Application Development Tutorial for Vestel MB70 Smart TV

1.Preface

This document is written to help Vestel Smart TV application developers who have used development languages such as HTML, CSS and JavaScript. JavaScript is a prototype-based scripting language that is dynamic, weakly typed and has first-class functions. It is a multi-paradigm language, supporting object-oriented, imperative and functional programming styles. JavaScript has a low learning curve. It is easy to start. But it can be strange to implement some classical design patterns on JavaScript by programmers who working on statically typed languages for a long time. If you want to improve your JavaScript skills, I recommend you to read *Pro JavaScript Techniques* by *John Resig* and *JavaScript: The Good Parts* by *Douglas Crockford*.

This document is not a reference document that covers every feature of Vestel Smart TV.

2.Getting Started

2.1 What is Vestel Smart TV?

Vestel smart Tv is a web based application platform. Concept is very likely to classic web platforms. Plus, Vestel Smart Tv’s browser has special objects which are explained in the OIPF document. It serves many opportunities for Vestel Smart Tv developers like streaming DVBS Channels.

2.2. Server Initiation

We need a web server to serve the content. I prepared this tutorial with Apache Server on Microsoft Windows 7.

First we will create a simple html page and serve it. In Hotel Tv mode, we need to put a “contest” folder under root. The Tv need that folder to check your URL is valid. So here is the hello page.

index.php :

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <meta http-equiv="content-Type" content="text/html; charset=utf-8" />  <title>Vestek Tutorial</title>  </head>  <body style="background:gray;">  Hello World !  </body>  </html> |

Default background of the browser and text color are black so background of the body is styled as gray. Otherwise the text can’t be seen. Check your server path before lunching it for Tv.

2.3. Tv Initiation

In Hotel TV mode, the Tv navigates to a start URL after waking up. You can set it directly in Tv’s file system. Make a telnet connection to the Tv as root user, password is “mb704725”. Write your URL to “/mnt/settings/browser/starturl.txt” file. Wake your Tv, you should see the “Hello World” page on screen.

3. A basic application

3.1 Making a menu

Smart Tv Application development has different concept than classic web applications. This section tries to explain the basic approaches to create an application menu. For sure, there is no limit for creation with this environment.

3.1.1 CSS Based Menu

The Tv’s browser can navigate across the link elements with arrow keys. It is one of the approaches that you can obtain. In this example we write four link and arrow keys can navigate across these links. You should specify style of focused link by using “focus” pseudo-class in CSS file. Here is the simple example.

index.php:

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <meta http-equiv="content-Type" content="text/html; charset=utf-8" />  <link href="main.css" rel="stylesheet" type="text/css" />  <title>Vestek Tutorial</title>  </head>  <body style="background:gray;">  <a href="#">Link 1 </a><br />  <a href="#">Link 2 </a><br />  <a href="#">Link 3 </a><br />  <a href="#">Link 4 </a><br />  </body>  </html> |

main.css:

|  |
| --- |
| a  {  text-decoration:none;  font-size:20px;  color:Black;  }  a:focus  {  text-decoration:none;  color:White;  font-size:30px;  } |

When you navigate across these links, you will see the focused link as white.

3.1.2 Javascript Based Menu

You can also write it with JavaScript. It can be decrease DOM dependencies and you can also customize navigation by writing your own key handler methods. Here is the basic example with JavaScript.

index.php:

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <meta http-equiv="content-Type" content="text/html; charset=utf-8" />  <script type="text/javascript" src="main.js"></script>  <link href="main.css" rel="stylesheet" type="text/css" />  <title>Vestek Tutorial</title>  </head>  <body>  <div id="menu"></div>  <script>  vestek.boot();  </script>  </body>  </html> |

main.css :

|  |
| --- |
| body  {  background:gray;  }  #menu  {  text-decoration:none;  font-size:20px;  color:Black;  }  #menu .selected  {  text-decoration:none;  color:White;  font-size:30px;  } |

main.js

|  |
| --- |
| var vestek = {};  vestek.menuContent = [{ "title" : "Link 1", "url" : "#"} ,{ "title" : "Link 2", "url" : "#"},{ "title" : "Link 3", "url" : "#"},{ "title" : "Link 4", "url" : "#"} ];  vestek.menuIndex = 0;  vestek.boot = function () {  for(index in vestek.menuContent){  vestek.menuContent[index].element = document.createElement("div");  vestek.menuContent[index].element.innerHTML = vestek.menuContent[index]['title'];  document.getElementById("menu").appendChild( vestek.menuContent[index].element);  }  vestek.menuContent[this.menuIndex].element.className = "selected";  document.onkeydown = function (evt) {  vestek.keyHandler(evt.keyCode);  };  };  vestek.keyHandler = function (code) {  switch (code){  case this.keyEnum.OK:  document.location.href = vestek.menuContent[this.menuIndex]['url'];  break;  case this.keyEnum.UP:  if(this.menuIndex != 0){  vestek.menuContent[this.menuIndex].element.className = "";  this.menuIndex--;  vestek.menuContent[this.menuIndex].element.className = "selected";  }  break;  case this.keyEnum.DOWN:  if(this.menuIndex != this.menuContent.length - 1){  vestek.menuContent[this.menuIndex].element.className = "";  this.menuIndex++;  vestek.menuContent[this.menuIndex].element.className = "selected";  }  break;  default:    break;  }  };  vestek.keyEnum = {  OK : 13,  UP : 38,  DOWN : 40,  RIGHT : 39,  LEFT : 37,  GREEN : 404,  RED : 403,  YELLOW : 405,  BLUE : 406,  BACKSPACE : 461,  OK : 13,  PLAY : 415,  PAUSE : 19,  FW : 417,  BW : 412,  STOP : 413,  LANGUAGE : 312,  HOME : 407,  HELP : 156,  BACK : 461,  ALARM : 309,  ZERO : 48,  ONE : 49,  TWO : 50,  THREE : 51,  FOUR : 52,  FIVE : 53,  SIX : 54,  SEVEN : 55,  EIGHT : 56,  NINE : 57,  PROGUP : 310,  PROGDOWN : 311,  FRONTPROGUP : 314,  FRONTPROGDOWN : 313,  }; |

