wrapper element. Then inside it add the required number of columns by adding the `slds-col` class to the child elements. We'll use `<div>` elements in this case. For example, here is a simple grid with three columns:

```
<div class="slds-grid">
 <div class="slds-col">Column 1</div>
 <div class="slds-col">Column 2</div>
 <div class="slds-col">Column 3</div>
</div>
```

Copy

Copy

This markup results in the following layout:


column 1

By default, the columns are sized relative to their contents. In this simple example, we should therefore see three identically spaced columns because they hold equal an amount of content. If more content was added to one of the columns, it would grow relative to the others.

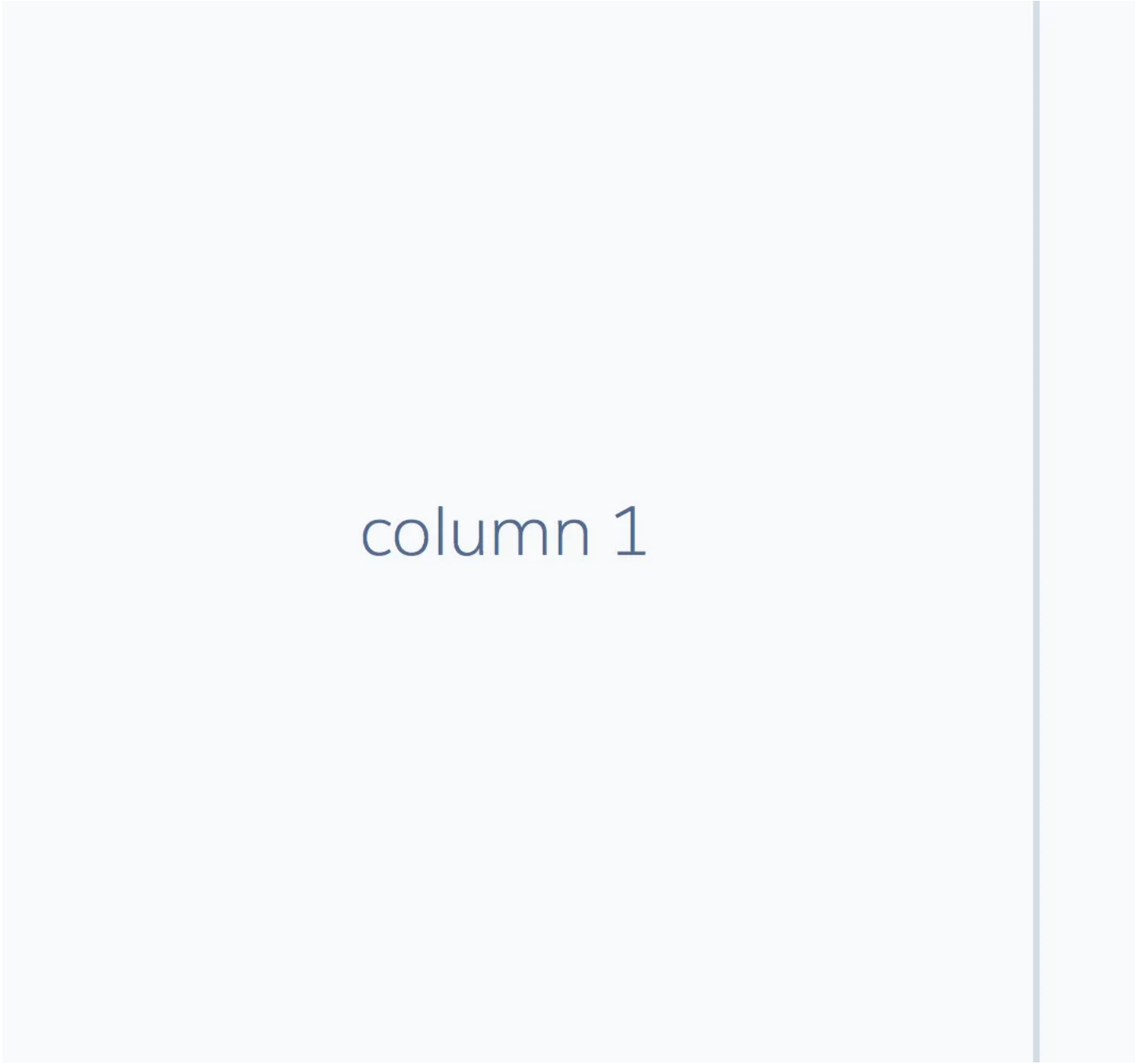
You can also specify the sizes of the columns manually using the sizing helper classes. These use a `slds-size_X-of-Y` format where X represents a fraction of the total space Y. For example, `slds-size_1-of-2` represents a width that is 50% of the available space. Using the manual sizing class helpers, you can specify column ratios across the following grids – 2, 3, 4, 5, 6, and 12.

