OnKeyDown Tutorial:

Using keys on the key using your keyboard keys to call functions.

NOTE: OnKeyPress and OnKeyDown are only slightly different. OnKeyDown is triggered when a key is pushed down. The release isn't required. OnKeyPress is triggered when a key is pressed (usually including release). OnKeyDown works on all browsers, but if you're having issues, feel free to replace it with onKeyPress.

Contents

Events: 1

Using onKeyDown: 2

Below is a table of keys and their keycodes: 2

Using keyDown to move the hero: 3

Your Turn: 4

(6 pts)Get the onKeyDown to work with your hero 4

Fun Extras with OnKeyDown: 4

Even Crazier fun: 4

Events:

JavaScript is designed to be interactive, meaning that it is designed so that users interact with the web page by clicking on buttons, running their mouse over and off of items, and even by pushing on keys on the keyboard. Each one of these actions is known as an **event**. If you've ever heard the term, "event driven", it means that the functions that are called in javaScript are a direct result of actions that the user took, like, for instance, typing on a key on the keyboard. Typing on the keyboard is considered an event in javaScript.

We've called javaScript functions using onClick. We've also written code that calls functions for us, for instance, by putting the name of a function inside of another function, or by using setTimeout.

But there are other actions a user can take that will result in calling a function. One is onKeyDown. onKeyDown calls a function when any key on your keyboard is pressed down. For example:

```
<body onKeyDown = "myfunc(event)">
```

Now whenever any key is pressed on the keyboard, the function *myfunc* is called. The event is a parameter, and it is the key that was pressed on the keyboard (in other words, it is what is keeping track of which key you pressed).

event means the activity the user takes on the web page (like pressing a key), and this event – what the user did – must be passed into the function as a parameter.

So when you call a function using onKeyPress, you must call it with a parameter representing the event (as seen in the example in which onKeyDown calls myfunc(event) above).

The event inside the function becomes whatever you named your parameter. In the example below, I called the event parameter keys:

```
function myfunc(par)
```

Now, inside the function, par hold the keystroke the user took. To identify which key the user pressed, the event has associated with it a .keycode. Each key on your keyboard has a keycode associated with it. So the keycode tells the function exactly which key you pressed.

Notice that the onKeyDown goes inside the body tag. This is so that you can type anywhere within the body of the web page and the onKeyDown will be called. If we just put it inside a button, you'd have to very carefully type within the button, and that's usually not what you want to happen. Usually we want to be able to type a key anywhere within the web page and have it recognized. For instance, if you want to move something, you want to be able to hit the up arrow anywhere in the web page and have it work.

Using onKeyDown:

Let's try using the onKeyDown to see how it works:

Step 1: In your web page, add a paragraph (this paragraph is just for testing purposes – you can get rid of it after we see how the keyDown works. So it doesn't matter where you place this paragraph). Give the paragraph an id of "keyp". I don't care about the content of the paragraph.

Step 2: Add to the body tag a call to the javaScript function getArrows(event) as follows:

```
<body onKeyDown = "getKeys(event)">
```

Step 3: Save the html code.

Step 4: Now in your .js file, add the function getKeyss(par) as follows:

```
function getKeys(par)
{
	if (par.keyCode === 39)
	{	document.getElementById("keyp").innerHTML = "You pushed the right arrow"
	}
	else if (par.keyCode === 37)
	{	document.getElementById("keyp").innerHTML = "You pushed the left arrow"
	}
	else if (par.keyCode === 38)
	{	document.getElementById("keyp").innerHTML = "You pushed the up arrow"
	}
	else if (par.keyCode === 40)
	{	document.getElementById("keyp").innerHTML = "You pushed the down arrow"
	}
}
```

Step 5: Save the .js file. Load the web page into a browser, and test by clicking on any of the four arrow keys. Depending on which arrow key you press, the paragraph should change to the appropriate innerHTML text.

A couple of notes: If this doesn't work, try changing onKeyDown to onKeyPress and test again.

If the web page starts jumping around when you press the arrows, that means that the browser associates the arrows with web page movement. You may not want to use the arrow keys to trigger a function. In the chart below, I've included every key on the keyboard and its corresponding keycode. Try using other keycodes.

