

# HTML5 Training for Web Developers

## HTML5 Forms

## Lesson 1, Activity 2: Modernizr

Before we get into HTML5 forms, we will re-introduce you to a great little JavaScript library called [Modernizr](#). Modernizr is used to check the browser for feature support so you can write code like this:

```
if (Modernizr.xyzFeature) {
  //do this;
} else {
  //do something different
}
```

We have included it in the [html5-common](#) folder in the root of the class files. Here's an example of how you can check for the new email input type:

**Code Sample:**

[html5-forms/Demos/modernizr.html](#)

```
<!DOCTYPE HTML>
---- C O D E   O M I T T E D ----
<script src="../../html5-common/modernizr.min.js" type="text/javascript"></script>
<script>
  if (Modernizr.inputtypes.email) {
    alert("woohoo!");
  } else {
    alert("boohoo!");
  }
</script>
---- C O D E   O M I T T E D ----
```

To see which HTML5 features it supports, open [html5-forms/Demos/modernizr-full-check.html](#) in your browser.

We use **modernizr** in some of the demos in this lesson.

### Lesson 1, Activity 3: New Input Types

HTML5 introduces thirteen new input types:

1. search
2. tel
3. url
4. email
5. datetime
6. date
7. month
8. week
9. time
10. datetime-local
11. number
12. range
13. color

Unfortunately, the browser implementation of these new input types is still spotty at best. Visit <http://www.findmehyip.com/linus> to see the current state of browser support for these input types.

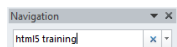
The good news is that these browsers all fall back to `type="text"` when they don't recognize an input type.

#### search

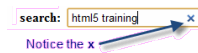
The new HTML5 `search` input type is only supported by:

1. Safari
2. Chrome
3. Firefox 4

Most input fields are meant to be filled out only one time and then submitted for processing. But a search box is a bit different. For example, consider Microsoft Word 2010's search box:



Notice the **x** used for clearing the box. If you look at search boxes in other applications, you'll notice many of them provide a simple way to clear the text. HTML5 browsers that support the `search` input type are taking the same approach. For example, here's Chrome's search box:



Note that the **x** doesn't show up until you have entered some text into the field.

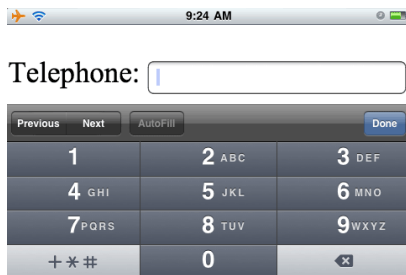
#### tel

The new HTML5 `tel` input type is only supported by:

1. Safari
2. Chrome
3. Firefox 4

Even in your computer-based supporting browsers, you don't really gain anything by using the `tel` input type. As telephone numbers can come in all different formats, there are no constraints on what can be entered here. You could, however, add your own custom validation to all telephone inputs and use the `type="tel"` as a means of finding them.

Also, user agents are free to provide a different/better means for filling out input fields based on their type. For example, the iPhone provides a more appropriate interface for filling out fields of the `tel` type:



#### url and email

The new HTML5 `url` and `email` input types are only supported by:

1. Safari
2. Chrome
3. Opera

Supporting browsers can provide validation for `type="url"` fields to make sure the user enters a valid URL.

Also, like with `type="tel"`, user agents are free to provide a different/better means for entering URLs and email addresses.

For example, for `url` types, the iPhone provides keys for ".", "/", and ".com" and does not provide a "space" key as spaces are not allowed in URLs.

For emails, the iPhone provides an "@" and "." keys. For some reason, it does not provide a ".com" key.

Interestingly enough, the iPhone also provides a "space" key for emails. This is because `email` input types can include a `multiple` attribute, which, when included, allows users to enter multiple emails delimited by spaces. If the iPhone were a little smarter, it would only include the "space" key when the "multiple" attribute was present.