Here is the above example modified so that the first column fills two-thirds of the screen.

```
<!-- BASIC GRID EXAMPLE -->
<div class="slds-grid">
 <div class="slds-col slds-size_4-of-6">Column 1</div>
 <div class="slds-col slds-size_1-of-6">Column 2</div>
 <div class="slds-col slds-size_1-of-6">Column 3</div>
</div>
```

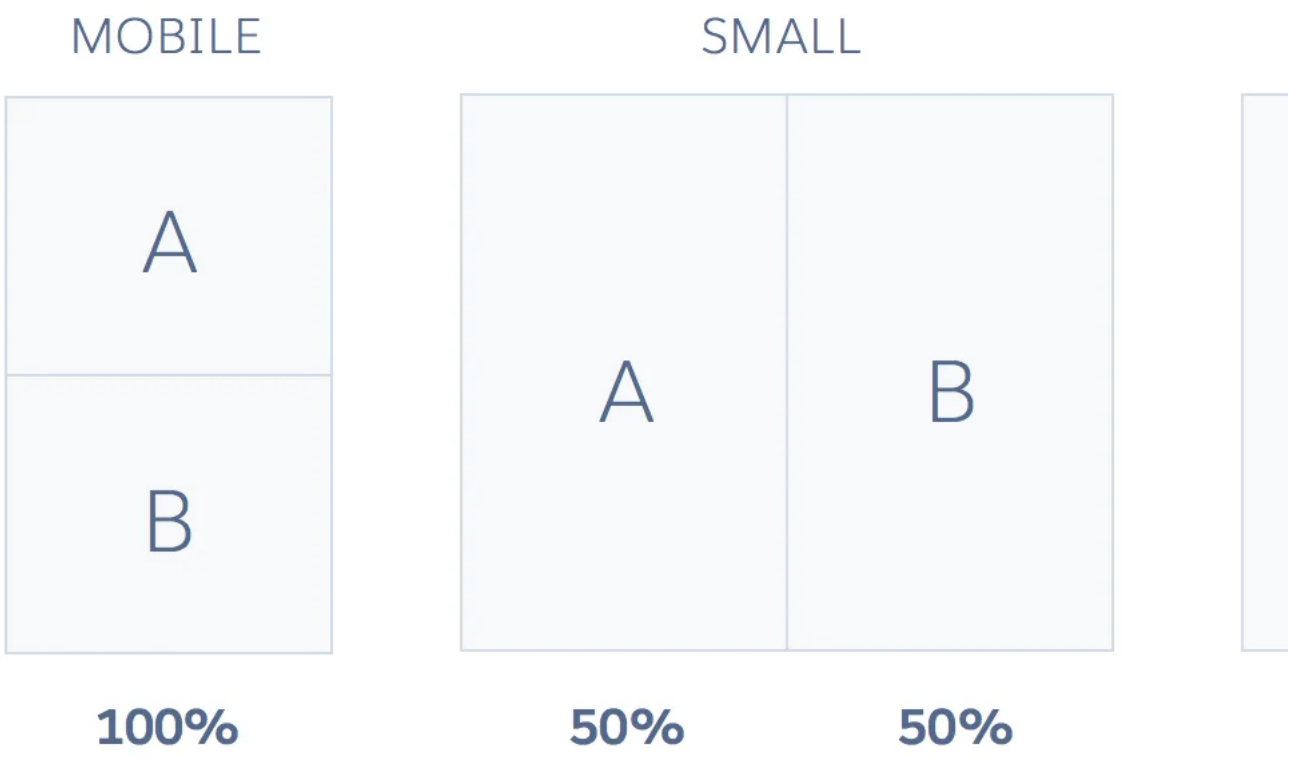
Copy

Copy



The sizing helpers can also be used to specify different layouts depending on screen size. For example, let's create a two column grid where the two columns are:

