Collapsible lists in JavaScript

Long nested lists on web pages can be difficult to understand, especially if the visitor must scroll to see the entire list. The *tree view*, a user interface widget that displays hierarchical data as nested lists, solves this problem by making lists collapsible and expandable; a list can be opened by or closed by clicking on its parent list item. CollapsibleLists is a JavaScript object that turns a normal HTML list into a tree view. In the example below, click on the items with plus and minus icons to expand and collapse their lists; in this example the top level deliberately cannot be collapsed.

- Collapsible lists
  - Actions
    - Creation
    - Toggling
      - Expanding/opening
      - Collapsing/closing
  - Uses

Download CollapsibleLists

Download one of the files below and upload it to your web server.

File	Size	Description
CollapsibleLists.js	4,925 bytes	Full version, with comments
CollapsibleLists.compressed.js	1,730 bytes	Compressed version

Link to the file using a script element in the head of your page:

```
1 <script type="text/javascript" src="CollapsibleLists.js"></script>
```

Using CollapsibleLists

CollapsibleLists works with normal HTML lists, such as the following:

```
<ul class="collapsibleList">
  <li>
    Parent item
    <ul>
      <li>Child item</li>
      <li>Child item</li>
    </ul>
  </li>
  <li>
    Parent item
    <ul>
      <li>Child item</li>
      <li>Child item</li>
    </ul>
  </li>
</ul>
```

The sub-lists need not be contained directly within their parent items. For example, each ul element could be inside a div element that is in turn inside the li element, allowing for more advanced styling.

The apply function turns any list with the class ‘collapsibleList’ into a tree view and collapses its sub-lists:

```
// make the appropriate lists collapsible
CollapsibleLists.apply();
```

This function should generally be called immediately after page load, using code such as runOnLoad. By default all sub-lists are also made collapsible; to prevent this, ‘true’ can be passed as a second parameter, causing sub-lists to be left in their expanded state unless they also have the class ‘collapsibleList’.

If a list is dynamically added to the page, it can be turned into a tree view using the applyTo function. The function should be passed the DOM node containing the list:

```
1 // make the list with the ID 'newList' collapsible
2 CollapsibleLists.applyTo(document.getElementById('newList'));
```

As with the apply function, an additional optional parameter can be set to ‘true’ to prevent sub-lists from being made collapsible.

Styling the lists

CollapsibleLists will apply the class ‘collapsibleListClosed’ to any list item whose sub-lists are closed, and ‘collapsibleListOpen’ to any list item whose sub-lists are open. List items without sub-lists will not have a class applied.

For improved usability, closed lists should indicate that they can be expanded, and open lists should indicate that they can be collapsed. One way of doing this is through the use of different mouse pointers and bullet point images. For example:

```
1  .collapsibleList li{
2    list-style-image:url('button.png');
3    cursor:auto;
4  }
5
6  li.collapsibleListOpen{
7    list-style-image:url('button-open.png');
8    cursor:pointer;
9  }
10
11 li.collapsibleListClosed{
12   list-style-image:url('button-closed.png');
13   cursor:pointer;
14 }
```

The image names in lines 2, 7, and 12 should be changed to reflect the name of your images. Lines 3, 8, and 13 ensure the mouse pointer changes to what CSS refers to as a ‘pointer’ (usually rendered as a hand) when the mouse can be

clicked to expand or collapse a sub-list.