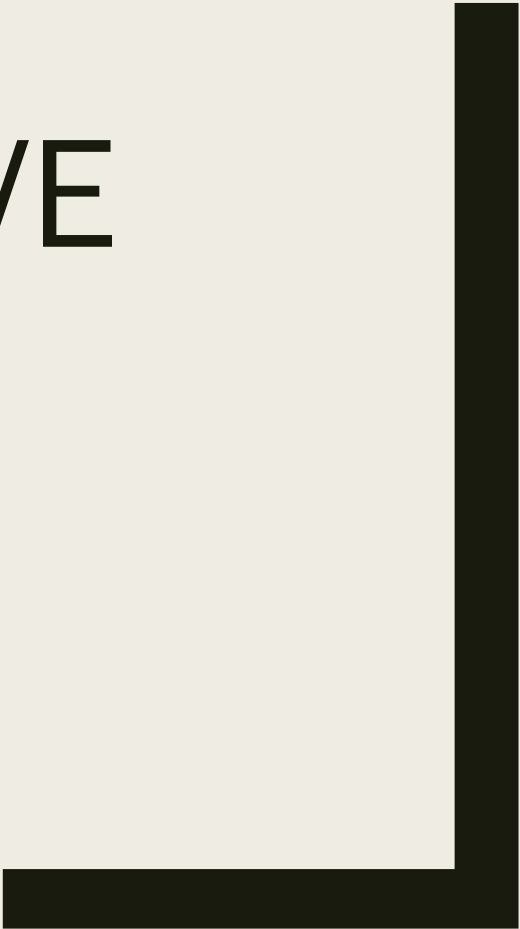


# RESPONSIVE DESIGN



# Responsive WebApps

- We would like our webapps to display well on devices that have limited screen resources.
- This is driven by the rise of mobile devices (smart phones)
  - *Mobile users tend to be more comfortable scrolling vertically as opposed to scrolling horizontally.*
  - *Compare that with a large form screen where we want to effectively use the width of the screen.*
- A design that looks good on a large scale device can have issues on a small scale device and vice verse.

# Responsive WebApps

- Why not just create a native mobile app?
  - *Excellent use of the mobile devices capabilities but expensive to maintain a mobile version (maybe two if you develop an app for both android and iOS) and a web based app.*
- Why not create a special set of pages for the small screen device?
  - *Twice as many pages to maintain and worry about consistency. More affordable*
- Why not keep the content in one place and apply different CSS for to respond to the device size?
  - *Most affordable and maintainable. Look on small screen may make the app harder to use.*

# FlexBox example

```
<!DOCTYPE html>
<html> <head> <style>
    .flex-container {
        display: flex; flex-direction: row;
        background-color: red;
    }

    .flex-container > div { /*div children of the flex
container*/
        background-color: #f1f1f1;
        margin: 10px; padding: 20px; font-size: 30px;
    }

    /* Responsive layout */
    @media (max-width: 700px) {
        .flex-container {
            flex-direction: column;
        }
    }
</style></head>
```

Trigger this rule at width  
Less than 700px

# FlexBox example

```
<body>
<div class="flex-container">
  <div>row item 1 with box sized to fit</div>
  <div>row item 2</div>
  <div>row item 3</div>
</div>
```

<p> We have a parent element with class flex-container holding children items. Div is used to break up an HTML document into divisions, but is a common target to hold other things that can be styled with CSS. In this case we have 3 items in our container. As the width is decreased a point will be reached where the 3 items are stacked vertically instead of horizontally. FlexBox is primarily used to linearly lay out items. </p>

<p>Direct child element(s) of the flexible container automatically becomes flexible items.</p>

```
</body> </html>
```

# Grid Example

```
<!DOCTYPE html>
<html> <head> <style>
.my-header { grid-area: header; }
.my-menu { grid-area: menu; }
.my-content { grid-area: main; }
.my-right-bar { grid-area: right; }
.my-footer { grid-area: footer; }

.grid-container {
  display: grid;
  grid-template-areas:
    'header header header header header'
    'menu main main main right'
    'menu main main main right'
    'footer footer footer footer footer';
  grid-gap: 10px;
  background-color: rgba(255, 174, 255, 0.8);
  padding: 10px;
}
```

# Grid Example

```
.grid-container > div {  
  background-color: rgba(255, 255, 255, 0.8);  
  text-align: center;  
  padding: 20px 0;  
  font-size: 30px;  
}  
</style> </head>
```

```
<body>  
<h1>Grid Layout</h1>
```

```
<p>This grid layout has 4 rows and five columns  
All of the items and the container are div  
elements and will  
be placed in the grid according to the template.  
</p>
```





# Bootstrap

- Bootstrap sets certain things for the basic HTML elements and its own defined classes
- Ex: Containers have 16px padding left/right and 0 padding top/bottom.
- Ex: Heading sizes are set relative to the base size. So h1 is 2.5rem which is 250% of the base.
- There are differences between the versions of Bootstrap. Bootstrap 5 has been available and stable for a while now, so that is the version we will use.
  - *New in Bootstrap 5 is that it no longer uses JQuery.*
  - *SHA codes are provided so that we can be more confident that what we get has not been tampered with.*