- Full width and arranged vertically on a mobile screen
- Sized 1:1 and arranged side by side on small screens (> 480px)
- Sized 3:1 and arranged side by side on larger screens (> 768px)



Here is the corresponding markup:

```
<!-- RESPONSIVE GRID EXAMPLE -->
<div class="slds-grid slds-wrap">
 <div class="slds-col slds-size_1-of-1 slds-small-size_1-of-2 slds-medium-size_3-of-4">A</div>
 <div class="slds-col slds-size_1-of-1 slds-small-size_1-of-2 slds-medium-size_1-of-4">B</div>
</div>
```

Copy

Copy

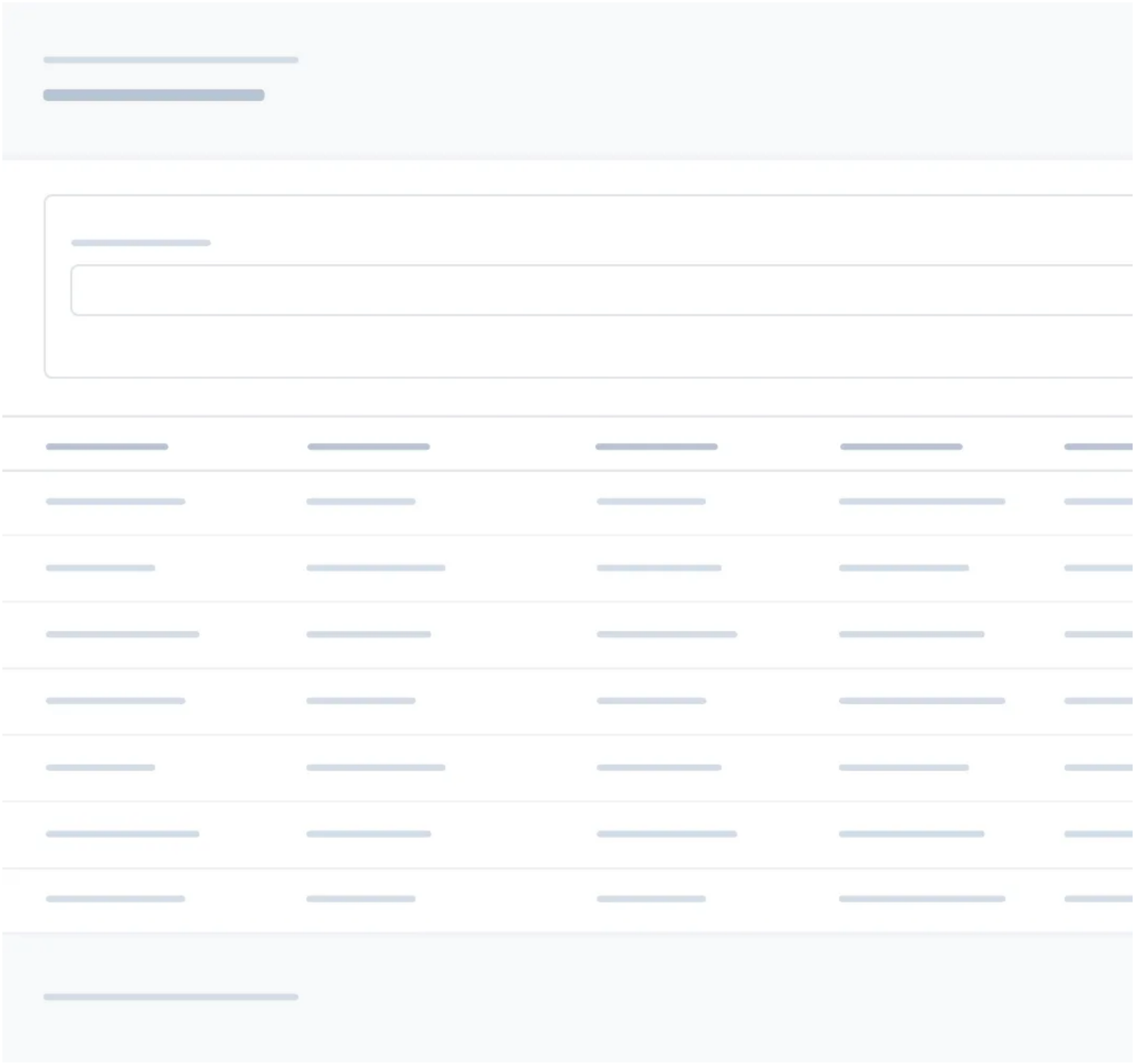
Using this mobile-first approach, you can control elements widths at specific breakpoints where needed. No more broken mobile pages!

