

Browser IDE Help Guide

Published 2014-10-28 |

Browser IDE is a powerful and flexible online integrated development environment for Smart TV applications on browser

Contents

[Introduction](#)

[Installation](#)

[User Account](#)

[Download for Browser IDE](#)

[Open Browser IDE](#)

[Sign up](#)

[Sign in](#)

[Log out](#)

[Change password](#)

[Project Dashboard](#)

[Create new project](#)

[Start editing project](#)

[Delete project](#)

[Get demonstration of project](#)

[Browser IDE Menu Bar](#)

[File](#)

[Edit](#)

[Selection](#)

[View](#)

[Run](#)

[Help](#)

[Source Editor](#)

[Syntax Highlight](#)

[AutoComplete](#)

[Snippets](#)

[Multi-lines Edit](#)

[Context menu](#)

[Settings](#)

[Project Explorer](#)

[Context menu](#)

[Image Viewer](#)

[Terminal](#)

Summary

Introduction

Browser IDE is a powerful and flexible online integrated development environment for Smart TV applications on browser. There are four main functions, including project manager, code editors, terminal, and running application in Smart TV emulator. Firstly, for project manager, basic information of project, such as public URL, is showed and description of project is available to edit. Secondly, there are three kinds of editors, HTML editor, JavaScript editor, CSS editor. They all provide highlighting syntax, automatic completion and some other powerful features. Thirdly, for terminal, it's more like a gnome terminal. What's more, there's a Smart TV emulator for developers to run application on the browser.

With Browser IDE, you can develop applications anywhere and work more efficiently.

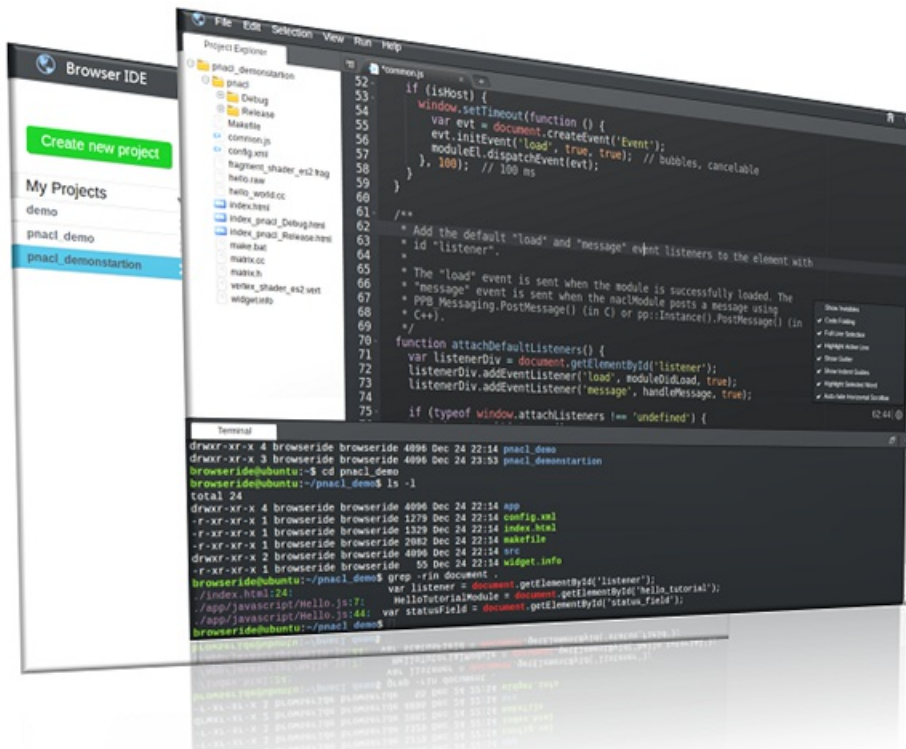


Figure 1: Browser IDE

Installation

The following software is used for this installation:

1. Import SDK BrowserIDE_Server_xxx.ova on your Virtual box

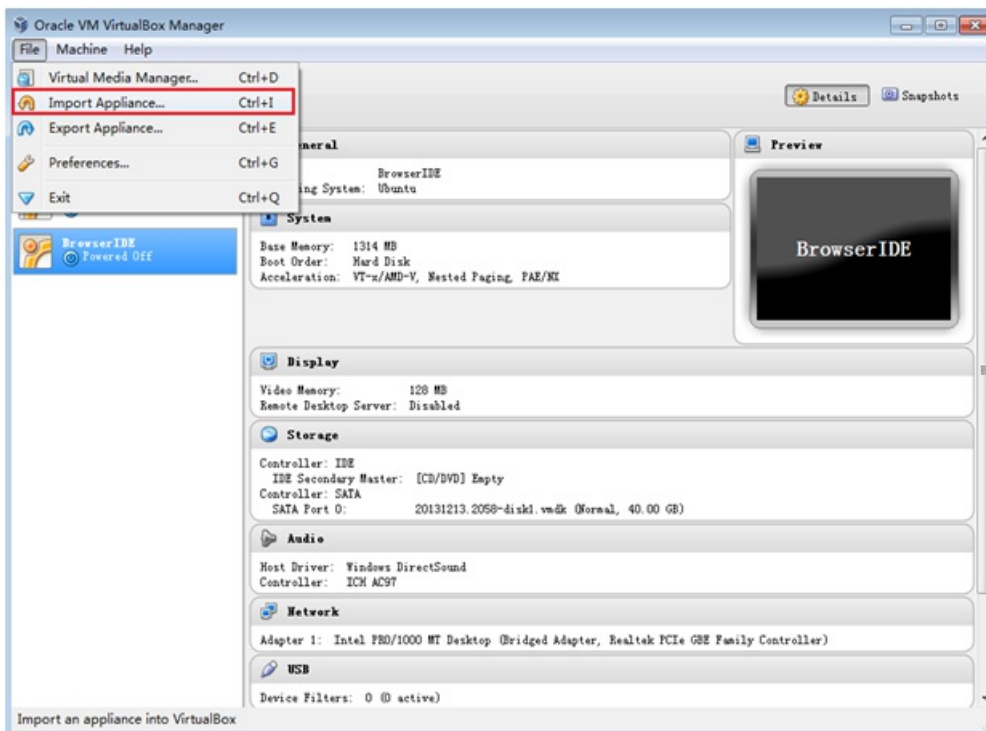


Figure 2: Import ova

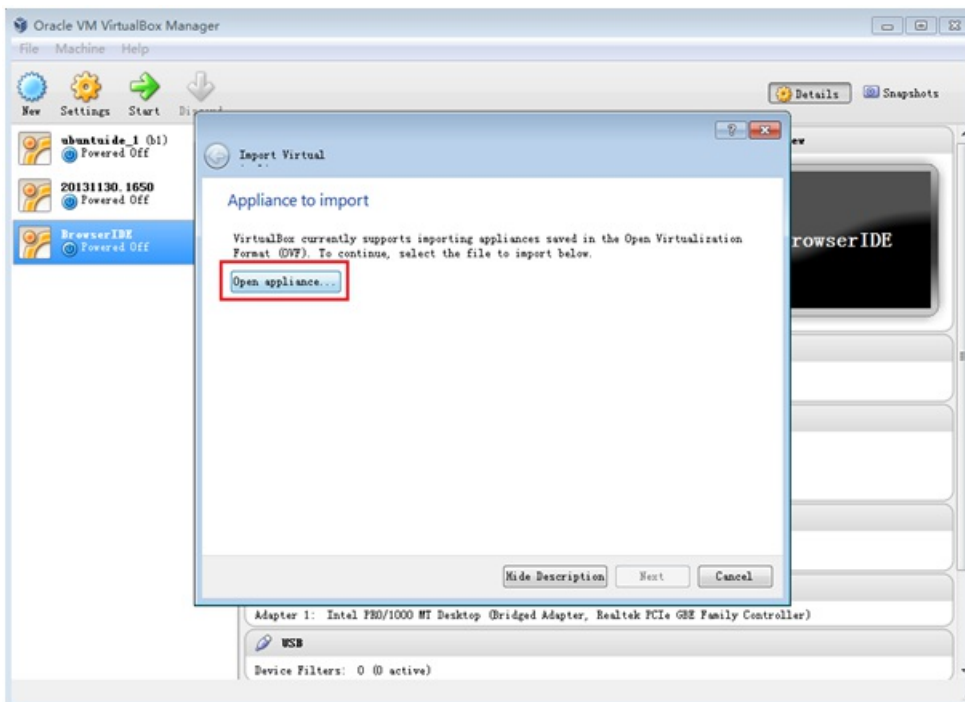