# Bootstrap – The pull

- We are going to pull in a lot of styles and code that we can use. Knowing what is available, is the key.

```
<html lang="en">
<head>
  <title>Bootstrap Example</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css"
    rel="stylesheet"
    integrity="sha384-EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCOmLASjC"
    crossorigin="anonymous">
  <script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.9.2/dist/umd/popper.min.js"
    integrity="sha384-IQsoLX15PILFhosVNubq5LC7Qb9DXgDA9i+tQ8Zj3iwWAwPtgFTxbJ8NT4GN1R8p"
    crossorigin="anonymous"></script>
  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.min.js"
    integrity="sha384-cVKIPhGWiC2Al4u+LWgxfKTRIcfu0JTxR+EQDz/bglldoEyl4H0zUF0QKbrJ0EcQF"
    crossorigin="anonymous"></script>
</head>
```

Viewport is needed so we know the area and initially we use what is there.

The script files are JavaScript.

# Bootstrap – The content

- We are going to use divs as convenient stylable containers.
- Root must have class container (fixed width) or container-fluid(full width)

```
<div class="container-fluid">  
  <h1>Responsive Columns</h1>  
  <p>Resize the browser window to see the effect.</p>  
  <p>The columns will automatically stack on top of each other when the screen is less than 576px  
  wide.</p>
```

```
<div class="row">
```

```
<div class="col-sm-2" style="background-color:lavender;">.col-sm-2 this will span 2 columns and is  
contained inside a div element</div>
```

```
<div class="col-sm-3" style="background-color:lavenderblush;">.col-sm-3</div>
```

```
<div class="col-sm-3" style="background-color:lavender;">.col-sm-3</div>
```

```
<div class="col-sm-3" style="background-color:lavenderblush;">.col-sm-3</div>  
</div>  
</div>
```

The items are stacked in a horizontal row, but can stack vertically responsively to the width of the view port .

# Sizing – Bootstrap Grid

- We design with 12 columns in mind. Element each element can span up to twelve columns and the total number of columns that the elements in a row can span is twelve or less.
- Each column is a share of the total available width.
- If the width is small enough, then the row will start to stack elements vertically.
- How to read a column style class: `col-sm-2`
  - *Horizontal good for this size or larger*
    - sm – 576px
    - md – 768px
    - lg – 992px
    - xl – 1200px
    - xxl – 1400px
  - *How many columns do we span?*
- 12 columns is convenient because it is easy to do 1, 2, 3, 4, and 6 equally weighted spans.

# Tables

- Bootstrap defines a number of style classes that can be applied to a table.
- The basic table styling is the class “table”. We can add in extra styling.

```
<table class="table table-striped table-bordered">
```

Gives the basic table styling, zebra stripes the table and applies a border to all elements.

# Buttons

- Bootstrap allows us to style buttons. (At this point we have not talked about how to make the buttons respond to a click yet.)
- Ex: Make a responsive row of buttons. We have other add in styles.

```
<div class="container">  
  <button type="button" class="btn btn-success">Success</button>  
  <button type="button" class="btn btn-warning">Warning</button>  
  <button type="button" class="btn btn-danger">Danger</button>  
  <button type="button" class="btn btn-link">Link</button>  
</div>
```

# Progress Bars

- Ex: We have a progress bar. No code – no change in the percentage.

```
<div class="progress">  
  <div class="progress-bar" style="width:40%"></div>  
</div>
```

# Spinners

- Ex: A spinner. We can use the text color utility to give the spinner a color. (Warning is yellow. Check out the other 7.)

```
<div class="container">  
  <p> Here is a spinner! </p>  
  <div class="spinner-border text-warning"></div>  
</div>
```



# Buttons with Spinners

- This gives us a button with a spinner and text.
- Uses a span element.

```
<button class="btn btn-danger">  
  <span class="spinner-border"></span>  
  Launch in 5  
</button>
```

# List groups

- Basically how unordered lists are styled.
- `class="list-group"` can be applied to a `<ul>` or to a `<div>` holding links.
- `class = "list-group-item"` is used with `<li>` or `<a>`.
- Add `list-group-horizontal` to the `list-group` class to display items in a row.

# Dropdown Menu

No actions yet, but can link out.

```
<div class="container">
  <div class="dropdown">
    <button type="button" class="btn btn-primary dropdown-toggle" data-
toggle="dropdown">
      Dropdown button
    </button>
    <div class="dropdown-menu">
      <div class="dropdown-item">First</div>
      <div class="dropdown-item">Second</div>
      <a class="dropdown-item" href="#">Link 3</a>
    </div>
  </div>
</div>
```

# Material Design

- [Material Design](#)
- Google created Material Design to standardize a high quality set of components, tools and guidelines for the creation of interfaces.
- Used in web and android.
- iOS in comparison tends to prefer a flatter style for components. (Less shadow, shading and color.)
- Can be combined with Bootstrap.

# References

- [BootStrap](#)
  - [List of bootstrap components](#) *Things like buttons, cards, and carousels.*
  - [List of bootstrap utilities](#) *Things like borders, shadow, and flex*
- [Material Design](#)
- [BootStrap & Material Design](#)