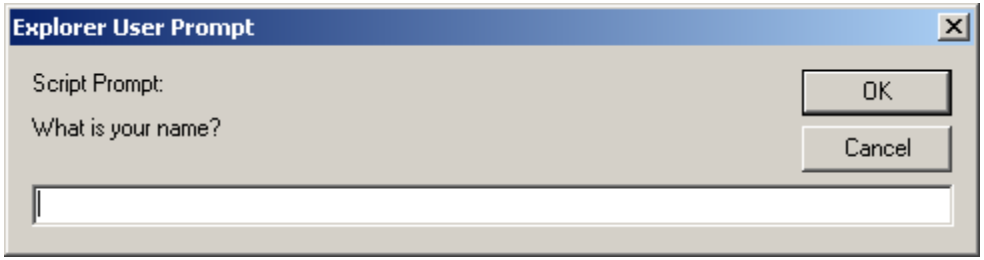
COMP 4021
Internet Computing

JavaScript 2

David Rossiter

Input

- You have already experienced prompt() as a method to get input from the user - also confirm()
- Both are very inflexible methods of input
- To have great flexibility for user input we need an understanding of events and event handling in JavaScript



Events

- Examples of events
  - The user presses a key on the keyboard
  - The user moves the mouse
  - The user clicks on the mouse button
- Whenever an event occurs you can tell the browser to run some JavaScript code to do something appropriate

Events - onLoad

- Here's an example:

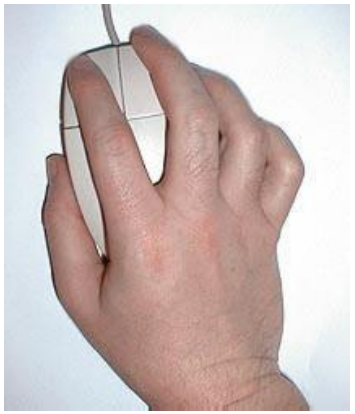
```
<body onload="new_game()">
. . .
</body>
```

- The *body* object is loaded when the web page is loaded: therefore the result of the above code is that when the web page is loaded the function new_game() is executed

Mouse Events

- The most commonly used mouse events:
  - onclick – when the user clicks on an object

    onclick = onmousedown followed by onmouseup
  - onmousedown – when the user presses down the button on the mouse (but hasn't let go yet)
  - onmouseup – when the user lets go of the mouse button



Mouse Events – onclick

```
<html>
<head>
<script language= "JavaScript">
<!--
function greet_the_user() {
   var user_name;
   user_name=prompt("What is your name?", "" );
   alert("Welcome to my page " + user_name + "!");
}
//-->
</script>
</head>
<body  onclick="greet_the_user()" >

<h1>Please click on this page!!</h1>

</body>
</html>
```

onclick event handler: When the mouse button is clicked execute the function greet_the_user()

Mouse Events - onmousedown

- We could change
  ```
  <body onclick="greet_the_user()">
  ```
  into
  ```
  <body onmousedown="greet_the_user()">
  ```

- Then the code will be executed as soon as the mouse button is pressed down
  (no need to also release the button)

Mouse Events - onmousedown

```
<html>
<head>
<script language= "JavaScript">
<!--
function greet_the_user() {
   var user_name;
   user_name=prompt("What is your name?", "" );
   alert("Welcome to my web page " + user_name +
   "!");
}
//-->
</script>
</head>
<body  onmousedown="greet_the_user()"  >

<h1>Please click on this page!!</h1>

</body>
</html>
```

onmousedown event: when the mouse button is pressed down execute greet_the_user()

