

Advanced JavaScript Programming

Dynamic Forms

Lesson 1, Activity 2: Jump Menus

A *jump menu* is a select menu that contains a list of websites or pages to visit. There are two main types of jump menus. One jumps to the selected page as soon as the user makes a selection. The other jumps to a page only after the user has made a selection and clicked on a "Go" button. We'll look at the latter type first and then create the former type in an exercise.

Code Sample:

[DynamicForms/Demos/JumpMenus.html](#)

```

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

<script type="text/javascript">
function jumpMenu(e) {
  var btn = getTarget(e);
  var select = btn.form.state;
  var i = select.selectedIndex;
  var selection = select.options[i].value;
  var url;
  if (i === 0) {
    alert("Please select a state.");
  } else {
    url = "http://www.50states.com/" + selection + ".htm";
    location.href = url;
  }
}

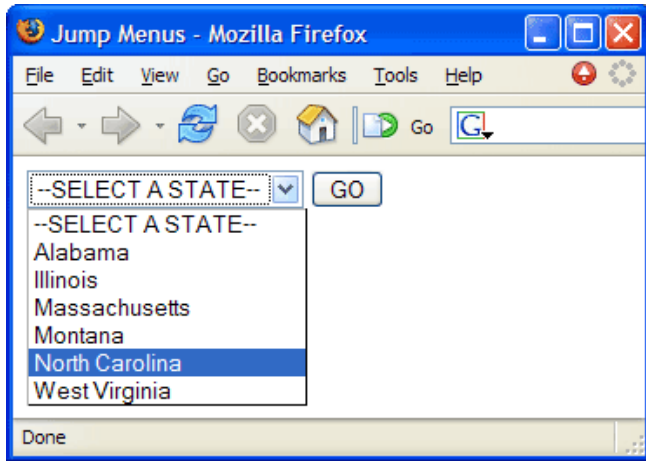
observeEvent(window, "load", function() {
  var btn = document.getElementById("stbtn");
  observeEvent(btn, "click", jumpMenu);
});
</script>
</head>
<body>
<form action="">
  <select name="state" id="state">
    <option value="0">--SELECT A STATE--</option>
    <option value="alabama">Alabama</option>

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

  </select>
  <input id="stbtn" name="stbtn" type="button" value="GO">
</form>
</body>
</html>

```

Viewed in a browser, the page looks like this:



The options[] Collection

Select menus contain a collection of options. Collections, like arrays, are base 0, meaning the first item has an index of 0. So, in this example, the first option, which reads "--SELECT A STATE--" is option 0.

The selectedIndex Property

Select menus have a `selectedIndex` property that holds the index of the currently selected option. For example, in the sample above, if North Carolina is selected, `state.selectedIndex` would hold 5. To see if the first option of a select menu is selected, check if `selectedIndex` is 0.

Let's look at the code above in detail.

1. The "GO" button has an event listener so that the `jumpMenu()` function is called when the user clicks it.
2. The `jumpMenu()` function does the following:
 1. Gets the target of the click event so that it can identify the `select` menu.
 2. Creates the variable `i` that holds the `selectedIndex` property of the `select` menu.
 3. Creates the variable `selection` that holds the value of the selected option.
 4. Checks to see if the first option is selected.
 - If it is, the function alerts the user to select an option.
 - If it is not, the function determines the destination page based on `selection`, assigns that value to the `url` variable, and loads that page by changing the `href` property of the `location` object to `url`.

Disabling Form Elements

As we have seen, form elements can be disabled by setting the element's `disabled` property to `true`. They can be re-enabled by setting the `disabled` property to `false`.

```
formElement.disabled = true;
formElement.disabled = false;
```

The example below is a modified version of the jump menu that disables the "GO" button unless a state is selected.

Code Sample:

DynamicForms/Demos/JumpMenus2.html

```

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

<script type="text/javascript">
function jumpMenu(e) {
    var btn = getTarget(e);
    var select = btn.form.state;
    var i = select.selectedIndex;
    var selection = select.options[i].value;
    var url = "http://www.50states.com/" + selection + ".htm";
    location.href = url;
}

function toggleButton(e) {
    var select = getTarget(e);
    var btn = select.form.stbtn;
    if (select.selectedIndex === 0) {
        btn.disabled = true;
    } else {
        btn.disabled = false;
    }
}

observeEvent(window,"load",function() {
    var btn = document.getElementById("stbtn");
    observeEvent(btn,"click",jumpMenu);
    btn.disabled = true;
    var stateSel = document.getElementById("state");
    observeEvent(stateSel,"change",toggleButton);
});
</script>
---- C O D E   O M I T T E D ----

```

Notice that the `jumpMenu()` function no longer checks if a state has been selected. It is only called when the user clicks the "Go" button, which is only enabled if a state is selected.

