**HTML forms: best practices**

**1 - Creating accessible forms**

Forms are commonly used to enable user interaction within Web sites and Web applications, for example, for login, registering, commenting, and purchasing.

Since HTML5 provides functionalities to assist with accessibility, developers should make a concerted effort to mark up Web based forms. The following two guidelines are to give you a good start to make your forms accessible:

1. For every form field, ensure that a descriptive **label** is provided and use the <label> element to identify each form control.
2. For larger or complex forms, use the <fieldset> and <legend> elements to respectively **group and associate** related form controls.

We will give usage examples for each of these two basic guidelines in the following pages.

**Further reading**

The WAI Web site hosts a [Forms tutorial](http://www.w3.org/WAI/tutorials/forms/) where you will find more guidelines to help make your forms truly accessible: Form Instructions, Validating Input, User Notifications, Multi-Page Forms, and Custom Controls.

**2 - Why is this important?**

Forms can be visually and cognitively complex and difficult to use. Accessible forms are easier to use for everyone, including people with disabilities.

* **People with cognitive disabilities** can more easily understand the form and how to complete it, as making forms accessible improves the layout structure, instructions, and feedback.
* **People using speech input** can use the labels via voice commands to activate controls and move the focus to the fields that they need to complete.
* **People with limited dexterity** benefit from large clickable areas that include the labels, especially for smaller controls, such as radio buttons and checkboxes.
* **People using screen readers** can identify and understand form controls more easily because they are associated with labels, field sets, and other structural elements.

#### 3 - Labeling controls

##### Labels need to describe the purpose of the form control

Form fields and other form controls usually have visible labels, such as "E-mail Address:" as the label for a text field (see figure below).

Visual of a text input field preceded by the mention "E-mail address:"

When these labels are marked up correctly, people can interact with them using only the keyboard, using voice input, and using screen readers. Also, the label itself becomes clickable, which enables a person who has difficulty clicking on small radio buttons or checkboxes to click anywhere on the label text.

##### Associating labels explicitly

Whenever possible, use the label element to explicitly associate text with form elements. The for attribute of the label must exactly match the id of the form control.

###### Example 1 (click on the label, not on the input field to see the effect)

Top of Form

First name: 

Source code:

Bottom of Form

1. <label for="first\_name">First name:</label>
2. <input id="first\_name" type="text" name="fname"/>

###### Alternative example 1

Note that you can also include the <input> element inside the <label>...</label> element, and also add a <span lang="en"> for example, to indicate the language used in the label. Sometimes, [nesting labels and inputs can also make CSS styling easier and produce better results with screen readers](https://developer.mozilla.org/en-US/docs/Web/Guide/HTML/Forms/How_to_structure_an_HTML_form).

Source code (with <input> inside the <label>):

1. **<label for="first\_name"><span lang=en">**First name**</span>**
2. <input id="first\_name" type="text" name="fname"/>
3. **</label>**

###### Example 2 (click on the label "Subscribe to newsletter" to see the effect)

Top of Form

First name:   
Subscribe to newsletter 

Source code:

Bottom of Form

1. <label for="firstname">First name:</label>
2. <input type="text" name="firstname" id="firstname"><br>
3. <label for="subscribe">Subscribe to newsletter</label>
4. <input type="checkbox" name="subscribe" id="subscribe">

**4 - Labeling buttons**

The label of a <button> element is set inside the element and can include markup. This allows advanced accessibility hints to be included, such as marking up language change.

Example: <button>Mon <span lang="fr">bouton</span></button>, for a button with a label in French.

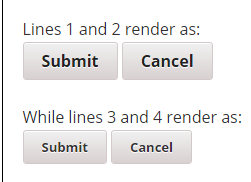
When using the <input> element to create buttons, the label is set in the value attribute of the element.

Example: <input type="submit" value="Please submit">, will be rendered as a button.

