COMP 4021
Internet Computing

More Web Browser Programming Techniques

CSS basics and more div and event handling

David Rossiter

Cascading Style Sheets - CSS

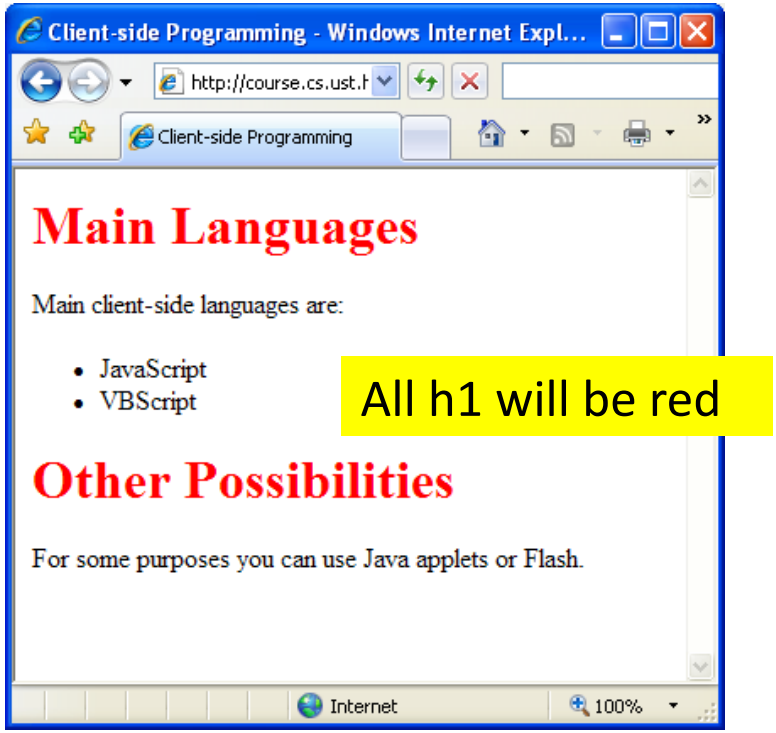
- CSS separates visual parameters (colour, spacing, etc.) from actual content (words)
- http://www.w3.org/Style/
- You have already seen how style can be used for individual elements:

```
<p style="color:red; background-color:yellow; font-size:46pt;
   font-style:italic“ >
A pretty paragraph.</p>
```

- But what if you want the same visual information to be used for *all* paragraphs in the web page?

Cascading Style Sheets - CSS

- Declare “style” once and apply it to all paragraphs
- You can put a style section at the top of the page



All h1 will be red

```
<style>
h1 { color: red }
</style>
… …
<body>
<h1>Main Languages</h1>
<p>Main client-side languages are:</p>
<ul>
<li>JavaScript</li>
<li>VBScript</li>
</ul>
<h1>Other Possibilities</h1>
<p>Java applets or Flash</p>
</body>
```

Rule:

h1 { color: red }

selector↑ ↑ style

Style Using a Relative Link

- An alternative is to separate style and data into two files:

File: my_style.css

```
h1 { color: red }
```

File: css_simple.html

```
<link rel="stylesheet" href="my_style.css"
    type="text/css"/>

<body>
<h1>Main Languages</h1>
<p>Main client-side languages are:</p>
<ul>
<li>JavaScript</li>
<li>VBScript</li>
</ul>
<h1>Other Possibilities</h1>
<p> Java applets or Flash.</p>
</body>
```

- The visual result is the same as before

Div and Span

- ‘div’ and ‘span’ are useful because they don’t have any default visual display parameters
- div is additionally useful because it can have any position
- ‘div’ is typically for a rectangular area of text/objects, i.e.

```
<div style="background-color: yellow;
    font-weight: bold; color: black;">
    <p>Here’s a paragraph</p>
    <p>Here is a second paragraph</p>
</div>
```

- ‘span’ is for a few “inline” words:

```
<p>This part is <span style="color:blue; background-
    color:red;">special</span> </p>
```

div introduces a line break but span does not

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 (change the visible boundaries of the div)
  - x, y offset (change 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 new style rule is created