We've only touched on the basics of grids here. For more documentation on grids refer to the Grid System and Sizing pages on the Design System website. Now onto some code!

Creating a Page Header

Let's create a list view page. I'm sure you've seen them in Salesforce here and there. The following wireframe shows what we're aiming for:

- Top header, appropriately laid out
- Main list area
- Base footer



Create a new Visualforce page called `Trailhead_SLDS_Listview` with the following code:

```
<apex:page showHeader="false" standardStylesheets="false" sidebar="false" applyHtmlTag="false" applyBodyTag="false" docType="html-5.0">
<html xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink" lang="en">
<head>
 <meta charset="utf-8" />
 <meta http-equiv="x-ua-compatible" content="ie=edge" />
 <title>Salesforce Lightning Design System Trailhead Module</title>
 <meta name="viewport" content="width=device-width, initial-scale=1" />
 <!-- Import the Design System style sheet -->
 <apex:slds />
</head>
<body>
 <!-- REQUIRED SLDS WRAPPER -->
 <div class="slds-scope">
 <!-- MASTHEAD -->
 <p class="slds-text-heading label slds-m-bottom small">
 Salesforce Lightning Design System Trailhead Module
 </p>
 <!-- / MASTHEAD -->
 <!-- PAGE HEADER -->
 <!-- / PAGE HEADER -->
 <!-- PRIMARY CONTENT WRAPPER -->
 <div class="myapp">
 </div>
 <!-- / PRIMARY CONTENT WRAPPER -->
 <!-- FOOTER -->
 <!-- / FOOTER -->
 </div>
 <!-- / REQUIRED SLDS WRAPPER -->
 <!-- JAVASCRIPT -->
 <!-- / JAVASCRIPT -->
</body>
</html>
</apex:page>
```

Copy

Copy

That's the skeleton outline for our list view page. Now we will make things significantly more interesting by adding a Page Header component. This component has a bunch of new markup and classes but don't worry, we'll go through it below. Replace the `<!-- PAGE HEADER -->` part of the markup with the following code:

```
<!-- PAGE HEADER -->
<div class="slds-page-header" role="banner">
 <div class="slds-grid">
 <div class="slds-col slds-has-flexi-truncate">
 <!-- HEADING AREA -->
 <p class="slds-text-title caps slds-line-height reset">Accounts</p>
 <h1 class="slds-page-header__title slds-truncate" title="My Accounts">My Accounts</h1>
 <!-- / HEADING AREA -->
 </div>
 <div class="slds-col slds-no-flex slds-grid slds-align-top">
 <button class="slds-button slds-button_neutral">New Account</button>
 </div>
 </div>
 <div class="slds-grid">
 <div class="slds-col slds-align-bottom slds-p-top_small">
 <p class="slds-text-body_small">COUNT items</p>
 </div>
 </div>
</div>
<!-- / PAGE HEADER -->
```

Copy

Copy

If you preview your page now, you will see the list view page starting to bloom, well the top part of it anyway. Now is a good time to ask yourself: *"How much CSS did I have to write to create this beautifully styled page header component?"*

Since this is one of the more advanced Design System components, we will step through it line by line. We recommend you have your source code open in your favorite development environment as we go through it:

The outer wrapper `<div>` has the class `slds-page-header` which applies page header styling. Inside that we have a two column Grid System.

