JavaScript
TUTORIAL

JavaScript select Element

Summary: in this tutorial, you will learn how to handle the `<select>` element in JavaScript.

Introduction to the HTML select elements

A `<select>` element provides you with a list of options. A `<select>` element allows you to select one or multiple options.

To create a `<select>` element, you use the `<select>` and `<option>` elements. For example:

```
<select id="framework">
    <option value="1">Angular</option>
    <option value="2">React</option>
    <option value="3">Vue.js</option>
    <option value="4">Ember.js</option>
</select>
```

Output

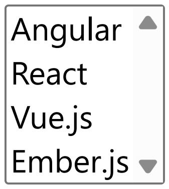
Angular

The above `<select>` element allows you to select a single option at a time.

To enable multiple selections, you add `multiple` attribute to `<select>` element as follows:

```
<select id="framework" multiple>
    <option value="1">Angular</option>
    <option value="2">React</option>
    <option value="3">Vue.js</option>
    <option value="4">Ember.js</option>
</select>
```

Output



The HTMLSelectElement type

To interact with `<select>` element in JavaScript, you use the `HTMLSelectElement` type.

The `HTMLSelectElement` type has the following useful properties:

- `selectedIndex` – returns the zero-based index of the selected option. The `selectedIndex` is `-1` if no option is selected. If the `<select>` element allows multiple selections, the `selectedIndex` returns the `value` of the first option.
- `value` – returns the `value` property of the first selected option element if there is one. Otherwise, it returns an empty string.
- `multiple` – returns `true` if the `<select>` element allows multiple selections. It is equivalent to the `multiple` attribute.

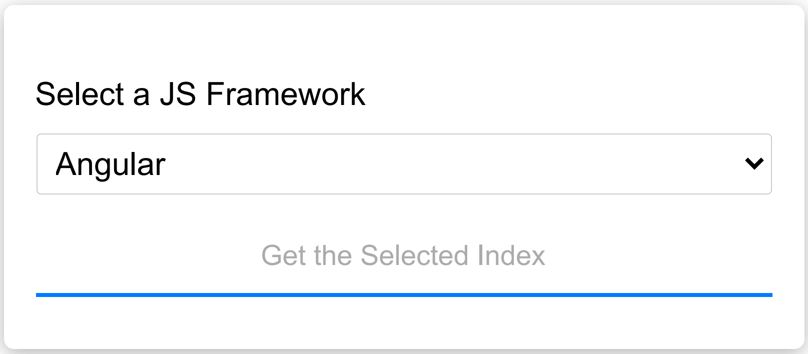
The selectedIndex property

To select a `<select>` element, you use the DOM API like `getElementById()` or `querySelector()` .

The following example illustrates how to get the index of the selected option:

```
<!DOCTYPE html>
<html>
<head>
    <title>JavaScript Select Element Demo</title>
    <link href="css/selectbox.css" rel="stylesheet">
</head>
<body>
    <form>
        <label for="framework">Select a JS Framework</label>
        <select id="framework">
            <option value="1">Angular</option>
            <option value="2">React</option>
            <option value="3">Vue.js</option>
            <option value="4">Ember.js</option>
        </select>
        <button id="btn">Get the Selected Index</button>
    </form>
    <script>
        const btn = document.querySelector('#btn');
        const sb = document.querySelector('#framework')
        btn.onclick = (event) => {
            event.preventDefault();
            // show the selected index
            alert(sb.selectedIndex);
        };
    </script>
</body>
</html>
```

Output



How it works:

- First, select the `<button>` and `<select>` elements using the `querySelector()` method.
- Then, attach a click event listener to the button and show the selected index using the `alert()` method when the button is clicked.