Declaring the Div

- The div is defined using the style rule:

Style rule created in the last slide is used

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

Dynamically Changing Layers

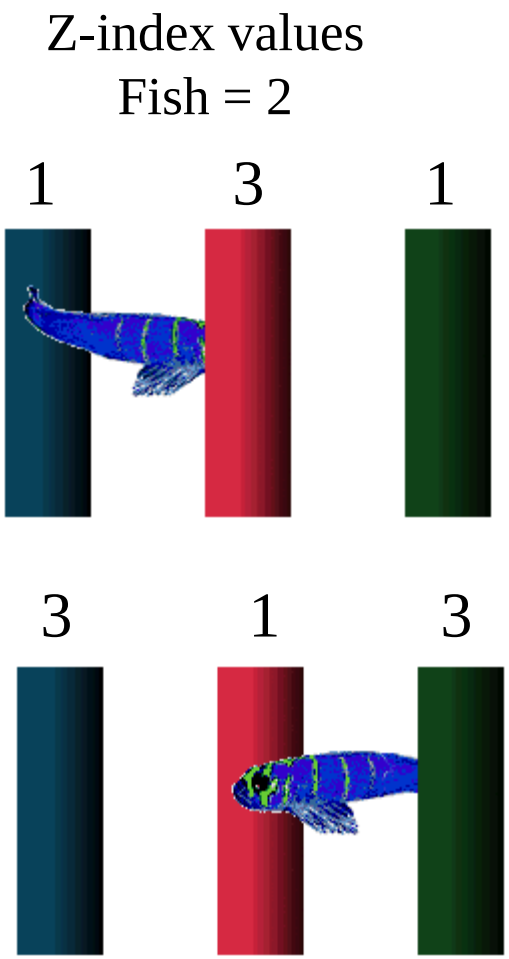
- JavaScript can be used to change the properties of a layer
  - E.g., to make the layer move, change the .left and .top properties of the layer
- In the following example (seen previously)
  - A layer is set up
  - When the mouse moves into the layer, it turns blue
  - When the mouse moves out of the layer, it turns red
- Techniques for dynamically changing layers (or any other HTML elements) is called 'Dynamic HTML' or DHTML
  - Today, by HTML, you implicitly include DHTML

Example - Changing Layer Colour

```
<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>
```

Changing z-index Example

- Each layer has a z-index property
- The greater the z-index, the 'closer' the layer is to the viewer
- The z-indexes of the layers are changed when the fish reaches the right hand side
- In JavaScript, the z-index parameter is called 'zIndex' and set in "style"



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

- Blue and Green have the same z-index value
- Red is swapped with Blue/Green in each call of the function

JavaScript - More on Event Handling

- When somebody clicks on something in a web page, the onclick event is handled by a function (called the event handler)
- How can the event handler code know which object caused the event?
- E.g., if there are 30 objects in the web page that can trigger the same onclick event handler code – how can the code know which object was clicked on?