Figure 3: Select ova

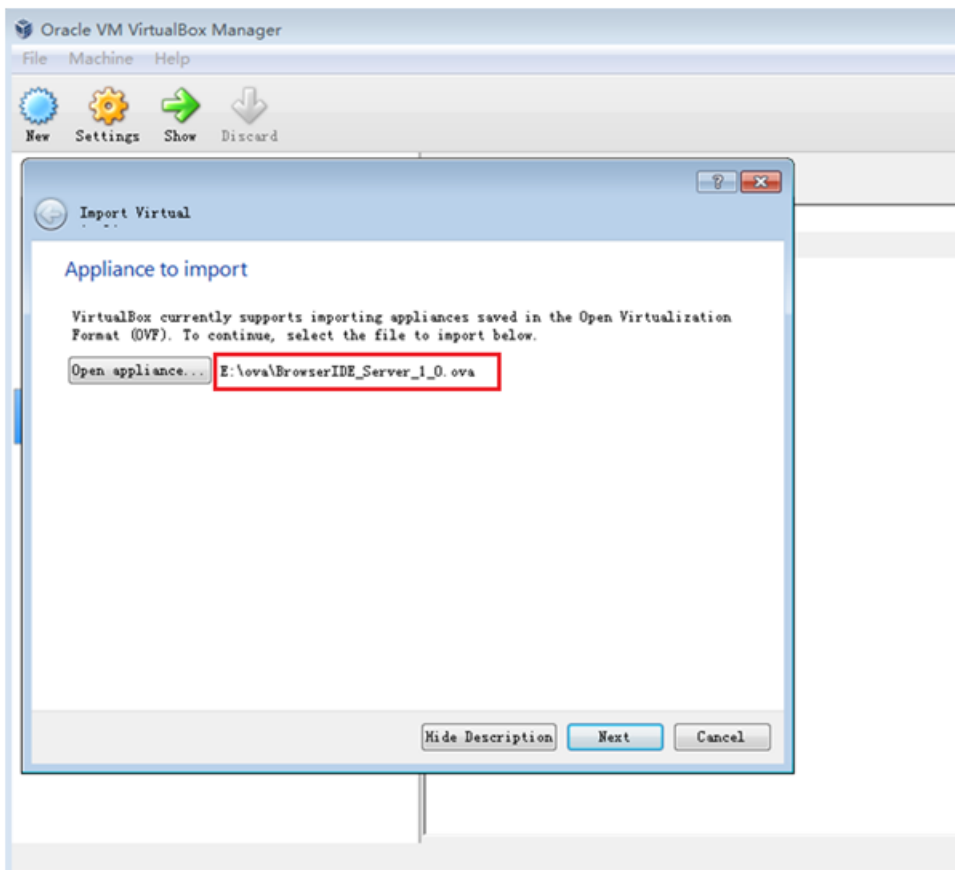


Figure 4: Select ova

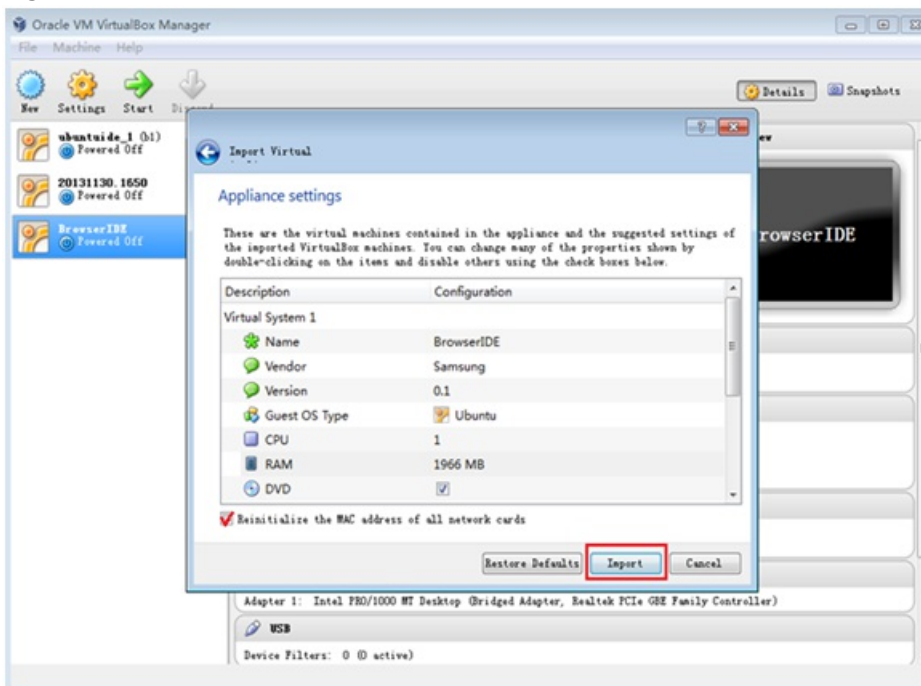


Figure 5: Import ova (reinitialize the MAC)

