



COMP 4021

Internet Computing

More Web Programming Techniques

Dr. Kenneth LEUNG

Slides created by Dr. David ROSSITER

Div Properties

- Div properties which are commonly controlled:
 - x and y position
 - width and height
 - Z index (=display precedence order)
 - background properties (i.e. colour, or use an image)
 - visible/ hidden
 - Clipping (=changing the visible boundaries of the div)
 - x, y offset (=changing the x, y position within the div)

Nice Way to Create a Div

- Typically you would first define the style information for a div (such as the position and colours):

```
<style type="text/css">
```

```
<!--
```

```
.layer_style1 {
```

```
    position:absolute; top:20px; left:5px;
```

```
    color:#CC00EE; width:200px;
```

```
}
```

```
//-->
```

```
</style>
```

A completely new style rule is
being created

Declaring the Div

- Then you would define the layer, using the style rule:

Here we use the style rule created in the last slide

```
<div id="layer_name1" class="layer_style1" >  
  <h1>Layer 1</h1>  
  <p>Content for layer 1 goes here.</p>  
</div>
```

Dynamically Changing Layers

- After you have set everything up, you can use JavaScript to change the properties of the layer(s)
- For example, to make the layer move, you change the .left and .top properties of the layer
- In the following example (seen previously)
 - A layer is set up
 - When the mouse moves into the layer, it turns blue
 - When the mouse moves out of the layer, it turns red
- The general description for dynamically changing layers (or any other HTML thing) is called 'Dynamic HTML'

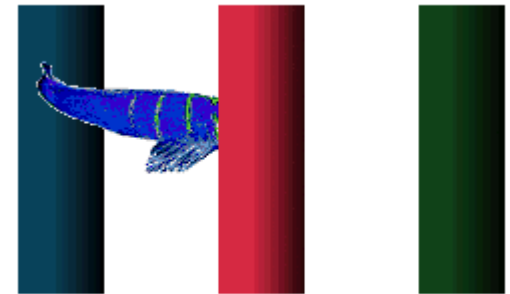
Example - Changing Layer Colour

```
<html>
<script language="JavaScript">
    function changeColor(newcolor) {
        document.getElementById("myrect").
            style.backgroundColor=newcolor;
        return false;
    }
</script>
<div id="myrect" style="position:absolute; background-color:red;
    width:200px; height:100px" onmouseover="changeColor('blue');"
    onmouseout="changeColor('red');">
    <p>Layer content...</p>
</div>
</html>
```

- Each layer has a z-index property
 - The greater the z-index, the 'closer' the layer is to the viewer
 - In this example several layer z-index values change when the fish reaches the right hand side
-
- The parameter is called 'z-index'
 - So you would refer to it as 'z-index' in a style instruction
 - However, the hyphen causes trouble if you type it directly in JavaScript code
 - So in that situation it has to be changed to 'zIndex'

capital letter here, hyphen has gone

Changing z-index Example



Changing zIndex - Example

```
function change_layer_order()  
{  
    var redpole = document.getElementById("redpole");  
    var bluepole = document.getElementById("bluepole");  
    var greenpole = document.getElementById("greenpole");  
  
    var tmp = redpole.style.zIndex;  
    redpole.style.zIndex = bluepole.style.zIndex;  
    bluepole.style.zIndex = tmp;  
    greenpole.style.zIndex = tmp;  
}
```

Swap z values {

JavaScript - More on Event Handling

- Let's say somebody clicks on something in a web page, and the onclick event is handled by a function (called the 'event handler')
- How can the event handler code know which object caused the event?
- For example, let's say there are 30 objects in the web page that can trigger the same onclick event handler code to be executed – how can the code know which object was clicked on?

Event Handling Example 1/2

```
<html>
<script language="JavaScript">

/* This is an example of one event handler being
   used with multiple objects – works in all browsers
   i.e. IE, Firefox, Chrome, etc */

function handle_event(obj) {
    obj.style.background="red"; // Change the bg colour
}
</script>
```

```
<body>  
<h2>Click on any object</h2>
```

```
<p style="background:lightgrey;"  
  onclick="handle_event(this)">
```

I'm a paragraph!

```
</p>
```

```
<br />
```

```
<div style="background:lightgrey;"  
  onclick="handle_event(this)">
```

I'm a div!

```
</div>
```

```
<br />
```

```
<span style="background:lightgrey;"  
  onclick="handle_event(this)">
```