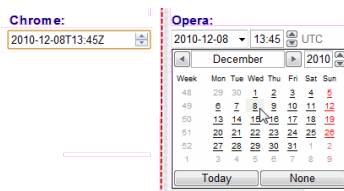
#### date/time input types

The new HTML5 `date`, `datetime`, `datetime-local`, `month`, `week`, and `time` input types are only supported by:

1. Opera
2. Chrome

The table above, which is based on Modernizr's reporting, indicates that Google Chrome doesn't support these date/time types. In fact, Chrome does support them and, according to [chromium.org](http://chromium.org), it *has* since version 5. However, according to [GitHub](https://github.com), Chrome's support is not yet fully implemented and allows badly formatted dates to pass validation.

We did not notice this in our own testing, but we do feel that Chrome 8's current implementation of the date fields is pretty lame. It does not provide nice widgets for entering dates and times, but rather just lets the user either type in a valid entry or scroll through dates and times using up and down arrows. The values shown by Chrome are not at all user friendly:



What user is going to know to enter "2010-12-08T13:45Z" for a date and time?

#### number

The new HTML5 `number` input type is only supported by:

1. Opera
2. Chrome

Note that although Safari isn't listed, it does recognize valid numbers to the same extent that Chrome and Opera do. It just doesn't provide a nice interface for entering numbers on the computer. On the iPhone, it does give you a number keypad, which is nice.

Both Opera and Chrome use up and down buttons (spinboxes) to scroll through numbers and they also allow you to use the up and down arrows on the keyboard. Opera's spinbox is shown below:

**Number:**

All three implementations (Opera, Chrome, and Safari) are subpar in that they don't accept decimals in their standard format ([a bug has been filed](#)). To see this:

1. Open [html5-forms/Demos/input-validity.html](#) in Chrome, Opera, or Safari.
2. In the number field, enter 3.3 or some other decimal.
3. Press the **Check Validity** button. You'll receive an alert reading "3.3 is NOT a valid number." Oops.

#### range

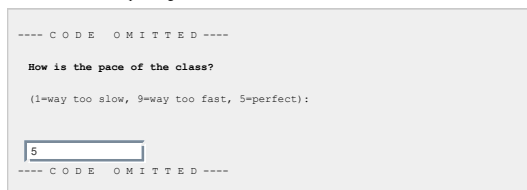
The new HTML5 `range` input type is only supported by:

1. Opera
2. Chrome
3. Safari

All supporting browsers currently represent `range` input types as sliders. This is nice, but the problem is that they don't indicate a value so the user doesn't know what they've selected. Consider the following code:

#### Code Sample:

[html5-forms/Demos/input-range.html](#)



The above code will render as follows (in Google Chrome):

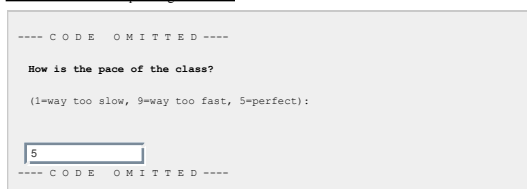
**How is the pace of the class?**  
(1=way too slow, 9=way too fast, 5=perfect):



The problem is that there is no indication of the current value. We can try fixing this with this simple little trick:

#### Code Sample:

[html5-forms/Demos/input-range-title.html](#)



The above code will render as follows (in Safari):

**How is the pace of the class?**  
(1=way too slow, 9=way too fast, 5=perfect):



This trick works pretty well in Safari, which displays the `title` value immediately after the mouse hovers over the field. However, with Chrome and Opera there is a slight delay, which makes this solution less than ideal. Also, the value is only visible when the cursor is over the field.

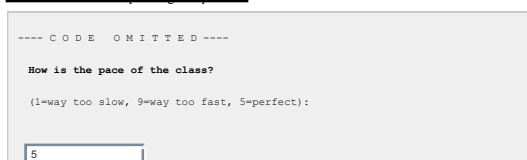
#### The output Element

There is a new HTML5 output element used to show output generated by a script on a page. For example, you could use it to show an error message or the result of a calculation based on values entered in form fields.