Below is a table of keys and their keycodes:

backspace 8	a 65	right window key 92
tab 9	b 66	select key 93
enter 13	c 67	f1 112
shift 16	d 68	f2 113
ctrl 17	e 69	f3 114
alt 18	f 70	f4 115
escape 27	g 71	f5 116
		f6 117

page up 33	h 72	f7 118
page down 34	i 73	f8 119
end 35	j 74	f9 120
home 36	k 75	f10 121
left arrow 37	l 76	f11 122
up arrow 38	m 77	f12 123
right arrow 39	n 78	num lock 144
down arrow 40	o 79	scroll lock 145
insert 45	p 80	semi-colon 186
delete 46	q 81	equal sign 187
0 48	r 82	comma 188
1 49	s 83	dash 189
2 50	t 84	period 190
3 51	u 85	forward slash 191
4 52	v 86	grave accent 192
5 53	w 87	open bracket 219
6 54	x 88	back slash 220
7 55	y 89	close braket 221
8 56	z 90	single quote 222
9 57	left window key 91	

Using keyDown to move the hero:

For the purposes of this final game,we want to use the keystroke to move the hero (because I'm guessing at this point you're all quite sick of clicking on those buttons). It is SOOOO much nicer to just hit a key to move the hero than to keep clicking that button! But clearly keystrokes can be used for many purposes outside of this game – make the font larger/smaller, make the volume go up/down, make the contrast or brightness go up or down, etc. etc. Just recently I've used keystrokes to allow users to jump between the areas of text in a long document that are relevant to answering a topical question. Clearly there are many reasons why you'd want to be able to detect the keystroke and use it to modify the behavior of a web page/app.

But for now we'll use it to move the hero:

Step 6: Modify the getKeys function so that for each if condition you call the appropriate Move function. So, for instance, before when you clicked on the right arrow, you called the MoveRight function with the id of your hero (mine was 'z1'), but yours could be whatever id you gave to your hero in your html code. Now, in the getKeys function, replace the line,

```
document.getElementById("keyp").innerHTML = "You pushed the right arrow"
```

with

```
MoveRight('z1')
```

(again, replace z1 with the id of your hero in your html code).

Step 7: Repeat for MoveLeft, MoveUp, and MoveDown.

(again, if you don't like usin the arrow keys to move your hero, you can use other keys by changing the keycodes in the function)

Step 8: Save your code and reload the html page. You should now be able to move your hero by pressing on keys anywhere within your web page.

Step 8b: Remove the paragraph used in steps 1-5 that you're no longer using.

Your Turn:

(6 pts)Get the onKeyDown to work with your hero

Fun Extras with OnKeyDown:

Either add anAdd an image to your html code of using one of the images already on your web page, add in-line style to the image that sets the opacity to something (maybe 0.5) to start with.

Step 1: Change the body tag so that, instead of calling the getKeys(event) method, you'll call a function changeOpacity(event).

Note that the body can only call one function, so it can either call getKeys() or changeOpacity() but not both at the same time.

However, if youwanted to, you could extend the getKeys function to include two new keys (maybe 'u' and 'd')

If you opted to call a new function, your body tag should now look like this:

```
<body onKeyDown = "changeOpacity(event)">
```

Step 2: Now in the .js file, add a global variable x, and initialize it to whatever you set your image's opacity to in the html file (so perhaps 0.5).

Step 3: Create a function changeOpacity(par) that increases the opacity if one key is pressed, and decreases opacity if another key is pressed. I, again, used the 'u' and 'd' keys, so mine looks like this:

```
x = .02
function changeopacity(par)
{       if (par.keyCode === 85)
        {       x = x + .03
                document.getElementById("zombie").style.opacity = x
        }
        else if (par.keyCode === 68)
        {       x = x - .03
                document.getElementById("zombie").style.opacity = x
        }
}
```

Step 4: Save and test your code. The image's opacity should go up and down when you click on the keys you've chosen.

Even Crazier fun:

This code detects where you clicked on the page, and then moves the image to that location in a continuous glide by slowly changing the position down from the top and over from the left based on the original top and left coordinates and the goal top and left coordinates.

Step 1: create a new html and .js file. Make sure they're connected.

Step 2: Place an image into your html file. Position it in-line and absolutely. Make sure the image is relatively small (maybe 100 by 100, or even smaller…). Give it an id of img3

Step 3: In your body, add the following function call:

```
<body onClick = "getClicks(event)" >
```

Step 4: In your .js, add the following code:

```
xcoord = 10.0;
ycoord = 10.0;
function getClicks(par)
{       x = par.clientX;
        y = par.clientY;
        xchange = (x-xcoord)/50.0;
        ychange = (y-ycoord) / 50.0;
        count = 0;
        Move(xchange,ychange)
}
function Move(xc, yc) {
        count = count + 1;
        if (count < 50) {
                ycoord = ycoord + yc;
                xcoord = xcoord + xc;
                y = Math.floor(ycoord)
                x = Math.floor(xcoord)
                document.getElementById('img3').style.position = "absolute"
                document.getElementById('img3').style.left = x + "px"
                document.getElementById('img3').style.top = y + "px"
                setTimeout(function() {Move(xc,yc)}, 30)
        }
}
```

Step 5: Save. Load the html file into your browser. Click on your web page. Hopefully something happens. Click again. Click as long as you want to.