

CSE 154

LECTURE 19: EVENTS AND TIMERS

The six global DOM objects

Every Javascript program can refer to the following global objects:

name	description
document	current HTML page and its content
history	list of pages the user has visited
location	URL of the current HTML page
navigator	info about the web browser you are using
screen	info about the screen area occupied by the browser
window	the browser window

The window object

the entire browser window; the top-level object in DOM hierarchy

- technically, all global code and variables become part of the `window` object
- properties:
 - `document`, `history`, `location`, `name`
- 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`

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 document object

the current web page and the elements inside it

- properties:

- anchors, body, cookie, domain, forms, images, links, referrer, title, URL

- methods:

- getElementById
- getElementsByName, getElementsByTagName
- querySelector, querySelectorAll
- close, open, write, writeln

The location object

the URL of the current web page

- properties:

- host, hostname, href, pathname, port, protocol, search

- methods:

- assign, reload, replace

The navigator object

information about the web browser application

- properties:
 - `appName`, `appVersion`, `browserLanguage`, `cookieEnabled`, `platform`, `userAgent`
- 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
```

- (this is poor style; you usually do not need to do this)

The screen object

information about the client's display screen

- properties:
 - `availHeight`, `availWidth`, `colorDepth`, `height`, `pixelDepth`, `width`

The history object

the list of sites the browser has visited in this window

- properties:
 - `length`
- methods:
 - `back`, `forward`, `go`
- sometimes the browser won't let scripts view `history` properties, for security

Setting a timer

method	description
<code>setTimeout(<i>function</i>, <i>delayMS</i>);</code>	arranges to call given function after given delay in ms
<code>setInterval(<i>function</i>, <i>delayMS</i>);</code>	arranges to call function repeatedly every <i>delayMS</i> ms
<code>clearTimeout(<i>timerID</i>);</code> <code>clearInterval(<i>timerID</i>);</code>	stops the given timer

- 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

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

HTML

```
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!";  
}
```

JS

Click me!

output

setInterval example

```
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!";
}
```

JS

Click me!

output

Passing parameters to timers

```
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);  
}
```

JS

Click me!

output

- 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?

```
setTimeout(multiply(6 * 7), 2000);
```

JS

Common timer errors

- many students mistakenly write `()` when passing the function

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

JS

- what does it actually do if you have the `()` ?
 - it calls the function immediately, rather than waiting the 2000ms!