The first grid column contains two elements, and one of them has a Text heading utility class, whereas the other one has a component specific CSS class, `slds-page-header__title`.

The second column is a bit more involved. It has multiple classes applied: `slds-col slds-no-flex slds-grid slds-align-top`. `slds-no-flex` is one of the Design System layout utility classes, and prevents the column from automatically resizing by removing its flex property. `slds-align-top` is an alignment utility class which adjusts the vertical placement of the column contents so they align to the top of it.

Inside the second column, we have a Button component that has a modifier class, slds-button--neutral, which applies minimal styling.

Underneath the grid, the second row of the page header is a simple placeholder for a count of the number of items in the listview.

Filling Out the Rest of the List View Outline

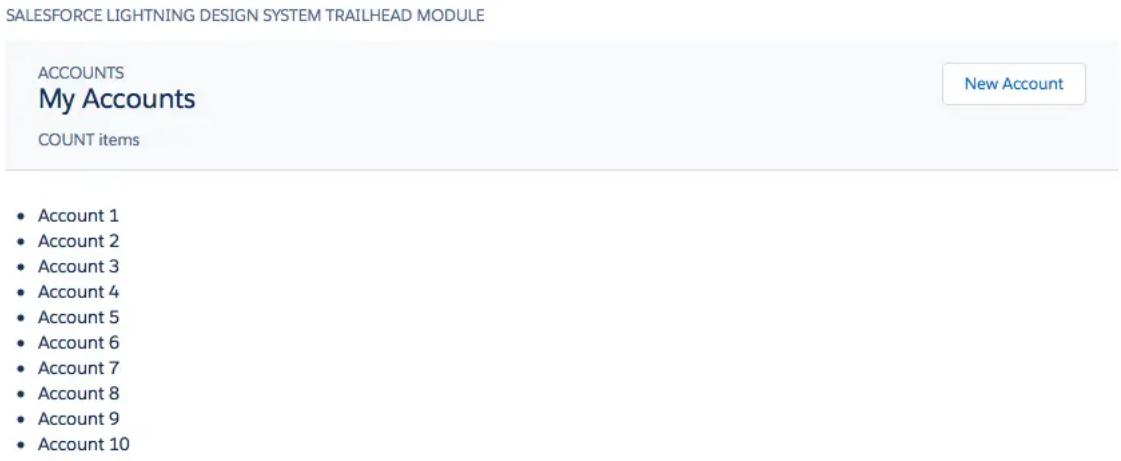
Next, we will add a simple unordered list as a placeholder for the glorious data table which is to come. Replace the `<!-- PRIMARY CONTENT WRAPPER -->` part of the markup with the following code:

```
<!-- PRIMARY CONTENT WRAPPER -->
<div class="myapp">
 <ul class="slds-list_dotted slds-m-top_large">
 <li>Account 1</li>
 <li>Account 2</li>
 <li>Account 3</li>
 <li>Account 4</li>
 <li>Account 5</li>
 <li>Account 6</li>
 <li>Account 7</li>
 <li>Account 8</li>
 <li>Account 9</li>
 <li>Account 10</li>
 </ul>
</div>
<!-- / PRIMARY CONTENT WRAPPER -->
```

Copy

Copy

Preview the page and you'll see the basic list view layout has taken form:



By default the content will fill the screen width. Use one of the slds-container_* helper classes if you would like to position your primary content horizontally on the screen or constrain it to a particular width.

Design System styling is applied to the `<ul>` with the slds-list_dotted and slds-m-top_large classes. Admittedly this is not very exciting primary content but just wait until the next unit…

Let's a take a brief aside. It is important to note that the Design System focuses on being an "application framework", rather than a "webpage framework". For this reason, we avoid adding extra styling by default. Instead the developer can choose to add additional styling manually such as in this example. This gives the developer much greater control over the exact layout, and can make a big difference where space is at a premium. As you get to know the Design System, you will see default styling assumptions are avoided on many other components such as **Button**.