Mouse Events

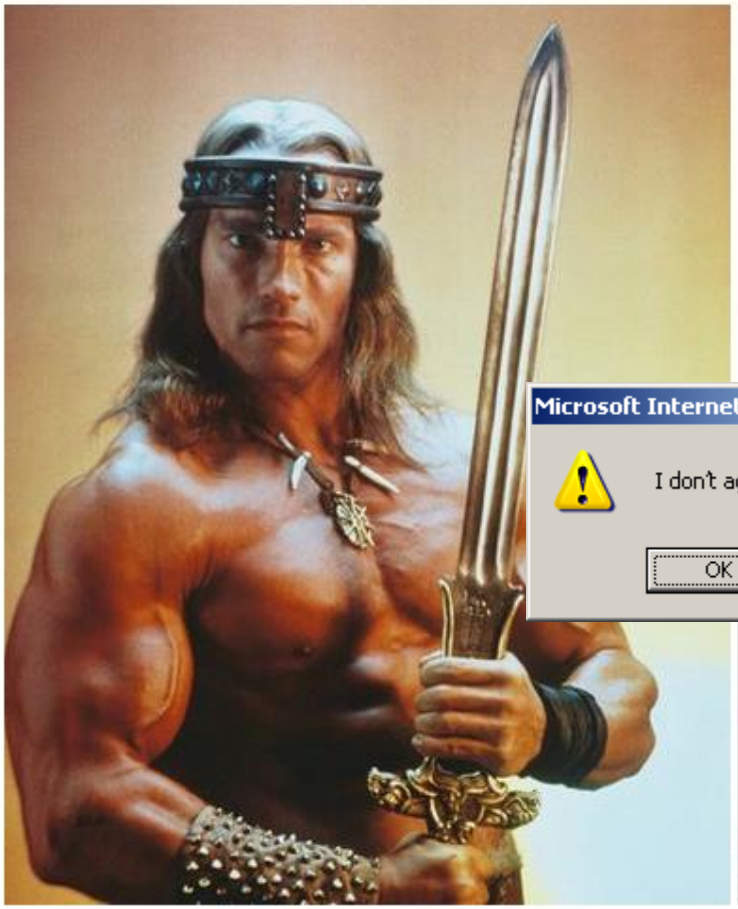
- Remember we can handle events when they happen to *any* object
- For example, on the following page the JavaScript function will be executed when the user clicks on the image

Mouse Events - onclick

```
<html>
<head>
<script language="JavaScript">
<!--
function good_choice() {
   alert("That's a good choice!");
}
function bad_choice() {
   alert("I don't agree!");
}
//-->
</script>
</head>
<body>
<h1>Click on the best action actor…</h1>

<img src="arnold.jpg" onclick="bad_choice();">
<img src="jackie_chan.jpg" onclick="bad_choice();">
<img src="jet_li.jpg" onclick="good_choice();">
</body>
</html>
```

The user sees a different message depending on which image he/she clicks on

Click on the best action actor...



Done

Local intranet

More Mouse Events

- ondblclick – when the user double clicks on an object
  (it is more sensible to use onclick)
- onmouseover – when the user moves the mouse over an object
  (almost the same: onmouseenter)
- onmouseout – when the user moves the mouse away from an object *(almost the same: onmouseleave)*

- oncontextmenu – when the user clicks on the right mouse button

onmouseover/ onmouseout

```
<html>
<head>
<script language= "JavaScript">
function change_colour( new_colour ) {
   document.getElementById("pretty_layer").style.background=new_colour;
}
</script>
</head>

<body>
<div id="pretty_layer"
   style=" position:absolute;
   background:yellow;
   left:300; top:100; width:100;
   font-size:22pt"
   onmouseover="change_colour('red');"
   onmouseout="change_colour('yellow');">
   Move mouse in and out this layer...
</div>
<div id="ugly_layer" … …> … … </div>
</body>
</html>
```

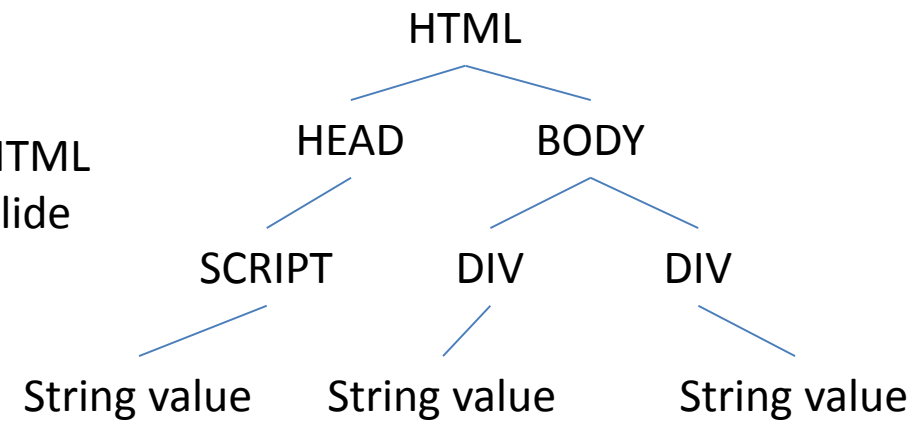
- document.getElementById finds the "pretty_layer" element
- Then, change the background colour

- Mouse over/out triggers the JavaScript function
- Color changes accordingly
- But there are 2 div's (2 display areas) …

Document Object Model (DOM)

- When a web page is loaded by the browser it stores the page in the DOM structure, which is a tree representing the nested structure of the elements (html tags)
- JavaScript code can access and change any of the things stored in memory
- When DOM is updated, the user will immediately see the the change in the web page

