Bootstrap Forms

In this tutorial you will learn how to create elegant forms with Bootstrap.

Creating Forms with Bootstrap

HTML forms are the integral part of the web pages and applications, but creating the form layouts or styling the form controls manually one by one using CSS are often boring and tedious. Bootstrap greatly simplifies the process of styling and alignment of form controls like labels, input fields, selectboxes, textareas, buttons, etc. through predefined set of classes.

Bootstrap provides three different types of form layouts:

- Vertical Form (default form layout)
- Horizontal Form
- Inline Form

The following section will give you the detailed overview of all these form layouts as well as the various form related Bootstrap components one by one. Well let's get started.

Creating Vertical Form Layout

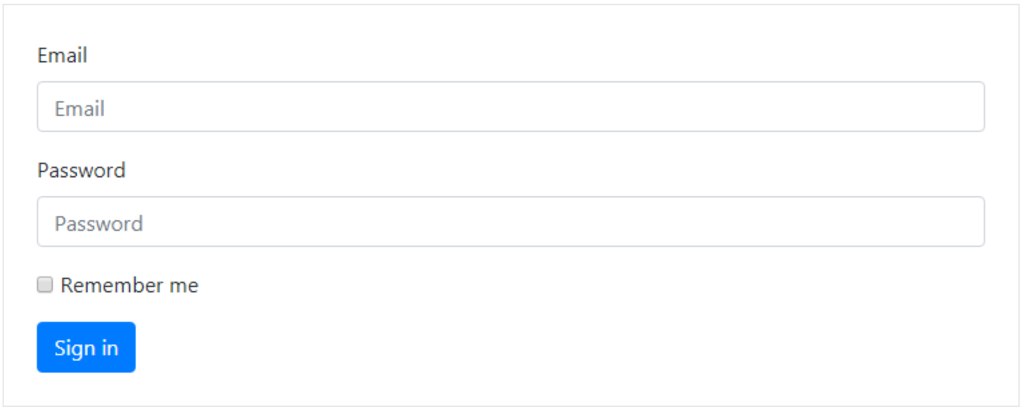
This is the default Bootstrap form layout in which styles are applied to form controls without adding any base class to the `<form>` element or any large changes in the markup.

The form controls in this layout are stacked with left-aligned labels on the top.

Example | Try this code »

```
<form>
    <div class="form-group">
        <label for="inputEmail">Email</label>
        <input type="email" class="form-control" id="inputEmail" placeholder="Email">
    </div>
    <div class="form-group">
        <label for="inputPassword">Password</label>
        <input type="password" class="form-control" id="inputPassword"
placeholder="Password">
    </div>
    <div class="form-group">
        <label class="form-check-label"><input type="checkbox"> Remember me</label>
    </div>
    <button type="submit" class="btn btn-primary">Sign in</button>
</form>
```

— The output of the above example will look something like this:



Note: All textual form controls, like `<input>`, `<textarea>`, and `<select>` require the class `.form-control` for general styling. The `.form-control` class also makes them 100% wide. To change their width or use them inline, you can utilize the predefined grid classes.

Creating Horizontal Form Layout

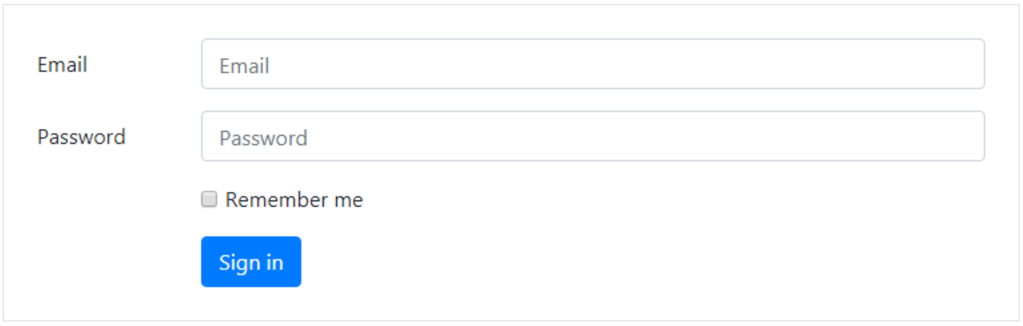
You can also create horizontal form layouts where labels and form controls are aligned side-by-side using the Bootstrap grid classes. To create a horizontal form layout add the class `.row` on form groups and use the `.col-*-*` grid classes to specify the width of your labels and controls.