Source code for an example of "Submit" and "Cancel" buttons:

1. <button type="submit">Submit</button>
2. <button type="button">Cancel</button>
4. <input type="submit" value="Submit">
5. <input type="button" value="Cancel">

These will produce the same results:



**5 - Labeling text areas**

Enter your address:  


Source code:

1. <label for="address">Enter your address:</label>
2. <br>
3. <textarea id="address" name="addresstext"></textarea>

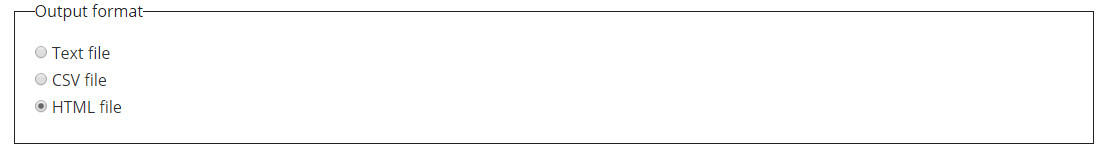
**6 - Grouping controls**

Groupings of form controls, typically groups of related checkboxes and radio buttons, sometimes require a higher level description. Grouping related form controls makes forms more understandable for all users, as related controls are easier to identify.

**Associating related controls with fieldset**

Grouping needs to be carried out visually and in the code, for example, by using the <fieldset> and <legend> elements to associate related form controls. The <fieldset> identifies the entire grouping and <legend> identifies the grouping's descriptive text.

**Example 1 - Radio buttons**

In the example below, there are three radio buttons that allow the user to choose an output format. Radio button groups should always be grouped using <fieldset>. 

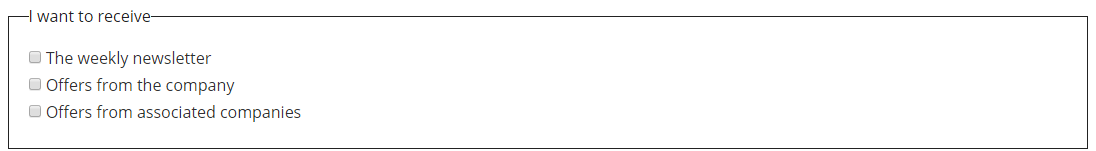
HTML file

Bottom of Form

Source code:

1. <fieldset>
2. <legend>Output format</legend>
3. <div>
4. <input type="radio" name="format" id="txt" value="txt" checked>
5. <label for="txt">Text file</label>
6. </div>
7. <div>
8. <input type="radio" name="format" id="csv" value="csv">
9. <label for="csv">CSV file</label>
10. </div>
11. […]
12. </fieldset>

**Example 2 - Checkboxes**

In the example below, there are three checkboxes that are all part of an opt-in function for receiving different types of information. 

Top of Form

Bottom of Form

Source code:

1. <fieldset>
2. <legend>I want to receive</legend>
3. <div>
4. <input type="checkbox" name="newsletter" id="check\_1">
5. <label for="check\_1">The weekly newsletter</label>
6. </div>
7. […]
8. </fieldset>

#### [ADVANCED] Associating related controls with WAI-ARIA

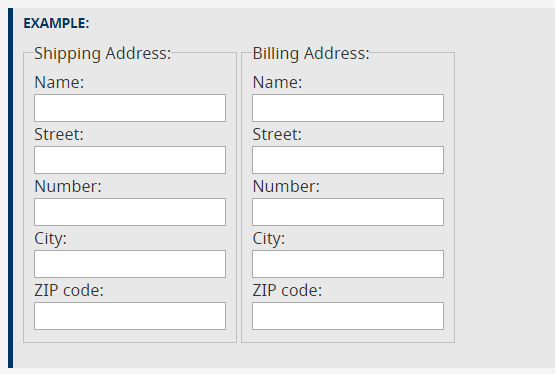
WAI-ARIA provides a grouping role that functions in a similar way to fieldset and legend. For example, a div element can have role=group to indicate that the contained elements are members of a group.

WAI-ARIA roles are very important in the accessibility world, and we invite you to see an example provided in the [associated WAI tutorial](http://www.w3.org/WAI/tutorials/forms/grouping/). This article by Oscar Cao gives an [Introduction to understanding WAI-ARIA 1.0 roles](http://oscarcao.com/blog/2015/06/08/introduction-to-understanding-wai-aria-1-0-roles/).

