**Lab04**

Objective

This lab looks at Semantic UI Containers, and the Semantic UI Grid System.

# Getting started with SemanticUI

We will link to the required Semantic UI files using CloudFlare, a content delivery network (CDN). Note that there are three links you need to include in each page in your website:

* Link to the SemanticUI CSS file, which should go in the <head> section of the page.
* Link to the SemanticUI JavaScript and jQuery files, which should go at the end of your HTML content, before the closing </body> tag.

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<meta http-equiv="X-UA-Compatible" content="IE=edge">

<meta name="viewport" content="width=device-width, initial-scale=1">

<title>Semantic UI</title>

<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/semantic-ui@2.4.2/dist/semantic.min.css">

</head>

<body>

<script src="https://code.jquery.com/jquery-3.1.1.min.js"

integrity="sha256-hVVnYaiADRTO2PzUGmuLJr8BLUSjGIZsDYGmIJLv2b8=" crossorigin="anonymous"></script>

<script src="https://cdn.jsdelivr.net/npm/semantic-ui@2.4.2/dist/semantic.min.js"></script>

</body>

</html>

# Semantic UI Containers

Containers provide the foundation for page layout. A container is an element designed to contain page elements to a reasonable maximum width based on the size of a user's screen. This is useful to couple with other UI elements like grid or menu to restrict their width to a reasonable size for display.

Semantic UI requires a containing element to wrap elements and contain its grid system (more on the grid system next). The Semantic UI container class was created specifically for this purpose.

## **Container Sizes**

Containers are designed to responsively adjust their maximum width based on the size of the screen on which they are appearing.

A screenshot of a cell phone

Description automatically generated

**Note:** There is another computer device (widescreen), which is for device width > 1920px.

## **UI Container**

An ui container is a fixed width element that wraps your site's content. It redivs a constant size and uses margin to center. Containers are the simplest way to center page content inside a grid.

<div class="ui container">

Example of a fixed width container which is set using the <code>.ui.container</code> class.

</div>

# Semantic UI Grid System

Grid systems enable you to create advanced layouts using rows and columns.

Semantic UI uses a **16 column grid**, where each column is of equal width with spaces in between. These spaces are called gutters. The use of grids is by far the most popular method for dividing and utilizing space. Regardless of which framework you choose to develop with, a solid understanding of the grid layout will allow you the most flexibility and control in your designs.

If you would like to divide your content into 4 sections across the page it's as simple as dividing the 16 total columns by 4, which equals 4. This means you'll need to specify that each column takes up a width of four in order to have four columns across the page.

A screenshot of a social media post

Description automatically generated

<div class="ui container">

<div class="ui grid">

<div class="four wide column">One</div>

<div class="four wide column">Two</div>

<div class="four wide column">Three</div>

<div class="four wide column">Four</div>

</div>

</div>

Here's another example of a Semantic UI grid: A screenshot of a cell phone

Description automatically generated

# Grid Columns and Rows

As we saw in our previous example, to create a grid structure we require the ui grid class. Using a grid makes content appear to flow more naturally on your page.

## **Columns**

Grids divide horizontal space into indivisible units called **columns**. All columns in a grid must specify their width as proportion of the total available row width.

All grid systems choose an arbitrary column count to allow per row. Semantic's default theme uses **16** columns.

The following example creates a grid with 6 columns (3 on the first line and 3 on the next line). The browser detects that the fourth column goes on the next line as the grid is a three column grid structure.

<div class="ui container">

<section class="ui three column grid">

<div class="ui column">

<h2>Frontend</h2>

<p>

...

</p>

</div>

<div class="ui column">

<h2>Serverside</h2>

<p>

...

</p>

</div>

<div class="ui column">

<h2>Databases</h2>

<p>

...

</p>

</div>

<div class="ui column">

<h2>Frontend</h2>

<p>

...

</p>

</div>

<div class="ui column">

<h2>Serverside</h2>

<p>

...

</p>

</div>

<div class="ui column">

<h2>Databases</h2>

<p>

...

</p>

</div>

</section>

</div>

## **Rows**

In the above example the new line is implicitly a new row. Most grids do not need to specify rows. Content will automatically flow to the next row when all the grid columns are taken in the current row.

We can explicitly specify a new row also. Rows are groups of columns which are aligned horizontally.

Rows can either be explicit, marked with an additional row element, or implicit, automatically occurring when no more space is left in a previous row.

<div class="ui container">

<section class="ui three column grid">

<section class="ui row">

<div class="ui column">

<h2>Frontend</h2>

<p>

...

</p>

</div>

<div class="ui column">

<h2>Serverside</h2>

<p>

...

</p>

</div>

<div class="ui column">

<h2>Databases</h2>

<p>

...

</p>

</div>

</section>

<section class="ui row">

<div class="ui column">

<h2>Frontend</h2>

<p>

...

</p>

</div>

<div class="ui column">

<h2>Serverside</h2>

<p>

...