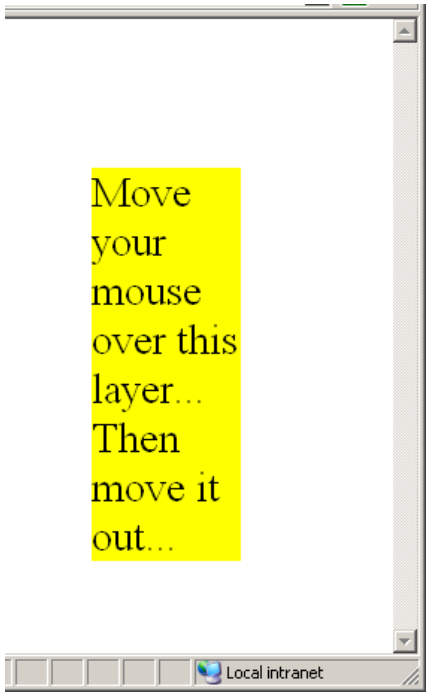
DOM of the HTML page on last slide



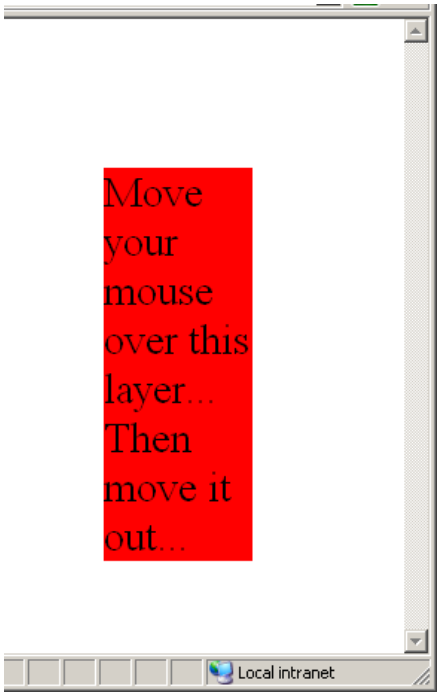
Finding - getElementById

- `getElementById("fun_thing")` means ‘find the object in the DOM structure which is called *fun_thing*’
- The object which you are searching for could be anything – i.e. a paragraph, a layer, a title, a list item, an image …
- After you find something you can change it/ delete it/ copy it

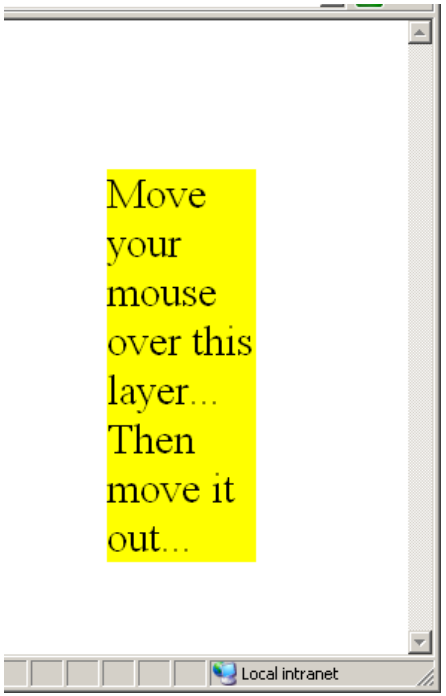
onmouseover/ onmouseout



- Move mouse over the div



- Move mouse away from the div



Other Methods of Access

- There are some special ‘shortcut’ methods for accessing some particular types of data:
- `document.images[]` – all images in the document
- `document.applets[]` – all applets in the document
- `document.links[]` – all links in the document

  … *others* …
- i.e. `document.images[0]` refers to the first image in the web page

Timers

- Timers are very useful for dynamic web page behaviour, i.e.

```
var the_timer;
the_timer = setTimeout("do_something()", 1000);
```

- This tells the browser to run the function do_something() 1 second later
- The value ‘1000’ is in milliseconds, so 1000=1 sec

Example of a Timer

```
<html>
<head>
<script language="JavaScript">
function set_things_up() {
  setTimeout("show_failure()", 3000 );
}
function show_failure() {
  alert("Too slow!!!");
}
</script>
</head>
<body onload="set_things_up()">

<h1>Warning!! You have 3 seconds to
  go to another page …</h1>
</body>
</html>
```

Another Example

- Here's another example
- This one is a kind of 'alarm clock'

Example of a Timer

```
<html>
<head>
<script language="JavaScript">
var wait_duration;

function set_things_up() {
  wait_duration=prompt("How long would you like to sleep?", "");
  setTimeout("show_wake_up_message()", wait_duration );
}
function show_wake_up_message() {
  alert("WAKE UP! WAKE UP! WAKE UP!!");
}
</script>
</head>

<body onload="set_things_up()">
<h1>Alarm clock example</h1>
</body>
</html>
```

For this example the user will enter a number in milliseconds i.e. 1000 means 1 second

