Web Programming Step by Step

Lecture 17
Events
Reading: 9.1 - 9.3

9.2: Event-Handling

- 9.1: The Prototype JavaScript Library
- **9.2: Event-Handling**

The keyword `this` (8.1.3)

```
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 inside the global `window` object
  - all global variables and functions you declare become part of `window`
- the `this` keyword refers to the current object

Event handler binding

```
function pageLoad() {
  $("ok").onclick = okayClick;   // bound to okButton here
}

function okayClick() {           // okayClick knows what DOM object
  this.innerHTML = "booyah";     // it was called on
}

window.onload = pageLoad;
```
JS

```
OK
```
output

- event handlers attached unobtrusively are **bound** to the element
- inside the handler, that element becomes `this` (rather than the `window`)

Fixing redundant code with `this`

```
<fieldset>
  <label><input type="radio" name="ducks" value="Huey"  /> Huey</label>
  <label><input type="radio" name="ducks" value="Dewey" /> Dewey</label>
  <label><input type="radio" name="ducks" value="Louie" /> Louie</label>
</fieldset>
```
HTML

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

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

More about events

abort	blur	change	click	dblclick	error	focu
keydown	keypress	keyup	load	mousedown	mousemove	mous
mouseover	mouseup	reset	resize	select	submit	unlo

- the `click` event (`onclick`) is just one of many events that can be handled
- **problem**: events are tricky and have incompatibilities across browsers
  - reasons: fuzzy W3C event specs; IE disobeying web standards; etc.
- **solution**: Prototype includes many event-related features and fixes

Attaching event handlers the Prototype way

```
~~element.onevent = function~~;
element.observe("event", "function");
```

JS

```
// call the playNewGame function when the Play button is clicked
$("play").observe("click", playNewGame);
```

JS

- to use Prototype's event features, you must attach the handler using the DOM element object's `observe` method (added by Prototype)
- pass the event of interest and the function to use as the handler
- handlers *must* be attached this way for Prototype's event features to work

- `observe` substitutes for `addEventListener` (not supported by IE)

Attaching multiple event handlers with `$$`

```
// listen to clicks on all buttons with class "control" that
// are directly inside the section with ID "game"
window.onload = function() {
  var gameButtons = $$("#game > button.control");
  for (var i = 0; i < gameButtons.length; i++) {
    gameButtons[i].observe("click", gameButtonClick);
  }
};

function gameButtonClick() { ... }
```

JS

- you can use `$$` and other DOM walking methods to unobtrusively attach event handlers to a group of related elements in your `window.onload` code

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:

method / property name	description
`type`	what kind of event, such as `"click"` or `"mousedown"`
`element()` *	the element on which the event occurred
`stop()` **	cancels an event
`stopObserving()`	removes an event handler

* replaces non-standard `srcElement` and `which` properties
** replaces non-standard `return false;`, `stopPropagation`, etc.

Mouse events (9.2.2)

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

clicking

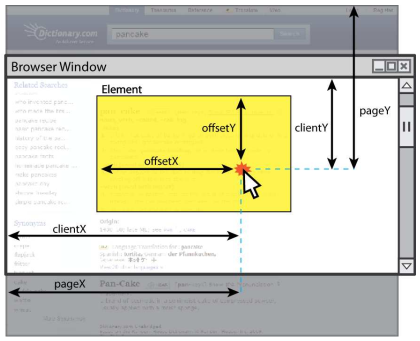
`mouseover`	mouse cursor enters this element's box
`mouseout`	mouse cursor exits this element's box
`mousemove`	mouse cursor moves around within this element's box

movement

Mouse event objects

The `event` parameter passed to a mouse event handler has the following properties:

property/method	description
`clientX`, `clientY`	coordinates in *browser window*
`screenX`, `screenY`	coordinates in *screen*
`offsetX`, `offsetY`	coordinates in *element*
`pointerX()`, `pointerY()` *	coordinates in *entire web page*
`isLeftClick()` **	`true` if left button was pressed



* replaces non-standard properties `pageX` and `pageY`
** replaces non-standard properties `button` and `which`

Mouse event example

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

```
window.onload = function() {
  $("target").observe("mousemove", showCoords);
};

function showCoords(event) {
  this.innerHTML =
      "pointer: (" + event.pointerX() + ", " + event.pointerY() + ")\n"
    + "screen : (" + event.screenX + ", " + event.screenY + ")\n"
    + "client : (" + event.clientX + ", " + event.clientY + ")";
}
```
JS

```
Move the mouse over me!
```
output