I'm a span!

```
</span>
```

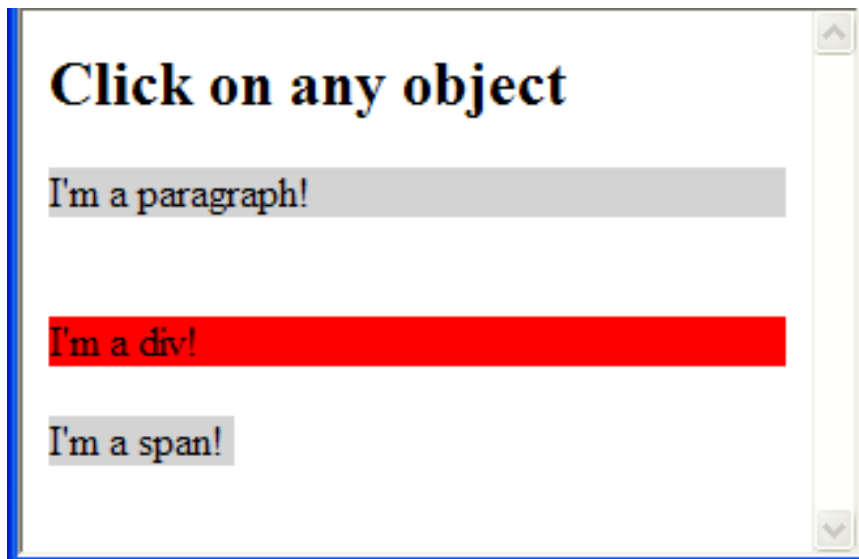
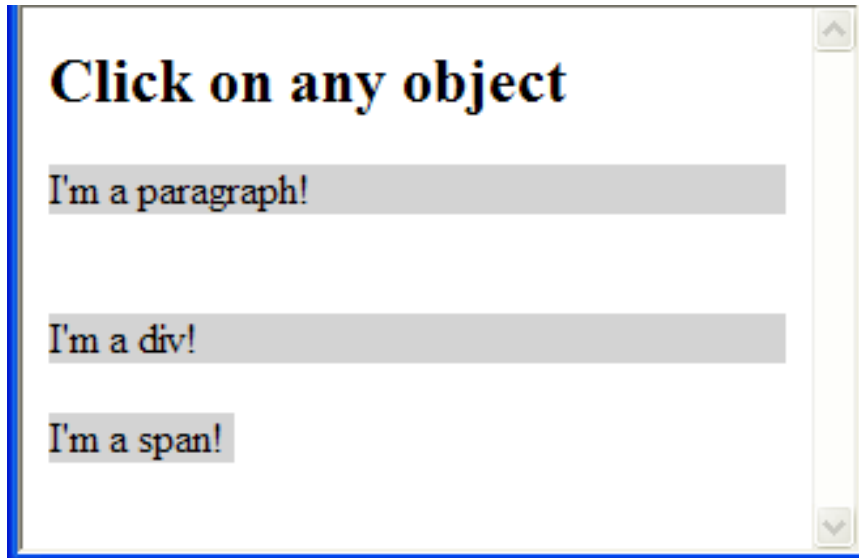
```
</body></html>
```

Events should always be written using all lower case letters i.e. write 'onclick' and not 'onClick'

Event Handling Example 2/2

In this example a p, div and span are used to demonstrate the event handling techniques

Example Execution



*After clicking
on the div*

- There is also another way to access the object that triggered the event – .target (W3C)
- This method doesn't use 'this'. An example:

```
function example_method2(e) {  
  if (e == null) e = window.event;  
  e.target.style.left = parseInt(e.target.style.left)+10;  
    // Move the layer 10 pixels further to the right  
}
```