Also, be sure to apply the class `.col-form-label` on the `<label>` elements, so that they're vertically centered with their associated form controls. Let's check out an example:

Example Try this code »

```
<form>
    <div class="form-group row">
        <label for="inputEmail" class="col-sm-2 col-form-label">Email</label>
        <div class="col-sm-10">
            <input type="email" class="form-control" id="inputEmail"
placeholder="Email">
        </div>
    </div>
    <div class="form-group row">
        <label for="inputPassword" class="col-sm-2 col-form-label">Password</label>
        <div class="col-sm-10">
            <input type="password" class="form-control" id="inputPassword"
placeholder="Password">
        </div>
    </div>
    <div class="form-group row">
        <div class="col-sm-10 offset-sm-2">
            <label class="form-check-label"><input type="checkbox"> Remember
me</label>
        </div>
    </div>
    <div class="form-group row">
        <div class="col-sm-10 offset-sm-2">
            <button type="submit" class="btn btn-primary">Sign in</button>
        </div>
    </div>
</form>
```

— The output of the above example will look something like this:



Creating Inline Form Layout

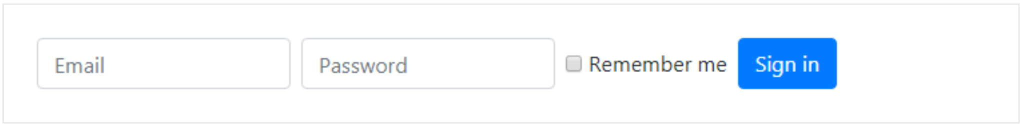
Sometimes you may want to display a series of labels, form controls, and buttons on a single horizontal row to compact the layout. You can do this easily by adding the class `.form-inline` to the `<form>` element. However, form controls only appear inline in viewports that are at least 576px wide.

Let's take a look at following example to see how it actually works:

Example **Try this code »**

```html
<form class="form-inline">
    <div class="form-group mr-2">
        <label class="sr-only" for="inputEmail">Email</label>
        <input type="email" class="form-control" id="inputEmail" placeholder="Email">
    </div>
    <div class="form-group mr-2">
        <label class="sr-only" for="inputPassword">Password</label>
        <input type="password" class="form-control" id="inputPassword" placeholder="Password">
    </div>
    <div class="form-group mr-2">
        <label><input type="checkbox" class="mr-1"> Remember me</label>
    </div>
    <button type="submit" class="btn btn-primary">Sign in</button>
</form>
```

— The output of the above example will look something like this:



Note: It is recommended to include a label for every form inputs otherwise screen readers will have trouble with your forms. However, in case of inline form layouts you can hide the labels using the `.sr-only` class, so that only screen readers can read it.

Check out the snippets section for examples of some beautifully designed Bootstrap forms.

Creating Static Form Control

There might be a situation when you just want to display a plain text value next to a form label instead of a working form control. You can do this easily by replacing the class `.form-control` with the `.form-control-plaintext` and applying the attribute `readonly`.

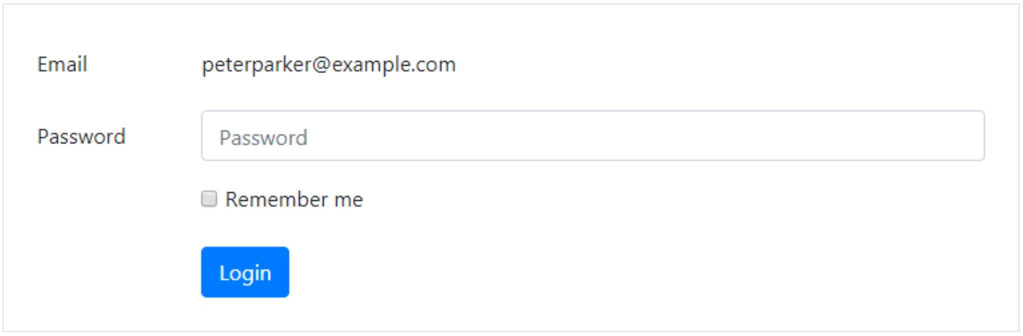
The `.form-control-plaintext` class removes the default styling from the form field, but preserves the correct margin and padding. Let's take a look at an example:

Example

Try this code »

```html
<form>
    <div class="form-group row">
        <label for="inputEmail" class="col-sm-2 col-form-label">Email</label>
        <div class="col-sm-10">
            <input type="text" readonly class="form-control-plaintext"
id="inputEmail" value="peterparker@example.com">
        </div>
    </div>
    <div class="form-group row">
        <label for="inputPassword" class="col-sm-2 col-form-label">Password</label>
        <div class="col-sm-10">
            <input type="password" class="form-control" id="inputPassword"
placeholder="Password">
        </div>
    </div>
    <div class="form-group row">
        <div class="col-sm-10 offset-sm-2">
            <div class="checkbox">
                <label><input type="checkbox"> Remember me</label>
            </div>
        </div>
    </div>
    <div class="form-group row">
        <div class="col-sm-10 offset-sm-2">
            <button type="submit" class="btn btn-primary">Sign in</button>
        </div>
    </div>
</form>
```

— The output of the above example will look something like this:



Placement of Checkboxes and Radios

Checkboxes and radio buttons can be placed either *stacked* or *inline*.

Stacked Checkboxes and Radios

To place the checkboxes or radio buttons vertically stacked i.e. line by line, just wrap all controls in a form group and apply the class `.d-block` on each `<label>`. Additionally, use the margin utility classes for proper spacing, as shown in the following example:

Example

Try this code »

```html
<!-- Vertically stacked checkboxes -->
<div class="form-group">
    <label class="d-block">
        <input type="checkbox" class="mr-1" name="sports"> Cricket
    </label>
    <label class="d-block">
        <input type="checkbox" class="mr-1" name="sports"> Football
    </label>
    <label class="d-block">
        <input type="checkbox" class="mr-1" name="sports"> Tennis
    </label>
</div>
<!-- Vertically stacked radios -->
<div class="form-group">
    <label class="d-block">
        <input type="radio" class="mr-1" name="gender"> Male
    </label>
    <label class="d-block">
        <input type="radio" class="mr-1" name="gender"> Female
    </label>
</div>
```

Inline Checkboxes and Radios

However, to place them inline i.e. side-by-side just place all form controls in a form group and use margin utility classes to ensure proper spacing. No need to use the `.d-block` class on the `<label>` element in this case. Let's check out the following example:

Example **Try this code »**

```
<!-- Inline checkboxes -->
<div class="form-group">
    <label class="mr-3">
        <input type="checkbox" class="mr-1" name="sports"> Cricket
    </label>
    <label class="mr-3">
        <input type="checkbox" class="mr-1" name="sports"> Football
    </label>
    <label class="mr-3">
        <input type="checkbox" class="mr-1" name="sports"> Tennis
    </label>
</div>
<!-- Inline radios -->
<div class="form-group">
    <label class="mr-3">
        <input type="radio" class="mr-1" name="gender"> Male
    </label>
    <label class="mr-3">
        <input type="radio" class="mr-1" name="gender"> Female
    </label>
</div>
```

