Riverside JS Workshop
JavaScript Skills Ladder, Lesson 2.13

Creating New DOM Elements

Lesson Objectives

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

- *create new elements and add them to the DOM.*
- *insert new elements where you want them.*

When we left our Movie application earlier, you had a web application that you could use to create a list of movies, but no good way to add those movies to the web page. We used `innerHTML` to update the content of a `<div>` element, but what we'd *really* like is a way to add new elements to the web page. In this example, it would be much better if we could create a real list of movies, using the `<ul>` and `<li>` elements.

Creating and Adding New Elements to the DOM

You can indeed create new elements using the *document* method `createElement()`, and add these new elements to a web page using the element object methods `appendChild()` and `insertBefore()`. First, let's look at how to use `appendChild()`. Take a look at the code below. We've updated the Movie code from the previous lesson, so make your changes carefully (or start a whole new file) and then we'll take a closer look at the code.

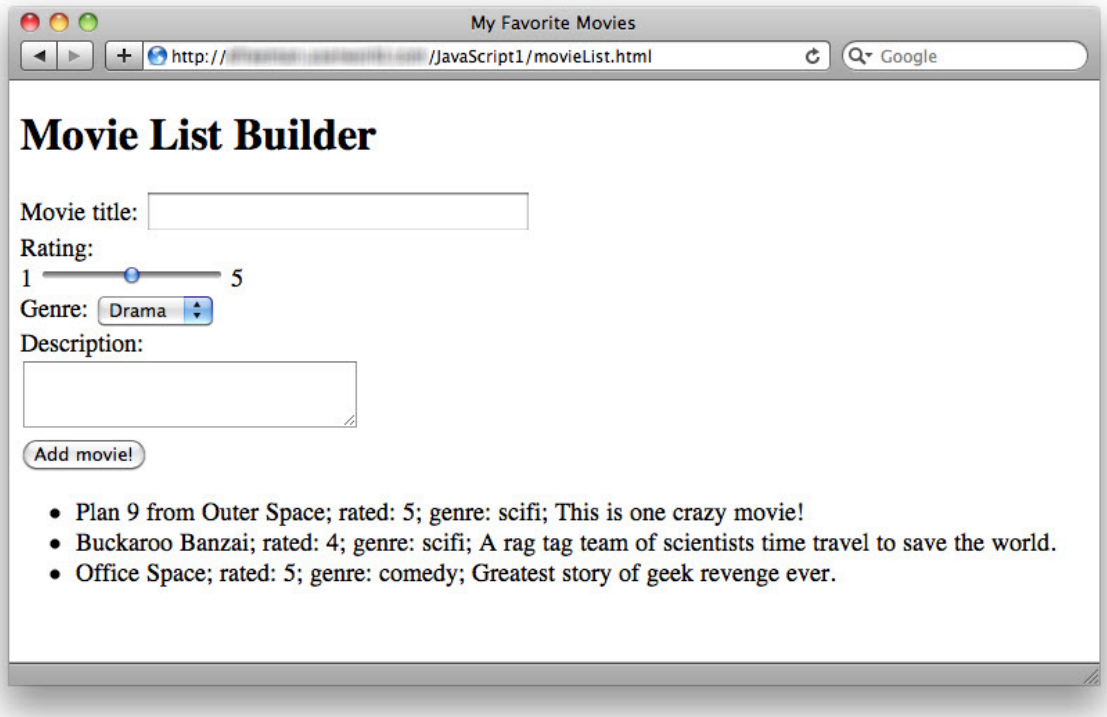
CODE TO TYPE:

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

      this.print = function() {
        var s = this.title + "; rated: " + this.rating + "; genre: " +
this.genre +
                "; " + this.description;
        return s;
```

```
      }
    }

    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.";
        addMovieToList(movie);

        var theForm = document.getElementById("theForm");
        theForm.reset();
      }
    }

    function addMovieToList(movie) {
      var movieList = document.getElementById("movieList");
      var li = document.createElement("li");
      li.innerHTML = movie.print();
      movieList.appendChild(li);
    }

