JavaScript: Dynamically Add & Remove Options

Summary: in this tutorial, you will learn how to dynamically add options to and remove options from a select element in JavaScript.

The `HTMLSelectElement` type represents the `<select>` element. It has the `add()` method that dynamically adds an option to the `<select>` element and the `remove()` method that removes an option from the `<select>` element:

- `add(option,existingOption)` adds a new `<option>` element to the `<select>` before an existing option.
- `remove(index)` removes an option specified by the index from a `<select>` .

Adding options

To add an option dynamically to a `<select>` element, you use these steps:

- First, create a new option.
- Second, add the option to the select element.

There are multiple ways to create an option dynamically and add it to a `<select>` in JavaScript.

1) Using the Option constructor and add() method

First, use the `Option` constructor to create a new option with the specified option text and value:

```
let newOption = new Option('Option Text','Option Value');
```

Then, call the `add()` method of the `HTMLSelectElement` element:

```
const select = document.querySelector('select');
select.add(newOption,undefined);
```

The `add()` method accepts two arguments. The first argument is the new option and the second one is an existing option.

In this example, we pass `undefined` in the second argument, the method `add()` method will add the new option to the end of the options list.

2) Using the DOM methods

First, construct a new option using DOM methods:

```
// create option using DOM
const newOption = document.createElement('option');
const optionText = document.createTextNode('Option Text');
// set option text
newOption.appendChild(optionText);
// and option value
newOption.setAttribute('value','Option Value');
```

Second, add the new option to the select element using the `appendChild()` method:

```
const select = document.querySelector('select');
select.appendChild(newOption);
```

Removing Options

There are also multiple ways to dynamically remove options from a select element.

The first way is to use the `remove()` method of the `HTMLSelectElement` type. The following illustrates how to remove the first option:

```
select.remove(0);
```

The second way to remove an option is to reference the option by its index using the `options` collection and set its value to `null` :

```
select.options[0] = null;
```

The third way is to use the `removeChild()` method and remove a specified option. The following code removes the first element of a the `selectBox` :

```
// remove the first element:
select.removeChild(selectBox.options[0]);
```

To remove all options of a select element, you use the following code:

```
function removeAll(select) {
    while (select.options.length > 0) {
        select.remove(0);
    }
}
```

When you remove the first option, the select element moves another option as the first option. The `removeAll()` function repeatedly removes the first option in the select element, therefore, it removes all the options.

Putting it all together

We'll build a application that allows users to add a new option from the value of an input text and remove one or more selected options.

Here's the project's structure:

```
├── css
|   └── style.css
├── js
|   └── app.js
└── index.html
```

The index.html:

```
<!DOCTYPE html>
<html>
    <head>
        <title>JavaScript Selected Value</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <link href="css/style.css" rel="stylesheet">
    </head>
    <body>
        <div id="container">
            <form>
                <label for="framework">Framework:</label>
                <input type="text" id="framework" placeholder="Enter a framework" autocomplet

                <button id="btnAdd">Add</button>

                <label for="list">Framework List:</label>
                <select id="list" name="list" multiple>

                </select>
                <button id="btnRemove">Remove Selected Framework</button>
            </form>
        </div>
        <script src="js/app.js"></script>
    </body>

</html>
```