</p>

</div>

<div class="ui column">

<h2>Databases</h2>

<p>

...

</p>

</div>

</section>

</section>

</div>

### **Divided Grid**

A grid can have dividers between its columns. It requires the ui row class.

<div class="ui three column divided grid">

<section class="ui row">

...

</section>

</div>

### **Vertically Divided Grid**

A grid can have dividers between rows. It also requires the ui row class.

<div class="ui three column vertically divided grid">

<section class="ui row">

...

</section>

</div>

### **Celled Grid**

A grid can have rows divided into cells.It also requires the ui row class.

<div class="ui three column celled grid">

<section class="ui row">

...

</section>

</div>

# Grid Sizes

We can vary the sizes of the columns in the grid and the columns do not have to be even sized. Column widths can be specified using (x) wide class names. If a column cannot fit in a row it will automatically flow to the next row. The following examples produce different sized columns.

<div class="ui container">

<div class="ui grid">

<div class="four wide column">Column A</div>

<div class="four wide column">Column B</div>

<div class="four wide column">Column C</div>

<div class="four wide column">Column D</div>

<div class="two wide column">Column E</div>

<div class="eight wide column">Column F</div>

<div class="six wide column">Column G</div>

</div>

</div>

<div class="ui container">

<div class="ui grid">

<div class="eight wide column">Column A</div>

<div class="eight wide column">Column B</div>

<div class="ten wide column">Column C</div>

<div class="six wide column">Column D</div>

<div class="four wide column">Column E</div>

<div class="twelve wide column">Column F</div>

<div class="two wide column">Column G</div>

<div class="fourteen wide column">Column H</div>

<div class="sixteen wide column">Column I</div>

<div class="twelve wide column">Column J</div>

<div class="ten wide column">Column K</div>

</div>

</div>

## **Nesting Grids**

Grids can be placed inside of other grids, letting you sub-divide columns.

<div class="ui container">

<div class="ui two column grid">

<div class="column">

<div class="ui three column grid">

<div class="column">Column A</div>

<div class="column">Column B</div>

<div class="column">Column C</div>

</div>

</div>

<div class="column">Column D</div>

<div class="column">Column E</div>

<div class="column">

<div class="ui grid">

<div class="ten wide column">Column F</div>

<div class="six wide column">Column G</div>

</div>

</div>

</div>

</div>

## **Coloured**

Grids can use named colours variations to add background colours, but only with padded grid that do not include negative margins.

To include a colour that is not available in the default palette its perfectly fine to use CSS.

<div class="ui container">

<div class="ui equal width center aligned padded grid">

<div class="olive column">

Olive

</div>

<div class="black column">

Black

</div>

<div class="column" style="background-color: #869D05;color: #FFF;">

Custom Column

</div>

<div class="black column">

Black

</div>

<div class="olive column">

Olive

</div>

</div>

</div>

## **Automatic Column Count**

The equal width variation will automatically divide column width evenly. This is useful with dynamic content where you do not know the column count in advance.

We can see this used in the previous example.

<div class="ui container">

<div class="ui equal width center aligned padded grid">

...

</div>

</div>

## **Centering Content**

If a row does not take up all sixteen grid columns, you can use a centered variation to center the column contents inside the grid.

<div class="ui container">

<div class="ui two column centered grid">

<div class="column">Column A</div>

<div class="four column centered row">

<div class="column">Column B</div>

<div class="column">Column C</div>

</div>

</div>

</div>

## **Significant Word Order**

Grids include many variations for adjusting things like vertical or horizontal alignment, text alignment, or default gutter sizes.

Some multi-word variations, like left floated or right aligned or middle aligned are word-order dependent and require you to use adjacent class names.

<div class="ui container">

<div class="ui grid">

<div class="ui four column row">

<div class="left floated column">Column A</div>

<div class="right floated column">Column B</div>

</div>

<div class="ui row">

<div class="three wide column">Column C</div>

<div class="eight wide column">Column D</div>

<div class="five wide column">Column E</div>

</div>

</div>

</div>

The following example uses floated and alignment options:

<div class="ui container">

<div class="ui grid">

<div class="left floated right aligned six wide column">

Left floated right aligned column

</div>

<div class="right floated left aligned six wide column">

Right floated left aligned column

</div>

<div class="two column center aligned row">

<div class="column">

Center aligned row

</div>

<div class="column">

Center aligned row

</div>

</div>

<div class="sixteen wide column">

Left Aligned Grid

</div>

<div class="sixteen wide right aligned column">

Right Aligned Grid

</div>

</div>

</div>

The following example illustrates an example using middle aligned:

<div class="ui container">

<div class="ui center aligned middle aligned grid ">

<div class="column">

<h1>My DIV</h1>

</div>

</div>

</div>

The grid has a specified height assigned.

.ui.grid {

height: 500px;

}

# Responsive Grids

## **Containers**

A container can be used alongside a grid to provide a responsive, fixed width container for wrapping the contents of a page.