  </script>
</head>
<body>
  <h1>Movie List Builder</h1>
  <form id="theForm">
    <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>
  <ul id="movieList">
  </ul>
</body>
</html>
```

Save it, and open it in a browser. Try adding a few movies. They should appear below the form in a list:



It's much more satisfying to see your movies show up in the web page, don't you agree?! So how does this work? Let's take a closer look.

First, notice that we've removed the "movies" `<div>`, and replaced it with an empty "movieList" unordered list, `<ul>`. This is where we'll add the movies to the page.

Next, we added a `print()` method to the Movie object. This method returns a string that contains information about the movie. We use that string in the `addMovieToList()` function; we'll get to that in a moment.

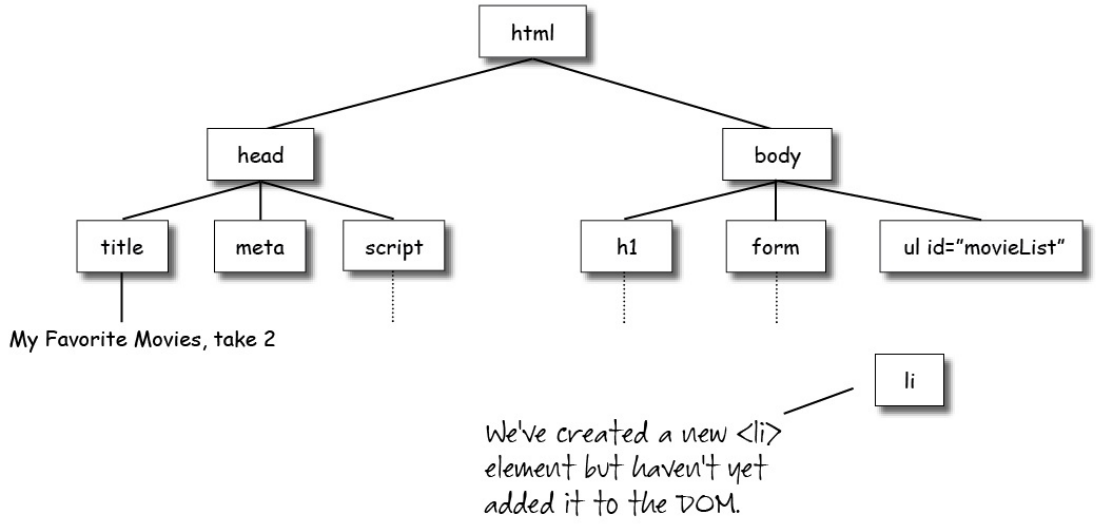
Next, in `getMovieData()`, instead of just updating a `<div>` with a message, we call a new function, `addMovieToList()`, passing the new movie object we created.

OBSERVE:

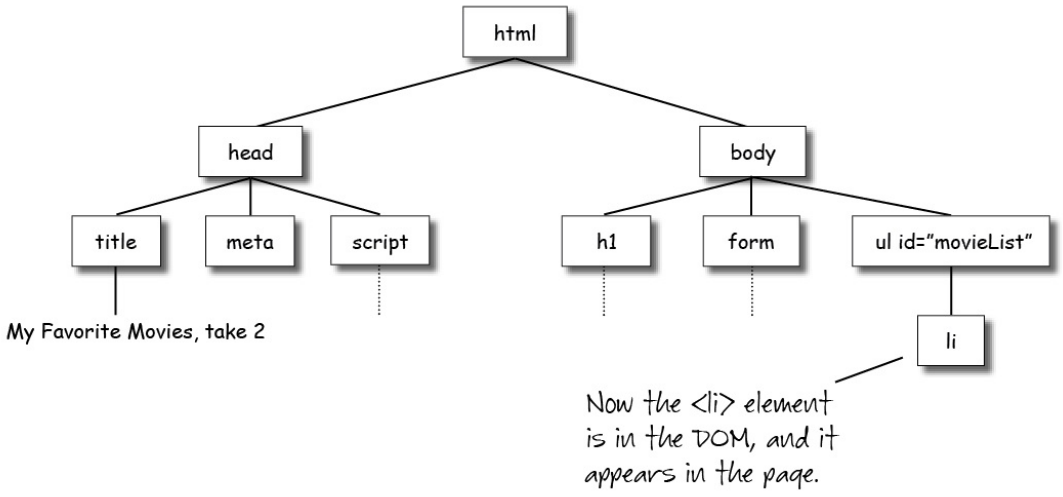
```
function addMovieToList(movie) {
    var movieList = document.getElementById("movieList");
    var li = document.createElement("li");
    li.innerHTML = movie.print();
    movieList.appendChild(li);
}
```

In the `addMovieToList()` function, we create a new `<li>` element to add to the list, using the `document.createElement()` method. We pass in a string with the tag name of the element we want to create, and we use the trusty `innerHTML` property to set the content of that `<li>` element; in this case, we set the content to the string that is returned from the movie object's `print()` method.

So, we have an `<li>` element, with the content we want, but it's just hanging out there. It's not actually part of the page until we add it to the DOM, so you won't see it until you add it.