Example provided in [associated WAI tutorial](http://www.w3.org/WAI/tutorials/forms/grouping/). - <https://www.w3.org/WAI/tutorials/forms/grouping/>

### Example 3: Related fields

This example shows form fields to enter shipping and billing addresses. As the labels in both groups have the same text, the fieldset element also helps to distinguish the form fields by their groups. In case the <legend> is not read by screen readers (see note below), labels for the first form control in each group should include the group’s name. This name can be hidden visually.



**CODE SNIPPET:**

<fieldset>

<legend>Shipping Address:</legend>

<div>

<label for="shipping\_name">

<span class="visuallyhidden">Shipping </span>Name:

</label><br>

<input type="text" name="shipping\_name" id="shipping\_name">

</div>

<div>

<label for="shipping\_street">Street:</label><br>

<input type="text" name="shipping\_street" id="shipping\_street">

</div>

[…]

</fieldset>

<fieldset>

<legend>Billing Address:</legend>

<div>

<label for="billing\_name">

<span class="visuallyhidden">Billing </span>Name:

</label><br>

<input type="text" name="billing\_name" id="billing\_name">

</div>

<div>

<label for="billing\_street">Street:</label><br>

<input type="text" name="billing\_street" id="billing\_street">

</div>

[…]

</fieldset>

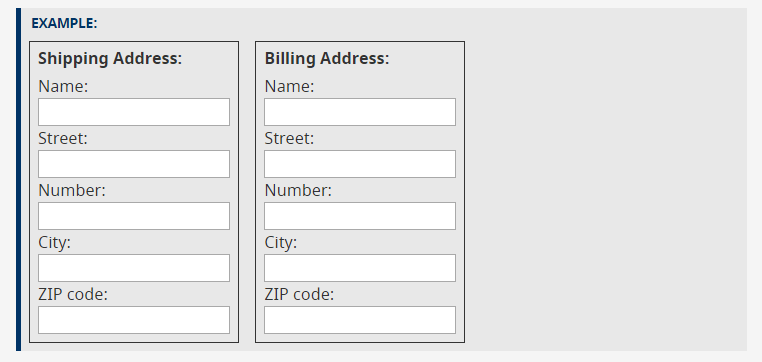
**Note:** Depending on the configuration, some screen readers read out the legend either with every form element, once, or, rarely, not at all. To accommodate this consider the following:

* Make the legend as short as possible for situations in which it is read together with the label each time.
* Make the individual labels sufficiently self-explanatory for situations in which legends are not read aloud, without repeating the legend in every label.

## Approach 2: Associating related controls with WAI-ARIA

WAI-ARIA provides a grouping role that functions similarly to fieldset and legend. In this example, the div element has role=group to indicate that the contained elements are members of a group and the aria-labelledby attribute references the id for text that will serve as the label for the group.

This technique provides additional styling possibilities.



CODE SNIPPET:

<div role="group" aria-labelledby="shipping\_head">

<div id="shipping\_head">Shipping Address:</div>

<div>

<label for="shipping\_name">

<span class="visuallyhidden">Shipping </span>Name:

</label><br>

<input type="text" name="shipping\_name" id="shipping\_name">

</div>

[…]

</div>

<div role="group" aria-labelledby="billing\_head">

<div id="billing\_head">Billing Address:</div>

<div>

<label for="billing\_name">

<span class="visuallyhidden">Billing </span>Name:

</label><br>

<input type="text" name="billing\_name" id="billing\_name">

</div>

[…]

