Riverside JS Workshop
JavaScript Skills Ladder, Lesson 1.11

Working with Forms

Lesson Objectives

When you complete this skills ladder lesson, you will be able to:

- *create a page with a form.*
- *add JavaScript to process the form.*
- *set up a click handler for the button input control.*
- *get the value of a text input control, a numeric input control, a select input control, and a text area control.*
- *validate form input and process the data.*
- *clear form controls.*

If you've used or written HTML Forms, you've probably run across JavaScript to *validate* a form: that is, code that checks to see if the values conform to a certain format or range before submitting the form to the back-end server.

You've already seen a couple of examples of using JavaScript with form controls: we've used `document.getElementById()` to get the value of a form input control, and we've used a click handler function with the `onclick` property to detect when a form button control was clicked. We'll expand on these examples in this lesson, and build a web application you can use to create a list of your favorite movies.

Create a Page with a Form

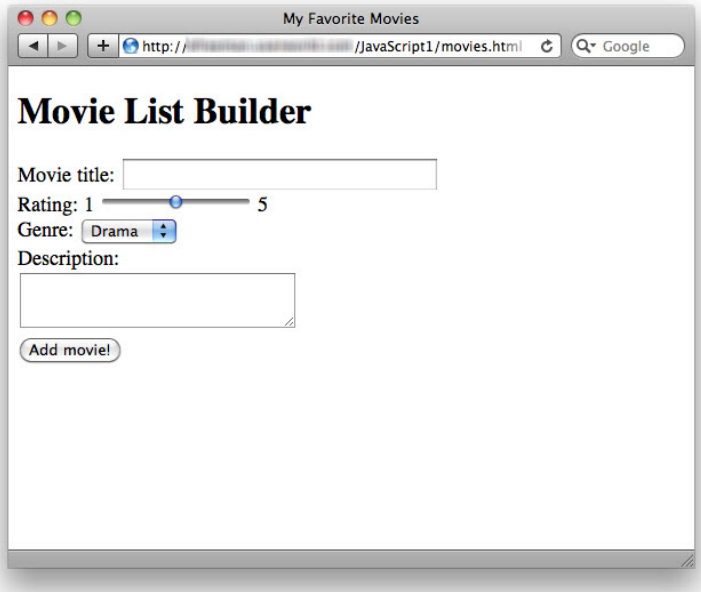
Start by typing in the HTML for a form that you can use to enter a movie title, a rating, a genre and a description. In addition to the controls for entering the movie data, we'll have a button to click to add a movie to the list. Each of these form controls is a different type, so be careful as you type them in. After you type the HTML, we'll look at the JavaScript you need to get values from each of these types of form controls.

CODE TO TYPE:

```
<!doctype html>
<html lang="en">
<head>
  <title> My Favorite Movies </title>
  <meta charset="utf-8">
  <script>
```

```
    // we'll add code here in the next step
  </script>
</head>
<body>
  <h1>Movie List Builder</h1>
  <form>
    <label for="title">Movie title: </label>
      <input id="title" name="title" type="text" size="30"
required><br>
    <label for="rating">Rating: </label>
      1 <input id="rating" name="rating" type="range" max="5" min="1">
5<br>
    <label for="genre">Genre: </label>
      <select id="genre" name="genre">
        <option value="drama" selected>Drama</option>
        <option value="action">Action</option>
        <option value="comedy">Comedy</option>
        <option value="scifi">Sci Fi</option>
        <option value="thriller">Thriller</option>
      </select><br>
    <label for="description">Description:</label><br>
      <textarea id="description" name="description"></textarea>
    <br>
    <input type="button" id="submitButton" value="Add movie!"><br>
  </form>
  <div id="movies">
  </div>
</body>
</html>
```

Save it in your work folder as *movies.html*, and open it in a browser. It will look like this:



We have four different kinds of form controls here:

OBSERVE:

```
  <form>
    <label for="title">Movie title: </label>
      <input id="title" name="title" type="text" size="30"
required><br>
    <label for="rating">Rating: </label>
      1 <input id="rating" name="rating" type="range" max="5" min="1">
5<br>
    <label for="genre">Genre: </label>
      <select id="genre" name="genre">
        <option value="drama" selected>Drama</option>
        <option value="action">Action</option>
        <option value="comedy">Comedy</option>
        <option value="scifi">Sci Fi</option>
        <option value="thriller">Thriller</option>
      </select><br>
    <label for="description">Description:</label><br>
      <textarea id="description" name="description"></textarea>
    <br>
    <input type="button" id="submitButton" value="Add movie!"><br>
  </form>
```

- **text input**: you can type in any kind of content.
- **range input**: you're restricted to whole numbers between the min and max specified.
- **select with options**: you must select one of the options. The default is "drama".
- **textarea**: you can type in any kind of content.