To complete our list view layout skeleton, we'll add a footer using another Grid System component. You should be becoming an expert at this by now! We use a couple of additional modifier classes:

- slds-p-around_large adds padding to the footer element
- slds-grid_align-spread spreads out the grid columns on the main axis, with the first column starting at the farthest left and last item ending at the farthest right.

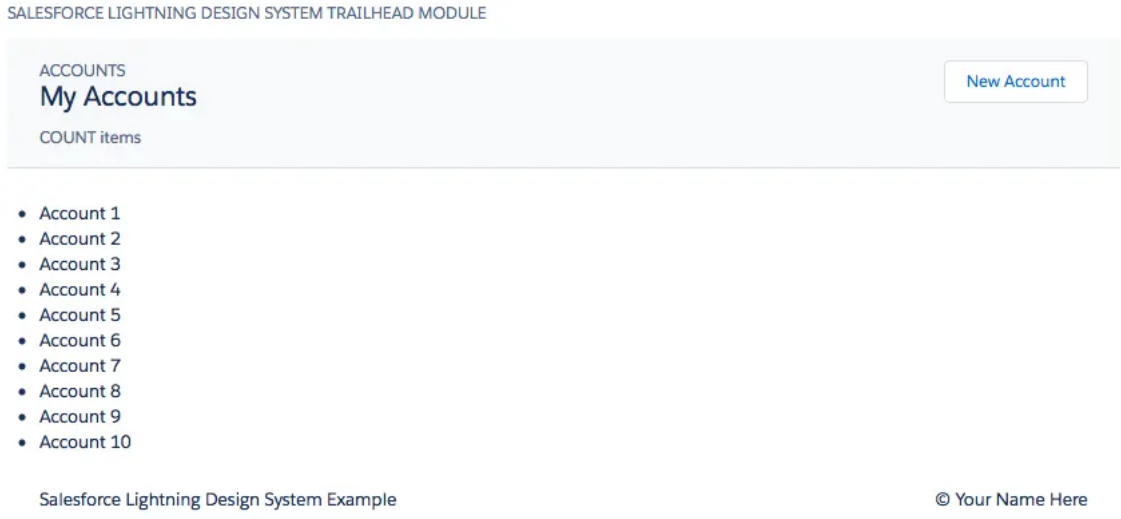
Here is the footer markup to go in between the `<!-- FOOTER -->` comments:

```
<!-- FOOTER -->
<footer role="contentinfo" class="slds-p-around_large">
 <!-- LAYOUT GRID -->
 <div class="slds-grid slds-grid_align-spread">
 <p class="slds-col">Salesforce Lightning Design System Example</p>
 <p class="slds-col">&copy; Your Name Here</p>
 </div>
 <!-- / LAYOUT GRID -->
</footer>
<!-- / FOOTER -->
```

Copy

Copy

Preview your page and you can see the list view taking shape. Note again how we didn't need to write any CSS to give us the latest Lightning UI styling. All we had to do was apply Design System classes in our HTML markup. In the next unit we'll hook in some real data.



One last note: remember that the Grid System may not always be the best component for implementing a particular layout. For example, it may be simpler just to use a Media Object or Tile component if you want an image and text side by side.

Resources

- *A Guide to Flexbox*

- *Grid System documentation*
- *Page Header documentation*

1In the Grid System component, the sizing class, slds-size--3-of-4, represents which of the following?

- A.75% of the available space.
- B.3 rows by 4 columns.
- C.4 columns by 3 rows.
- D.The 3rd of 4 columns.
- E.None of the above

2The Design System is primarily:

- A.For building web pages.
- B.For building applications.
- C.For building web pages and applications.
- D.None of the above.

3Which class should be applied to the outer wrapper of a Grid System component?

- A.gridsystem
- B.slds-grid--system
- C.grid
- D.slds-grid