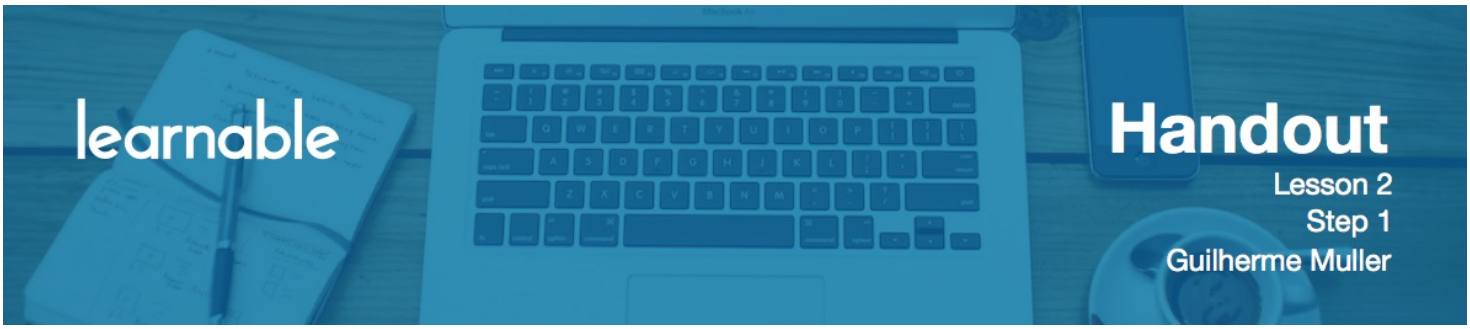

Introduction

In this lesson you are going to discover the main form tags and practice by building a simple contact form. To prepare for the lesson open the *"forms-begin.html"* file in the editor and in the browser. Inside we have a sample contact/job application form.

The form tag and its main attributes

Building a form starts with a `form` tag:

```
<form>
    Contact Info
    Name
    E-mail
    Telephone
    City
    State
    Other Info
    Gender
    Certifications
    CV
    Message
    Send
</form>
```

This tag accepts many attributes, the most important of them being `action` and `method`. The `action` attribute tells the form where to send the data to. In HTML5 you can omit this attribute and then the data will be sent to the current page. A programming language like PHP can then receive this data on server's side, and do something with it, like send it by email. Let's just put a hash there for now:

```
<form action="#">
    […]
</form>
```

The other attribute – `method` – specifies the method of sending the data. The most common values are **GET** and **POST**.

GET sends the data through the URL. For example, when you search something on Google, there's some data in the URL after the question mark – this is the GET data. POST is another method – it will send data using a more discreet approach. It can still be intercepted, but it's less visible to the user.

For this example we are going to use POST:

```
<form action="#" method="POST">
    […]
</form>
```

Fieldset and legend tags

The `form` is divided into two sections: contact info and other info. There's a tag for grouping form fields called `fieldset`:

```
<form action="#" method="POST">
    <fieldset>
        Contact Info
        Name
        E-mail
        Telephone
        City
```

```
        State
        </fieldset>
        <fieldset>
        Other Info
        Gender
        Certifications
        CV
        Message
        Send
    </fieldset>
</form>
```

Along with the `fieldset` we can also use another tag called `legend` to specify a title for each `fieldset`:

```
<form action="#" method="POST">
    <fieldset>
        <legend>Contact Info</legend>
        [...]
    </fieldset>
</form>
```

Now you may check the default appearance of these elements in the browser.

Form field markup and the label and input tags

As for the form fields themselves, there are many ways of approaching the mark up. Let's use `div`s for each field; it will have its description in a `label` tag that is the most appropriate tag for this job.

```
<div>
    <label>Name</label>
</div>
```

For form fields we have three main tags in HTML: `input`, `select` and `textarea`.

`input` does the most work as you'll see soon. `select` is used for selection boxes, and `textarea` displays a bigger area for longer text content.

```
<div>
    <label>Name</label>
    <input type="text" name="name" id="name">
</div>
```

Let's talk about each attribute a bit:

- The `type` attribute defines the type of input that will appear to the user. The `text` type is the simplest but there are others that we will see shortly.
- The `name` attribute defines the name of the field. This is important for the script that will process this form
- The `id` attribute is used for identification, so we can tie this form field with its label.

The `label` tag, in turn, has a `for` attribute, which allows us to link a label and a field together:

```
<div>
    <label for="name">Name</label>
    <input type="text" name="name" id="name">
</div>
```

Now this label is linked to the input so if you click on the label in the browser, you will notice that the cursor goes to the field. Such connection is good for accessibility as well.

Next, we have the email field.

```
<div>
    <label for="email">E-mail</label>
    <input type="text" name="email" id="email">
</div>
```

However we can employ a new HTML5 feature here: an input type called `email`:

```
<div>
    <label for="email">E-mail</label>
```

```
    <input type="email" name="email" id="email">
</div>
```

If you test this in the browser you'll see no noticeable changes. But if you access the page with a phone or tablet, the keyboard may change to the one more appropriate for entering emails.