In this example, the only input required from the user is the movie title; if the user doesn't change the rating or genre, they'll get the default values of 3 and "drama," and the description can be empty. Using a range type and a select with options makes it easier to validate the form because we know for sure that the user must select a number in the given range (for the range input) and one of the valid options (for the select input). So the text input for the title is really the only unknown factor in this example.

Notice also that we're using a *button* input type, rather than a *submit* input type. That's because we want to get the values from the form using JavaScript rather than submitting the values to the server script straight away. And in this example, we don't actually submit the values to the server script at all; but if you wanted to, you could do that using JavaScript after you've validated and processed all the form data. (We've included the `name` attribute on all the form controls in case you want to try submitting this to a server-side script; if you don't know how to do this, don't worry about it, you don't need to know for this course!)

Add JavaScript to Process the Form

Now it's time to add some JavaScript to process the form data. Notice that we're reusing the *Movie* object from a previous lab that you did. Take a few minutes to review it. We also have an empty array, `movieList`, that we'll use to store all the movies you enter when you test the page.

Go ahead and type this in now and we'll step through the rest of the code after you've tested it.

CODE TO TYPE:

```
<!doctype html>
<html lang="en">
<head>
  <title> My Favorite Movies </title>
  <meta charset="utf-8">
  <script>
    function Movie(title, rating, genre, description) {
      this.title = title;
      this.rating = rating;
      this.genre = genre;
      this.description = description;
    }

    var movieList = [];

    window.onload = init;

    function init() {
      var submitButton = document.getElementById("submitButton");
      submitButton.onclick = getMovieData;
    }

    function getMovieData() {
      var titleInput = document.getElementById("title");
      var title = titleInput.value;

      var ratingInput = document.getElementById("rating");
      var rating = parseInt(ratingInput.value);

      var genreSelect = document.getElementById("genre");
      var genreOption =
genreSelect.options[genreSelect.selectedIndex];
      var genre = genreOption.value;

      var descriptionTextarea =
document.getElementById("description");
      var description = descriptionTextarea.value;
```

```
      if (title == null || title == "") {
        alert("Please enter a movie title");
        return;
      }
      else {
        var movie = new Movie(title, rating, genre, description);
        movieList.push(movie);
        var movies = document.getElementById("movies");
        movies.innerHTML = "Added " + movie.title + " to the list.";
      }
    }
  </script>
</head>
<body>
  <h1>Movie List Builder</h1>
  <form>
    <label for="title">Movie title: </label>
      <input id="title" type="text" size="30" required><br>
    <label for="rating">Rating: </label>
      1 <input id="rating" type="range" max="5" min="1"> 5<br>
    <label for="genre">Genre: </label>
      <select id="genre">
        <option value="drama">Drama</option>
        <option value="action">Action</option>
        <option value="comedy">Comedy</option>
        <option value="scifi">Sci Fi</option>
        <option value="thriller">Thriller</option>
      </select><br>
    <label for="description">Description:</label><br>
      <textarea id="description"></textarea>
    <br>
    <input type="button" id="submitButton" value="Add movie!"><br>
  </form>
  <div id="movies">
  </div>
</body>
</html>
```

Save it, and open it in a browser. Try adding a movie. Add a few movies, trying different values for all the form controls. We agree. It's not that satisfying because we're not actually adding the movies to the page yet, but we're getting there! If you want to see the movies in your `movieList` array, you can always open up the console and type `movieList` at the console prompt. Since `movieList` is a global variable, you can inspect it using the console.

Let's step through the code now.

Set Up a Click Handler for the Button Input Control

Remember from earlier that the first thing we need to do is set up a *click handler* function that will be called when the *Add movie!* button is clicked. We've got that in the `init()` function as usual. When you click the button with the id "`submitButton`", we call the `getMovieData()` function.

