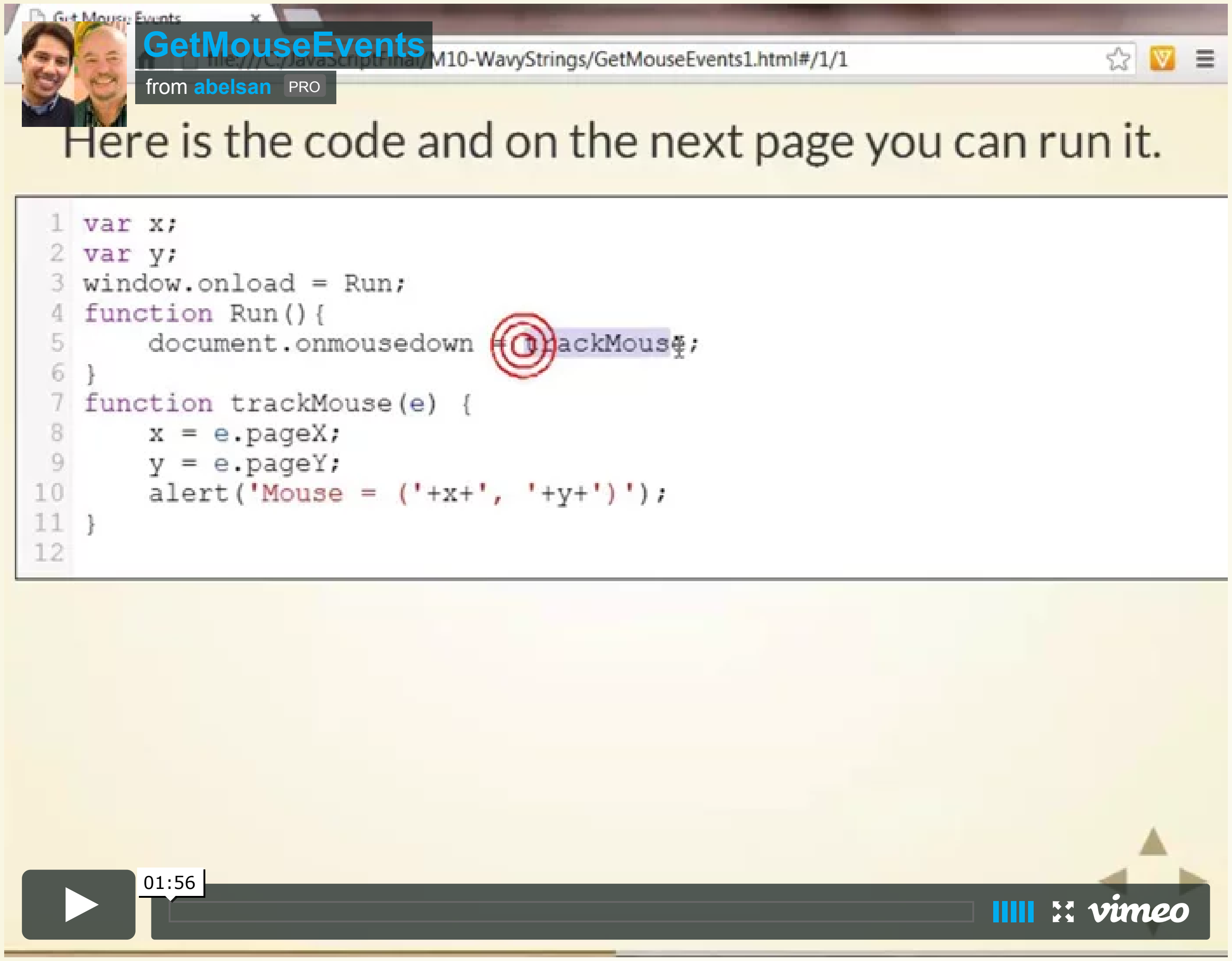
GET MOUSE EVENTS

John R. Williams and Abel Sanchez


GetMouseEvents
from abelsan PRO
file:///C:/JavaScriptFinal/M10-WavyStrings/GetMouseEvents1.html#/1/1
01:56
vimeo

Here is the code and on the next page you can run it.

```
var x;
var y;
window.onload = Run;
function Run(){
    document.onmousedown = trackMouse;
}
function trackMouse(e) {
    x = e.pageX;
    y = e.pageY;
    alert('Mouse = ('+x+', '+y+')');
}
```

The browser and HTML5 provide several ways in which mouse events can be handled. We have already seen that we can call a "Run()" function when a button is clicked. These functions are often called "listeners" because they wait listening for events. We note that events are associated with HTML elements, such as a button element, document element or canvas element. We'll in the lesson how to add "listeners" to events on a specific HTML element, such as an image or the canvas.

```
<input type="button" id="myButton" value="Run" onclick="Run()">
```

MOUSE EVENTS

We need to capture the mouse events. The browser gives us an easy way to do this. First lets listen for events on the whole 'document'. Every time the mouse is clicked on the document the browser fires an 'onmousedown' event. There are two ways we can add "listeners" to an HTML element. They are shown below and we can use either. Notice that the listener gets passed the "event" as its first argument.

```
<SCRIPT>
document.onmousedown(trackMouse);    // notice this uses onmousedown
document.addEventListener('mousedown',trackMouse); // this uses mousedown
function trackMouse(e){
        // the event e is the mousedown event
}



</SCRIPT>
```

Here is some simple code to track the mouse down event. On the next page you can run it.

CLICK YOUR MOUSE BUTTON TO SEE COORDINATE IN ALERT BOX

MOVE YOUR MOUSE AROUND ON THE PAGE

Now lets look at the code to track your mouse by drawing a trail of circles. To do this we are going to use a new element on the HTML page called "canvas". We will use the id to get hold of the canvas and attach listeners to it.

```
<body>
<canvas id="myCanvas" width="900" height="300"></canvas>
</body>
```

Here is the JavaScript that adds an event listener to the canvas element. It also draws little red circles on the canvas.

```
<SCRIPT>
var context;
var canvas;
window.onload=getCanvas;
function getCanvas(){
canvas = document.getElementById("myCanvas"); // get a handle on canvas
context = canvas.getContext("2d");   // we need context to draw on canvas
Run();
}
function Run(){
canvas.addEventListener('mousemove',trackMouse);
}
function trackMouse(e) {
var posx = e.pageX;
var posy = e.pageY;
DrawCircle(posx,posy, 3);
}
function DrawCircle(x, y, radius){  // notice the use of context to draw
context.fillStyle = "rgba(255, 0, 0, 1)";
context.beginPath();
context.arc(x, y, radius, 0, Math.PI*2, true);
context.closePath();
context.fill();
}
</SCRIPT>
```

SIMPLE EVENTS

The following are simple mouse events:

- mousedown - Triggered by an element when a mouse button is pressed down over it
- mouseup- Triggered by an element when a mouse button is released over it
- mouseover- Triggered by an element when the mouse comes over it
- mouseout- Triggered by an element when the mouse goes out of it
- mousemove- Triggered by an element on every mouse move over it.

COMPLEX EVENTS

The browser provides more complex events:

- click- Triggered by a mouse click: mousedown and then mouseup over an element
- contextmenu- Triggered by a right-button mouse click over an element.
- dblclick- Triggered by two clicks within a short time over an element There is also a mousewheel event, but it’s not used. The scroll event is used to track scrolling instead. It occurs on any scroll, including keyboard scroll.

THE END