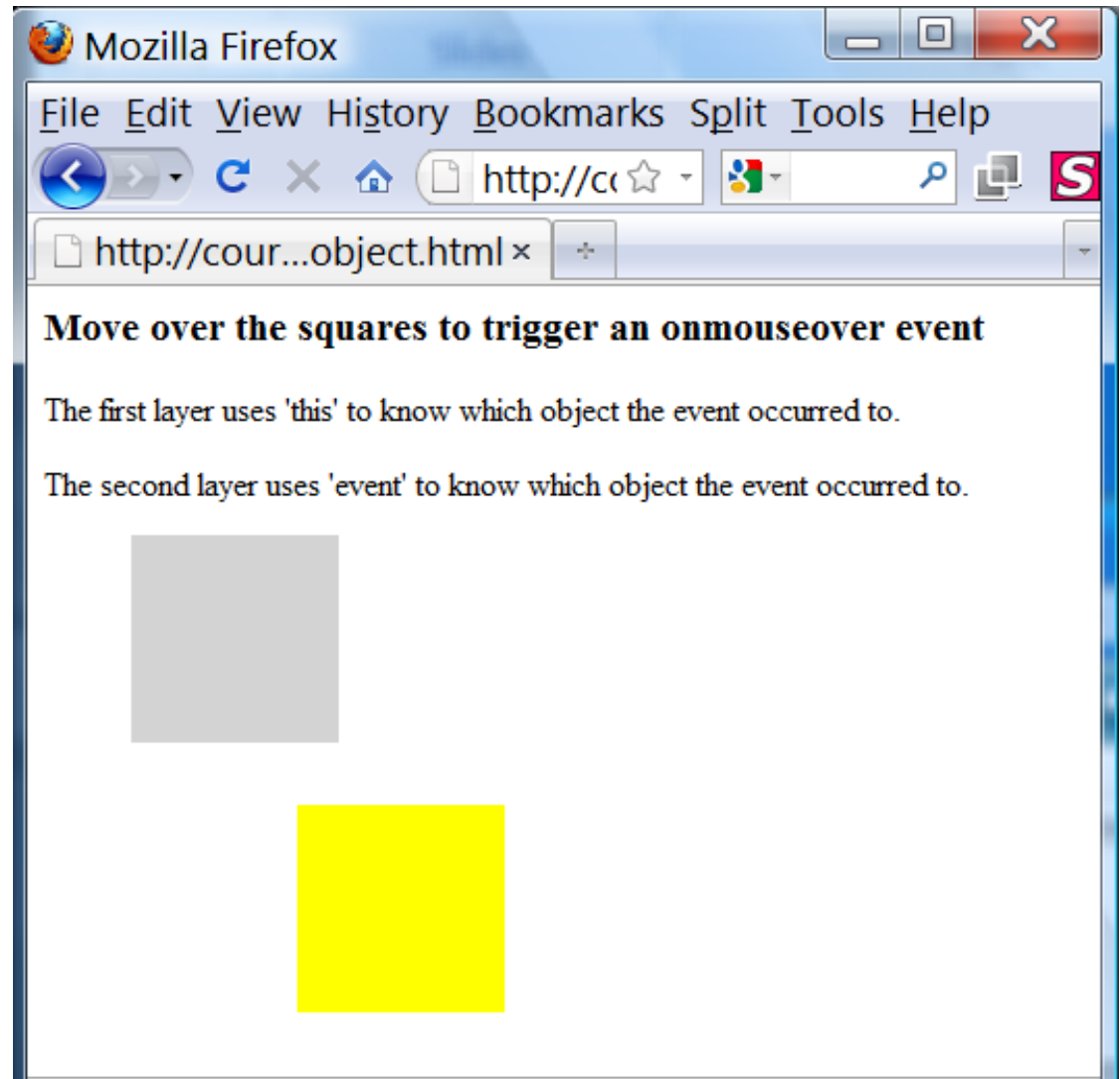
*parseInt is used to convert
the string into a number*

Another Way to Access the Object

```
<div id="text_layer2"  
  style="position:absolute; top:200; left:130; width:100;  
    height:100; background:yellow;"  
  onmouseover="example_method2(event)">  
</div>
```

...