2. Configure Virtual box

2.1

System configuration

Base Memory: ≥ 1024 MB

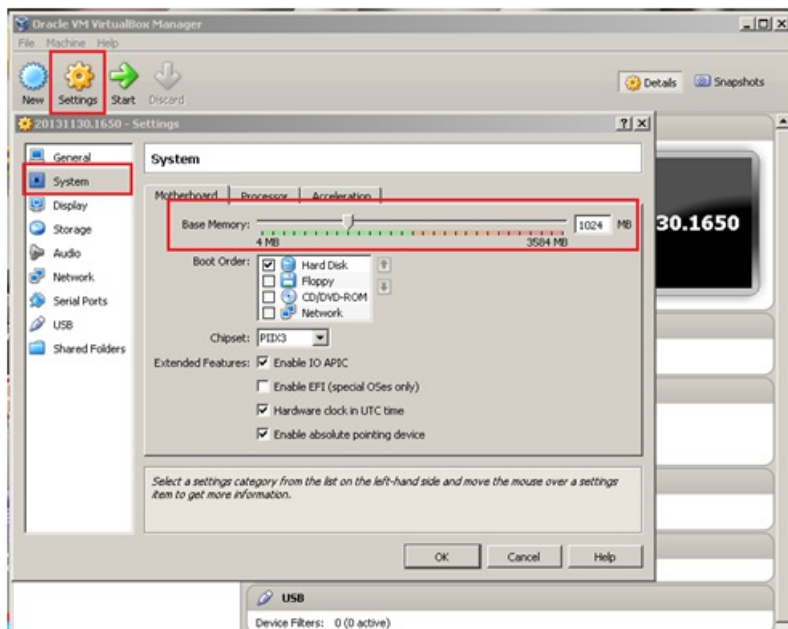


Figure 6: System configuration

2.2

Network configuration

Support Bridged connection method (NAT method not supported)

MAC address should be set if it's binding with IP address

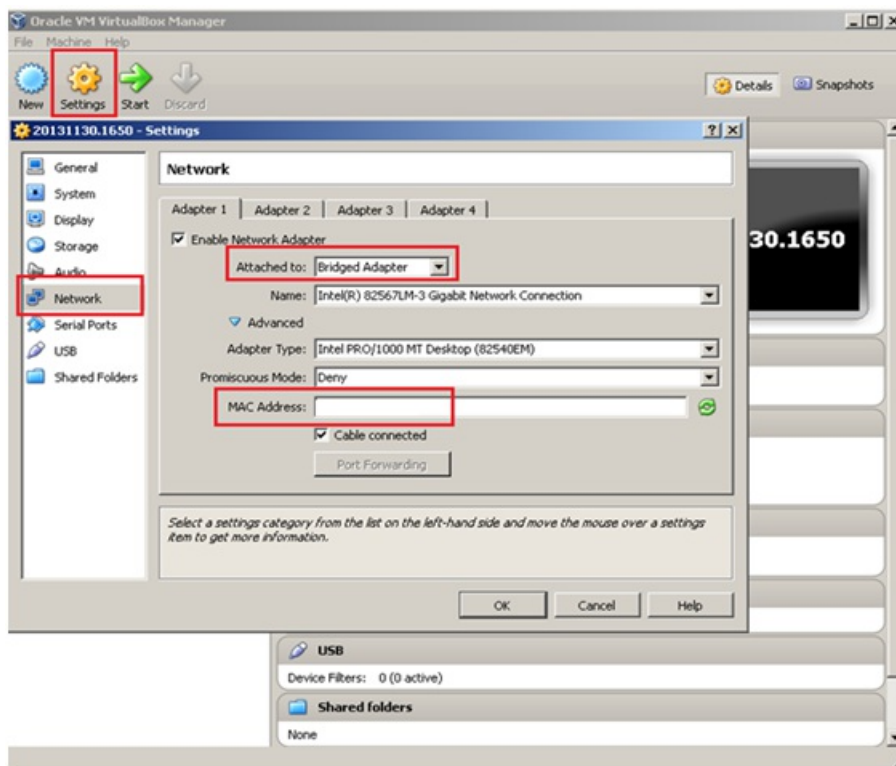


Figure 7: Network configuration

After network configuration, then start the Browser Server, system will show IP Setting page of Browser IDE.

Figure 8: Ip setting

Input IP Address and DNS Server and click confirm button.

Figure 9: Reboot server question

Recommend select OK to reboot Browser Server.

User Account

Each user has own account to facilitate managing their projects or files.

Download for Browser IDE

Before installing Browser IDE, VirtualBox must be installed first. To access Admin Tools Module in Browser IDE, Chrome or Firefox must be installed. Download "BrowserIDE_Server_1_0.ova" inside of the package named, "Browser IDE Server Image for Virtual Box" from <https://developer.samsung.com/smarttv/develop/extension-libraries/smart-view-sdk/download.html>.