Event Handling Example 1/2

```
<script language="JavaScript">

/* One event handler is used for multiple objects */

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

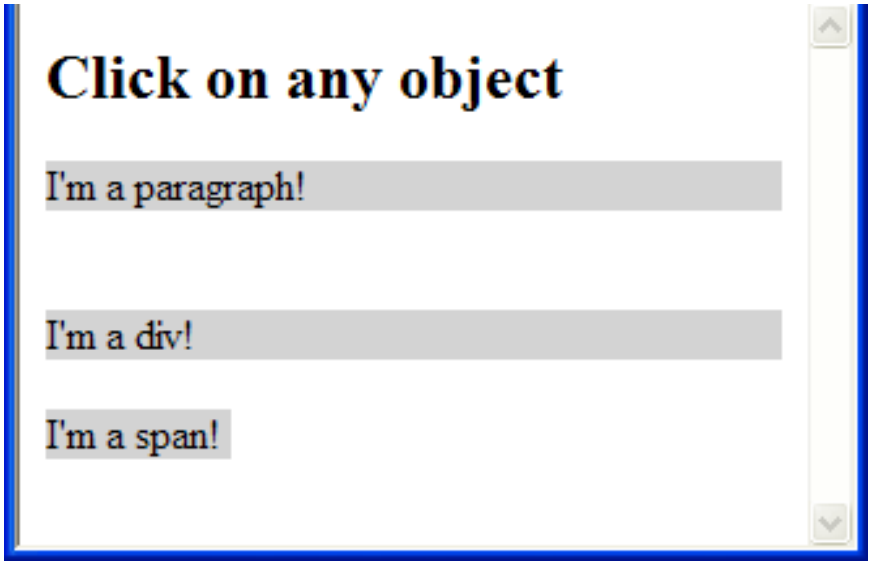
Parameter is an object pointer/ID

Event Handling Example 2/2

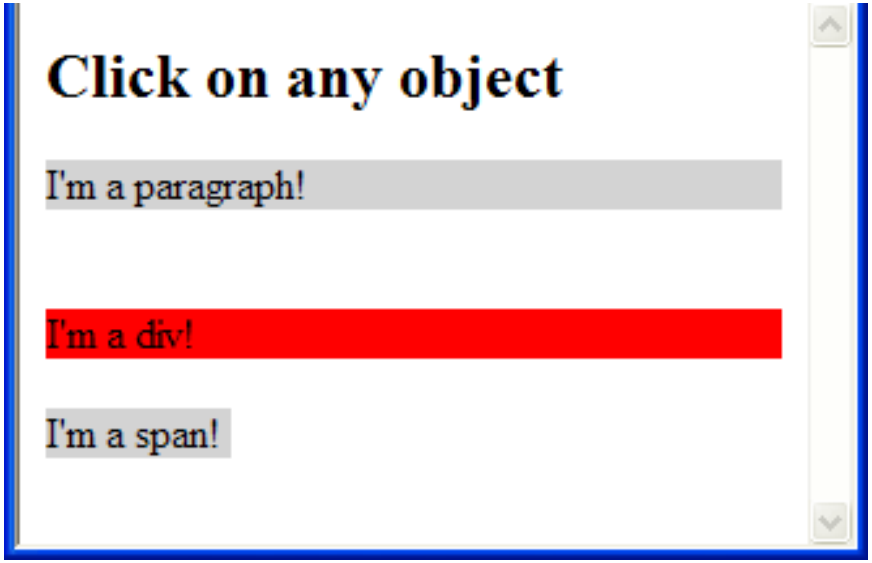
```
<h2>Click on any object</h2>

<p style="background:lightgrey;"
  onclick="handle_event(this)">
I'm a paragraph! </p>

<div style="background:lightgrey;"
  onclick="handle_event(this)">
I'm a div! </div>

