



Scripting and Web Applications EE1081

Lecture 5 JavaScript

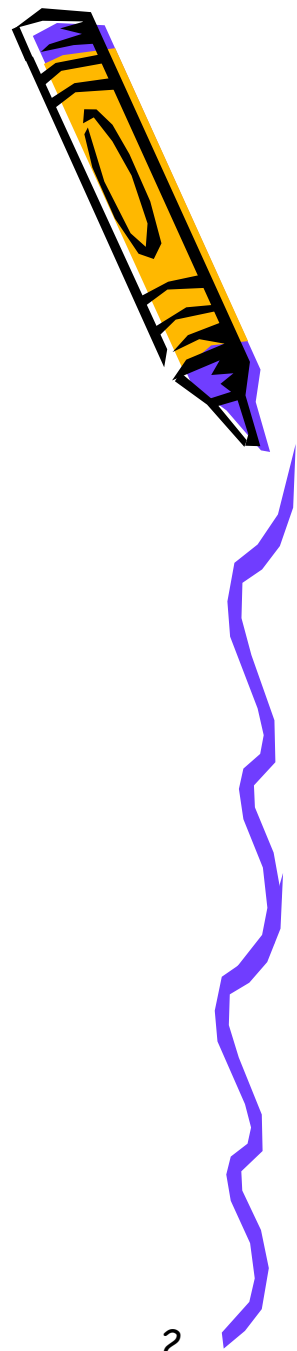
By: A. Mousavi

SERG, School of Engineering Design, Brunel University, UK



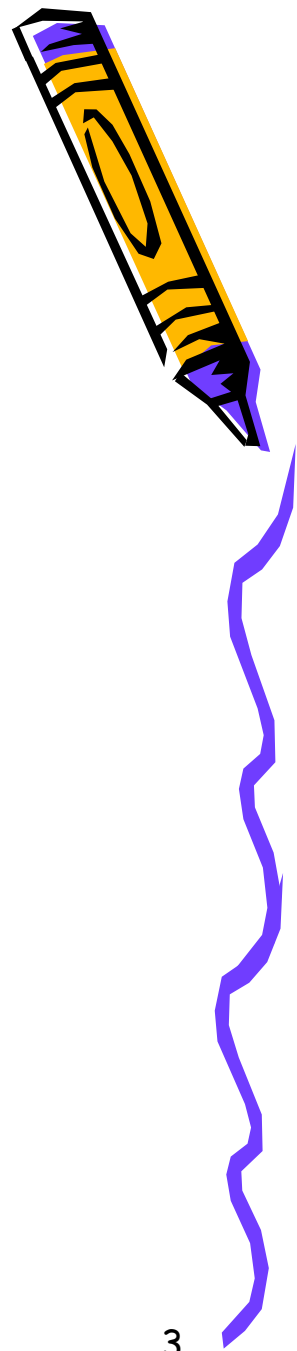
today

- Event Handling in JavaScript
- Client-Side JavaScript
- Examples



Events

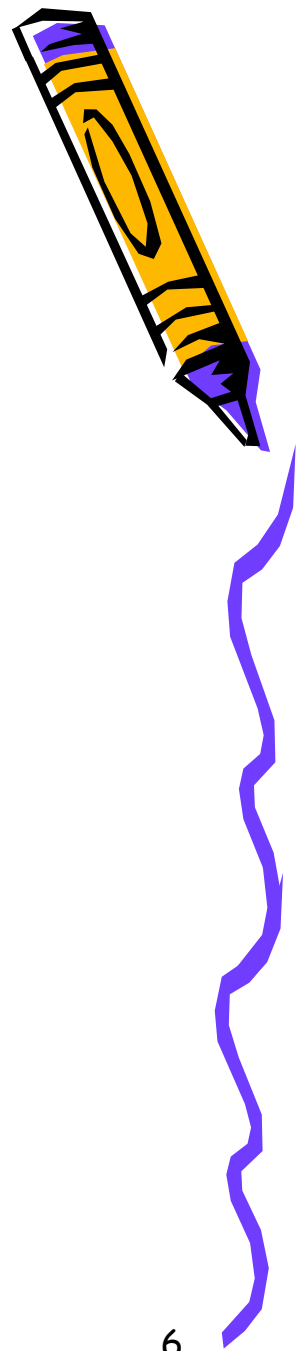
- **Events** are user-driven implementation of functions or operations in an application. For example:
 - Click on a button or mouse
 - Mouse going over an area of the document
 - Pressing on a button on the keyboard
 - ...
- *Events* trigger a predefined operation or function



Events cont.