OBSERVE:

```
window.onload = init;

function init() {
    var submitButton = document.getElementById("submitButton");
    submitButton.onclick = getMovieData;
}
```

Get the Value of a Text Input Control

The `getMovieData()` function is where all the real work happens. First, we get the value of the title text input form control:

OBSERVE:

```
var titleInput = document.getElementById("title");
var title = titleInput.value;
```

We get the "title" form element from the DOM and store it in the titleInput variable. To get the string value from this form element, which is a simple text input, we use the value property of the form element. Because this is a required field, we also need to make sure the user entered something, which we do later in the function:

OBSERVE:

```
if (title == null || title == "") {
    alert("Please enter a movie title");
    return;
}
```

There, we check to see if title is null OR the empty string "", and if it is, we alert the user to enter a movie title and return. The return ends the function, so nothing else happens until the user enters a title.

A Slight Sidetrack: Logical Operators

If you're paying close attention you probably noticed the || in the code:

OBSERVE:

```
if (title == null || title == "") {
    alert("Please enter a movie title");
    return;
}
```

and you're wondering what that is. This is a *logical operator* and it means "or." This operator allows you to test two things at once: if the first test is true, then the whole conditional expression is true. Same for the second test; if the second test is true, then the whole conditional expression is true. Likewise if *both* tests are true, then the whole conditional expression is true. The only case where the conditional expression is false is if both tests are false.

Another logical operator you'll use frequently is &&, which means "and." In this case, *both* conditional tests must be true! So, you might use it like this:

```
if (color != null && color == "blue") { ... }
```

In this case it must be true that `color` is not null, *AND* that `color` equals "blue" in order for the conditional expression to be true. There are other logical operators, but these are the two you'll use most frequently.

Get the Value of a Numerical Input Control

Back up a bit in the function, below where we just got the movie title, we get the rating. The rating input is a *range* input control, so we don't have to worry about this input value being empty (like we did for the movie title), but we do need to convert the value to a number (not all browsers do this automatically for this input type yet). We can use the parseInt() function to convert the value we get from the range into an integer number. In this example, we don't actually need the value as a number, but if you want to, say, sort the movies by their rating, it would be handy to have this value as a number rather than a string.

OBSERVE:

```
var ratingInput = document.getElementById("rating");
var rating = parseInt(ratingInput.value);
```

Notice that we're again using the value property of the form element to get the value of the range input, and then using the `parseInt()` function to convert that value to an integer to store in the rating variable.

Note If you want your code to work on older browsers, you will need to use a text input control instead. The range type was added in HTML5 and older browsers don't support it. If you use a text input, you'll need to make sure the user entered a number. Remember that you can use `isNaN()` to check for a number.

Get the Value of a Select Input Control

Next, we get the value of the select input control. This takes an extra step because we need to figure out which option in the select was selected. We do this using the selectedIndex property of the `<select>` element object, and then use that index to get the selected option from the options array.

OBSERVE:

```
var genreSelect = document.getElementById("genre");
var genreOption = genreSelect.options[genreSelect.selectedIndex];
var genre = genreOption.value;
```

Let's walk through an example so you can see how this code to get the selected option works. Suppose you choose "Comedy" from the genre menu. The genreSelect variable holds the entire `<select>` element, and a `<select>` element has a special property, `options`, that contains an array of all the options in the `<select>`. In our case, that array will contain five values, the various movie genres. To find out which item is selected, use the `genreSelect.selectedIndex` property. You selected "Comedy," the third item in the list, which means it's got index 2 in the array of options (because arrays start with an index of 0), so `genreSelect.selectedIndex` equals 2. `genreSelect.options[2]` is the option you selected, and we store that in the genreOption variable. We can then use the `value` property to get the value of the option, which results in the string "Comedy" being stored in the *genre* variable.

Get the Value of a Text Area Control

Finally, we get the value of the textarea control. This is very similar to getting the value of a text input control.

OBSERVE:

```
var descriptionTextarea = document.getElementById("description");
var description = descriptionTextarea.value;
```

Validate Form Input and Process the Data

Now that we've got all the data from the form, we can check the data. As we noted above, the only input we need to check is the "title" text input, and all we're going to do in this example is make sure it's not null or empty.