Lesson 1, Activity 4: **Modifying the Jump Menu**

Duration: 15 to 25 minutes.

In this exercise, you will modify the jump menu from the demo so that it jumps to the selected page as soon as the user makes a selection.

1. Open [DynamicForms/Exercises/JumpMenus.html](#) for editing.
2. Remove the "GO" button from the form and remove all related JavaScript.
3. Modify the JavaScript so that when the user selects a state, the `jumpMenu()` function is called immediately. Note that the target of the event is no longer the button, but the `select` menu itself, so you will have to modify the `jumpMenu()` function accordingly.
4. Test the solution in a browser.
5. Press the **Back** button and select the "SELECT A STATE" option. Did it try to go to <http://www.50states.com/0.htm>? Fix the function so that it doesn't try to change the location if the first option is selected.

Challenge

The `jumpMenu()` function isn't reusable. Why not? Fix the function so that it is portable and then modify the form accordingly.

Solution:

[DynamicForms/Solutions/JumpMenus.html](#)

```
<!DOCTYPE HTML>
<html>
<head>
<meta charset="UTF-8">
<title>Jump Menus</title>
<script type="text/javascript" src="../../lib.js"></script>
<script type="text/javascript">
function jumpMenu(e) {
    var select = getTarget(e);
    var i = select.selectedIndex;
    if (i > 0) {
        var selection = select.options[i].value;
        var url = "http://www.50states.com/" + selection + ".htm";
        location.href = url;
    }
}

observeEvent(window, "load", function() {
    var stateSel = document.getElementById("state");
    observeEvent(stateSel, "change", jumpMenu);
});
</script>
</head>
<body>
<form id="jumpform">
<select name="state" id="state">
<option value="0">--SELECT A STATE--</option>
<option value="alabama">Alabama</option>
```

```

<option value="illinois">Illinois</option>
<option value="massachu">Massachusetts</option>
<option value="montana">Montana</option>
<option value="ncarolin">North Carolina</option>
<option value="wvirgini">West Virginia</option>
</select>
</form>
</body>
</html>

```

Challenge Solution:

[DynamicForms/Solutions/JumpMenus-challenge.html](#)

```

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

<script type="text/javascript">
function jumpMenu(e) {
  var select = getTarget(e);
  var i = select.selectedIndex;
  if (i > 0 ) {
    var url = select.options[i].value;
    location.href = url;
  }
}

observeEvent(window,"load",function() {
  var stateSel = document.getElementById("state");
  observeEvent(stateSel,"change",jumpMenu);
});
</script>
</head>
<body>
<form id="jumpform">
<select name="state" id="state">
  <option value="0">--SELECT A STATE--</option>
  <option value="http://www.50states.com/alabama.htm">Alabama</option>
  <option value="http://www.50states.com/illinois.htm">Illinois</option>
  <option value="http://www.50states.com/massachu.htm">Massachusetts</option>
  <option value="http://www.50states.com/montana.htm">Montana</option>
  <option value="http://www.50states.com/ncarolin.htm">North Carolina</option>
  <option value="http://www.50states.com/wvirgini.htm">West Virginia</option>
</select>
</form>
</body>
</html>

```

Lesson 1, Activity 5: Adding Options to a Select Menu

As we learned previously, select menus have an `options` collection. You can create a new option using the `Option()` constructor and then add it to the `options` collection using the `add()` method of the `select` object.

Select objects have a corresponding `remove(position)` method used to remove the option at the passed-in position (index). You will see this used in the last demo of the lesson.