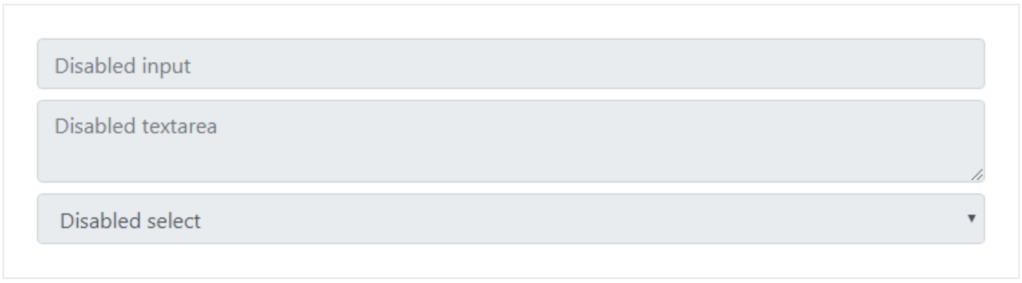
Creating Disabled Form Controls

To disable individual form controls such as `<input>`, `<textarea>`, `<select>` just add the attributes `disabled` to them and Bootstrap will do the rest. Here's an example:

Example **Try this code »**

```
<input type="text" class="form-control mb-2" placeholder="Disabled input" disabled>
<textarea class="form-control mb-2" placeholder="Disabled textarea" disabled>
</textarea>
<select id="disabledSelect" class="form-control" disabled>
    <option>Disabled select</option>
</select>
```

— The output of the above example will look something like this:

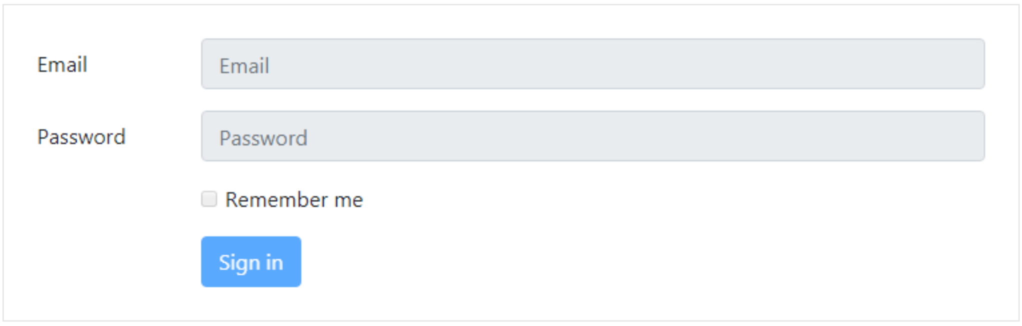


However, if you want to disable all controls within a `<form>` at once place them inside a `<fieldset>` element and apply the attribute on it, as shown in the following example:

Example **Try this code »**

```html
<form>
    <fieldset disabled>
        <div class="form-group row">
            <label for="inputEmail" class="col-sm-2 col-form-label">Email</label>
            <div class="col-sm-10">
                <input type="email" class="form-control" id="inputEmail" placeholder="Email">
            </div>
        </div>
        <div class="form-group row">
            <label for="inputPassword" class="col-sm-2 col-form-label">Password</label>
            <div class="col-sm-10">
                <input type="password" class="form-control" id="inputPassword"
```

— The output of the above example will look something like this:



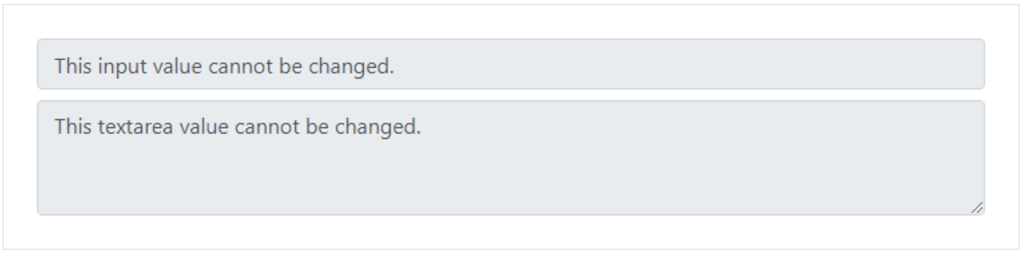
Creating Readonly Inputs

You can also add the `readonly` boolean attribute on an input or textarea to prevent the modification of its value. Read-only inputs appear in lighter background (just like disabled inputs), but it retain the standard text cursor. Check out the following example to see how it works:

Example **Try this code »**

```html
<input type="text" class="form-control mb-2" value="This input value cannot be changed." readonly>
<textarea rows="3" class="form-control" readonly>This textarea value cannot be changed.</textarea>
```

— The output of the above example will look something like this:


This input value cannot be changed.