To do that, we use the `appendChild()` method and add it as a child of the "movieList" `<ul>` element. As soon as we call `appendChild()`, the `<li>` element, with the movie information, appears like magic in our web page!

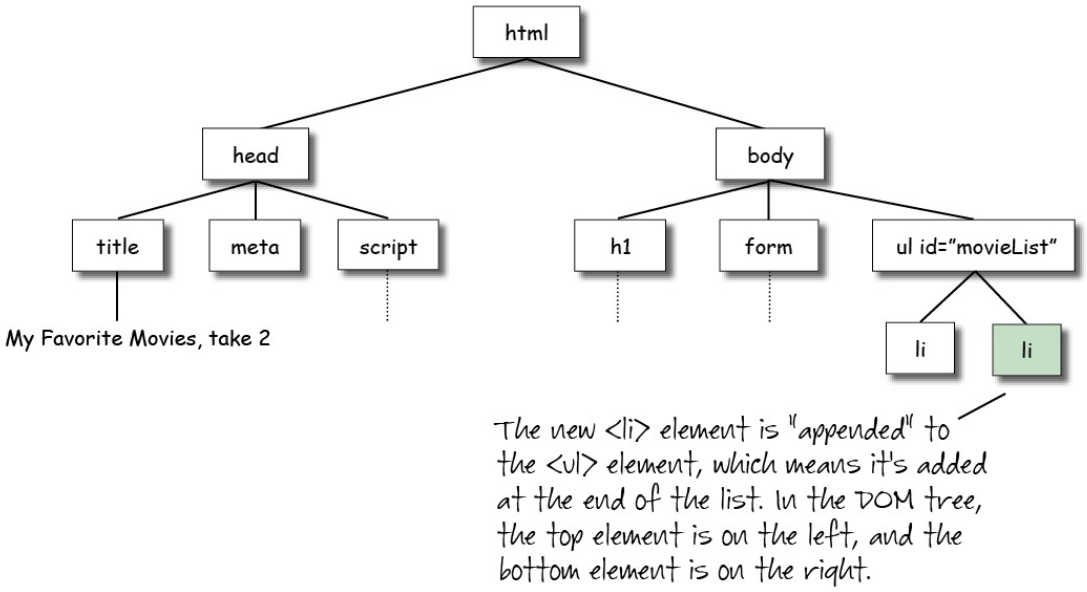


Notice (in `movieList.appendChild(li)`) that the `appendChild()` method is a method of the "movieList" `<ul>` element object.

Every element has this method (so you can add a "child" element to any element using this method). You pass in the element you're adding as a child as the argument to the method; in this case, we pass in the new `<li>` element we just created. The end result is a DOM that contains a new `<li>` element (just as if you'd typed it in the HTML!). Remember, the enclosing element (in this case, the `<ul>` element) is the *parent* element, and the elements enclosed within the parent are the *child* elements (in this case, the `<li>` elements that we are adding with the JavaScript code). In the DOM tree, it's just like a family tree: all the elements directly below an element object in the tree are that element's children.

Adding More Elements

Try adding another movie. Notice that each new movie you add is added at the *end* of the list. Let's take a look behind the scenes when you add a movie to the page and another `<li>` gets added to the DOM:



The second (and third, and so on) movie you add is *appended* to the list because you are using `movieList.appendChild(li)`. `appendChild()` adds the new element to the end of the list. So, what happens if you want to add a new element to the *beginning* of the list instead?

Inserting Elements

To add a child element above the other child elements in the parent element (to the *left* of the other child elements in the DOM tree), use the `insertBefore()` method.

CODE TO TYPE:

