vPedal Javascript Browser API Programmer Notes

Programmer Notes:

The above Javascript page is included in the free API. The Browser API is necessary to give today's programmer with the utmost power to control the interactions between the vPedal and any internet platform, including embedded devices.

The concept behind the API is very simple. Due to the limitation of various security programs and browsers, a COM/ActiveX method was replaced with a much smarter alternative. Each browser in Windows operates as a Window, whether it be **Microsoft Edge**, **Internet Explorer**, **FireFox**, **Chrome**, **Safari, Opera and all variants including the earlier Spartan version in Windows 10.**

vPedal Listener Tray Icon

Browser in the Active Window

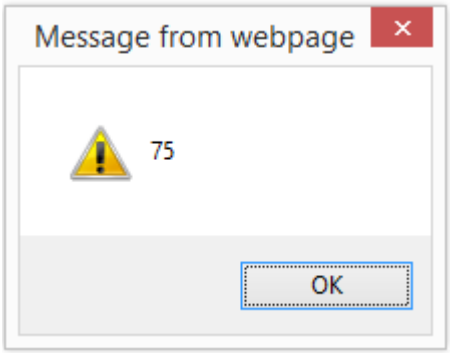
Javascript code and the events exposed

A challenge was presented to the programmer. How can one solution control the active application's web page and actually fire a JavaScript event, allowing the programmer to essentially – do anything?

The answer was developed via vPedal and its developers and looks like this to Windows.

Essentially, the vPedal Listener Tray Icon is responsible for authenticating and allowing the vPedal to operate. The Browser in the Active Window is automatically detected. When you enter any key into a browser, whether it be a text box or any object, an event is triggered. For example if you type in "K" into a text box, the object in focus would receive 3 events. **onKeyDown**, **onKeyPress**, and **onKeyUp**, each with an event parameter passed with the ASCII **Keycode** of "K", which is of course Character **75**. So therefore any Javascript programmer would know that they can have an event on the **HTML DOM** element **onKeyUp**="alert(event.keyCode);"

So, you would expect to see:



Javascript was chosen as it is more universal than any other scripting language. Despite the power of VBScript being capable, it is not universal to all browsers or usable on multiple platforms and devices, including mobile devices.

Now, obviously the **onkeyUp** event, using the event object gives you the ability to run code. This is triggered by the user. In this case, the **vPedal** must be the one to tell the browser. Without any plugins, how can this be so?

The **vPedal** Javascript API must be able to communicate with

the active browser and simulate the above. Hence, on various events of the **vPedal**, the window events are passed the active window. Take the following script for example. This script gets fired on the <BODY> onkeyUp event.

Naturally in Windows it would be quite silly to rely on the K key to be a FootDown event. Therefore the API has been programmed to specifically require a reasonably impossible combination of keys that could not possibly interfere with either Windows or the Active Browser.

Therefore we rely on the alt, ctrl, and the shift key to be held at the same time *"Virtually"*. So the following code is the listener, and executes various code on combinations of inbound key events sent from the vPedal tray icon.


```
<SCRIPT type="text/javascript">
		var isVpedalActive;
		isVpedalActive=false; //Global Variable
		function processkeycode(e) {
			var rewinding=false;
			var fastforwarding=false;
			var myVar;
			if (e.altKey==true && e.ctrlKey==true && e.shiftKey) {
			if (e.keyCode==49){ExecuteCommand("1");};
			if (e.keyCode==50){ExecuteCommand("10");};
			if (e.keyCode==51){ExecuteCommand("2");};
			if (e.keyCode==52){ExecuteCommand("20");};
			if (e.keyCode==53){ExecuteCommand("3");};
			if (e.keyCode==54){ExecuteCommand("30");};
			if (e.keyCode==55){ExecuteCommand("4");};
			return;
		}
	}
</SCRIPT>
```

Therefore this is the virtual listener that then goes and chooses what to do at certain events. This is what happens. This code says if the user presses **alt**, **ctrl**, **shift** and **1-6** then run the respective commands. The parameter to the ExecuteCommand is not relevant to the functioning, but it distinguishes an action. Review the following code.

```
<SCRIPT>

function ExecuteCommand(command) {
switch (command) {
isVpedalActive=true;
setTimeout(function(){ isVpedalActive=false; }, 30000);
    case '1':
        document.getElementById('eventmessage').innerHTML='FOOTDOWN (Play) - code runs here';
        break;
    case '10':
document.getElementById('player a').currentTime=document.getElementById('player a').currentTim
e-1;
        break;
    case '2':
        document.getElementById('eventmessage').innerHTML='FOOTDOWNLEFT (Rewind) - code runs
here';
myVar=setInterval(function(){document.getElementById('player_a').currentTime=document.getEleme
ntById('player_a').currentTime-1;}, 250);
        break;
    case '20':
        rewinding=false;
        document.getElementById('eventmessage').innerHTML='FOOTUPLEFT (StopRewinding) - code
runs here';
        break;
    case '3':
        document.getElementById('eventmessage').innerHTML='FOOTDOWNRIGHT (FastForward) - code
runs here';
myVar=setInterval(function(){document.getElementById('player_a').currentTime=document.getEleme
ntById('player_a').currentTime+1;}, 250);
```


```
        break;
    case '30':
        document.getElementById('eventmessage').innerHTML='FOOTUPRIGHT (StopFastForwarding) -
code runs here';
        break;
     case '4':
        document.getElementById('eventmessage').innerHTML='The programmer is notified that the
vPedal is active and in use.';
myVar=setInterval(function(){document.getElementById('player_a').currentTime=document.getEleme
ntById('player_a').currentTime+1;}, 250);
        break;

    default:

       break;
 }
}
</SCRIPT>
```

This code captures the event and in this case, changes the contents of a Div. **Also note:** The absense n the above example of receiving a message in 30 seconds, sets the global variable **isVpedalActive** to False, otherwise True.

The above example is also written to do other media based events with **player_a** but for the purposes of the programmer, the main exercise here is to understand that:

ALT	**CTRL**	**SHIFT**	**1** ASC(49)	FOOTDOWN (Play)
ALT	**CTRL**	**SHIFT**	**2** ASC(50)	FOOTUP (Stop)
ALT	**CTRL**	**SHIFT**	**3** ASC(51)	FOOTDOWNLEFT (SetRewindFlag)
ALT	**CTRL**	**SHIFT**	**4** ASC(52)	FOOTUPLEFT (StopRewindFlag)
ALT	**CTRL**	**SHIFT**	**5** ASC(53)	FOOTDOWNRIGHT (SetFastForwardFlag)
ALT	**CTRL**	**SHIFT**	**6** ASC(54)	FOOTUPRIGHT (StopFastForwardFlag)
ALT	**CTRL**	**SHIFT**	**7** ASC(55)	HEARTBEAT – TO DETECT THE PEDAL EXISTS

ISVPEDALACTIVE IS TRUE ON ANY EVENT AND IS SET TO WAIT 30 SECONDS UNTIL IT IS FALSE, UNLESS IT IS PRESSED, THEN THE TIME STARTS AGAIN. THE PROGRAMMER COULD THEREFORE SOUND AN ALARM IF THE PEDAL IS NOT USED FOR A CERTAIN PERIOD OF TIME.

NOTE: To save the programmer, the API automatically runs a rewind and fast forward action time so you don't have to by pressing the relevant button at the set frequency defined when you right mouse click the taskbar icon.

VPEDAL