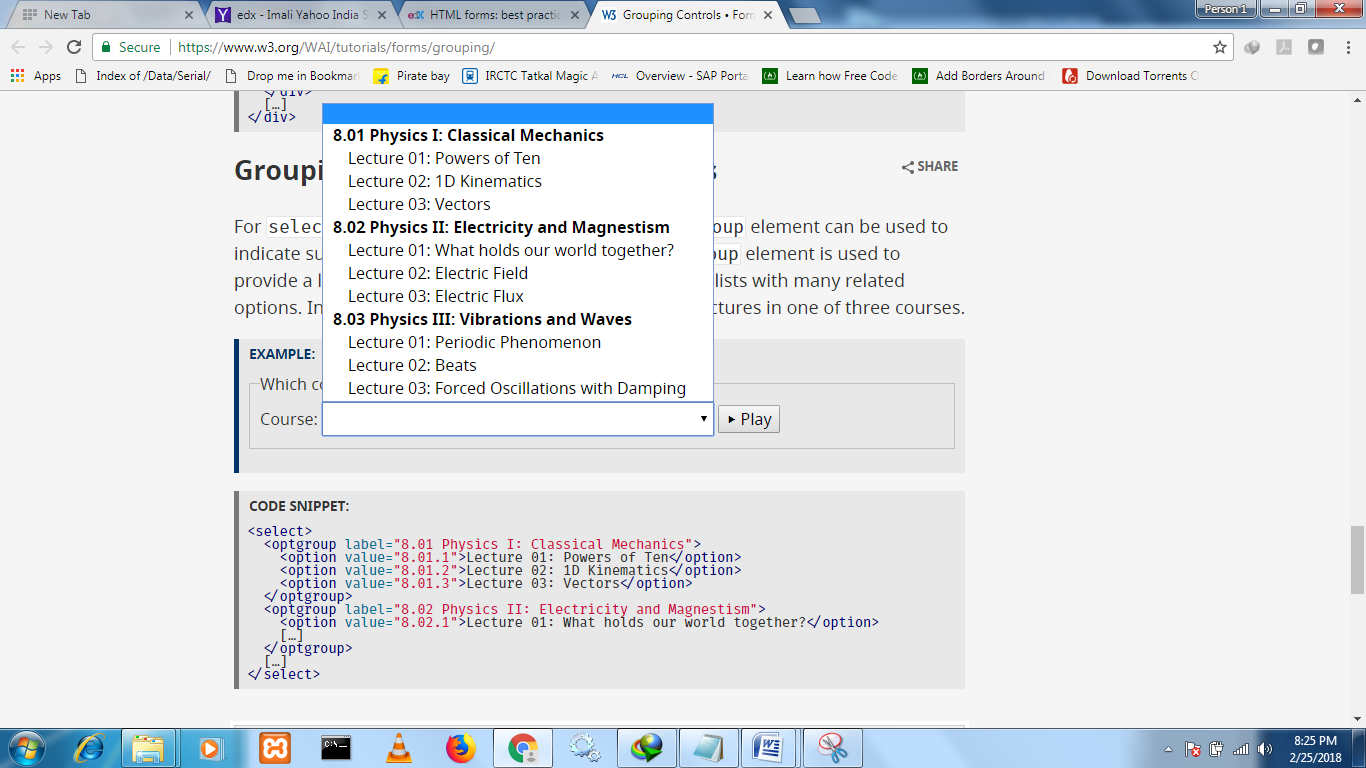
</div>

[**SHARE**](https://www.w3.org/WAI/tutorials/forms/grouping/#grouping-items-in-select-elements)

## Grouping items in select elements

For select elements with groups of options, the optgroup element can be used to indicate such groups. The label attribute of the optgroup element is used to provide a label for the group. This is especially useful for lists with many related options. In the example below, users can choose from lectures in one of three courses.