We can use the `output` element to create a better solution for making our `range` slider more user friendly:

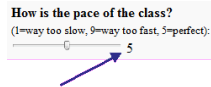
#### Code Sample:

[html5-forms/Demos/input-range-output.html](#)

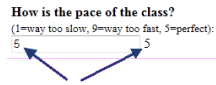


```
5
---- CODE OMITTED ----
```

The above code will render as follows (in Safari):



Notice the number 5 immediately after the slider telling us the current value of the slide. This number will update when the slider value changes. This solution works in browsers that support `range`. However, it creates a problem with non-supporting browsers. Here's how Internet Explorer 9 displays this page:

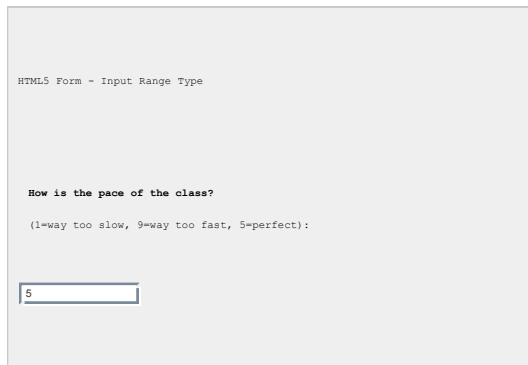


Notice that 5 is repeated twice: once within the text field and once after it.

To fix this, we need to populate the output element dynamically if and only if the `range` input type is supported. Here's a possible solution:

#### Code Sample:

[html5-forms/Demos/input-range-output-dynamic.html](http://html5-forms/Demos/input-range-output-dynamic.html)



This solves the problem. The code, which runs when the page is loaded:

1. Sets the `innerHTML` of the output element to the value of the `range` input.
2. Attaches an event listener to catch any changes of that value and update the output accordingly.
3. And it only runs if the browser supports the `range` input type. We used Modernizr to check for that.

#### min, max, and step attributes

The `min`, `max`, and `step` attributes can be used on the `number`, `range`, and `datepicker` elements.

`min` and `max` are intuitive. They are used to set the minimum and maximum possible values.

The `step` attribute is used to indicate possible values in between the `min` and `max` values.

These attributes are only supported by:

1. Opera
2. Chrome

#### Code Sample:

[html5-forms/Demos/input-number-step.html](http://html5-forms/Demos/input-number-step.html)



To see how this works:

1. Open [html5-forms/Demos/input-number-step.html](http://html5-forms/Demos/input-number-step.html) in Chrome or Opera.
2. Use the spinner to go up and down. Notice only even numbers are shown and that you cannot go below 0 or above 100.
3. Type in 15 or some other odd number. Notice the field turns red.

Safari can validate the numbers based on `min`, `max`, and `step`, but doesn't provide an easy way to pick a number.

#### color

No browser yet supports the new HTML5 `color` input type, but, again, as browsers all fall back to `type="text"` when they don't recognize an input type, there is no harm in using the `color` input type now. When browsers start implementing it, your visitors will likely get a nice color picker.

Chrome and Safari recognize valid color names and hexadecimal color formats, but don't provide an easy way to pick them. As we saw earlier with the `number` type, you can check the validity of the `color` input type with JavaScript. To see this:

1. Open [html5-forms/Demos/input-validity.html](http://html5-forms/Demos/input-validity.html) in Chrome, Opera, or Safari.
2. In the color field, enter "red" or "#ff0000".
3. Press the **Check Validity** button. You'll receive an alert reading "red is a valid color."
4. Now try it with a bad color name (e.g., "foo") and you'll get an alert reading "foo is NOT a valid color."

## Lesson 1, Activity 5: HTML5 New Form Attributes

HTML5 introduces two new attributes of the `form` element:

1. `autocomplete` - "on" or "off". When set to "off" the browser should not use built-in features to help a user auto-fill the form.
2. `novalidate` - Boolean. If included, the form should not validate on submission.

