Agenda:

- Forms in Bootstrap
- Flexbox in Bootstrap
- Introduction to Tailwind css

In this session we are going to learn about Forms and Flexbox in Bootstrap and have an introduction about another UI Library called Tailwind.

Forms:

We can create responsive and consistently styled forms using bootstrap form classes.

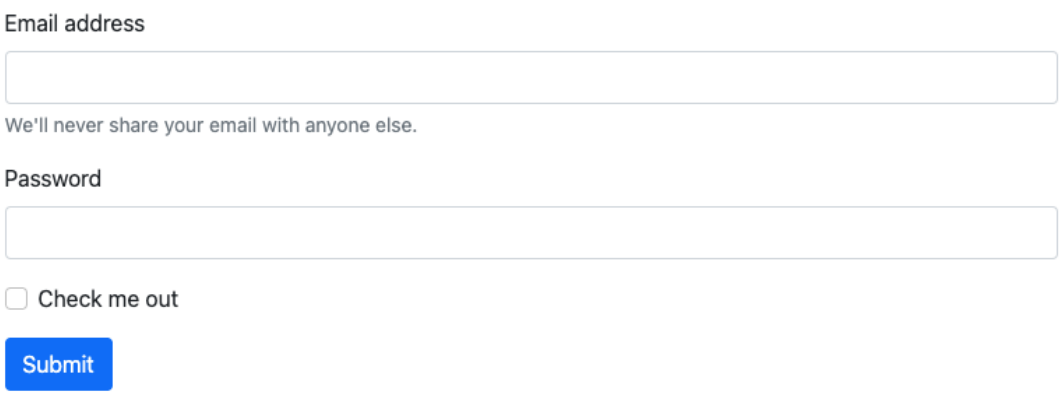
Bootstrap's form controls expand on our rebooted form styles with classes. Use these classes to opt into their customized displays for a more consiste rendering across browsers and devices.

Be sure to use an appropriate `type` attribute on all inputs (e.g., `email` for email address or `number` for numerical information) to take advantage newer input controls like email verification, number selection, and more.

Consider the example shown below:

```
<form>
  <div class="mb-3">
    <label for="exampleInputEmail1" class="form-label">Email address</label>
    <input type="email" class="form-control" id="exampleInputEmail1"
           aria-describedby="emailHelp">
    <div id="emailHelp" class="form-text">We'll never share your email
           with anyone else.</div>
  </div>
  <div class="mb-3">
    <label for="exampleInputPassword1" class="form-label">Password</label>
    <input type="password" class="form-control" id="exampleInputPassword1">
  </div>
  <div class="mb-3 form-check">
    <input type="checkbox" class="form-check-input" id="exampleCheck1">
    <label class="form-check-label" for="exampleCheck1">Check me out</label>
  </div>
  <button type="submit" class="btn btn-primary">Submit</button>
</form>
```

Output:



There are different additional components within the form element in bootstrap let's discuss about them one by one:

Form Control:

We can give textual form controls like `<input>` s and `<textarea>` s an upgrade with custom styles, sizing, focus states, and more.

We can set heights using classes like `.form-control-lg` and `.form-control-sm` .

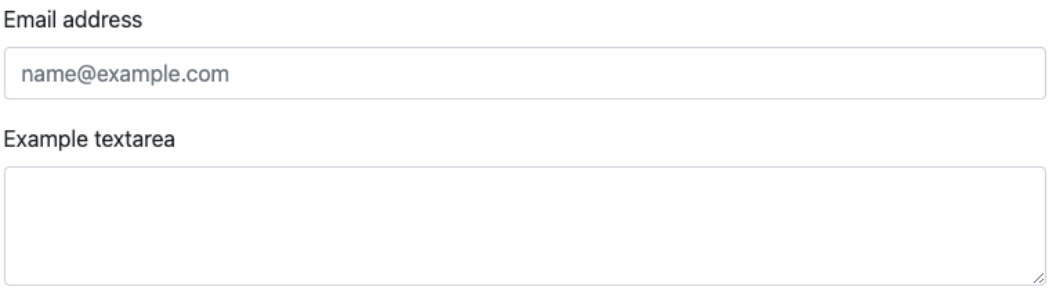
We can add the `disabled` boolean attribute on an input to give it a grayed out appearance and remove pointer events.

We can also add the `readonly` boolean attribute on an input to prevent modification of the input's value.

Consider the example shown below:

```
<div class="mb-3">
  <label for="exampleFormControlInput1" class="form-label">Email address</label>
  <input type="email" class="form-control" id="exampleFormControlInput1"
     placeholder="name@example.com">
</div>
<div class="mb-3">
  <label for="exampleFormControlTextarea1" class="form-label">
    Example textarea</label>
  <textarea class="form-control" id="exampleFormControlTextarea1"
    rows="3"></textarea>
</div>
```

Ouput:



Select:

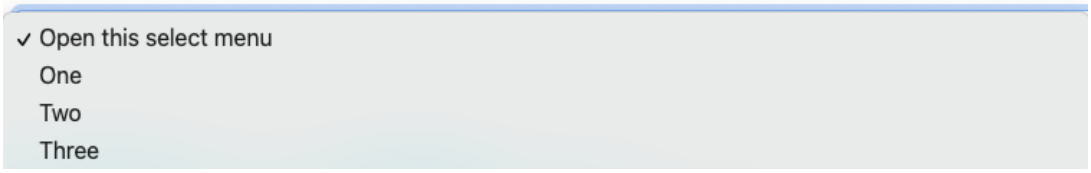
We can customize the the native `<select>` s with custom CSS that changes the element's initial appearance.