Using a Timer to Move an Image

- Many games use moving images
- To make a moving image using HTML and JavaScript, the image must first be put in a layer:

```
<div id="image_layer" style="position:absolute">
   <img src="dave.png">
</div>
```

As discussed before, this part tells the browser that this layer can be positioned anywhere on the web page by the client side code

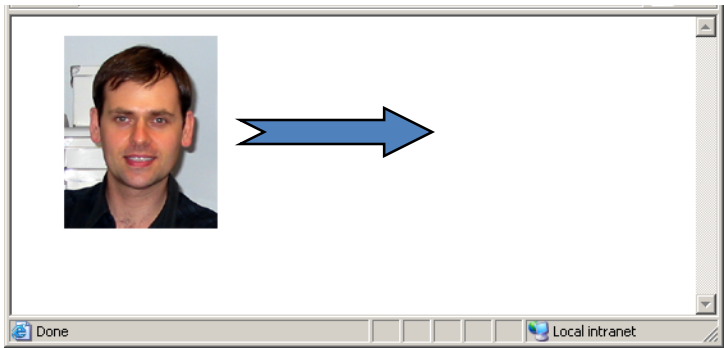
Using a Timer to Move an Image

- Then in order to move the image you have to move the layer
- Because the image is inside the layer, you will see the image move
- In the following example a timer is used to move the layer slowly from left to right across the screen

Moving an Image

```
<html>
<head>
<script language="JavaScript">
<!--
var the_timer;  var x_position=0;  var the_layer;

function set_timer() {

the_layer=document.getElementById("image_layer"); // Find the layer
x_position=x_position+1;                  // Move the position to the right
the_layer.style.left=x_position;          // Now put the div at that x position

// Now start the timer again so this function will run again.
// It will run every 0.1 second. (100 milliseconds = 0.1 second).
the_timer=setTimeout( "set_timer()",  100   );
}
//-->
</script>
</head>

<body onload=" set_timer() ">
<div id="image_layer" style="position:absolute;
        <img src="dave.png">
</div>
</body>
</html>
```

The same function will be run again in 0.1 second



Randomly Moving an Image

```
<html>
<head>
<script language="JavaScript">
<!--
var the_timer;  var x_position;  var the_layer;

function set_timer() {
x_position=Math.floor( Math.random() * 400 ); // Random position from 0 to 400

the_layer=document.getElementById("image_layer"); // Find the layer

the_layer.style.left=x_position; // Now set the x position of the layer

// Now start the timer again so this function will run again.
// It will run between 0 and 1 second later. (1000 milliseconds = 1 second).
the_timer=setTimeout( "set_timer()",   Math.floor( Math.random() * 1000)   );
}
//-->
</script>
</head>
<body onload=" set_timer() ">

<div id="the_layer" style="position:absolute;">
   <img src="dave.png">
</div>
</body>
</html>
```

The same function will be run again a bit later (a random time between 0 and 1 second)

Stopping the Timer

- What if you want to stop the timer before it finishes?

- Use: `clearTimeout(the_timer);`
  where 'the_timer' is the variable which was used to start the timer

- Here's a reminder of how a timer gets started:

```
the_timer=setTimeout('set_timer()', 1000
);
```

Example: clearTimeout

```
<html>
<head>
<script language="JavaScript">
var the_timer;
function set_timer() {
        the_timer=setTimeout("show_message()", 500 ); }

function reset_timer() {
        clearTimeout(the_timer); }

function show_message() {
        alert("Too slow! Are you a snail?"); }
</script>
</head>

<body onload="set_timer()" onclick="reset_timer()">
<h1>Click on this page within 0.5 seconds!</h1>
</body>
</html>
```

Intervals

- An alternative to setTimeout is setInterval
- setTimeout is for something that happens once
- setInterval is for something that happens repeatedly
- As before, the time value is in thousandths of a second
- For example: `setInterval("update_display()",3000)`
- The function *update_display()* will be executed every 3 seconds
- This will continue until the web page is unloaded, or until the interval is cleared using *clearInterval()*

Including Other JavaScript Files

- You can use ‘script’ to load multiple JavaScript files
- For example, you could split your code into different files, then in your main file have this:

```
. . .
<script language="JavaScript"
        src="audio_functions.js"
        type="text/javascript">
</script>
<script language="JavaScript"
        src="layer_functions.js"
        type="text/javascript">
</script>
<script language="JavaScript"
        src="form_functions.js"
        type="text/javascript">
</script>
. . .
```

Take Home Message

- JavaScript runs on browser to interact with users via:
  - Input and output via dialog box, text box, etc.
  - Detect and respond to mouse and keyboard events
- JavaScript manipulates the DOM structure of a web page, including reading and modifying existing DOM nodes, inserting new and deleting old DOM nodes
- We learnt event handling, including mouse events, keyboard events, page events, and timers