Prototype and Events

CSE 190 M (Web Programming), Spring 2008
University of Washington

Reading: Chapter 4

Lecture Outline

- the Prototype JavaScript library
  - useful additional objects, methods, and compatibility fixes
- global DOM objects
  - DOM objects for accessing the document, browser window, etc.
- more events
  - richer event handling, keyboard/mouse events, etc.

Prototype JavaScript library

A set of useful additional objects, methods, and cross-browser compatibility fixes

Problems with JavaScript

JavaScript is a powerful language, but it has many flaws:

- The DOM can be clunky to use
- Several potentially useful objects and methods are missing
- The same code doesn't always work the same way in every browser
  - code that works great in Firefox, Safari, ... will fail in IE and vice versa
- Many web developers work around these problems with hacks:

```
// check if browser is IE (bad style!)
if (navigator.appName === "Microsoft Internet Explorer") { ...
```

(Some) Things that break in IE

- CSS:
  - the CSS box model, in many ways
  - fixed positioning
- JavaScript:
  - getting the `.value` of many DOM controls (unless set explicitly)

    ```
    <option value="Bike">Bike</option>
    ```

  - `String.split` (some incompatibilities)
  - timers with `setTimeout` (some incompatibilities)
  - accessing String characters using `str[i]` notation
  - lots of DOM stuff
  - lots of event-handling stuff
  - Ajax programming (seen later)
  - ...

Prototype

```
<script src="http://www.cs.washington.edu/education/courses/cse190m/08sp/prototype.js"
 type="text/javascript"></script>

<!-- or, -->
<script src="http://prototypejs.org/assets/2008/1/25/prototype-1.6.0.2.js"
 type="text/javascript"></script>
```

- Prototype JavaScript library adds many useful features to JavaScript:
  - many useful extensions to the DOM
  - added methods to String, Array, Date, Number, Object
  - improves event-driven programming
  - many cross-browser compatibility fixes
  - makes Ajax programming easier (seen later)

Prototype methods

- methods added to Arrays
  - `clear`, `clone`, `compact`, `each`, `first`, `flatten`, `from`, `indexOf`, `inspect`, `last`, `reduce`, `reverse`, `size`, `toArray`, `toJSON`, `uniq`, `without`
- methods added to Numbers
  - `abs`, `ceil`, `floor`, `round`, `succ`, `times`, `toColorPart`, `toJSON`, `toPaddedString`
- methods added to all Objects
  - `clone`, `extend`, **`inspect`**, `isArray`, `isElement`, `isFunction`, `isHash`, `isNumber`, `isString`, `isUndefined`, `keys`, `toHTML`, `toJSON`, `toQueryString`, `values`
- methods added to Strings
  - `blank`, `camelize`, `capitalize`, `dasherize`, `empty`, **`endsWith`**, **`escapeHTML`**, `evalJSON`, `evalScripts`, `extractScripts`, `gsub`, `include`, `inspect`, `interpolate`, `isJSON`, `parseQuery`, `scan`, **`startsWith`**, **`strip`**, `stripScripts`, **`stripTags`**, `sub`, `succ`, `times`, `toArray`, `toJSON`, `toQueryParams`, `truncate`, `underscore`, **`unescapeHTML`**, `unfilterJSON`

Some Prototype features

-
  ```
  $("id")
  ```
  returns the DOM object representing the element with the given `id`

-
  ```
  $$("class")
  ```
  returns an array of DOM objects representing elements that match the given CSS selector
- Prototype **extends** each DOM object you fetch with the above functions (adds methods to it)

Prototype DOM element methods

Prototype adds the following methods to every DOM element object:

- absolutize, **addClassName**, addMethods, adjacent, ancestors, **childElements**, **classNames**, cleanWhitespace, clonePosition, cumulativeOffset, cumulativeScrollOffset, descendantOf, **descendants**, **down**, empty, extend, fire, firstDescendant, getDimensions, **getElementsByClassName**, **getElementsBySelector**, getHeight, getOffsetParent, **getStyle**, getWidth, **hasClassName**, **hide**, identify, immediateDescendants, insert, inspect, makeClipping, makePositioned, match, **next**, nextSiblings, observe, positionedOffset, **previous**, previousSiblings, readAttribute, recursivelyCollect, relativize, **remove**, **removeClassName**, replace, **scrollTo**, select, setOpacity, setStyle, **show**, **siblings**, stopObserving, toggle, toggleClassName, undoClipping, undoPositioned, **up**, update, viewportOffset, visible, wrap, writeAttribute

Prototype in action

```
function makeFontBigger() {
  $("text").style.fontSize = parseInt(
    $("text").getStyle("font-size")) + 2 + "pt";
}
```
JS

- `$` function makes accessing elements easy
- `getStyle` function added to DOM object allows accessing existing styles
- works in all browsers!