**autocomplete**

Browsers have different ways of choosing data to use for `autocomplete`. For example, Chrome does it based on previous form entries, while Opera uses its built-in contact list. Users can manage autocomplete in their browser's settings. When `autocomplete` is on, you see something like the behavior shown below in Firefox 4:



When `autocomplete` is off, that behavior is blocked.

**novalidate**

The purpose of the `novalidate` attribute is to allow users to submit their forms even if the form data is invalid. For example, perhaps they're filling out an online application and you want to allow them to save the current state of their application even though some fields might not be valid.

But only Opera and Firefox 4 currently support validation on submission, so all other browsers are currently exhibiting the `novalidate` behavior by default. However, we can use Opera to illustrate.

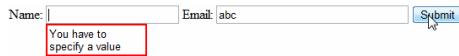
Take a look at the following code.

**Code Sample:**

<html5-forms/Demos/form-attributes.html>

```
<!DOCTYPE HTML>
---- C O D E   O M I T T E D ----
<form method="post" action="" autocomplete="on">
<!--novalidate attribute not included-->
<label for="fullname">Name: </label>
<input type="text" name="fullname" id="fullname" required>
<label for="email">Email: </label>
<input type="email" name="email" id="email">
<input type="submit">
</form>
---- C O D E   O M I T T E D ----
```

The screenshot below shows this page in Opera. Notice that the `fullname` field is required, but left empty, and the `email` field is of the email type, but is not a valid email. So, both fields are currently invalid. When submitting this form in Opera, you get the following results:



If we were to include the `novalidate` attribute, the form would submit without erroring.

But Opera's (and Firefox's) behavior is a bit ugly as it only reports the first error found. So at this stage, the promise of validation without JavaScript is yet to be fulfilled. And, unless browsers do a better job than Opera has in reporting form errors on submission, it may remain unfulfilled even after all the browsers technically support the specification.

## Lesson 1, Activity 6: Some Other New Form Field Attributes

**required**

The `required` attribute is used to indicate that a form field must contain data. It is only supported by:

1. Opera
2. Chrome
3. Firefox 4

**Code Sample:**

<html5-forms/Demos/required.html>

```

---- CODE OMITTED ----
Name:

---- CODE OMITTED ----

```

There is a bug in WebKit browsers (Chrome and Safari) that prevents `required` from working on `color` input types. Open <html5-forms/Demos/webkit-color-required-bug.html> in Chrome or Safari to see the bug.

**placeholder**

The `placeholder` attribute is used to add placeholder text to the form field. It is only supported by:

1. Opera
2. Chrome
3. Firefox 4

**Code Sample:**

<html5-forms/Demos/placeholder.html>

```

Name:


---- CODE OMITTED ----

```

Here's what it looks like in Firefox 4:

Name:

**autofocus**

The `autofocus` attribute can only go in one field on the page. It instructs the browser to place focus on that field allowing the user to begin typing as soon as the page loads.

**Code Sample:**

<html5-forms/Demos/autofocus.html>

```

---- CODE OMITTED ----
Name:

Email:


---- CODE OMITTED ----

```

We used to accomplish this by adding `onload="document.getElementById('fullname').focus();"` to the tag or some other similar scripting method. The HTML5 way is easier and comes with at least two added benefits:

1. It works when scripting is turned off.
2. Browser makers could potentially provide a preference setting for disabling autofocus.

Unfortunately, there's an associated bug in Google Chrome. When one field in a form contains the `autofocus` attribute, Chrome's number spinboxes break. Check out the following code:

**Code Sample:**

<html5-forms/Demos/chrome-autofocus-bug.html>

```

---- CODE OMITTED ----
Your name:


Your age:

---- CODE OMITTED ----

```

That `autofocus` attribute in the `fullname` field messes up the `age` field's spinbox in Chrome (testing on v. 8.0.552.215 on Windows 7). When you click on the up or down arrow, nothing happens until the field loses focus. To test this:

1. Click on the age field's up arrow three times. Nothing happens.
2. Click somewhere else on the page to remove focus from the element. The value updates to 2.

Chrome registers all three clicks, but doesn't update the value until focus is removed from the element.

Very strange. If you remove the `autofocus` attribute from the `fullname` element, everything works swimmingly.