OBSERVE:

```
if (title == null || title == "") {
    alert("Please enter a movie title");
    return;
}
else {
    var movie = new Movie(title, rating, genre, description);
    movieList.push(movie);
    var movies = document.getElementById("movies");
    movies.innerHTML = "Added " + movie.title + " to the list.";
}
```

If the "title" input data is okay, we create a new movie object using the Movie constructor, and pass in the four pieces of data we retrieved from the form: title, rating, genre, and description.

If the user didn't enter a description, then the *description* argument will be the empty string, which is fine. Once we've created the new movie object, we add it to the `movieList` array using the array `push()` method, which just appends the item to the end of the array.

Finally, we update the content of the movies `<div>` to display the title of the movie we just added.

Clearing Form Controls

After you click the button to add a movie to the list, all the values in the form controls stay in the form controls, so to add another movie, you need to change the values? That can be a pain... so let's *reset* the form. You might have done this previously by adding a Reset button to your HTML, like this:

```
<input type="reset" value="Reset">
```

You can do a reset using JavaScript instead. The best time to do it is after you've collected all the form data and made sure it's correct. So update the `getMovieData()` function to add this code at the bottom, in the `else` clause. And don't forget to add an id "`theForm`" to the form element in the HTML.

CODE TO TYPE:

```
<!doctype html>
<html lang="en">
<head>
  <title> My Favorite Movies </title>
  <meta charset="utf-8">
  <script>
    function Movie(title, rating, genre, description) {
      this.title = title;
      this.rating = rating;
      this.genre = genre;
      this.description = description;
    }

    var movieList = [];

    window.onload = init;

    function init() {
      var submitButton = document.getElementById("submitButton");
      submitButton.onclick = getMovieData;
    }

    function getMovieData() {
      var titleInput = document.getElementById("title");
```

```
      var title = titleInput.value;

      var ratingInput = document.getElementById("rating");
      var rating = parseInt(ratingInput.value);

      var genreSelect = document.getElementById("genre");
      var genreOption =
genreSelect.options[genreSelect.selectedIndex];
      var genre = genreOption.value;

      var descriptionTextarea =
document.getElementById("description");
      var description = descriptionTextarea.value;

      if (title == null || title == "") {
        alert("Please enter a movie title");
        return;
      }
      else {
        var movie = new Movie(title, rating, genre, description);
        movieList.push(movie);
        var movies = document.getElementById("movies");
        movies.innerHTML = "Added " + movie.title + " to the list.";

        var theForm = document.getElementById("theForm");
        theForm.reset();
      }
    }
  </script>
</head>
<body>
  <h1>Movie List Builder</h1>
  <form id="theForm">
    <label for="title">Movie title: </label>
      <input id="title" type="text" size="30" required><br>
    <label for="rating">Rating: </label>
      1 <input id="rating" type="range" max="5" min="1"> 5<br>
    <label for="genre">Genre: </label>
      <select id="genre">
        <option value="drama">Drama</option>
        <option value="action">Action</option>
        <option value="comedy">Comedy</option>
        <option value="scifi">Sci Fi</option>
        <option value="thriller">Thriller</option>
    </select><br>
    <label for="description">Description:</label><br>
      <textarea id="description"></textarea>
    <br>
    <input type="button" id="submitButton" value="Add movie!"><br>
  </form>
  <div id="movies">
  </div>
```

```
</body>
</html>
```

Save it, and open it in a browser. Try entering a movie. The form resets after you click *Add movie!* Try adding a few more movies.

Notice that we only reset the form when we've successfully got all the data. It the user enters other values, but forgets the title, we don't want to reset in that case, because that would force them to re-enter those other values. It's a good idea to consider carefully where in your code you reset a form. And we're done! In the next lesson, you'll learn about form collections and how to submit a form.

More about the material in this lesson

The Riverside JS Workshop would like to acknowledge the generosity of O'Reilly Media, Inc. for making this material available to us through the Creative Commons License. We would also like to acknowledge the great work of Elisabeth Robson who authored this content. She is one of our favorites. Elisabeth has written a number of *Head First* programming books for web developers. We recommend them highly. You can browse her work here.