The value property

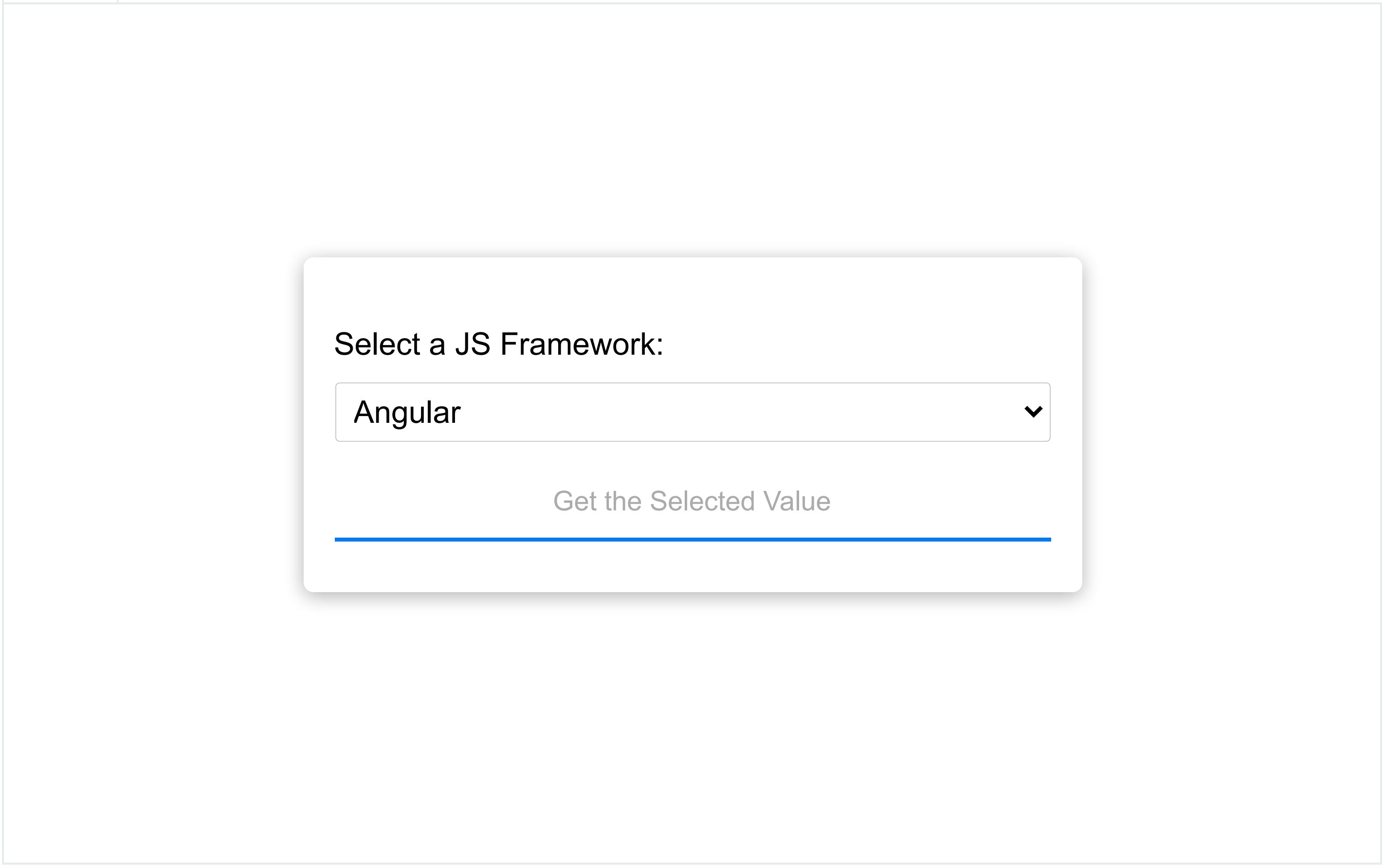
The `value` property of the `<select>` element depends on the `<option>` element and its HTML `multiple` attribute:

- If no option is selected, the `value` property of the select box is an empty string.
- If an option is selected and has a `value` attribute, the `value` property of the select box is the value of the selected option.
- If an option is selected and has no `value` attribute, the `value` property of the select box is the text of the selected option.
- If multiple options are selected, the `value` property of the select box is derived from the first selected option based on the previous two rules.