Menu content is just a JSON. It also gives you a flexibility to write Ajax based application. All default key values are written as enum. You can abstract key values from your application. You can also change key map via object api which I will mention later. You can find more information about key mapping in the OIPF document.

3.2. Making a basic video player

Object element defines an embedded object in HTML. As I mention before, Vestel Smart Tvs has some special objects. OIPF declares some standards for these objects. In this page we will use “video/mp4” type object. If we select this element, we can use its native properties and methods which are declared in the OIPF document.

HTML:

|  |
| --- |
| <object id='video' type='video/mp4' ></object> |

You should specify the style of the video object. Size and location of the playing video depends on it.

JavaScript:

|  |
| --- |
| var video = documen.getElemenById("video"); |

Now we are ready to use video object. Here is the basic example:

|  |
| --- |
| video.stop();  video.data = "http://itv.ard.de/video/timecode.php/video.mp4";  video.play(1); |

It is just that simple. In this example a MP4 file is played from HTTP source. It also supports many network protocols and media formats. Please read spec document for full list. Here is another example for TS format from RTSP.

|  |
| --- |
| video.data = "rtsp://192.168.1.2:121/aVideo.ts"; |

In play method, a number argument is passed to specify the play speed. In example code, argument is 1 to play in normal speed. This argument is also used for trick modes. If you pass a number bigger than 1, video is moved forward. It is moved rewind for negative numbers. Video is paused for zero. For more details please read the OIPF document. Example:

|  |
| --- |
| video.play(4); //plays x4 speed forward  video.play(-2); //plays x2 speed rewind  video.play(0); //pause |

Video player also keep its state. You can give handler method to the object.

|  |
| --- |
| document.getElementById("video").onPlayStateChange = function () {  // here is your new player state, document.getElementById("video").playState  } |

“playState” property will give us new state. Now you can determine any handler method for player state changes. Play state values and descriptions are listed in the OIPF document.

3.3 Making a basic broadcast page

Previous section we use “video/mp4” type objects. Now we will use “video/broadcast” object for TV page. You can configure DVB-IP, DVB-S, DVB-T, DVB-C and Analog channels to be broadcasted.

3.3.1 Configuring the TVs

First you should serve a XML from your server. This tutorial also includes an example xml file (channel\_list\_example.xml). The format is explained in Vestel Quick Start Guide.

You should give service URL to TV. Make a telnet connection to the TV as root user, password is “mb704725”. After, write your service URL to "/mnt/settings/atrium/atrium\_config.xml" in your TV's file system. Now your TV's video API can reach the channel list.

3.3.2 TV page

If you give the channel list to the TV correctly, you can reach the channels when you call the broadcast object.

HTML:

|  |
| --- |
| <object id='video' type='video/broadcast' style='position: absolute; left: 0px; top: 0px; width: 1280px; height: 720px;'></object> |

JavaScript:

|  |
| --- |
| var myChannelList = document.getElementById("video").getChannelConfig().channelList; |

Now you get channel list as an array. This array contains channels that TV gets from your service URL. A channel object can be an IP channel or analog channel. But there are some abstract properties that cover both. You can find all properties of channel object in the OIPF document.

TV channel can be set from your local array. We should use set channel method for this. Here is function definition:

|  |
| --- |
| void setChannel( Channel channel, Boolean trickplay, String contentAccessDescriptorURL, Integer offset ) |

If you want to have the TV to broadcast first channel of your channel array, here is sample call:

|  |
| --- |
| document.getElementById("video").setChannel(myChannelList[0], false); |

In this approach you gave directly the channel entity to API. If you want to bind the current channel directly you can also call “bindToCurrentChannel” method:

|  |
| --- |
| document.getElementById("video").bindToCurrentChannel(); |

And if you want to change channel you use next and previous channel methods. They change the index of the current channel and bind to current.

|  |
| --- |
| document.getElementById("video").nextChannel(); // increase channel index and bind  document.getElementById("video").prevChannel(); // decrease channel index and bind |

As you can see style is specified for object in given HTML example. Style gives the size and location of the video frame. If you call full screen method with false parameter, it fills the frame. If you call the with true parameter, video is set to full screen but it is only visible from given size. If you play video in small frame as fitted, you should call it with false parameter:

|  |
| --- |
| document.getElementById("video").setFullScreen(false); |

There are several methods and properties of broadcast object, they are declared as standard in the OIPF document.