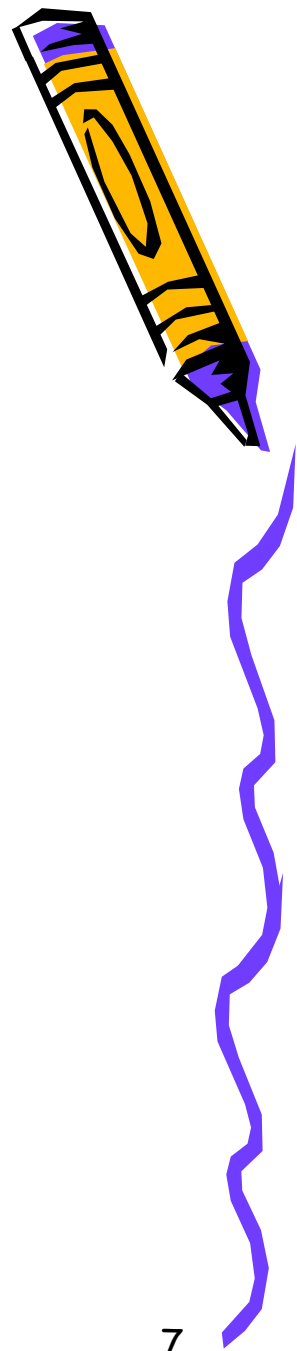
- Browsers cater for the events that are handled by JavaScript

```
*)
  - Names can be assigned to a variable  
*var imgName = "myPhoto";*  
*document.images[imgName];*

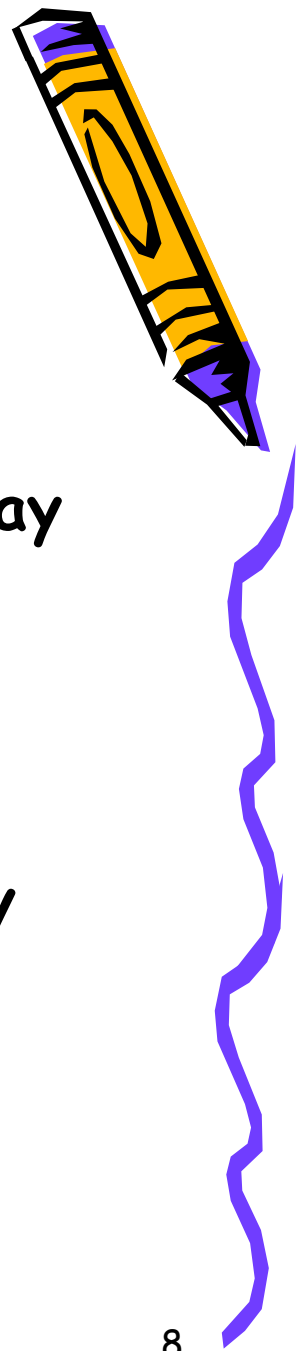


# Another Exercise

A rollover image with a link



# Three Methods to access Images and Image Objects



## 1. *images[ ]* array using a string index

```
document.images["ImageName"].src = "ImageName.png";

```

## 2. Using the numeric index in the *images[ ]* array