EXAMPLE:



Top of Form

Bottom of Form

CODE SNIPPET:

<select>

<optgroup label="8.01 Physics I: Classical Mechanics">

<option value="8.01.1">Lecture 01: Powers of Ten</option>

<option value="8.01.2">Lecture 02: 1D Kinematics</option>

<option value="8.01.3">Lecture 03: Vectors</option>

</optgroup>

<optgroup label="8.02 Physics II: Electricity and Magnestism">

<option value="8.02.1">Lecture 01: What holds our world together?</option>

[…]

</optgroup>

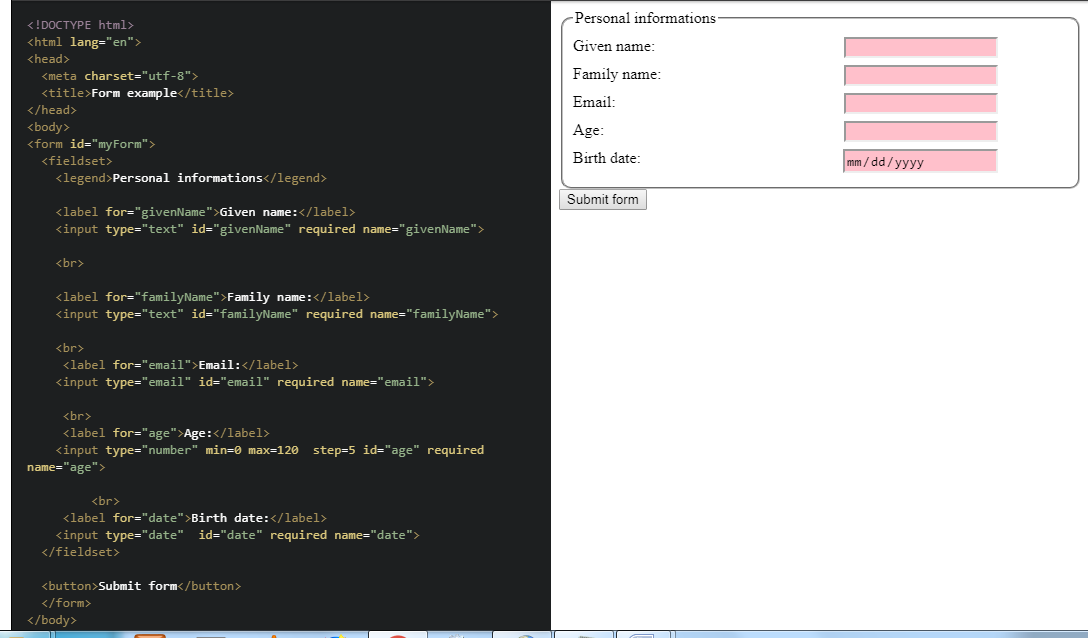
[…]