The `Option()` constructor takes four parameters: `text`, `value`, `defaultSelected`, and `selected`. Only `text` is required. The following demo shows how to dynamically add color options to a select list:

Code Sample:

<DynamicForms/Demos/adding-options.html>

```
<!DOCTYPE HTML>
<html>
<head>
<meta charset="UTF-8">
<title>Adding Options</title>
<link rel="stylesheet" href="adding-options.css" type="text/css">
<script type="text/javascript" src="../../lib.js"></script>
<script type="text/javascript">
function addOption() {
    var sel = document.getElementById("color-list");
    var name = document.getElementById("new-option-name").value;
    var value = document.getElementById("new-option-value").value;
    var option = new Option(name,value);
    try {
        sel.add(option,null);
    } catch (e) {
        //for IE7 and earlier
        if (e.name == "TypeError") {
            sel.options[sel.options.length] = option;
        } else {
            throw e;
        }
    }
}

observeEvent(window,"load",function() {
    var btn = document.getElementById("add-option");
    var sel = document.getElementById("color-list");
    observeEvent(btn,"click",addOption);
    observeEvent(sel,"change",function() {
        if(sel.selectedIndex == 0) {
            document.bgColor="#ffffff";
        } else{
            document.bgColor=sel.options[sel.selectedIndex].value;
        }
    });
});
</script>
```

```

</head>
<body>
  <h1>Adding Options</h1>
  <form action="">
    <label>Colors:</label>
    <select id="color-list" name="color-list">
      <option value="">Please select</option>
    </select>
    <hr>
    <label>Name:</label>
    <input type="text" id="new-option-name" name="new-option-name" value="Red">
    <br>
    <label>Value:</label>
    <input type="text" id="new-option-value" name="new-option-value" value="#ff0000">
    <br>
    <input type="button" id="add-option" value="Add Option">
  </form>

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

</body>
</html>

```

When the page loads, we begin observing click events on the buttons and change events on the `select` list.

When the user clicks the button, we get the values for `name` and `value` from the form elements and create a new option element using `var option = new Option(name,value);`.

We then try to add that option to the `select` object using the standard `add()` method, like this:

`sel.add(option,null);` The second argument is the option which this new option should come before. Specifying `null` makes the new option get appended to the end of the `options` collection. (Note that the second argument is required in Firefox).

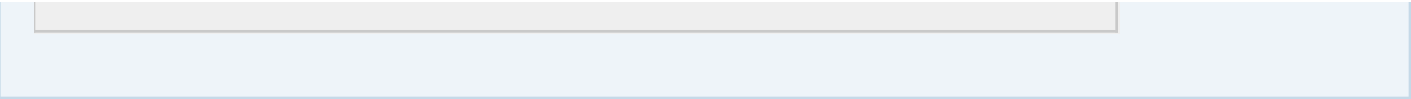
Internet Explorer 7 and earlier do not support adding options in this way, which is why we must put the code in a `try` block. If it fails, we add the option the way IE used to do it: `sel.options[sel.options.length] = option;`

Our [lib.js](#) file contains the following cross-browser function for appending options to select lists:

```

function appendOptionToSelect(sel,opt) {
  try {
    sel.add(opt,null);
  } catch (e) {
    //for IE7 and earlier
    if (e.name == "TypeError") {
      sel.options[sel.options.length] = opt;
    } else {
      throw e;
    }
  }
}

```

Lesson 1, Activity 6: Cascading Select Menus

Cascading select menus allow you to populate one select menu based on a choice made in another select menu. For example, a menu of states might change based on the country that was selected in another menu, as shown in the following example:

Code Sample:

[DynamicForms/Demos/cascading.html](#)

```
<!DOCTYPE HTML>
<html>
<head>
<meta charset="UTF-8">
<title>Cascading Dropdowns</title>
<script type="text/javascript" src="../../lib.js"></script>
<script type="text/javascript">
var statesPerCountry = {
  "usa" : ["New York", "Florida", "Iowa", "California"],
  "canada" : ["Ontario", "Quebec", "Manitoba", "Yukon"]
};

function countryChanged() {
  var sel = document.getElementById("country");
  var country = sel.options[sel.selectedIndex].value;
  updateStateList(country);
}

function updateStateList(country) {
  var states = statesPerCountry[ country ];
  var list = document.getElementById("state");
  var opt;
  removeAllChildren(list);
  for (var i in states) {
    opt = new Option(states[i]);
    appendOptionToSelect(list,opt);
  }
}

observeEvent(window,"load",function() {
  var country = document.getElementById("country");
  observeEvent(country,"change",countryChanged);
});
</script>
</head>
<body>
<h1>Cascading Dropdowns</h1>
<form action="">
  <label for="country">Country:</label>
  <select id="country" name="country">
    <option value="">Please select</option>
    <option value="usa">USA</option>
    <option value="canada">Canada</option>
  </select>
  <label for="state">State/Province:</label>
  <select id="state"></select>
</form>
```

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

Let's look at the code above in detail.

