CSE 154

LECTURE 11: MORE EVENTS

Problems with reading/changing styles

```
<button id="clickme">Click Me</button>
```
HTML

```
window.onload = function() {
  document.getElementById("clickme").onclick = biggerFont;
};
function biggerFont() {
  var button = document.getElementById("clickme");
  var size = parseInt(button.style.fontSize);
  button.style.fontSize = (size + 4) + "pt";
}
```
JS

Click Me

output

- style property lets you set any CSS style for an element
- problem: you cannot read existing styles with it
  (you can read ones you set using the DOM .style, but not ones that are set in the CSS file)

Accessing elements' existing styles

```
window.getComputedStyle(element).propertyName
```
JS

```
function biggerFont() {
  // turn text yellow and make it bigger
  var clickMe = document.getElementById("clickme");
  var size = parseInt(window.getComputedStyle(clickMe).fontSize);
  clickMe.style.fontSize = (size + 4) + "pt";
}
```
JS

Click Me

output

- getComputedStyle method of global window object accesses existing styles

Removing a node from the page

```
function slideClick() {
  var bullet = document.getElementById("removeme");
  bullet.parentNode.removeChild(bullet);
}
```
JS

- odd idiom: *obj*.parentNode.remove(*obj*);

The keyword this

```
this.fieldName                      // access field
this.fieldName = value;             // modify field

this.methodName(parameters);        // call method
```
JS

- all JavaScript code actually runs inside of an object
- by default, code runs in the global `window` object (so `this` === `window`)
  - all global variables and functions you declare become part of `window`
- the `this` keyword refers to the current object

Event handler binding

```
window.onload = function() {
  document.getElementById("textbox").onmouseout = booyah;
  document.getElementById("submit").onclick = booyah;
                                  // bound to submit button here
};

function booyah() { // booyah knows what object it was called on
  this.value = "booyah";
}
```

JS

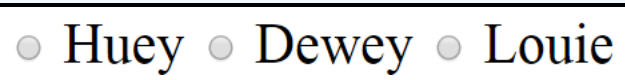
output

- event handlers attached unobtrusively are **bound** to the element
- inside the handler, that element becomes `this`

Fixing redundant code with this

```
<input id="huey"  type="radio" name="ducks" value="Huey"  /> Huey
<input id="dewey" type="radio" name="ducks" value="Dewey" /> Dewey
<input id="louie" type="radio" name="ducks" value="Louie" /> Louie
```
HTML

```
function processDucks() {
  if (document.getElementById("huey").checked) {
    alert("Huey is checked!");
  } else if (document.getElementById("dewey").checked) {
    alert("Dewey is checked!");
  } else {
    alert("Louie is checked!");
  }
  alert(this.value + " is checked!");
}
```
JS



output

- if the same function is assigned to multiple elements, each gets its own bound copy

JavaScript events

abort	blur	change	click	dblclick	error	focus
keydown	keypress	keyup	load	mousedown	mousemove	mouseout
mouseover	mouseup	reset	resize	select	submit	unload

- the `click` event (`onclick`) is just one of many events that can be handled

The event object

```
function name(event) {
  // an event handler function ...
}
```
JS

- Event handlers can accept an optional parameter to represent the event that is occurring. Event objects have the following properties / methods:

property name	description
type	what kind of event, such as "click" or "mousedown"
target	the element on which the event occurred
timeStamp	when the event occurred

Mouse events

click	user presses/releases mouse button on the element
dblclick	user presses/releases mouse button twice on the element
mousedown	user presses down mouse button on the element
mouseup	user releases mouse button on the element

clicking

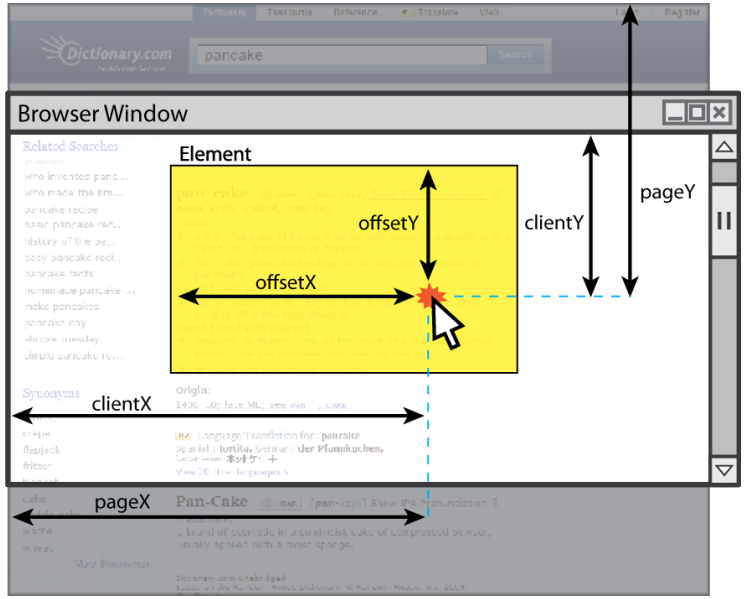
mouseover	mouse cursor enters the element's box
mouseout	mouse cursor exits the element's box
mousemove	mouse cursor moves around within the element's box