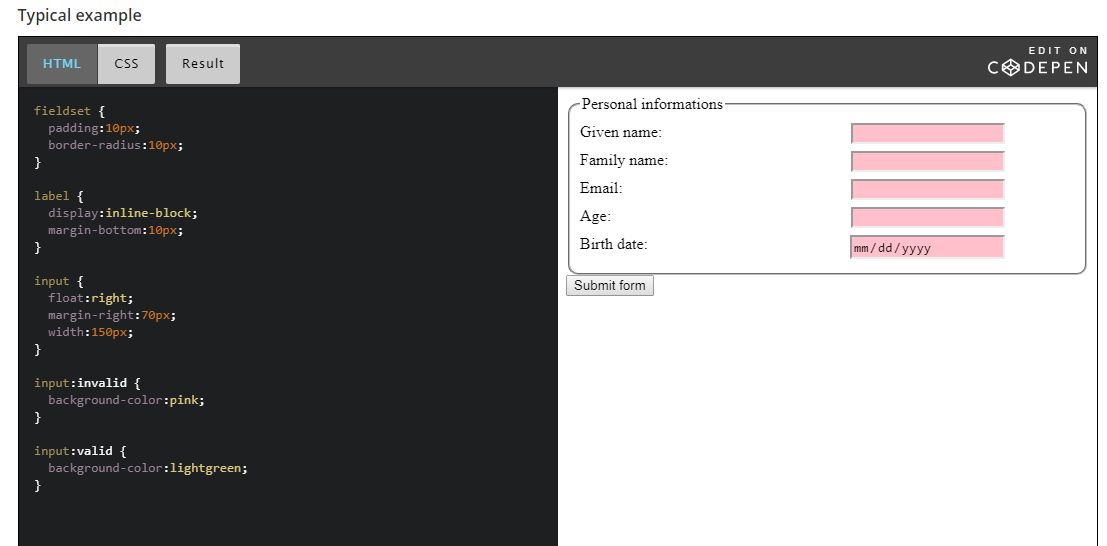
</select>

### [HTML forms and JavaScript](https://courses.edx.org/courses/course-v1:W3Cx+JS.0x+3T2017/courseware/b108897ce3c4449c8326cda572490b8f/3fd025df64df4a5fa2818efe2f1c689d/?activate_block_id=block-v1%3AW3Cx%2BJS.0x%2B3T2017%2Btype%40sequential%2Bblock%403fd025df64df4a5fa2818efe2f1c689d)

Forms are a way to get user input which is either sent to a remote server, or processed locally, or both.

This section of the course will only cover local processing and the client-side part, with a focus on JavaScript processing.





Here invalid/valid is pseudo class of input element(pseudo class can be for any CSS selector of element.)

<https://developer.mozilla.org/en-US/docs/Web/CSS/Pseudo-classes>

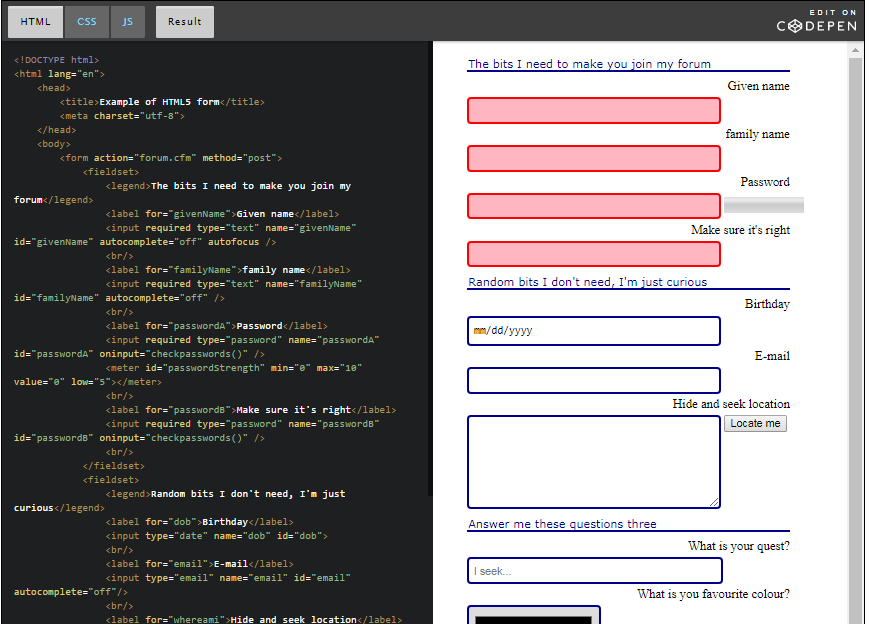
**HTML form input can be sent to a server without JavaScript**

If a form's content is sent to a remote server, on the server side, you may have PHP, Java, C#, Ruby, Python, etc. components. There are several ways to collect server-side data from a form in a Web page: REST Web services, servlets, Microsoft ASP pages, etc.

On the client side, the forms indicate to which server and how the data should be sent,  using the action and method attributes respectively. A <button type="submit"> or an <input type=submit> field is used to submit the form content.

For example: <form action="myServerCode.php" method="POST">...</form>. Here, we set the URL of the server side code (myServerCode.php), and the HTTP method that will be used by the browser for sending the form content (POST).