<span style="background:lightgrey;"
  onclick="handle_event(this)">
I'm a span! </span>
```

- Events should always be written using all lower case letters, i.e., write ‘onclick’ and not ‘onClick’
- When the p, div and span elements are clicked, the same event handler is called but with the respective object pointer is passed to the handler

Example Execution



After clicking on the div



Another Way to Access the Object

- When an event triggers an event handler:
  - W3C: For an event object e, e.target is the element that triggered e
  - IE8 or earlier: does NOT pass event object to event handler, but updates window.event object with window.event.srcElement recording the triggering element, i.e., e.target = window.event.srcElement

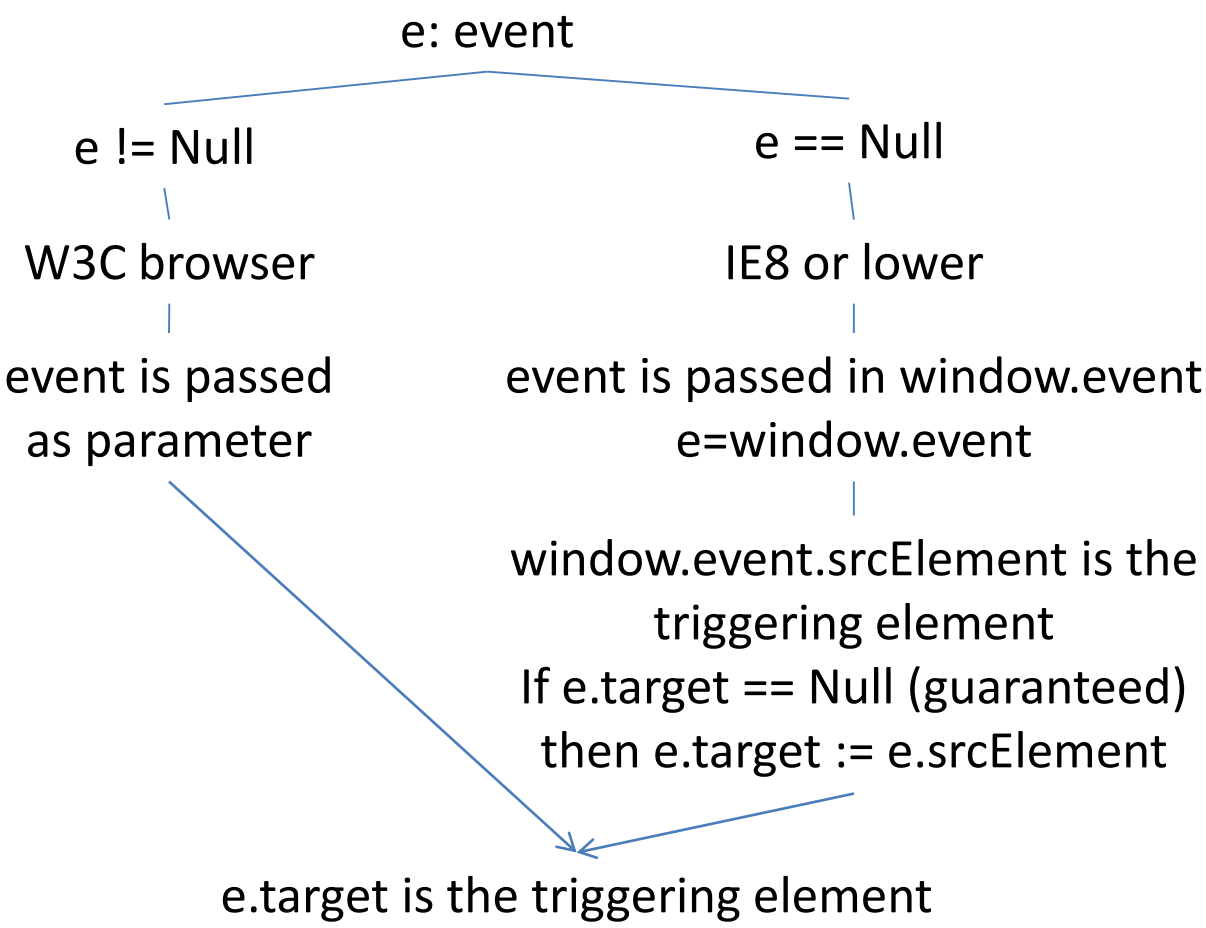
```
function example_method2(e) {
  If (e == null) e = window.event;
  If (e.target == null) e.target = e.srcElement;
