CS472 WAP

JavaScript Timers

JavaScript Programming Environment

Timer events

method	description
setTimeout (*function*, *delayMS*);	arranges to call given function after given delay in ms
setInterval (*function*, *delayMS*);	arranges to call function repeatedly every *delayMS* ms
clearTimeout(*timerID*); clearInterval(*timerID*);	stops the given timer so it will not call its function

- 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 onclick="delayMsg();">Click me!</button>
<span id="output"></span>

function delayMsg() {
 setTimeout(booyah, 5000);
 document.getElementById("output").innerHTML = "Wait for 
 it...";
}
function booyah() {
 // called when the timer goes off
 document.getElementById("output").innerHTML = "BOOYAH!";
}
```

setInterval example

```
timer = null; // stores ID of interval timer
function delayMsg2() {
 if (timer === null) {
   timer = setInterval(rudy, 1000);
 } else {
   clearInterval(timer); // cancel the timer
   timer = null;
 }
}

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

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);
}
```

- any parameters after the delay are eventually passed to the timer function
- why not just write this? setTimeout(**multiply(6, 7)**, 2000);

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);
```

- what does it actually do if you have the ()? -- e.g., first and 3rd examples above
  - a) Does nothing
  - b) It is an error to call a function as a parameter
  - c) It is an error if the function returns a function
  - d) Calls the function after 2 seconds
  - e) it calls the function immediately
  - f) Calls the function immediately and attempts to call the return value of the function after 2 seconds

IMPORTANT!!!

CONNECTING THE PARTS OF KNOWLEDGE WITH THE WHOLENESS OF KNOWLEDGE

Actions in Accord with the Laws of Nature

1. Client-side JavaScript is included in HTML pages and executes on the browser when it loads.
2. JavaScript reacts to browser events and manipulates the web page using the HTML DOM API

---

3. **Transcendental consciousness** is the source of thought and the home of all the laws of nature.
4. **Impulses within the transcendental field** spontaneously are in accord with all the laws of nature.
5. **Wholeness moving within itself:** In unity consciousness, one enjoys all perceptions and knowledge in terms of the source of all the laws of nature, pure bliss consciousness. This is parallel to the good feeling and confidence that we experience when we have a deep appreciation and understanding of the environment in which JavaScript runs.

MAHARISHI INTERNATIONAL UNIVERSITY

HIGHER EDUCATION FOR HIGHER CONSCIOUSNESS