So, I think the takeaway for now is that we cannot yet use `autofocus` in any forms that include `number` types.

Autofocusing on a form element (the HTML5 way or through script) can cause problems for people using screen readers. For sighted people, it's generally okay if we provide one focus point for the keyboard (i.e., autofocus) and another one for the eyes (e.g., instructions for filling out the form), but for people using screen readers, there is only one focus point. So be careful not to skip over important contextual content when directing focus to the first form field.

**autocomplete**

The `autocomplete` attribute is used to override the browser's or form element's autocomplete behavior on a field-by-field basis. It is widely supported by HTML5-compliant browsers.

**form**

The `form` attribute is used to associate a form element with a form (by the form's `id`) in which it is **not** nested. This would typically be used for styling purposes; for example, if you wanted to have one or more form elements appear separately from the main form.

Note that the `form` attribute can be used with the `label` and `fieldset` elements as well as all the form fields (e.g., `input`, `textarea`, `select`,...). Unfortunately, only Opera supports it.

**pattern**

The `pattern` attribute is used to force a specific pattern (via a regular expression) within a form field. It is only supported by:

1. Opera
2. Chrome
3. Firefox 4

**Code Sample:**

---

<html5-forms/Demos/pattern.html>

```
---- C O D E   O M I T T E D ----
Telephone:

---- C O D E   O M I T T E D ----
```

Open this file in one of the supporting browsers and enter data in the field. It should remain red until the value is a valid 10-digit U.S.-style phone number. It will allow for parentheses around the area code and for dashes, spaces, and dots as separators.



## Lesson 1, Activity 7: New Form Elements

HTML5 introduces these new elements, which are often associated with forms:

1. output
2. datalist
3. progress
4. meter

We covered [the output element](#) earlier in this lesson.

The others are covered below.

**datalist**

The `datalist` element combined with the `list` attribute is used to provide a list of suggestions for an input field. It differs from the `select` element in that the associated `input` can accept values that are not included in the `datalist`. It is only supported by:

1. Opera
2. Firefox 4

**Code Sample:**

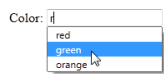
[html5-forms/Demos/datalist.html](#)

```

---- C O D E   O M I T T E D ----
<label for="color">Color: </label>
<input type="color" name="color" id="color" list="color-list">
<datalist id="color-list">
  <option value="red">
  <option value="blue">
  <option value="green">
  <option value="yellow">
  <option value="orange">
</datalist>
---- C O D E   O M I T T E D ----

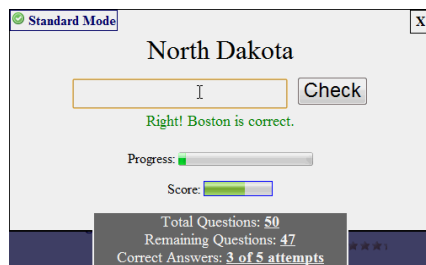
```

Here's what it looks like in Firefox 4:

**progress and meter**

HTML5 introduces the new `progress` and `meter` elements. They sometimes get confused, so we'll illustrate with an example.

We use both `progress` and `meter` in our [HTML5 Cards](#) application as shown below:



The `progress` element is used to show the progress through some action. In this case, through completing the "deck" of flash cards. Another example would be a file download or completion of a form.

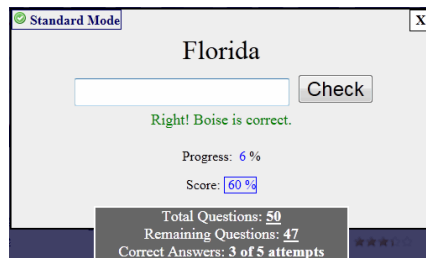
The `meter` element, according to the [spec](#), "represents a scalar measurement within a known range". So the `meter` element is only used when you know the minimum and maximum values. We use it in the above HTML5 cards example to show the user's current score (between 0% and 100%). This is different from `progress`, which shows how much of the deck has been completed. The HTML markup for the two tags is shown below:

```

Progress: <progress id="amountComplete" max="100" value="0"><span>0</span></progress>
Score: <meter id="score" max="100" value="0"><span>0</span></meter>