1. As the page loads, we create a `statesPerCountry` object that contains two properties: `usa` and `canada`, each of which holds an array of states (provinces).
2. The body of the page contains a form with two select menus: `country` and `state`
3. In the JavaScript, we call the `countryChanged()` function when a change event occurs on `country` select.
4. The `countryChanged()` function determines the name of the country selected and passes it into the `updateStateList()` function, which removes all elements from the `state` select list: `removeAllChildren(list)`; and then adds new options getting the values by looping through the array associated with the passed-in country.

Making the Code Modular

A problem with the code in `DynamicForms/Demos/cascading.html` is that it would need to be modified in several places to add or change options. This next example, though more complex, is much more modular, and hence, reusable.

Code Sample:

DynamicForms/Demos/cascading-refactored.html

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

<script type="text/javascript">
function selChanged(sel,data,dependentSel) {
    var selection = sel.options[sel.selectedIndex].value;
    var arrOptions = data[selection];
    var opt;
    removeAllChildren(dependentSel);
    for (var i in arrOptions) {
        opt = new Option(arrOptions[i]);
        appendOptionToSelect (dependentSel,opt);
    }
}

observeEvent(window,"load",function() {
    var statesPerCountry = {
        "usa" : ["New York", "Florida", "Iowa", "California"],
        "canada" : ["Ontario", "Quebec", "Manitoba", "Yukon"]
    };
    var country = document.getElementById("country");
    var state = document.getElementById("state");
    observeEvent(country,"change",function() {
        selChanged(country,statesPerCountry,state);
    });

    var employeesPerCompany = {
        "Webucator" : ['Nat', 'Connie', 'Brian', 'Steve'],
        "Microsoft" : ['Bill', 'Paul', 'Steve']
    };
});
```

```

var company = document.getElementById("company");
var employee = document.getElementById("employee");
observeEvent(company, "change", function() {
    selChanged(company, employeesPerCompany, employee);
});
});
</script>
</head>
<body>
<h1>Cascading Dropdowns</h1>
<form action="">
<label for="country">Country:</label>
<select id="country" name="country">
<option value="">Please select</option>
<option value="usa">USA</option>
<option value="canada">Canada</option>
</select>
<label for="state">State:</label>
<select id="state"></select>
<hr>
<label for="company">Company:</label>
<select id="company" name="company">
<option value="">Please select</option>
<option value="Webucator">Webucator</option>
<option value="Microsoft">Microsoft</option>
</select>
<label for="employee">Employee:</label>
<select id="employee"></select>
</form>
</body>
</html>

```

Now we have two different objects: `statesPerCountry` and `employeesPerCompany`, each of which holds a couple of arrays to populate the cascading select menus. We observe `change` events on both the country and company select menus. When they change, we call the `selChanged()` function passing in the select menu itself, the relevant object with its arrays, and the dependent select menu.

The `selChanged()` function gets the value of the option selected in the main list, uses that to select the appropriate array from the passed-in object of arrays, and uses that array to populate the dependent select list.

Again, the nice thing about this code is we can reuse it for other cascading (or interdependent) select lists.

Lesson 1, Activity 8: Creating Cascading Select Menus

Duration: 20 to 30 minutes.

In this exercise, you will modify [DynamicForms/Exercises/cascading.html](#), so that when a rock band is selected from the first menu, the second menu provides a list of artists in that band. The bands and artists are shown below:

- Beatles
 - Paul McCartney
 - John Lennon
 - George Harrison
 - Ringo Starr
- Rolling Stones
 - Mick Jagger
 - Keith Richards
 - Charlie Watts
 - Bill Wyman
- Genesis
 - Phil Collins
 - Peter Gabriel
 - Mike Rutherford
- Eagles
 - Don Henley
 - Joe Walsh
 - Glenn Frey

1. Open [DynamicForms/Exercises/cascading.html](#) for editing.
2. Write code so that after the page loads, change events on the rockbands select menu cause the artist menus options to be updated appropriately.

Code Sample:

DynamicForms/Exercises/cascading.html

```
<!DOCTYPE HTML>
<html>
<head>
<meta charset="UTF-8">
<title>Cascading Select Menus</title>
<script type="text/javascript" src="../../lib.js"></script>
<script type="text/javascript">
function selChanged(sel,arr,dependentSel) {
    var selection = sel.options[sel.selectedIndex].value;
    var arrOptions = arr[selection];
    var opt;
    dependentSel.options.length = 0;
    for (var i in arrOptions) {
        opt = new Option(arrOptions[i]);
        appendOptionToSelect(dependentSel,opt);
    }
}
```