This textarea value cannot be changed.

Column Sizing of Inputs, Textareas and Select Boxes

You can also match the sizes of your form controls to the Bootstrap grid column sizes. Just wrap your form controls (i.e. `<input>`, `<textarea>`, and `<select>`) in grid columns, or any custom element and apply the grid classes on it, as shown in the following example:

Example **Try this code »**

```
<div class="form-row">
    <div class="form-group col-sm-6">
        <label for="inputCity">City</label>
        <input type="text" class="form-control" id="inputCity">
    </div>
    <div class="form-group col-sm-4">
        <label for="inputState">State</label>
        <select id="inputState" class="form-control">
            <option>Select</option>
        </select>
    </div>
    <div class="form-group col-sm-2">
        <label for="inputZip">Zip</label>
        <input type="text" class="form-control" id="inputZip">
    </div>
</div>
```

Tip: You can alternatively use the class `.form-row` in place of `.row` while creating form layouts. The `.form-row` class is a variation of standard Bootstrap grid `.row` which overrides the default column gutters for tighter and more compact layouts.

Height Sizing of Inputs and Select Boxes

You can easily change the height of your text input and select boxes to match the button sizes. Use the form control height sizing classes like `.form-control-lg`, `.form-control-sm` on the `<input>` and `<select>` boxes to create it's larger or smaller sizes.

Also, be sure to apply the class `.col-form-label-sm` or `.col-form-label-lg` on the `<label>` or `<legend>` elements to correctly resize the label accordion to the form controls.

Example **Try this code »**

```
<form>
    <div class="form-group row">
        <label class="col-sm-2 col-form-label col-form-label-lg">Email</label>
        <div class="col-sm-10">
            <input type="email" class="form-control form-control-lg"
placeholder="Large input">
        </div>
    </div>
    <div class="form-group row">
        <label class="col-sm-2 col-form-label">Email</label>
        <div class="col-sm-10">
```

```
            <input type="email" class="form-control" placeholder="Default input">
        </div>
    </div>
```

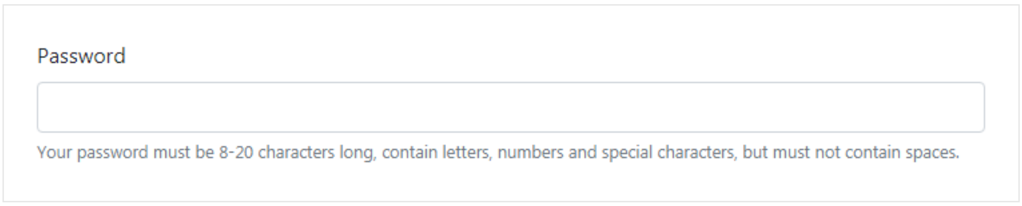
Placing Help Text around Form Controls

Placing help text for the form controls in an efficient way to guide users to enter the correct data in a form. You can place block level help text for a form control using the class `.form-text`. The block help text is typically displayed at the bottom of the control. Here's an example:

Example | **Try this code »**

```
<div class="form-group">
    <label>Password</label>
    <input type="password" class="form-control">
    <small class="form-text text-muted">
        Your password must be 8-20 characters long, contain letters, numbers and special characters, but must not contain spaces.
    </small>
</div>
```

— The output of the above example will look something like this:


Password

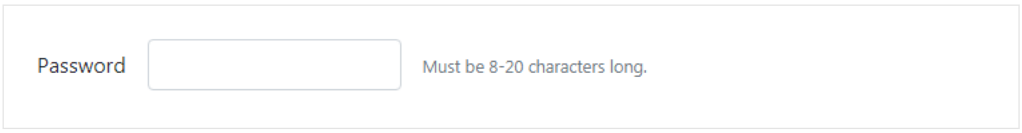
Your password must be 8-20 characters long, contain letters, numbers and special characters, but must not contain spaces.