```

Only Chrome and Opera support the `progress` and `meter` tags, so we use the nested `<span>` tags in combination with JavaScript to provide similar functionality in other browsers. For example, here's what we show in IE9:

**The Form Element's change Event Handler**

The `progress` and `meter` elements can be updated using the new `change` event handler on the `Form`, rather than capturing changes on each individual form element:

```
document.getElementById('my-form').addEventListener('change', updateProgress, false);
```

## Lesson 1, Activity 10: An HTML5 Quiz

Duration: 30 to 45 minutes.

In this exercise, you will create an HTML5 quiz that validates form entries and reports the percentage of both the valid (but not necessarily correct) answers and the percentage of correct answers.

1. Open [html5-forms/Exercises/quiz.html](#) in your editor.
2. Make the following changes to the form:
  1. Add placeholders to all questions.
  2. Make all questions required.
  3. Question 1 should only accept valid colors.
  4. Question 2 should only accept integers greater than or equal to 20.
  5. Question 3 should only accept the pattern shown in the footnote below (don't look if you want to figure out the pattern yourself).
  6. Question 4 should only accept valid dates.
  7. Question 5 should only accept valid URLs and should provide a list of common search engines to choose from, but should not limit the answer to those shown in the list.
3. At the bottom of the form:
  1. Add a bar showing the percentage of valid (but not necessarily correct) answers answered. Give it an id of "quiz-progress".
  2. Add a bar showing the percentage of correct answers answered. Give it an id of "quiz-success".
4. Finish the `updateMeasures()` function so that it correctly updates the two bars added above on every form change.

**Code Sample:**[html5-forms/Exercises/quiz.html](#)

```
<!DOCTYPE HTML>
<html>
<head>
<meta charset="UTF-8">
<title>HTML5 Quiz</title>
<link href="style.css" rel="stylesheet" type="text/css">
<script>
if (window.addEventListener) {
  window.addEventListener("load", addLoadEvents, false);
}
function addLoadEvents() {
  document.getElementById('quiz').addEventListener("change",function() {
    updateMeasures();
  }, false);
}

function updateMeasures() {
  var questions = document.getElementsByTagName("input");
  var numQuestions = questions.length;
  var numAnswers = 0;
  var numCorrectAnswers = 0;
  var answers = ["orange","23","99+9/9","1963-11-22","http://www.google.com"];
  //finish this function
}
</script>
</head>
<body>
<h1>Quiz</h1>
<form method="post" action="process.xyz" id="quiz">
<ol>
<li>
<label for="q1">What color do you get when you mix red and yellow?</label>
<input type="text" name="q1" id="q1">
</li>
<li>
<label for="q2">What is the first primary number greater than 20?</label>
<input type="text" name="q2" id="q2">
</li>
<li>
<label for="q3">Using exactly four 9s and no other digits, write an equation w
evaluates to 100. You may use addition (+), subtraction (-), multiplication (*)
division (/). Do not include spaces.</label>
<input type="text" name="q3" id="q3">
</li>
<li>
<label for="q4">What date was John F. Kennedy assassinated?</label>
<input type="text" name="q4" id="q4">
</li>
<li>
<label for="q5">What is the world's most popular search engine?</label>
<input type="text" name="q5" id="q5">
</li>
</ol>
<!--add bar showing percentage of valid (but not necessarily correct) answers-->
<!--add bar showing percentage of correct answers-->
</form>
</body>
</html>
```

**Challenge**

1. Add code so that the result of the formula the user enters in question 3 is displayed next to the input field like this:
 

9\*9+9+9
 

99
2. Fix the two bars at the bottom of the form so that they present as follows in Opera and other browsers that do not support the `progress` and `meter` elements:

**Quiz**

1. What color do you get when you mix red and yellow?
2. What is the first primary number greater than 20?
3. Using exactly four 9s and no other digits, write an equ
4. What date was John F. Kennedy assassinated?
5. What is the world's most popular search engine?