```

/*
Beatles
  - Paul McCartney, John Lennon, George Harrison, Ringo Starr
Rolling Stones
  - Mick Jagger, Keith Richards, Charlie Watts, Bill Wyman
Genesis
  - Phil Collins, Peter Gabriel, Mike Rutherford
Eagles
  - Don Henley, Joe Walsh, Glenn Frey
*/
</script>
</head>
<body>
<form name="Menus">
  <label for="band">Band:</label>
  <select id="band" name="band">
    <option value="">Please select</option>
    <option value="beatles">Beatles</option>
    <option value="stones">Rolling Stones</option>
    <option value="genesis">Genesis</option>
    <option value="eagles">Eagles</option>
  </select>
  <label for="artist">Artist:</label>
  <select id="artist"></select>
</form>
</body>
</html>

```

Challenge

Using the `rockBands` object in the provided [rockBands.js](#) file, further modify the code such that when an option is selected from the artist `select` menu, the page redirects to the corresponding URL for that artist.

Code Sample:

DynamicForms/Exercises/rockBands.js

```

var rockBands = {
  "beatles" : [
    {
      "name" : "Paul McCartney",
      "value" : "http://www.paulmccartney.com"
    },
    {
      "name" : "John Lennon",
      "value" : "http://www.johnlennon.it"
    },
    {
      "name" : "George Harrison",
      "value" : "http://www.georgeharrison.com"
    },
    {
      "name" : "Ringo Starr",
      "value" : "http://www.ringostarr.com"
    }
  ]
}

```

```

],
"stones" : [
  {
    "name" : "Mick Jagger",
    "value" : "http://www.mickjagger.com"
  },
  {
    "name" : "Keith Richards",
    "value" : "http://www.keithrichards.com"
  },
  {
    "name" : "Charlie Watts",
    "value" : "http://www.rosebudus.com/watts"
  },
  {
    "name" : "Bill Wyman",
    "value" : "http://www.billwyman.com"
  }
],
"genesis" : [
  {
    "name" : "Phil Collins",
    "value" : "http://www.philcollins.co.uk"
  },
  {
    "name" : "Peter Gabriel",
    "value" : "http://www.petergabriel.com"
  },
  {
    "name" : "Mike Rutherford",
    "value" : "http://www.mike-and-the-mechanics.co.uk"
  }
],
"eagles" : [
  {
    "name" : "Don Henley",
    "value" : "http://www.donhenley.com"
  },
  {
    "name" : "Joe Walsh",
    "value" : "http://www.joewalsh.com"
  },
  {
    "name" : "Glenn Frey",
    "value" : "http://www.imdb.com/name/nm0004940"
  }
]
};

```

Solution:

[DynamicForms/Solutions/cascading.html](#)

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

```
<script type="text/javascript">
```

```

function selChanged(sel,data,dependentSel) {
    var selection = sel.options[sel.selectedIndex].value;
    var arrOptions = data[selection];
    var opt;
    dependentSel.options.length = 0;
    for (var i in arrOptions) {
        opt = new Option(arrOptions[i]);
        appendOptionToSelect (dependentSel,opt);
    }
}

observeEvent(window,"load",function() {
    var rockBands = {
        "beatles" : ["Paul McCartney", "John Lennon", "George Harrison", "Ringo Starr"],
        "stones" : ["Mick Jagger", "Keith Richards", "Charlie Watts", "Bill Wyman"],
        "genesis" : ["Phil Collins", "Peter Gabriel", "Mike Rutherford"],
        "eagles" : ["Don Henley", "Joe Walsh", "Glenn Frey"]
    };
    var band = document.getElementById("band");
    var artist = document.getElementById("artist");
    observeEvent(band,"change",function() {
        selChanged(band,rockBands,artist);
    });
});
</script>

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

```

Challenge Solution:

[DynamicForms/Solutions/cascading-challenge.html](#)