<div class="ui grid container">

<div class="four wide column">Column A</div>

<div class="four wide column">Column B</div>

<div class="four wide column">Column C</div>

<div class="four wide column">Column D</div>

<div class="four wide column">Column E</div>

<div class="four wide column">Column F</div>

<div class="four wide column">Column G</div>

<div class="four wide column">Column H</div>

</div>

## **Stackable**

A stackable grid will automatically stack rows to a single columns on mobile devices.

<div class="ui container">

<div class="ui stackable four column grid">

<div class="column">Column A</div>

<div class="column">Column B</div>

<div class="column">Column C</div>

<div class="column">Column D</div>

</div>

</div>

## **Reverse Order**

Semantic includes special reversed variations that allow you to reverse the order of columns or rows by device.

In the following example, the order is reversed for mobile devices:

<div class="ui grid container">

<div class="ui mobile reversed equal width row">

<div class="column">

First

</div>

<div class="column">

Second

</div>

<div class="column">

Third

</div>

</div>

</div>

In the following example, the order is reversed for both tablet and mobile devices:

<div class="ui grid container">

<div class="ui mobile reversed tablet reversed equal width row">

<div class="column">

First

</div>

<div class="column">

Second

</div>

<div class="column">

Third

</div>

</div>

</div>

## **Doubling**

A doubling grid will double column widths for each device jump.

<div class="ui center aligned grid container">

<div class="doubling eight column row">

<div class="column"><p><span>1</span></p></div>

<div class="column"><p><span>2</span></p></div>

<div class="column"><p><span>3</span></p></div>

<div class="column"><p><span>4</span></p></div>

<div class="column"><p><span>5</span></p></div>

<div class="column"><p><span>6</span></p></div>

<div class="column"><p><span>7</span></p></div>

<div class="column"><p><span>8</span></p></div>

</div>

<div class="doubling six column row">

<div class="column"><p><span>1</span></p></div>

<div class="column"><p><span>2</span></p></div>

<div class="column"><p><span>3</span></p></div>

<div class="column"><p><span>4</span></p></div>

<div class="column"><p><span>5</span></p></div>

<div class="column"><p><span>6</span></p></div>

</div>

<div class="doubling four column row">

<div class="column"><p><span>1</span></p></div>

<div class="column"><p><span>2</span></p></div>

<div class="column"><p><span>3</span></p></div>

<div class="column"><p><span>4</span></p></div>

</div>

</div>

## **Manual Tweaks**

Although design patterns like doubling or stackable are useful at simplifying responsive styling, you can also manually tweak device presentation by specifying (x) wide device or device only columns or rows.

<div class="ui centered stackable grid container">

<div class="computer only row">

<div class="column">Column 1</div>

</div>

<div class="five wide tablet eight wide computer column">Column A</div>

<div class="five wide tablet eight wide computer column">Column B</div>

<div class="five wide tablet eight wide computer column">Column C</div>

<div class="five wide tablet eight wide computer column">Column D</div>

<div class="five wide tablet eight wide computer column">Column E</div>

</div>

<div class="ui center aligned grid container">

<div class="widescreen only row">

<div class="column"><p><span>Widescreen</span></p></div>

</div>

<div class="large screen only row">

<div class="column"><p><span>Large Screen</span></p></div>

</div>

<div class="computer only row">

<div class="column"><p><span>Computer</span></p></div>

</div>

<div class="tablet computer only row">

<div class="column"><p><span>Tablet &amp; Computer</span></p></div>

</div>

<div class="two column tablet only row">

<div class="column"><p><span>Tablet</span></p></div>

<div class="column"><p><span>Tablet</span></p></div>

</div>

<div class="three column mobile only row">

<div class="column"><p><span>Mobile</span></p></div>

<div class="column"><p><span>Mobile</span></p></div>

<div class="column"><p><span>Mobile</span></p></div>

</div>

<div class="three wide mobile six wide tablet twelve wide computer column sixteen wide widescreen column"><p><span>Different width based on screen size</span></p></div>

</div>

Semantic UI Grid - Exercise One

* open the Lab04\_Material\_A folder.
* A web page has been created for you, add Semantic UI code that will lay out the web page as follows:

A screenshot of a cell phone

Description automatically generated

You will notice that when you resize the browser to small viewport size, the 4 columns of text appear very narrow.

* Adjust the code for the 4 columns such that the columns of text are size 8 on very small viewports (mobile), and size 4 on all other devices.

A screenshot of text

Description automatically generated

# Semantic UI Grid - Exercise Two

* open the Lab04\_Material\_B folder in Week09.
* A web page has been created for you. Add the necessary tags to render index.html as follows:

A screenshot of a social media post

Description automatically generated

Now, adjust the necessary tags to render index.html as follows when the screen viewport is at mobile viewport:

A screenshot of a social media post

Description automatically generated