Example of HTML5 form that will not be sent if invalid input fields are present. Notice that the JavaScript part is only used for giving feedback while entering the password. No JavaScript is used for sending the form data, or for complex, global validation:

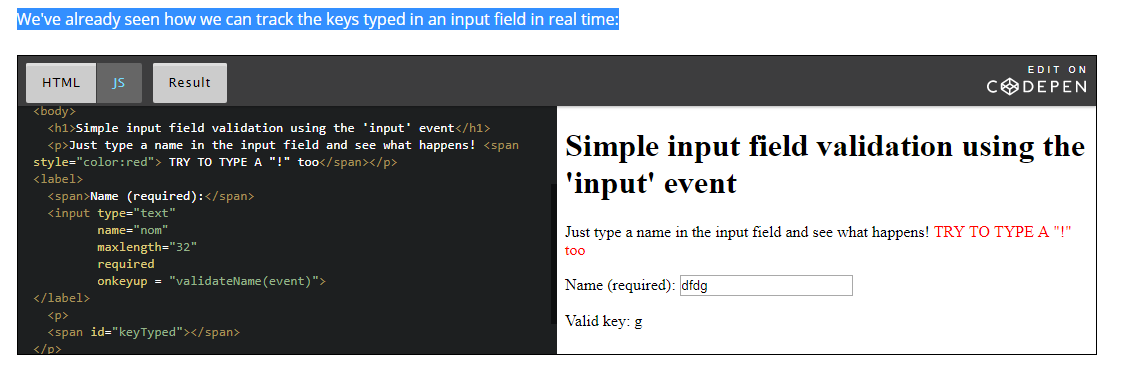


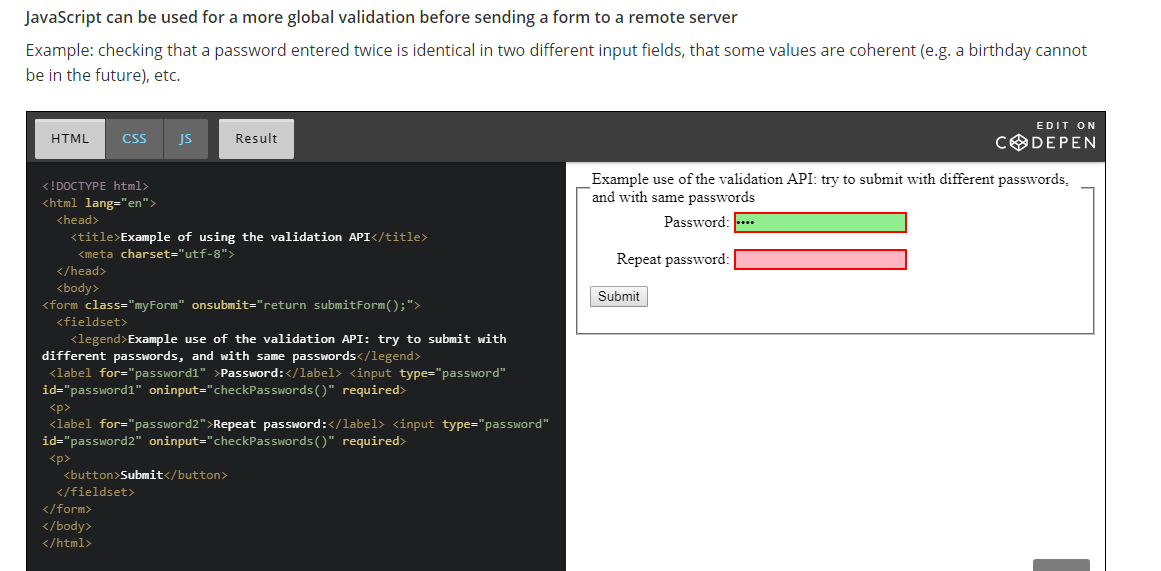
**HTML form input can be sent using Ajax / JavaScript**

Another approach is to use JavaScript for sending the form content with Ajax. This will be covered in the JavaScript advanced course, to be be available on W3Cx.

**JavaScript can be used for validating user input "on the fly"**

While one is typing or selecting a color, or moving a slider, JavaScript event listeners can be used to track the user's interactions in real time, and perform some validation steps along with giving visual feedback.





**JavaScript can be used to make a WebApp that uses form data locally, perhaps with some client-side persistence API**

For example, a contact manager that will work offline, saving data locally, in a database inside the browser. Data will be displayed in a dynamic HTML table, without the need for a remote database.

This is the small project we will build together at the end of the course. :-)