Similarly, you can also place inline help text using the `<small>` element. No need to use `.form-text` in this case. The following example shows how to implement this:

Example | **Try this code »**

```
<form class="form-inline">
    <div class="form-group">
        <label>Password</label>
        <input type="password" class="form-control mx-sm-3">
        <small class="text-muted">Must be 8-20 characters long.</small>
    </div>
</form>
```

— The output of the above example will look something like this:


Password Must be 8-20 characters long.

Bootstrap Form Validation

Bootstrap 4 provides an easy and quick way to validate web forms on client-side. It uses browser's native form validation API to validate the form. Form validation styles are applied via CSS `:invalid` and `:valid` pseudo-classes. It applies to `<input>`, `<select>`, and `<textarea>` elements.

Let's check out the following example to see how it actually works:

Example | **Try this code »**

```
<form class="needs-validation" novalidate>
    <div class="form-group">
```

```
        <label for="inputEmail">Email</label>
        <input type="email" class="form-control" id="inputEmail" placeholder="Email" required>
        <div class="invalid-feedback">Please enter a valid email address.</div>
    </div>
    <div class="form-group">
        <label for="inputPassword">Password</label>
        <input type="password" class="form-control" id="inputPassword" placeholder="Password" required>
        <div class="invalid-feedback">Please enter your password to continue.</div>
    </div>
    <div class="form-group">
        <label class="form-check-label"><input type="checkbox"> Remember me</label>
    </div>
    <button type="submit" class="btn btn-primary">Sign in</button>
</form>
```

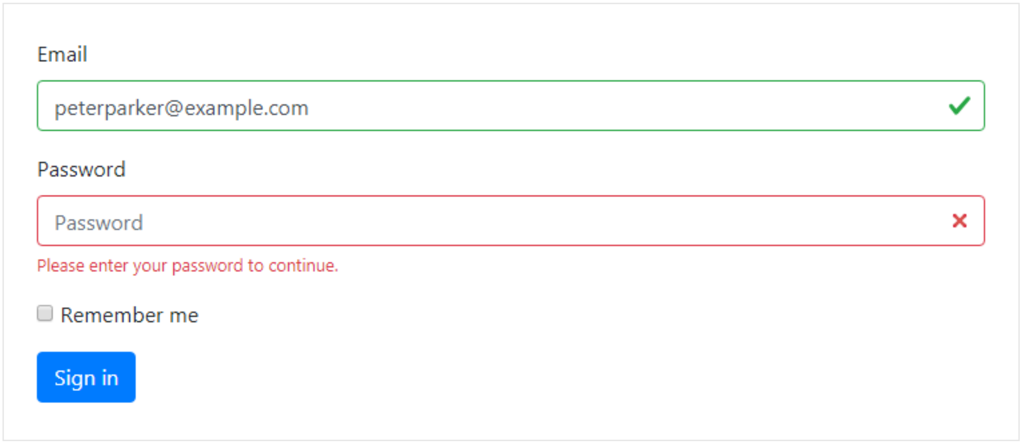
Note: For custom Bootstrap form validation messages, you'll need to disables the browser default feedback tooltips by adding the `novalidate` boolean attribute to the `<form>` element. However, it still provides access to the form validation APIs in JavaScript.

Here's the custom JavaScript code that displays error messages and disables form submission if there are invalid fields. See the JavaScript closures chapter to learn about self-executing function.

Example Try this code »

```
<script>
    // Self-executing function
    (function() {
        'use strict';
        window.addEventListener('load', function() {
            // Fetch all the forms we want to apply custom Bootstrap validation styles to
            var forms = document.getElementsByClassName('needs-validation');
            // Loop over them and prevent submission
            var validation = Array.prototype.filter.call(forms, function(form) {
                form.addEventListener('submit', function(event) {
                    if (form.checkValidity() === false) {
                        event.preventDefault();
                        event.stopPropagation();
                    }
                    form.classList.add('was-validated');
                }, false);
            });
        }, false);
    })();
</script>
```

— The output of the above example will look something like this:


Email

peterparker@example.com ✓

Password

Password ✗

Please enter your password to continue.