```

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

<script type="text/javascript" src="rockbands.js"></script>
<script type="text/javascript">
    /*
    * function triggers when the select menu option changes
    * Param sel refers to the select menu itself, in this case: "band"
    * Param data refers to the data being used, in this case: the rockband object in rockband.js
    * Param dependentSel refers to the secondary select menu, in this case: "artist"
    */
    function selChanged(sel,data,dependentSel) {
        //get the selected band
        var selection = sel.options[sel.selectedIndex].value;
        //get the corresponding artists array based on the band selected
        var arrOptions = data[selection];
        var opt;
        dependentSel.options.length = 0;
        /*
        * let's add the options to the secondary select menu (the artists).
        * notice the appendOptionToSelect() function from lib.js
        * 1st we add a "Please Select" option
        * then we loop the arrOptions, adding the option on each iteration
        */
        appendOptionToSelect(dependentSel,new Option("Please select",""));
        for (var i in arrOptions) {
            opt = new Option(arrOptions[i].name,arrOptions[i].value);

```



```

    appendOptionToSelect(dependentSel,opt);
}
//add the "on change" event listener to the newly populated "artists" select menu
observeEvent(dependentSel,"change",function() {
    var i = dependentSel.selectedIndex;
    //if the selected index in the "artists" menu is grater than 0, so is not "Please select", then...
    if (i > 0) {
        /*
        * location.href redirects the page to the URL given.
        * the url to redirect to is gathered from the rockband object.
        * Notice the "value" property here, which refers to the "value" property in the different
        * objects in rockbands.js
        */
        location.href=dependentSel.options[dependentSel.selectedIndex].value;
    }
});
}

observeEvent(window,"load",function() {
    var band = document.getElementById("band");
    var artist = document.getElementById("artist");
    observeEvent(band,"change",function() {
        selChanged(band,rockBands,artist);
    });
});
</script>

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

```

Lesson 1, Activity 9: Creating a JavaScript Timer

JavaScript timers can be used to create timed quizzes or events. Timers are started and stopped with the following four methods of the `Window` object:

1. `setTimeout (code_to_execute, wait_time_in_milliseconds)`
2. `clearTimeout (timer)`
3. `setInterval (code_to_execute, interval_in_milliseconds)`
4. `clearInterval (interval)`

The example below uses `setInterval ()` and `clearInterval ()` to start and stop a timer.

Code Sample:

DynamicForms/Demos/Timer.html

```
<!DOCTYPE HTML>
<html>
<head>
<meta charset="UTF-8">
<title>JavaScript Timer</title>
<script type="text/javascript" src="../../lib.js"></script>
<script type="text/javascript">
function resetTimer(seconds){
    var btnStart = document.getElementById("btnStart");
    var btnReset = document.getElementById("btnReset");
    document.getElementById("timeLeft").value = seconds;
    clearInterval(timer);
    btnStart.disabled = false;
    btnReset.disabled = true;
}

function decrementTimer(){
    var timeLeft = document.getElementById("timeLeft");
    var btnStart = document.getElementById("btnStart");
    var btnReset = document.getElementById("btnReset");
    timesUp = false;
    btnStart.disabled = true;
    btnReset.disabled = false;
    timeLeft.value--;
    if (timeLeft.value < 0) {
        resetTimer(10);
        alert("Time's up!");
    }
}

var timer, timesUp;
observeEvent(window,"load",function() {
    var timeLeft = document.getElementById("timeLeft");
    var btnStart = document.getElementById("btnStart");
    var btnReset = document.getElementById("btnReset");
    observeEvent(btnStart,"click",function() {
        timer=setInterval(decrementTimer, 1000);
    });
    observeEvent(btnReset,"click",function() {
        resetTimer(10);
    });
});
```

```

observeEvent(timeLeft,"focus",function() {
  timeLeft.blur();
});
});

</script>
</head>
<body>
<form name="Timer" onsubmit="return false;">
  <label for="TimeLeft">Timer:</label>
  <input type="text" id="timeLeft" name="timeLeft"
    size="2" value="10"><br>
  <input type="button" name="btnStart" id="btnStart" value="Start">
  <input type="button" name="btnReset" id="btnReset" value="Reset"
    disabled="disabled">
</form>
</body>
</html>

```

Let's look at the code above in detail.

1. As the page loads, two global variables are created: `timer` - to hold the timer, and `timesUp` - to flag if the timer has run out.
2. The body of the page contains a form with a text field holding the number of seconds left, a "Start" button and a "Reset" button.
3. When the page loads, we begin observing events on the `btnStart` and `btnReset` buttons.
4. When the user clicks on the "Start" button, the `decrementTimer()` function is called. This function does the following:
 - Sets `timesUp` to `false`.
 - Disables the "Start" button.
 - Enables the "Reset" button.
 - Decrements the value of the `timeLeft` field by one.
 - Checks to see if the value of the `timeLeft` field is less than zero and calls `resetTimer()` if it is.
5. The `resetTimer()` function is called when the user clicks on the "Reset" button or the timer runs out. This function does the following:
 - Sets the value of the `timeLeft` field to the number passed into the function.
 - Clears the `timer` interval.
 - Enables the "Start" button.
 - Disables the "Reset" button.

Lesson 1, Activity 11: Adding a "Pause" Button to the Timer

Duration: 10 to 20 minutes.

In this exercise, you will add a "Pause" button to the timer.

1. Open [DynamicForms/Exercises/Timer.html](#) for editing.
2. This page is the same as [DynamicForms/Demos/Timer.html](#), except that it has a new "Pause" button that, when clicked, calls the `pauseTimer()` function. Your job is to create this `pauseTimer()` function and to modify the script so that the right buttons are enabled at the right times. The diagram below shows the three different phases:

Phase 1

Timer:

Phase 2

Timer:

Phase 3

Timer:

3. Test your solution in a browser.

Solution:

[DynamicForms/Solutions/Timer.html](#)