```
document.images[0].src= "ImageName1";
document.images[1].src = "ImageName2";

```

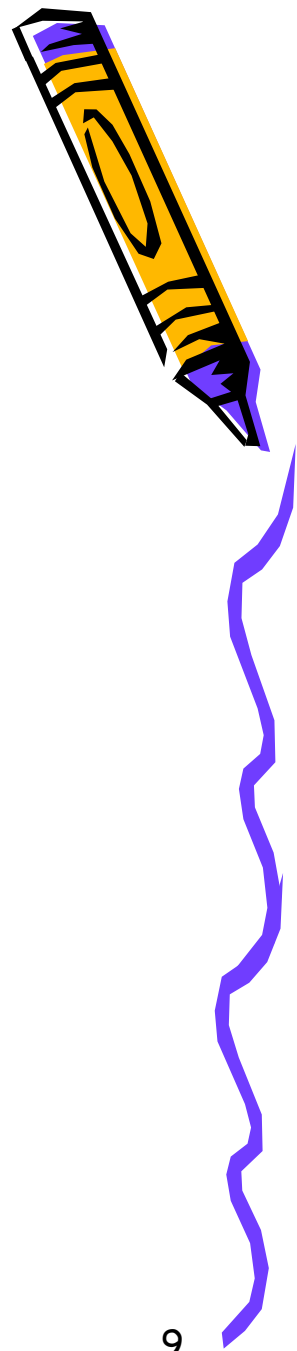
## 3. Using the *<img>* name on the *images[ ]* array object

```
document.images.ImageName.src = "ImageName.png";
```



# Timer Event

Timer events are important features in JavaScript. You can design applications such as: animation, displayable clocks, rotating advertisements, ...



# *setTimeout()* function

- *setTimeout()* is a method of the browser window object
- Two arguments are allowed:
  - A string semicolon separated statements
  - The delay in milliseconds

Example:

*// run a function called imageRotate with 0.1 second delay  
timer = setTimeout( "imageRotate( );" , 100);*

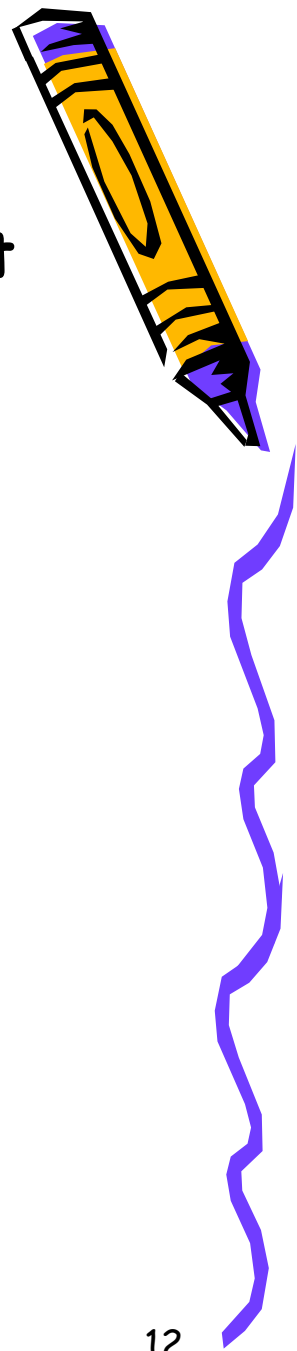


# Examples



**E5-3 (start and stop a timer) and E5-4 a small animation.**

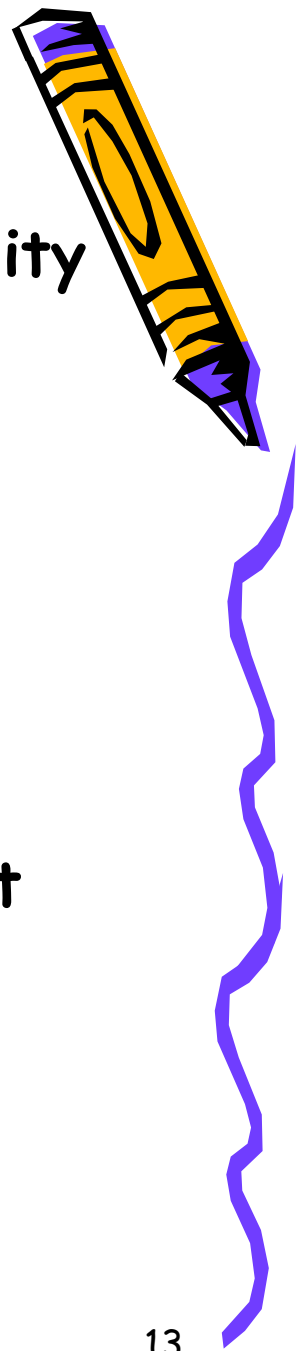
# Managing your JavaScript Applications



1. **Make sure that your web pages work without JavaScript (graceful degradation)**
2. **Separating structure from behaviour**
3. **Making sure that older versions of browsers handle your pages gracefully (backward compatibility)**

Also see DOM Scripting by J. Keith

# Graceful Degradation



- Modern Browser applications have the capability to block JavaScript (i.e. pop-up pages)

*window.open(url, name, features)*

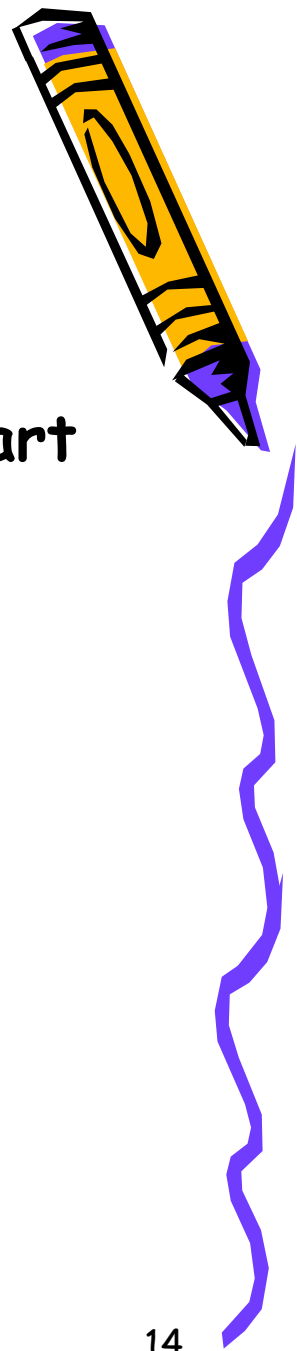
Example