movement

Mouse event objects

The `event` passed to a mouse handler has these properties:

property/method	description
clientX clientY	coordinates in *browser window*
screenX screenY	coordinates in *screen*
offsetX offsetY	coordinates in *element* (non-standard)
button	integer representing which button was pressed (0=Left, 1=Middle, 2=Right)



Mouse event example

```html
<pre id="target">Move the mouse over me!</pre>
```

HTML

```
window.onload = function() {
  var target = document.getElementById("target");
  target.onmousemove = target.onmousedown = showCoords;
};

function showCoords(event) {
  document.getElementById("target").innerHTML =
    + "screen : (" + event.screenX + ", " + event.screenY + ")\n"
    + "client : (" + event.clientX + ", " + event.clientY + ")\n"
    + "button : "  + event.button;
}
```

JS

```
screen : (333, 782)
client : (222, 460)
button : 0
```

output

Keyboard/text events

name	description
focus	this element gains keyboard **focus** (attention of user's keyboard)
blur	this element loses keyboard focus
keydown	user presses a key while this element has keyboard focus
keyup	user releases a key while this element has keyboard focus
keypress	user presses and releases a key while this element has keyboard focus
select	this element's text is selected or deselected

Key event objects

property name	description
keyCode	ASCII integer value of key that was pressed (convert to char with String.fromCharCode)
altKey, ctrlKey, shiftKey	true if Alt/Ctrl/Shift key is being held

- issue: if the event you attach your listener to doesn't have the focus, you won't hear the event
  - possible solution: attach key listener to entire page body, `document`, an outer element, etc.

Key event example

```
document.getElementById("textbox").onkeydown = textKeyDown;
...
function textKeyDown(event) {
  var key = String.fromCharCode(event.keyCode);
  if (key == 's' && event.altKey) {
    alert("Save the document!");
    this.value = this.value.split("").join("-");
  }
}
```
JS

- each time you push down any key, even a modifier such as Alt or Ctrl, the `keydown` event fires
- if you hold down the key, the `keydown` event fires repeatedly
- `keypress` event is a bit flakier and inconsistent across browsers

Some useful key codes

keyboard key	event keyCode
Backspace	8
Tab	9
Enter	13
Escape	27
Page Up, Page Down, End, Home	33, 34, 35, 36
Left, Up, Right, Down	37, 38, 39, 40
Insert, Delete	45, 46
Windows/Command	91
F1 - F12	112 - 123

Page/window events

name	description
contextmenu	the user right-clicks to pop up a context menu
error	an error occurs when loading a document or an image
load, unload	the browser loads the page
resize	the browser window is resized
scroll	the user scrolls the viewable part of the page up/down/left/right
unload	the browser exits/leaves the page

- The above can be handled on the `window` object

Multiple listeners to the same event

```
element.addEventListener("event", function);
```
JS

```
var button = document.getElementById("mybutton");
button.addEventListener("click", func1);
                    // button.onclick = func1;
button.addEventListener("click", func2);
                    // button.onclick = func2;
```
JS

- if you assign `onclick` twice, the second one replaces the first
- `addEventListener` allows multiple listeners to be called for the same event
- *(note that you do not include "on" in the event name!)*

Multiple window.onload listeners

```
window.onload = function;
window.addEventListener("load", function);
```
JS

- it is considered bad form to directly assign to `window.onload`
- multiple .js files could be linked to the same page, and if they all need to run code when the page loads, their `window.onload` statements will override each other
- by calling `window.addEventListener` instead, all of them can run their code when the page is loaded

Stopping an event

event method name	description
preventDefault	stops the browser from doing its normal action on an event; for example, stops the browser from following a link when <a> tag is clicked, or stops browser from submitting a form when submit button is clicked
stopPropagation	stops the browser from showing this event to any other objects that may be listening for it

- you can also `return false;` from your event handler to stop an event

Stopping an event, example

```
<form id="exampleform" action="http://foo.com/foo.php">...</form>
```

```
window.onload = function() {
  var form = document.getElementById("exampleform");
  form.onsubmit = checkData;
};

function checkData(event) {
  if (document.getElementById("state").length != 2) {
    alert("Error, invalid city/state.");  // show error message
    event.preventDefault();
    return false;                         // stop form submission
  }
}
```

JS