Custom `<select>` menus need only a custom class, `.form-select` to trigger the custom styles. Custom styles are limited to the `<select>` initial appearance and cannot modify the `<option>` s due to browser limitations.

Consider the example shown below:

```
<select class="form-select" aria-label="Default select example">
  <option selected>Open this select menu</option>
  <option value="1">One</option>
  <option value="2">Two</option>
  <option value="3">Three</option>
</select>
```

Output:



Input Group:

Using input-group class we can easily extend form controls by adding text, buttons, or button groups on either side of textual inputs, custom selects, ar custom file inputs.

Consider the example shown below:

```
<div class="input-group mb-3">
  <span class="input-group-text" id="basic-addon1">@</span>
  <input type="text" class="form-control" placeholder="Username"
        aria-label="Username" aria-describedby="basic-addon1">
</div>
```

Output:

Flexbox in Bootstrap :

The Flexible Box Layout Module, makes it easier to design flexible responsive layout structure without using float or positioning.

We can add the following classes :

- To create a flexbox container and to transform direct children into flex items, use the `d-flex` class.
- To create an inline flexbox container, use the `d-inline-flex` class.
- Use `.flex-row` to display the flex items horizontally (side by side). This is the default behaviour.
- Use `.flex-column` to display the flex items vertically (on top of each other), or `.flex-column-reverse` to reverse the vertical direction.
- Use the `.justify-content-*` classes to change the alignment of flex items. Valid classes are `start` (default), `end`, `center`, `between` or `around`.
- Use `.flex-fill` on flex items to force them into equal widths.
- Use `.flex-grow-1` on a flex item to take up the rest of the space.
- We can change the visual order of a specific flex item(s) with the `.order` classes. Valid classes are from 0 to 5, where the lowest number has highest priority (order-1 is shown before order-2, etc..).
- We can control how flex items wrap in a flex container with `.flex-nowrap` (default), `.flex-wrap` or `.flex-wrap-reverse`.
- We can also control the vertical alignment of **gathered** flex items with the `.align-content-*` classes. Valid classes are `.align-content-start` (default), `.align-content-end`, `.align-content-center`, `.align-content-between`, `.align-content-around` and `.align-content-stretch`.
- We can control the vertical alignment of **single rows** of flex items with the `.align-items-*` classes. Valid classes are `.align-items-start`, `.align-items-end`, `.align-items-center`, `.align-items-baseline`, and `.align-items-stretch`.
- We can control the vertical alignment of **a specified flex item** with the `.align-self-*` classes. Valid classes are `.align-self-start`, `.align-self-end`, `.align-self-center`, `.align-self-baseline`, and `.align-self-stretch`.

Introduction to Tailwind css :

Tailwind CSS is a utility-based framework based on CSS. It provides a catalog of CSS classes that makes the process of styling more convenient.

Tailwind is a bit different from frameworks like Bootstrap in that it's not a UI kit.

It comes with a menu of predesigned widgets to build your site with, but doesn't impose design decisions that are difficult to undo.

Let's try to understand it with container component that Tailwind css has:

Container :

Note that unlike container component in bootstrap, Tailwind's container is a componet for fixing an element's width to the current breakpoint.

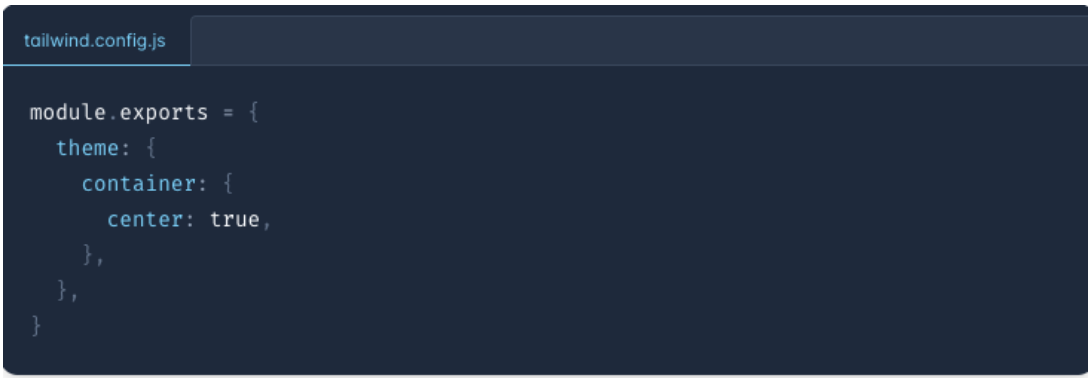
The `container` class sets the `max-width` of an element to match the `min-width` of the current breakpoint. This is useful if you'd prefer to desig for a fixed set of screen sizes instead of trying to accommodate a fully fluid viewport.

The `container` class also includes responsive variants like `md:container` by default that allow you to make something behave like a container only a certain breakpoint and up.

Consider the example below:

```
<!-- Full-width fluid until the `md` breakpoint, then lock to container -->
<div class="md:container md:mx-auto">
  <!-- ... -->
</div>
```

In Tailwind css, we can define our own customized theme instead of the default theme. For example, To center containers by default, s the `center` option to `true` in the `theme.container` section of your config file:



There are so many other things in tailwind css that you can find here : https://tailwindcss.com/docs/

Interview Questions

> In Bootstrap 4, what is flexbox?

Flexbox is a layout module for flexible boxes. Without using float or positioning, you can quickly create a flexible layout design with flexbox

> Explain the concept of creating a basic form in Bootstrap.

First, add a
element. Then, inside the form element, wrap labels and controls in a

element with the .form-group class. Next, the .form-control class to text input elements like ,

---

Thank You !

Thank You !