```

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

<script type="text/javascript">
function resetTimer(seconds){
    var btnStart = document.getElementById("btnStart");
    var btnPause = document.getElementById("btnPause");
    var btnReset = document.getElementById("btnReset");
    document.getElementById("timeLeft").value = seconds;
    clearInterval(timer);
    btnStart.disabled = false;
    btnPause.disabled = true;
    btnReset.disabled = true;
}

function decrementTimer(){
    var timeLeft = document.getElementById("timeLeft");
    var btnStart = document.getElementById("btnStart");
    var btnPause = document.getElementById("btnPause");
    var btnReset = document.getElementById("btnReset");
    timesUp = false;
    btnStart.disabled = true;
    btnPause.disabled = false;
    btnReset.disabled = false;
    timeLeft.value--;
    if (timeLeft.value < 0) {

```

```

    resetTimer(10);
    alert("Time's up!");
}
}

function pauseTimer() {
    var btnStart = document.getElementById("btnStart");
    var btnPause = document.getElementById("btnPause");
    clearInterval(timer);
    btnPause.disabled = true;
    btnStart.disabled = false;
}

var timer, timesUp;
observeEvent(window,"load",function() {
    var timeLeft = document.getElementById("timeLeft");
    var btnStart = document.getElementById("btnStart");
    var btnReset = document.getElementById("btnReset");
    observeEvent(btnStart,"click",function() {
        timer=setInterval(decrementTimer, 1000);
    });
    observeEvent(btnPause,"click",function() {
        pauseTimer();
    });
    observeEvent(btnReset,"click",function() {
        resetTimer(10);
    });
    observeEvent(timeLeft,"focus",function() {
        timeLeft.blur();
    });
});
</script>
---- C O D E   O M I T T E D ----

```

Lesson 1, Activity 12: A Sample Quiz Tool

The following example brings together the concepts learned in this lesson to create a quiz tool. The quiz looks like this:

Category:	Addition ▼
Question:	--Please Choose-- ▼
Answer:	<input type="text"/> <input type="button" value="Check Answer"/>
Timer:	<input type="text"/> seconds left
Good luck!	

Code Sample:

<DynamicForms/Demos/MathQuiz.html>

```
<!DOCTYPE HTML>
<html>
<head>
<meta charset="UTF-8">
<title>Math Quiz</title>
<link href="MathQuiz.css" rel="stylesheet" type="text/css">
<script type="text/javascript" src="../../lib.js"></script>
<script type="text/javascript" src="questions.js"></script>
<script type="text/javascript">
var timer, timesUp, timePerQuestion=10;
observeEvent(window,"load",init);

function init() {
var category = document.getElementById("category");
var question = document.getElementById("question");
var btnCheck = document.getElementById("btnCheck");
var timeLeft = document.getElementById("timeLeft");
observeEvent(category,"change",function() {
selChanged(category,categories,question);
resetTimer(timePerQuestion);
});
observeEvent(question,"change",function() {
questionChanged();
});
observeEvent(btnCheck,"click",function() {
checkAnswer();
});
observeEvent(timeLeft,"focus",function() {
timeLeft.blur();
});
selChanged(category,categories,question);
resetTimer(timePerQuestion);
}

function selChanged(sel,data,dependentSel) {
var selection = sel.options[sel.selectedIndex].value;
var arrOptions = data[selection];
var opt;
```