Global DOM objects

Objects provided by the browser that let you learn about the current document, browser window, URL, ...

The six global objects

Every Javascript program can refer to the following global objects:

- `document` : current HTML page object model
- `window` : the browser window
- `location` : URL of the current HTML page
- `navigator` : info about the web browser you're using
- `screen` : info about the screen area occupied by the browser
- `history` : list of pages the user has visited

The `document` object

- represents the URL of the current web page
- properties:
  - `anchors`, body, `cookie`, `domain`, `forms`, `images`, `links`, `referrer`, `title`, `URL`
- methods (* means provided by Prototype):
  - `getElementById` (a.k.a. `$` *)
  - `getElementsByName`
  - `getElementsByTagName`
  - `getElementsByClassName` * (a.k.a. `$$` *)
  - `close`, `open`, `write`, `writeln`
- complete list

The `window` object

- represents the entire browser window; the top-level object in DOM hierarchy
- technically, all global code and variables become part of the `window` object
- methods:
    - `alert`, `confirm`, `prompt` (popup boxes)
    - `setInterval`, `setTimeout` `clearInterval`, `clearTimeout` (timers)
    - `open`, `close` (popping up new browser windows)
    - `blur`, `focus`, `moveBy`, **`moveTo`**, `print`, `resizeBy`, `resizeTo`, **`scrollBy`**, **`scrollTo`**,
- properties:
    - `document`, `history`, `location`, `name`

Popup windows with `window.open`

```
window.open("http://foo.com/bar.html", "My Foo Window",
            "width=900,height=600,scrollbars=1");
```
JS

- `window.open` pops up a new browser window
- THIS method is the cause of all the terrible popups on the web!
- some popup blocker software will prevent this method from running

The `location` object

- represents the URL of the current web page
- properties:
    - `host`, `hostname`, `href`, `pathname`, `port`, `protocol`, `search`
- methods:
    - `assign`, `reload`, `replace`
- complete list

The `navigator` object

- information about the web browser application
- properties:
  - `appName`, `appVersion`, `browserLanguage`, `cookieEnabled`, `platform`, `userAgent`
  - complete list
- Some web programmers examine the `navigator` object to see what browser is being used, and write browser-specific scripts and hacks:

```
if (navigator.appName === "Microsoft Internet Explorer") { ...
```
JS

  - (careful programming and using Prototype reduce the need for this)

The `screen` object

- information about the client's display screen
- properties:
  - `availHeight`, `availWidth`, `colorDepth`, `height`, `pixelDepth`, `width`
  - complete list

The `history` object

- list of sites the browser has visited in this window
- properties:
  - `length`
- methods:
  - `back`, `forward`, `go`
- complete list
- sometimes the browser won't let script code view `history` properties, for security

Events

handling more user events such as mouse and keyboard actions

Mouse events

XHTML elements have the following events:

- clicking
  - `onclick` : user presses/releases mouse button on this element
  - `ondblclick` : user presses/releases mouse button *twice* on this element
  - `onmousedown` : user presses down mouse button on this element
  - `onmouseup` : user releases mouse button on this element
- movement
  - `onmouseover` : mouse cursor enters this element's box
  - `onmouseout` : mouse cursor exits this element's box
  - `onmousemove` : mouse cursor moves around within this element's box

```
<div onmousemove="myFunction();">...</div>
```
JS

Mouse event example

```
<div id="target" onmouseover="colorIt();">I'm OVER you!</div>
```
HTML

```
function colorIt() {
  $("target").style.backgroundColor = "red";
}
```
JS

I'm OVER you!

Handling multiple mouse events

```html
<div id="dare" onmousedown="colorIt();" onmouseup="uncolorIt();">
  Click me ... I dare you!
</div>
```

HTML

```javascript
function colorIt() {
  $("dare").style.backgroundColor = "red";
}
function uncolorIt() {
  $("dare").style.backgroundColor = "white";
}
```

JS

Click me ... I dare you!