For the telephone the process is the same:

```
<div>
    <label for="tel">Telephone</label>
    <input type="text" name="tel" id="tel">
</div>
```

There's another new input type in HTML5, called `tel`:

```
<div>
    <label for="tel">Telephone</label>
    <input type="tel" name="tel" id="tel">
</div>
```

It also does not change much when viewing the page from a regular computer, but on a mobile device you'll see that the keyboard also changes.

For the city the process is once again the same:

```
<div>
    <label for="city">City</label>
    <input type="text" name="city" id="city">
</div>
```

Select, option and optgroup tags

Next we have the state. In this case we want to present the user a list of options – the `select` tag is perfect for that. Inside the `select` tag we build the list using `option` tag.

```
<div>
    <label for="state">State</label>
    <select name="state" id="state">
        <option value="">-- Choose --</option>
        <option value="">State 1</option>
        <option value="">State 2</option>
        <option value="">State 3</option>
        <option value="">State 4</option>
        <option value="">State 5</option>
        <option value="">State 6</option>
        <option value="">State 7</option>
        <option value="">State 8</option>
        <option value="">State 9</option>
    </select>
</div>
```

The value is passed to the persisting script, but we'll leave that empty. Now check how this looks in the browser.

Suppose now that we have two countries in the list each with its own states, and that we wanted to separate them in the list. We can do that with the `optgroup`.

```
<div>
    <label for="state">State</label>
    <select name="state" id="state">
    <option value="">-- Choose --</option>
    <optgroup label="Country 1">
        <option value="">State 1</option>
        <option value="">State 2</option>
        <option value="">State 3</option>
    </optgroup>
    <optgroup label="Country 2">
        <option value="">State 4</option>
        <option value="">State 5</option>
        <option value="">State 6</option>
```

```
    </optgroup>
    <option value="">State 7</option>
    <option value="">State 8</option>
    <option value="">State 9</option>
    </select>
</div>
```

Check the result in the browser.

Input type radio and input type checkbox

For fields like gender you could also use a `select` tag, but there is another option that's more suitable – **radio buttons**.

```
<div>
    <label>Gender</label>
    <label>
        <input type="radio" name="gender" value="m"> M
    </label>
    <label>
        <input type="radio" name="gender" value="f"> F
    </label>
</div>
```

This is another way of linking an input to a label and for radio buttons and check boxes this is the method I recommend to use. This way, when clicking either the label or the radio button, the option is selected.

Also notice that I have to use the same name for both inputs, otherwise the browser does not know how many options are there in this group.

We are going to do a similar thing to the certifications field, but this time we want to let the user choose one or more options. This is done with a `checkbox` input:

```
<div>
    <label>Experience</label>
    <label>
        <input type="checkbox" name="experience" value="1"> Option 1
    </label>
    <label>
        <input type="checkbox" name="experience" value="2"> Option 2
    </label>
    <label>
        <input type="checkbox" name="experience" value="3"> Option 3
    </label>
</div>
```

Check the result in the browser.

Input type file

For the CV field we can use a `file` input so that the users can send their curriculum:

```
<div>
    <label for="file">CV</label>
    <input type="file" name="file" id="file">
</div>
```

The file input appearance will vary a lot depending on the OS and browser.

Textarea tag

The message field is an example of where to use the `textarea` tag. The `textarea` tag provides a large multi-line space where the user can type in a longer message.

```
<div>
    <label for="message">Message</label>
    <textarea name="message" id="message" cols="30" rows="10"></textarea>
</div>
```

This tag's format is very different in comparison to a `text` type input. The tag opens and closes, and its size can be controlled, with the `cols` and `rows` attributes. For a `text` type input you can control size by using the `size` attribute. In general you should use CSS to style these tags, but it's important to know that these attributes exist.

Creating buttons

Lastly we have the "Send" button. We'll take a look at the two ways of creating a button: one uses the `input` tag and the other uses the `button` tag. The outcome is basically the same but in my opinion the `button` tag is easier to work with CSS later.

```
<input type="submit" value="Send">
```

The `value` attribute here defines what's written on the button.

The `button` tag is used like this:

```
<button type="submit">Send</button>
```

The default appearance of these buttons can vary a lot between browsers, OS's and devices.

Now let's talk about the new HTML5 features for forms. They have a good support in all major browsers but it's always important to check if some JavaScript is needed as well to provide the backwards compatibility for older software.

If you want a field to have a brief explanation of what the user has to type in it, you can use a `placeholder` attribute:

```
<input type="text" name="name" id="name" placeholder="Your Name">
```

This text will appear when there's nothing typed in. When a user starts typing, the placeholder disappears, and when the field is empty again, the placeholder comes back.

If you want a field to be focused first when the user loads the page, use the boolean attribute `autofocus`:

```
<input type="text" name="name" id="name" placeholder="Your Name" autofocus>
```

Boolean attributes do not need values: the browser simply checks whether they are present in a tag or not. Now when the user loads the page, the field will be already selected. Be careful about the accessibility problems though: this is only useful if the main interest in the page is the form. Even so, a person using a screen reader might have problems so it's important to analyse the situation, and see what is the best for the users.

Finally, you can determine the required fields in the form. Of course, this is very easily breakable so you still need to validate the form on server's side. However this is still useful. Just put the `required` boolean attribute:

```
<input type="email" name="email" id="email" required>
```

If the user tries to send an empty form, the browsers that support this feature will notify the user about the errors.