CSE 154

LECTURE 8: EVENTS AND TIMERS
attribute
Setting a timer

<table>
<thead>
<tr>
<th>method</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>setTimeout(function, delayMS);</td>
<td>arranges to call given function after given delay in ms</td>
</tr>
<tr>
<td>setInterval(function, delayMS);</td>
<td>arranges to call function repeatedly every delayMS ms</td>
</tr>
<tr>
<td>clearTimeout(timerID);</td>
<td>stops the given timer</td>
</tr>
<tr>
<td>clearInterval(timerID);</td>
<td></td>
</tr>
</tbody>
</table>

- both `setTimeout` and `setInterval` return an ID representing the timer
- this ID can be passed to `clearTimeout/Interval` later to stop the timer
setTimeout example

```html
<button id="clickme">Click me!</button>
<span id="output"></span>

window.onload = function() {
    document.getElementById("clickme").onclick = delayedMessage;
};

function delayedMessage() {
    document.getElementById("output").innerHTML = "Wait for it...";
    setTimeout(sayBooyah, 5000);
}

function sayBooyah() {
    // called when the timer goes off
    document.getElementById("output").innerHTML = "BOOYAH!";
}
```

Click me!
setInterval example

```javascript
var timer = null;  // stores ID of interval timer

function delayMsg2() {
    if (timer === null) {
        timer = setInterval(rudy, 1000);
    } else {
        clearInterval(timer);
        timer = null;
    }
}

function rudy() {
    // called each time the timer goes off
    document.getElementById("output").innerHTML += " Rudy!";
}
```

Passing parameters to timers

```javascript
function delayedMultiply() {
    // 6 and 7 are passed to multiply when timer goes off
    setTimeout(multiply, 2000, 6, 7);
}

function multiply(a, b) {
    alert(a * b);
}
```

• any parameters after the delay are eventually passed to the timer function
  • doesn't work in IE; must create an intermediate function to pass the parameters
• why not just write this?

```javascript
setTimeout(multiply(6 * 7), 2000);
```
Common timer errors

• many students mistakenly write ( ) when passing the function

```javascript
setTimeout(booyah(), 2000);
setTimeout(booyah, 2000);
setTimeout(multiply(num1 * num2), 2000);
setTimeout(multiply, 2000, num1, num2);
```

• what does it actually do if you have the ( )?
  • it calls the function immediately, rather than waiting the 2000ms!
Checkboxes: <input>

yes/no choices that can be checked and unchecked (inline)

```html
<input type="checkbox" name="lettuce" /> Lettuce
<input type="checkbox" name="tomato" checked="checked" /> Tomato
<input type="checkbox" name="pickles" checked="checked" /> Pickles
```

• none, 1, or many checkboxes can be checked at same time
• when sent to server, any checked boxes will be sent with value on:
  • http://webster.cs.washington.edu/params.php?tomato=on&pickles=on
• use checked="checked" attribute in HTML to initially check the box
Radio buttons: `<input>`

*sets of mutually exclusive choices (inline)*

```html
<input type="radio" name="cc" value="visa" checked="checked" /> Visa
<input type="radio" name="cc" value="mastercard" /> MasterCard
<input type="radio" name="cc" value="amex" /> American Express
```

- grouped by name attribute (only one can be checked at a time)
- must specify a value for each one or else it will be sent as value on
Text labels: `<label>`

```html
<label><input type="radio" name="cc" value="visa" checked="checked" /> Visa</label>
<label><input type="radio" name="cc" value="mastercard" /> MasterCard</label>
<label><input type="radio" name="cc" value="amex" /> American Express</label>
```

- associates nearby text with control, so you can click text to activate control
- can be used with checkboxes or radio buttons
- label element can be targeted by CSS style rules
Drop-down list: `<select>`, `<option>`

*menus of choices that collapse and expand (inline)*

```html
<select name="favoritecharacter">
  <option>Jerry</option>
  <option>George</option>
  <option selected="selected">Kramer</option>
  <option>Elaine</option>
</select>
```

- `option` element represents each choice
- `select` optional attributes: disabled, multiple, size
- Optional selected attribute sets which one is initially chosen
Using `<select>` for lists

```html
<select name="favoritecharacter[]" size="3" multiple="multiple">
  <option>Jerry</option>
  <option>George</option>
  <option>Kramer</option>
  <option>Elaine</option>
  <option selected="selected">Newman</option>
</select>
```

- optional multiple attribute allows selecting multiple items with shift- or ctrl-click
  - must declare parameter's name with [] if you allow multiple selections
- option tags can be set to be initially selected
Option groups: `<optgroup>`

```html
<select name="favoritecharacter">
  <optgroup label="Major Characters">
    <option>Jerry</option>
    <option>George</option>
    <option>Kramer</option>
    <option>Elaine</option>
  </optgroup>
  <optgroup label="Minor Characters">
    <option>Newman</option>
    <option>Susan</option>
  </optgroup>
</select>
```

What should we do if we don't like the bold appearance of the optgroups?
Grouping input: `<fieldset>, <legend>`

*groups of input fields with optional caption (block)*

```html
<fieldset>
  <legend>Credit cards:</legend>
  <input type="radio" name="cc" value="visa" checked="checked" /> Visa
  <input type="radio" name="cc" value="mastercard" /> MasterCard
  <input type="radio" name="cc" value="amex" /> American Express
</fieldset>
```

- Credit cards:
  - Visa ○ MasterCard ○ American Express

• fieldset groups related input fields, adds a border; legend supplies a caption
Styling form controls

- attribute selector: matches only elements that have a particular attribute value
- useful for controls because many share the same element (input)
The innerHTML property

- can change the text inside most elements by setting the `innerHTML` property
Abuse of innerHTML

```javascript
// bad style!
var paragraph = document.getElementById("welcome");
paragraph.innerHTML =
  
  
  "<p>text and <a href="page.html">link</a>";
```

- `innerHTML` can inject arbitrary HTML content into the page
- however, this is prone to bugs and errors and is considered poor style
- we forbid using `innerHTML` to inject HTML tags; inject plain text only
  - (later, we'll see a better way to inject content with HTML tags in it)
The six global DOM objects

Every Javascript program can refer to the following global objects:

<table>
<thead>
<tr>
<th>name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>document</td>
<td>current HTML page and its content</td>
</tr>
<tr>
<td>history</td>
<td>list of pages the user has visited</td>
</tr>
<tr>
<td>location</td>
<td>URL of the current HTML page</td>
</tr>
<tr>
<td>navigator</td>
<td>info about the web browser you are using</td>
</tr>
<tr>
<td>screen</td>
<td>info about the screen area occupied by the browser</td>
</tr>
<tr>
<td>window</td>
<td>the browser window</td>
</tr>
</tbody>
</table>