```
<!doctype html>
<html lang="en">
<head>
  <title> My Favorite Movies, take 2 </title>
  <meta charset="utf-8">
  <script>
    function Movie(title, rating, genre, description) {
      this.title = title;
      this.rating = rating;
      this.genre = genre;
      this.description = description;

      this.print = function() {
        var s = this.title + "; rated: " + this.rating + "; genre: " +
this.genre +
                "; " + this.description;
        return s;
      }
    }
```

```
    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);
        addMovieToList(movie);
        var theForm = document.getElementById("theForm");
        theForm.reset();
      }
    }

    function addMovieToList(movie) {
      var movieList = document.getElementById("movieList");
      var li = document.createElement("li");
      li.innerHTML = movie.print();
      movieList.appendChild(li);
      if (movieList.childElementCount == 0) {
        movieList.appendChild(li);
      }
      else {
        movieList.insertBefore(li, movieList.firstChild);
      }
    }
  </script>
</head>
<body>
  <h1>Movie List Builder</h1>
  <form id="theForm">
    <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>
  <ul id="movieList">
  </ul>
</body>
</html>
```

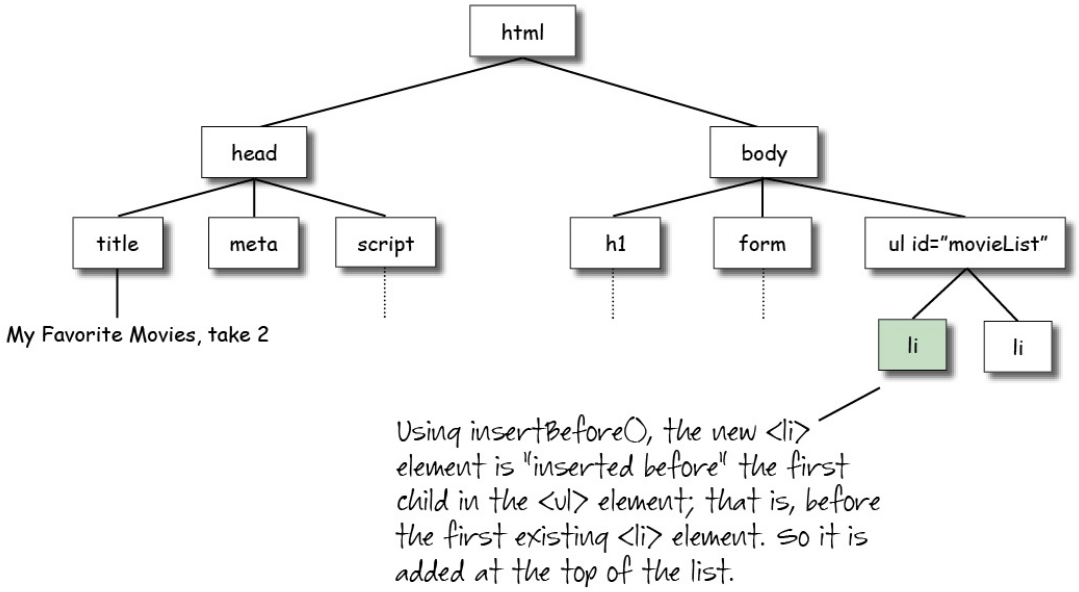
Save it, and open it in a browser. Try adding a few movies. Your new movies are added to the top of the list. Look at the code where we add the new element using `insertBefore()`:

OBSERVE:

```
        if (movieList.childElementCount == 0) {
          movieList.appendChild(li);
        }
        else {
          movieList.insertBefore(li, movieList.firstChild);
        }
```

`insertBefore()` is (again) a method of the parent element, in this case, the "movieList" `<ul>` element, and that it takes two arguments: the first is the new element you're adding, and the second is the existing *first child* in the list. That means we can use `insertBefore()` only if there is already a child element in the "movieList" list.

So, we first check to see if the list has no children using the element object property `childElementCount`. If the movieList has zero children, then we need to add the first child `using appendChild()` as we did before. But, if the `movieList` already has children, then we can use `insertBefore()` to add the new element *before* the first element (which means the new element now becomes the first child). We use the property `firstChild` to get the first child of the parent element. Here's how it looks in the DOM tree:



Now you know how to add elements to the DOM. Try adding other kinds of elements. For instance, try making a page with a `<div>` element and add some `<p>` elements to it.

How would you add an `<img>` element? Can you think of a missing piece you need to properly add this element to the page that we haven't talked about yet? You'll find out what the missing piece is and how to add it using JavaScript in the next lesson.

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.