```

dependentSel.options.length = 1;
for (var i in arrOptions) {
    opt = new Option(arrOptions[i].q, arrOptions[i].a);
    appendOptionToSelect (dependentSel, opt);
}
}

function resetTimer(seconds) {
    document.getElementById("timeLeft").value = seconds;
    clearInterval(timer);
}

function decrementTimer() {
    var timeLeft = document.getElementById("timeLeft");
    timeLeft.value--;
    if (timeLeft.value < 0) {
        resetTimer(timePerQuestion);
        msg("Time's up! The answer is " + getAnswer() + ".");
        removeOption();
    }
}

function checkAnswer() {
    var userAnswer = document.getElementById("answer").value;
    var correctAnswer = getAnswer();

    if (userAnswer === correctAnswer) {
        msg("Right! The answer is " + correctAnswer + ".");
    } else {
        msg("Sorry. The correct answer is " + correctAnswer + ".");
    }
    removeOption();
    questionChanged();
}

function getAnswer() {
    var i = document.Quiz.question.selectedIndex;
    var answer = document.Quiz.question[i].value;
    return answer;
}

function removeOption() {
    var category = document.Quiz.category;
    var question = document.Quiz.question;
    question.remove(question.selectedIndex);
    if(question.options.length == 1) {
        category.remove(category.selectedIndex);
        if (category.options.length == 0) {
            endQuiz();
        } else {
            selChanged(category, categories, question);
        }
    }
    resetTimer(timePerQuestion);
}

function questionChanged() {
    document.Quiz.answer.value="";
    if (document.Quiz.question.selectedIndex === 0) {
        document.Quiz.btnCheck.disabled = true;
    }
}

```

```

    document.Quiz.answer.disabled = true;
    resetTimer(timePerQuestion);
} else {
    document.Quiz.btnCheck.disabled = false;
    document.Quiz.answer.disabled = false;
    document.Quiz.answer.focus();
    timer = setInterval(decrementTimer,1000);
}
}

function endQuiz() {
    var category = document.Quiz.category;
    var question = document.Quiz.question;
    resetTimer(timePerQuestion);
    alert("Thanks for playing! The quiz will now reload.");
    appendOptionToSelect(category,new Option("Addition","addition"));
    appendOptionToSelect(category,new Option("Subtraction","subtraction"));
    appendOptionToSelect(category,new Option("Multiplication","multiplication"));
    appendOptionToSelect(category,new Option("Division","division"));
    selChanged(category,categories,question);
    resetTimer(timePerQuestion);
}

function msg(text) {
    document.getElementById("msg").innerHTML=text;
}
</script>
</head>
<body>
<form name="Quiz" onsubmit="return false;">
<table>
<tr>
<td>Category:</td>
<td>
<select name="category" id="category">
<option value="addition">Addition</option>
<option value="subtraction">Subtraction</option>
<option value="multiplication">Multiplication</option>
<option value="division">Division</option>
</select>
</td>
</tr>
<tr>
<td>Question:</td>
<td>
<select name="question" id="question">
<option value="0">--Please Choose--</option>
</select>
</td>
</tr>
<tr>
<td>Answer:</td>
<td>
<input type="text" name="answer" id="answer"
size="2" disabled="disabled">
<input type="button" name="btnCheck" id="btnCheck"
value="Check Answer" disabled="disabled">
</td>
</tr>
<tr>

```



```

<td>Timer:</td>
<td><input type="text" name="timeLeft" id="timeLeft" size="2"> seconds left</td>
</tr>
<tr>
<td id="msg" colspan="2">Good luck!</td></tr>
</tr>
</table>
</form>
</body>
</html>

```

Here's how the quiz works:

1. The `Question` select menu is always populated with questions in the indicated category.
2. The "Check Answer" button and the `Answer` text field are only enabled when a question is selected.
3. The timer starts when a question is selected and stops when it runs out or when the "Check Answer" button is clicked.
4. When the "Check Answer" button is clicked, the user is informed whether or not the answer is correct and the question is removed from the question menu.
5. When all questions in a category are gone, the category is removed from the category menu.
6. When all questions in all categories have been completed, the user is thanked for taking the quiz and the entire quiz is restarted.

Spend some time reviewing this code. You shouldn't see anything new, except in the way the code is designed.

You will need to have a good understanding of this code in the Dynamic HTML lesson.

As an optional exercise, you may want to go through the uncommented version and add comments yourself to make sure you understand what each line of code is doing. You can then compare it to our commented version at <DynamicForms/Demos/MathQuiz-commented.html>