Progress: 40%    Success: 20%

You'll need to change both your HTML and JavaScript to make this work.

**Solution:**

[html5-forms/Solutions/quiz.html](#)

```

<script>
---- C O D E   O M I T T E D ----
function updateMeasures() {
    var questions = document.getElementsByTagName("input");
    var numQuestions = questions.length;
    var numAnswers = 0;
    var numCorrectAnswers = 0;
    var answers = ["orange","23","99+9/9","1963-11-22","http://www.google.com"];
    for (var i=0; i < numQuestions; i++) {
        if (questions[i].validity.valid && questions[i].value.length>0) {
            numAnswers++;
        }
        if (questions[i].value==answers[i]) {
            numCorrectAnswers++;
        }
    }
    var progress=Math.round(numAnswers/numQuestions * 100);
    var score=Math.round(numCorrectAnswers/numQuestions * 100);
    document.getElementById("quiz-progress").value=progress;
    document.getElementById("quiz-success").value=score;
}
</script>
</head>
<body>
<h1>Quiz</h1>
<form method="post" action="process.xyz" id="quiz">
<ol>
<li>
<label for="q1">What color do you get when you mix red and yellow?</label>
<input type="color" name="q1" id="q1" required placeholder="Enter Color">
</li>
<li>
<label for="q2">What is the first primary number greater than 20?</label>
<input type="number" name="q2" id="q2" min="20" step="1" required placeholder="Enter Number">
</li>
<li>
<label for="q3">Using exactly four 9s and no other digits, write an equation which evaluates to 100. You may use addition (+), subtraction (-), multiplication (*), and division (/). Do not include spaces.</label>
<input type="text" name="q3" id="q3" required placeholder="Enter Equation" pattern="[0-9+\-\/]{4,7}">
</li>
<li>
<label for="q4">What date was John F. Kennedy assassinated?</label>
<input type="date" name="q4" id="q4" required placeholder="Enter Date">
</li>
<li>
<label for="q5">What is the world's most popular search engine?</label>
<input type="url" name="q5" id="q5" required placeholder="Enter URL" list="q5list">
<datalist id="q5list">
<option value="http://www.yahoo.com">
<option value="http://www.google.com">
<option value="http://www.excite.com">
<option value="http://www.dogpile.com">
</datalist>
</li>
</ol>
<strong>Progress:</strong>
<progress id="quiz-progress" min="0" max="100" title="Shows percentage of valid answers"></progress>
<strong>Success:</strong>
<meter id="quiz-success" min="0" max="100" title="Shows percentage of correct answers"></meter>
</form>
</body>
</html>

```

#### Challenge Solution:

[html5-forms/Solutions/quiz-challenge.html](#)

```

---- C O D E   O M I T T E D ----
function addLoadEvents() {
    document.getElementById('q3').addEventListener("change",function() {
        document.getElementById('q3output').innerHTML = eval(this.value);
    }, false);
    document.getElementById('quiz').addEventListener("change",function() {
        updateMeasures();
    }, false);
}

function updateMeasures() {
---- C O D E   O M I T T E D ----

    document.getElementById("quiz-progress").value=progress;
    document.getElementById("quiz-progress").getElementsByName("span")[0].innerHTML=progress;
    document.getElementById("quiz-success").value=score;
    document.getElementById("quiz-success").getElementsByName("span")[0].innerHTML=score;
}
---- C O D E   O M I T T E D ----
<input type="text" name="q3" id="q3" required placeholder="Enter Equation" pattern="[0-9+\-\/]{4,7}">
<output id="q3output"></output>
</li>
<li>
---- C O D E   O M I T T E D ----
<strong>Progress:</strong>
<progress id="quiz-progress" min="0" max="100" title="Shows percentage of valid answers"><span>0</span></progress>
<strong>Success:</strong>
<meter id="quiz-success" min="0" max="100" title="Shows percentage of correct answers"><span>0</span></meter>
---- C O D E   O M I T T E D ----

```