Remember me

Sign in

Tip: To reset the appearance of the form programmatically, remove the class `.was-validated` class from the `<form>` element after submission. This class is applied automatically on the form by the Bootstrap when you click the submit button.

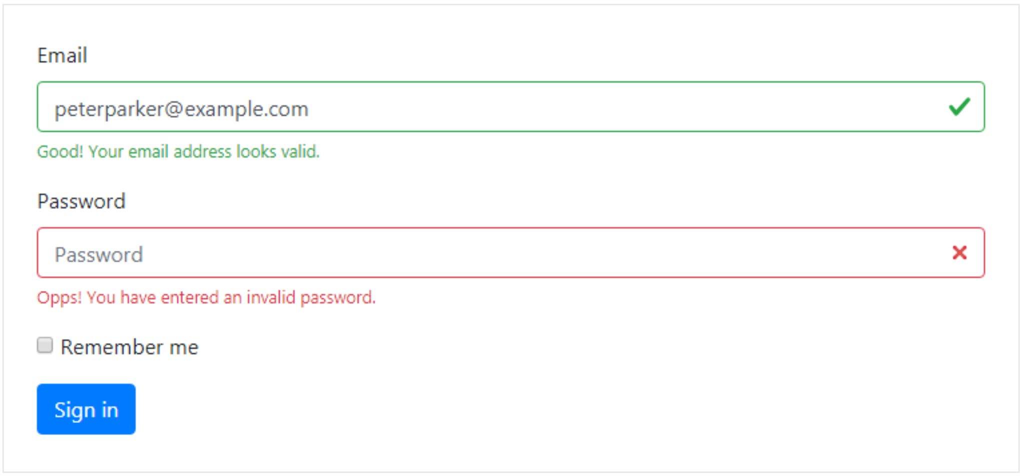
If you require server-side validation, you can indicate invalid and valid form fields with the `.is-invalid` and `.is-valid`. The `.invalid-feedback` and `.valid-feedback` are also supported with these classes. Try out the following example to see how it works:

Example

Try this code »

```
<form>
    <div class="form-group">
        <label for="inputEmail">Email</label>
        <input type="email" class="form-control is-valid" id="inputEmail"
placeholder="Email" value="peterparker@example.com" required>
        <div class="valid-feedback">Good! Your email address looks valid.</div>
    </div>
    <div class="form-group">
        <label for="inputPassword">Password</label>
        <input type="password" class="form-control is-invalid" id="inputPassword"
placeholder="Password" required>
        <div class="invalid-feedback">Opps! You have entered an invalid password.
</div>
    </div>
    <div class="form-group">
        <label class="form-check-label"><input type="checkbox"> Remember me</label>
    </div>
    <button type="submit" class="btn btn-primary">Sign in</button>
</form>
```

— The output of the above example will look something like this:


Email
peterparker@example.com
Good! Your email address looks valid.
Password
Password
Opps! You have entered an invalid password.
Remember me
Sign in

You can alternatively swap the `.{valid|invalid}-feedback` classes for `.{valid|invalid}-tooltip` classes to display the validation feedback text in a tooltip style.

Also, be sure to apply the style `position: relative` or class `.position-relative` on the parent element for proper feedback tooltip positioning. Here's an example:

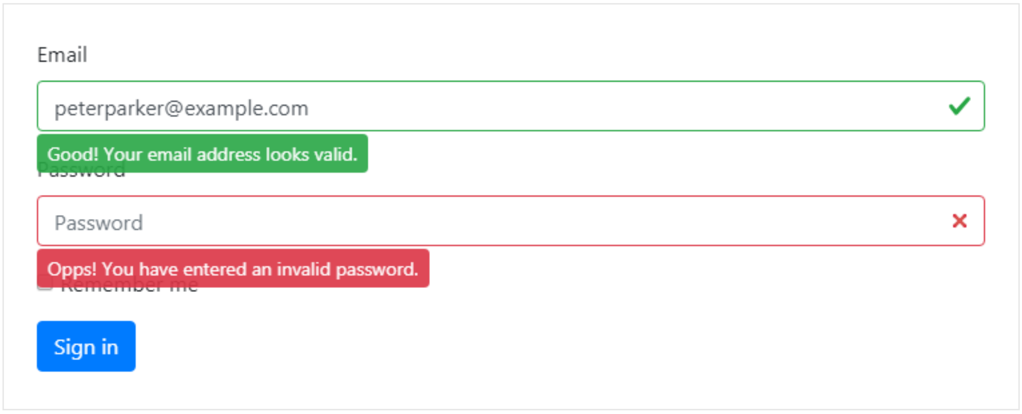
Example

Try this code »

```
<form>
    <div class="form-group position-relative">
        <label for="inputEmail">Email</label>
        <input type="email" class="form-control is-valid" id="inputEmail"
placeholder="Email" value="peterparker@example.com" required>
        <div class="valid-tooltip">Good! Your email address looks valid.</div>
    </div>
    <div class="form-group position-relative">
```

```
        <label for="inputPassword">Password</label>
        <input type="password" class="form-control is-invalid" id="inputPassword" placeholder="Password" required>
        <div class="invalid-tooltip">Opps! You have entered an invalid password.</div>
    </div>
    <div class="form-group">
        <label class="form-check-label"><input type="checkbox"> Remember me</label>
    </div>
    <button type="submit" class="btn btn-primary">Sign in</button>
</form>
```

— The output of the above example will look something like this:


Email

peterparker@example.com ✓

Good! Your email address looks valid.

Password

Password ✗

Opps! You have entered an invalid password.

Remember me

Sign in

Note: Background icons for `<select>` elements only work properly with `.custom-select`, not with `.form-control`. We will learn about Bootstrap custom forms in next chapter.

Supported Form Controls in Bootstrap

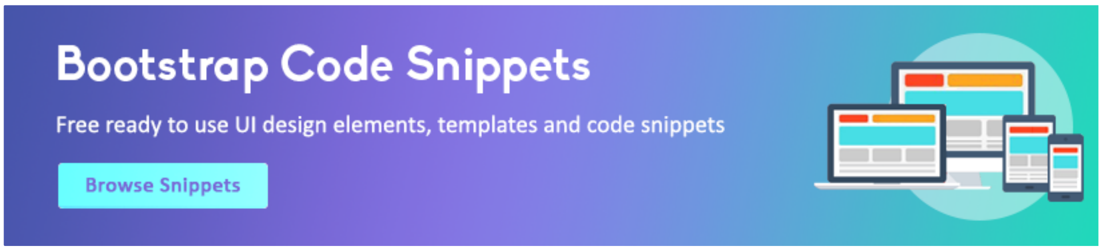
Bootstrap includes support for all standard HTML form controls as well as new HTML5 input types such as datetime, number, email, url, search, range, color, url, and so on. The following example will show you the usages of standard form controls with Bootstrap.

Example **Try this code »**

```
<form>
<div class="form-group row">
    <label class="col-sm-3 col-form-label" for="firstName">First Name:</label>
    <div class="col-sm-9">
        <input type="text" class="form-control" id="firstName" placeholder="First Name" required>
    </div>
</div>
<div class="form-group row">
    <label class="col-sm-3 col-form-label" for="lastName">Last Name:</label>
    <div class="col-sm-9">
        <input type="text" class="form-control" id="lastName" placeholder="Last Name" required>
    </div>
```

« PREVIOUS PAGE NEXT PAGE »

ADVERTISEMENTS


Bootstrap Code Snippets
Free ready to use UI design elements, templates and code snippets
Browse Snippets

Is this website helpful to you? Please give us a like, or share your feedback to help us improve. Connect with us on Facebook and Twitter for

ABOUT US

Our Story

Terms of Use

Privacy Policy

CONTACT

Contact Us

Report Error

Advertise

INTERACTIVE TOOLS

Title & Meta Length Calculator

Bootstrap Button Generator

SQL Playground

HTML Formatter

HTML Color Picker

HTML Editor

Sh