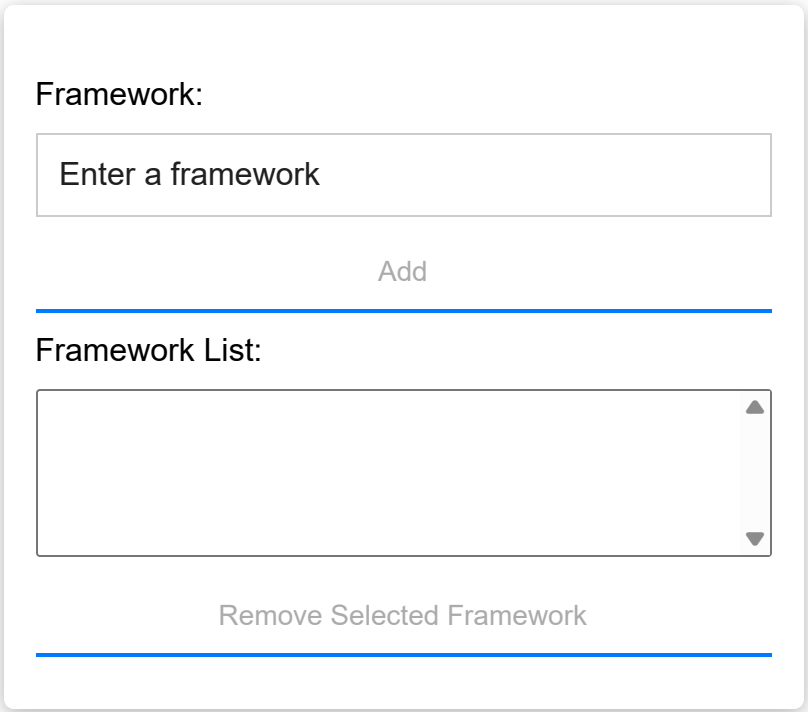
js/app.js

```
const btnAdd = document.querySelector('#btnAdd');
const btnRemove = document.querySelector('#btnRemove');
const listbox = document.querySelector('#list');
const framework = document.querySelector('#framework');

btnAdd.onclick = (e) => {
  e.preventDefault();
```

```
  // validate the option
  if (framework.value == '') {
    alert('Please enter the name.');
    return;
  }
  // create a new option
  const option = new Option(framework.value, framework.value);
  // add it to the list
  listbox.add(option, undefined);

  // reset the value of the input
  framework.value = '';
  framework.focus();
};

// remove selected option
btnRemove.onclick = (e) => {
  e.preventDefault();

  // save the selected options
  let selected = [];

  for (let i = 0; i < listbox.options.length; i++) {
    selected[i] = listbox.options[i].selected;
  }

  // remove all selected option
  let index = listbox.options.length;
  while (index--) {
    if (selected[index]) {
      listbox.remove(index);
    }
  }
};
```

The style can be found here.

Output



How it works:

First, use the `querySelector()` method to select elements including the input text, button, and selection box:

```
const btnAdd = document.querySelector('#btnAdd');
const btnRemove = document.querySelector('#btnRemove');
const listbox = document.querySelector('#list');
const framework = document.querySelector('#framework');
```

Second, attach the click event listener to the `btnAdd` button.

If the value of the input text is blank, we show an alert to inform the users that the name is required. Otherwise, we create a new option and add it to the selection box. After adding the option, we reset the entered text of the input text and set the focus to it.

```
btnAdd.addEventListener('click', (e) => {
  e.preventDefault();

  // validate the option
  if (framework.value.trim() === '') {
    alert('Please enter the name.');
    return;
  }
  // create a new option
  const option = new Option(framework.value, framework.value);

  // add it to the list
  listbox.add(option, undefined);

  // reset the value of the input
  framework.value = '';
  framework.focus();
});
```

Third, register a click event listener to the `btnRemove` button. In the event listener, we save the selected options in an array and remove each of them.

```
btnRemove.addEventListener('click', (e) => {
  e.preventDefault();

  // save the selected options
  let selected = [];

  for (let i = 0; i < listbox.options.length; i++) {
    selected[i] = listbox.options[i].selected;
  }

  // remove all selected option
  let index = listbox.options.length;
  while (index--) {
    if (selected[index]) {
      listbox.remove(index);
    }
```

```
  }
});
```

Summary

- JavaScript uses the `HTMLSelectElement` type to represent the `<select>` element.
- Use the `add()` method of the `HTMLSelectElement` to add an option to the `<select>` element.
- Use the `remove()` method of the `HTMLSelectElement` to remove an option from the `<select>` element.