Event Handling Example

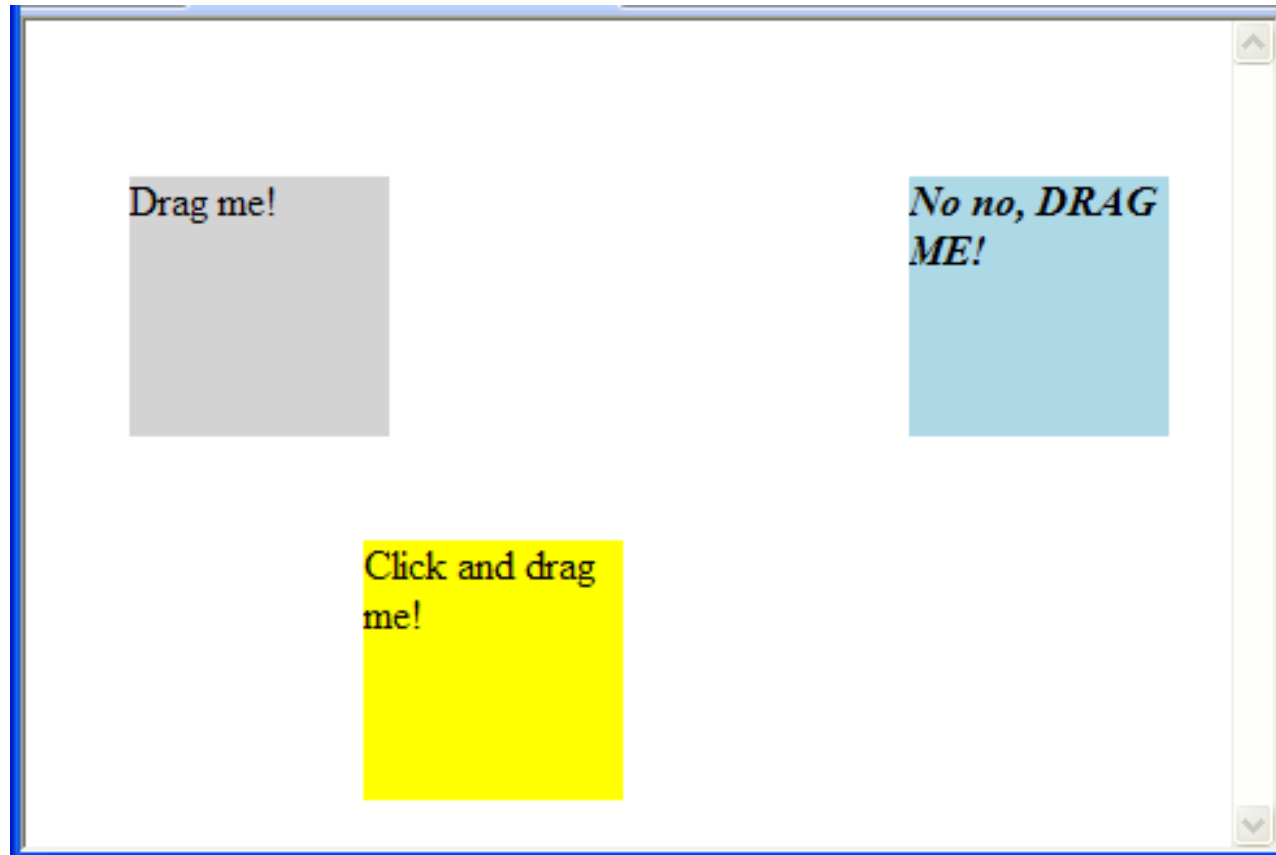


Click and Drag

There are new events in HTML5 for handling click and drag – but they are not yet implemented in all browsers. Here we look at the ‘traditional’ way of handling click and drag in a web page.

- ‘Click and drag’ interfaces can be created using Divs
- One approach is:
 - Put the things you want to be dragged into a layer
 - When the layer receives an ‘onmousedown’ event, drag and drop mode begins
 - In this mode, when there is an ‘onmousemove’ event, change layer position according to mouse position (need to use an x, y offset)
 - When the layer receives an ‘onmouseup’ event, drag and drop mode ends

Click and Drag Example



Example Click and Drag (1/4)

```
<html> <script language="JavaScript">
```

```
/*
```

This is an example of click & drag code
for multiple objects (HTML).

```
*/
```

```
var offset_y=0, offset_x=0;    // Store the position of the mouse  
    // cursor relative to the top left hand corner of the  
    // object being dragged
```

```
var dragmode=false;           // Simple boolean variable
```

```
function start_drag_mode(e) {  
    offset_x=e.clientX-parseInt(e.target.style.left);  
    offset_y=e.clientY-parseInt(e.target.style.top);  
    dragmode=true;  
}
```

Example Click and Drag (2/4)

```
function stop_drag_mode() {  
    dragmode=false;    // Turn off the mode  
}  
  
function update(e) {  
    if (dragmode===true) {  
        // If we are in drag mode, update the position of the object,  
        // taking into account the offset position when  
        // the mouse button was clicked down on the object.  
        e.target.style.left =e.clientX - offset_x ;  
        e.target.style.top  =e.clientY - offset_y ;  
    }  
}  
</script>
```

Example Click and Drag (3/4)

```
<body>
<div id="text_layer1"
  style="position:absolute; top:200; left:130; width:100;
  height:100; background:yellow;"
  onmousedown="start_drag_mode(event)"
  onmouseup="stop_drag_mode()"
  onmousemove="update(event)">
  Click and drag me!
</div>
```

```
<div id="text_layer2"
  style="position:absolute; top:60; left:40; width:100;
  height:100; background:lightgrey;"
  onmousedown="start_drag_mode(event)"
  onmouseup="stop_drag_mode()"
  onmousemove="update(event)">
  Drag me!
</div>
```

Example Click and Drag (4/4)

```
<div id="text_layer3"
  style="position:absolute; top:60; left:340; width:100;
  height:100; background:lightblue;"
  onmousedown="start_drag_mode(event)"
  onmouseup="stop_drag_mode()"
  onmousemove="update(event)">
  <i><b>No no, DRAG ME!</b></i>
</div>
```

```
</body>
</html>
```

- In this example we use three layers to demonstrate the click and drag technique
- The technique shown here could be used for as many layers as you need, not just three