```
function popUp(winURL) {
 window.open(winURL, "popup", "width = 400, height=200");
}
```

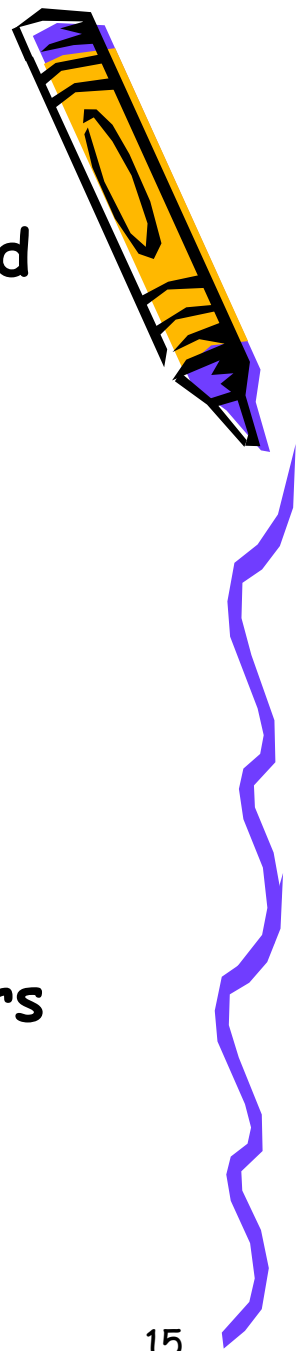
- You have to make sure that if the JavaScript is blocked your web pages and web site could be navigable - **Graceful Degradation**

# Separating structure from behaviour

- Nowhere more applicable than CSS
- Keep styles in an external file rather than part of the document
- This separation also allows for graceful degradation



# Backward compatibility



- Quiz the browser if it support JavaScript and if so to what level

```
// use a condition statement
window.onload = function () {
 if (!document.getElementById
 return false;
 }
```

- Browser sniffing - trying to read the browsers properties. Not very reliable.