Examining the mouse event object

```javascript
function colorIt(event) {
    $("dare").style.backgroundColor = "red";
    $("dare").innerHTML = "You clicked (" + event.screenX +
        ", " + event.screenY + ")");
}
```

JS

Click me ... I dare you!

- a handler can accept an optional parameter representing the event
- event object holds several properties about the event that occurred

Event object properties

- `type` : what kind of event, such as `"click"` or `"mousedown"`
  - same as event property name without `on` prefix
  - useful if you use the same handler to handle multiple events
- `clientX`, `clientY` : coordinates from top/left of *page*
- `screenX`, `screenY` : coordinates from top/left of *screen*
- complete list

Browser incompatibilities with events

- fuzzy W3C specs and browser wars have led to event differences between browsers
- IE6 sucks and doesn't support accepting `event` as a parameter
    - instead uses non-standard property `window.event`
    - some properties inside this object are non-standard
- even mighty Firefox is missing some standard properties (gasp!)
- a cross-browser script can handle both

Poorly supported event properties

- `offsetX`, `offsetY` : coordinates from top/left of *element*
    - Firefox uses non-standard `layerX`, `layerY` properties instead
- `button` : which mouse button was pressed/released, if any
    - IE returns 1/2/4 for left/right/middle button; Firefox returns 0/1/2 (standard)
    - Firefox also uses non-standard `which` property instead
- `srcElement` : element that fired the event
    - Firefox uses non-standard `target` property instead
- more incompatibilities

Click me: Which properties are supported?

Prototype and events

```
function name(event) {
    Event.extend(event);
    ...
}
```
JS

- calling Prototype's `Event.extend` repairs many event incompatibilities:
- *methods* added to Events
    - `element` (replaces `which` / `srcElement` properties)
    - `isLeftClick` (replaces `button` / `which` properties)
    - `pointerX`, `pointerY` (replace `clientX`, `clientY` properties)
    - `findElement`, `stop`, `stopObserving`, `unloadCache`

Keyboard events

DOM objects for HTML elements have the following properties:

- `onkeydown` : user presses a key while this element has keyboard focus
- `onkeyup` : user releases a key while this element has keyboard focus
- `onkeypress` : user presses and releases a key while this element has keyboard focus
- `onfocus` : this element gains keyboard focus
- `onblur` : this element loses keyboard focus
- **focus**: the attention of the user's keyboard (given to one element at a time)

Key event object properties

- `keyCode`: ASCII numeric value of key that was pressed
  - to convert to a letter: `String.fromCharCode(event.keyCode)`
  - list of key values
- `altKey` : `true` if Alt key is being held
- `ctrlKey` : `true` if Ctrl key is being held
- `shiftKey` : `true` if Shift key is being held

Which key event properties does your browser support?

Prototype and keyboard events

```
function name(event) {
    Event.extend(event);
    ...
}
```
JS

- calling Prototype's `Event.extend` adds these useful key code constants:
  - `Event.KEY_BACKSPACE`, `Event.KEY_DELETE`, `Event.KEY_DOWN`, `Event.KEY_END`, `Event.KEY_ESC`, `Event.KEY_HOME`, `Event.KEY_LEFT`, `Event.KEY_PAGEDOWN`, `Event.KEY_PAGEUP`, `Event.KEY_RETURN`, `Event.KEY_RIGHT`, `Event.KEY_TAB`, `Event.KEY_UP`,
- (otherwise, you'd need to know what integer key code mapped to each of the above keys! which would be a pain...)

Detecting Enter key on a text field

```html
<input type="text" onkeypress="keyPress();" />
```
HTML

```js
function keyPress(event) {
  Event.extend(event);
  if (event.keyCode == Event.KEY_RETURN) {
    // the user pressed Enter
    alert("You pressed the Enter key!");
  }
}
```
JS

Text box events

these are supported by `<input type="text">`, `<textarea>`

- `onselect` : text within a text box is selected
- `onchange` : content of a text box changes

Practice problem: Draggable map

One of the coolest features of Google Maps is the ability to drag the map to move it around. Write a program with a draggable map of Middle Earth using Javascript mouse event handlers. (See the background CSS properties from the end of the CSS slides.)