See the following example:

```
<!DOCTYPE html>
<html>
<head>
    <title>JavaScript Selected Value</title>
    <link href="css/selectbox.css" rel="stylesheet">
</head>
<body>
    <div id="container">
        <form>
            <label for="framework">Select a JS Framework:</label>
            <select id="framework">
                <option value="">Angular</option>
                <option value="1">React</option>
                <option value="2">Vue.js</option>
                <option>Ember.js</option>
            </select>
            <button id="btn">Get the Selected Value</button>
        </form>
    </div>
    <script>
        const btn = document.querySelector('#btn');
        const sb = document.querySelector('#framework')
        btn.onclick = (event) => {
            event.preventDefault();
            alert(sb.value);
        };
    </script>
</body>
</html>
```

Output



In this example:

- If you select the first option, the `value` property of the `<select>` is empty.
- If you select the last option, the `value` property of the select box is `Ember.js` because the selected option has no `value` attribute.
- If you select the second or third option, the value property will be `"1"` or `"2"` .

The HTMLOptionElement type

In JavaScript, the `HTMLOptionElement` type represents the `<option>` element.

The `HTMLOptionElement` type has the following handy properties:

- `index` – the index of the option inside the collection of options.
- `selected` – returns `true` when the option is selected. You set this property to `true` to select an option.
- `text` – returns the option's text.
- `value` – returns the HTML value attribute.

The `<select>` element has the `options` property that allows you to access the collection options:

```
selectBox.options
```

For example, to access the `text` and `value` of the second option, you use the following:

```
const text = selectBox.options[1].text;
const value = selectBox.options[1].value;
```

To get the selected option of a `<select>` element with a single selection, you use the following code:

```
let selectedOption = selectBox.options[selectBox.selectedIndex];
```

Then you can access the `text` and `value` of the selected option via `text` and `value` properties:

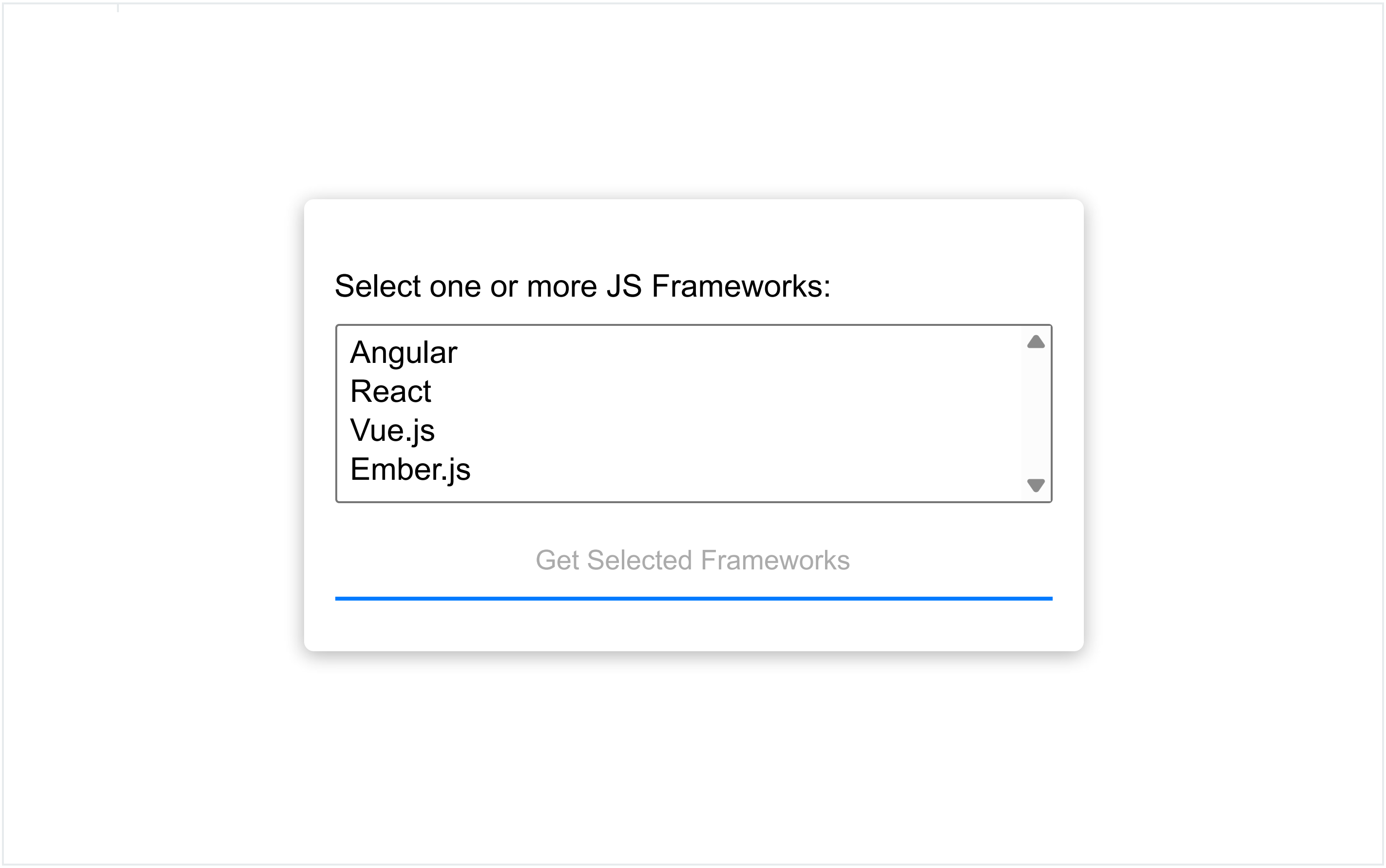
```
const selectedText = selectedOption.text;
const selectedValue = selectedOption.value;
```

When a `<select>` element allows multiple selections, you can use the `selected` property to determine which options are selected:

```html
<!DOCTYPE html>
<html>
<head>
    <title>JavaScript Select Box</title>
    <link rel="stylesheet" href="css/selectbox.css">
</head>
<body>
    <div id="container">
        <form>
            <label for="framework">Select one or more JS Frameworks:</label>
            <select id="framework" multiple>
                <option value="1">Angular</option>
                <option value="2">React</option>
                <option value="3">Vue.js</option>
                <option value="4">Ember.js</option>
            </select>
            <button id="btn">Get Selected Frameworks</button>
        </form>
    </div>
    <script>
        const btn = document.querySelector('#btn');
        const sb = document.querySelector('#framework');

        btn.onclick = (e) => {
            e.preventDefault();
            const selectedValues = [].filter
                .call(sb.options, option => option.selected)
                .map(option => option.text);
            alert(selectedValues);
        };
    </script>
</body>
</html>
```

Output



In this example, the `sb.options` is an array-like object, so it doesn't have the `filter()` methods like an `Array` object.

To borrow these methods from the `Array` object, you use the `call()` method. For example, the following returns an array of selected options:

```
[].filter.call(sb.options, option => option.selected)
```

And to get the `text` property of the options, you chain the result of the `filter()` method with the `map()` method, like this:

```
.map(option => option.text);
```

Summary

- The `<select>` element allows you to select one or multiple options. Add the `multiple` attribute to the `<select>` element to enable multiple selections.

- The `HTMLSelectElement` represents the `<select>` element. Use the `selectedIndex` and `value` to get the index and value of the selected option.
- The `HTMLOptionElement` represents the `<option>` element. If the option is selected, the `selected` property is true. The `selectedText` and `selectedValue` properties return the `text` and `value` of the selected option.