  e.target.style.left = parseInt(e.target.style.left)+10; }
```

Browser is IE

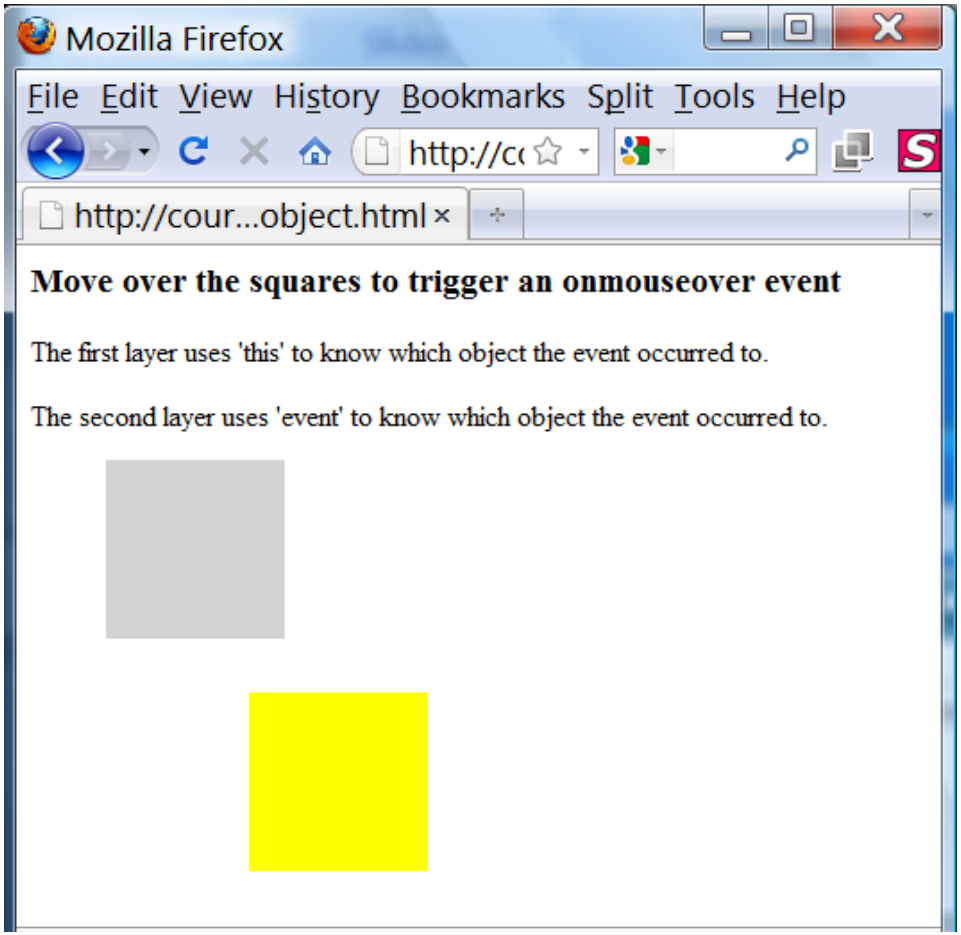
Move the layer 10 pixels to the right

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

Flow Chart


e: event
e != Null
e == Null
W3C browser
IE8 or lower
event is passed
as parameter
event is passed in window.event
e=window.event
window.event.srcElement is the
triggering element
If e.target == Null (guaranteed)
then e.target := e.srcElement
e.target is the triggering element

Event Handling Example

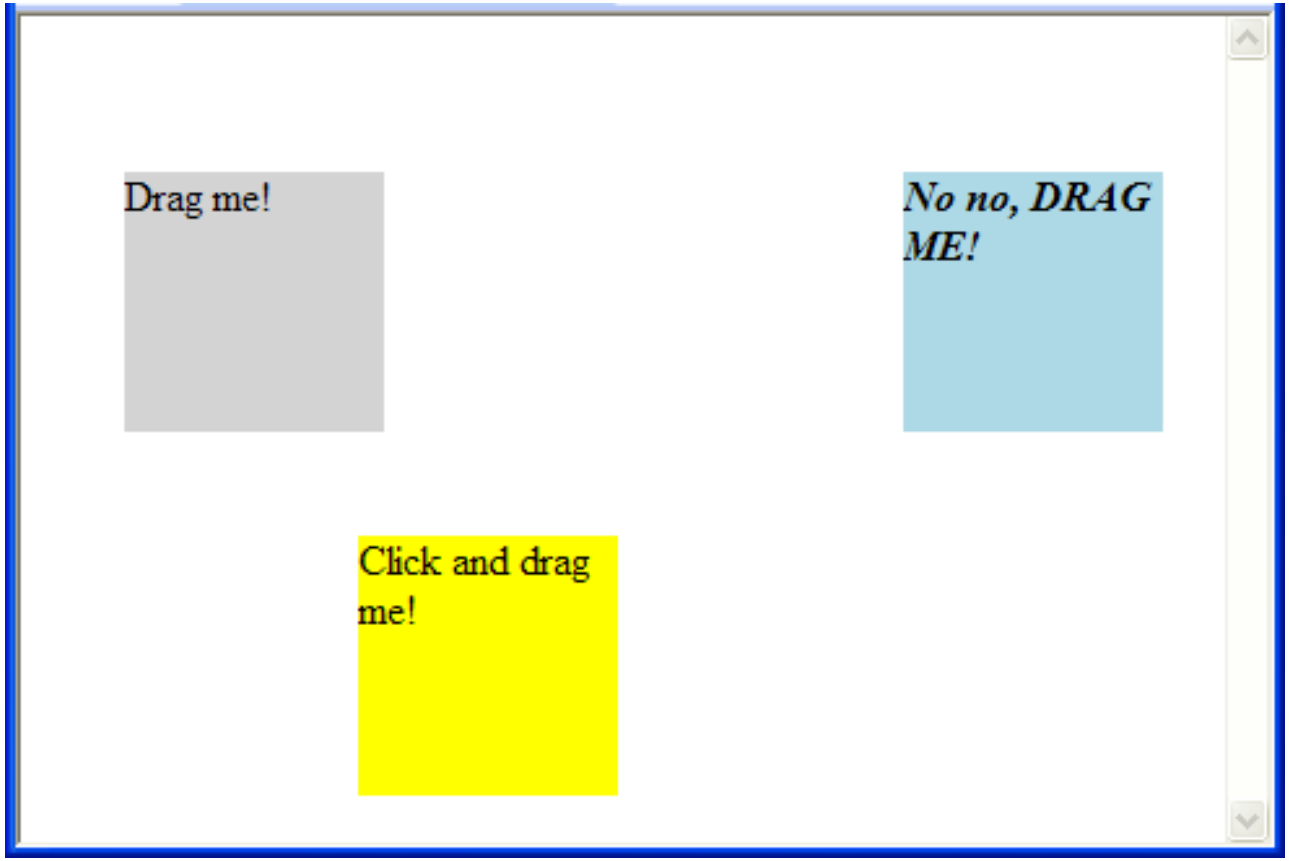


Click and Drag

- ‘Click and drag’ interfaces can be created using div’s
  - 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

- HTML5 provides new events for handling click and drag
- Here we look at the ‘traditional’ way of handling click and drag in

Click and Drag Example



Click and Drag for Multiple Objects (1/4)

```
<script language="JavaScript">
var offset_y=0, offset_x=0;
var dragmode=false;

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;
}
```

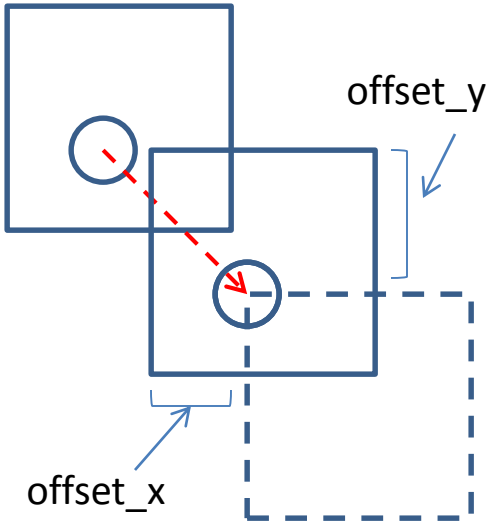
For storing the position of the mouse cursor relative to the top left hand corner of the object being dragged

x and y coordinates of clicked point

Example Click and Drag (2/4)

```
function stop_drag_mode() {
    dragmode=false;   // Turn off the mode   }

function update(e) {
    if (dragmode==true) {
      e.target.style.left = e.clientX - offset_x ;
      e.target.style.top  = e.clientY - offset_y ;    }
}
</script>
```

Update div position only **after** movedown



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.

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>
```

- 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

Take Home Message

- JavaScript is fun and powerful when combined with DOM
- Events make interactivity between users and application possible