Open Browser IDE

Please use chrome to visit the Admin Tools module. The available address will be "http://NetworkIP".

Note

Please use the network IP which you have set. If you don't remember, please login to Admin Tools module inside VirtualBox and see your current IP at "Network"->"IP Setting" page.

Sign up

To use Browser IDE, a user account is needed. First, open homepage, click the Signup button at the top-right corner and users will be taken to signup page. All information fields are required.

For these information input fields, there're some notices as follows:

1. User name: only contains letters and numbers, and the length is from 6 to 20.
2. Password: only contains letters and numbers, and the length is from 6 to 20.
3. Secret question: only contains letters and numbers, and the length is from 4 to 20.
4. User name: only contains letters and numbers, and the length is from 4 to 20.

Sign in

If you already have a Browser IDE account, just go to homepage and sign in. There's a keep user name checkbox, check it and user name will be kept in input box when you login next time with this browser.

A screenshot of a sign-in form. It features two input fields: "Enter the user name" and "Enter the password". To the right of these fields are two buttons: "Sign in" (blue) and "Sign up" (orange). Below the "Enter the user name" field is a checkbox labeled "Keep user name" which is checked. To the right of the checkbox is a link labeled "Forget password?".

Figure 10: Sign in

Log out

There're three methods to sign out, which are as follows:

1. Click log out button in homepage
2. Click log out submenu of user menu at the top-right corner of workspace page.
3. Click log out submenu of setting menu at the top-right corner of main page.

Change password

If you forget your password or want to modify it, click forget password link in homepage and users will be taken to find password page.

A screenshot of a form for finding a password. It features two input fields: "Enter the user name" and "Enter the password". To the right of these fields are two buttons: "Sign in" (blue) and "Sign up" (orange). Below the "Enter the user name" field is a checkbox labeled "Keep user name" which is checked. To the right of the checkbox is a link labeled "Forget password?" which is highlighted with a red rectangle.

Figure 11: Forget password

The process of retrieving password is as follows:

1. Input user name to get secret question.
2. Input secret answer, check it.
3. Reset user's password.

Project Dashboard

The project dashboard provides basic operations to master the project.

Create new project

After clicking on "Create new project", users are taken to the dialog which to choose the project type.

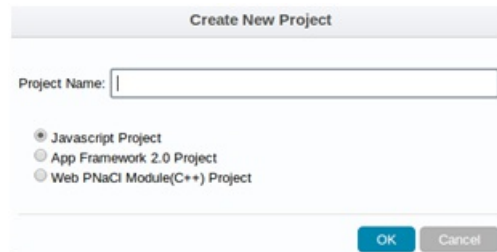
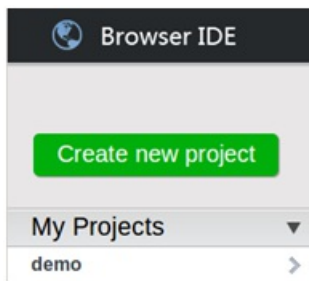


Figure 12: Create new project **Figure 13:** Choose the project type
Here's the introduction to different project types available on Browser IDE.

1. JavaScript Project

The JavaScript APIs provide low-level platform functions that give users greater control over application tasks and processes.

2. App Framework 2.0 Project

Apps Framework 2.0 is developed for creating web based application for Samsung Smart TV easily. Users can create Smart TV app with less hours, and can concentrate on improving their business logic and services.

3. Web PNaCl Module(C++) Project

Create fast and powerful application using C/C++ with support for Google's Portable Native Client (PNaCl) feature. PNaCl applications can be developed in the IDE and then executed in the Smart TV Emulator.

Fill input the project name and select the project type then press OK, after the project successful created, it will be added into the "My Projects" list.

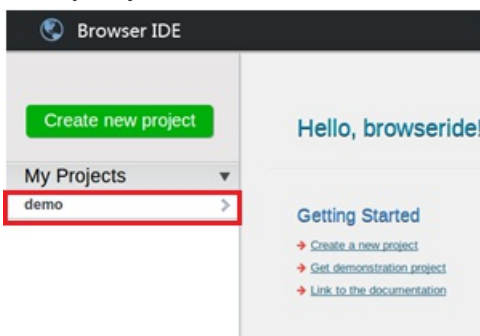


Figure 14: My Project

You can see the detail information at the right part of dashboard after you clicking it in "My Projects" list. You can also update the description of project through the "Project Description" text box.

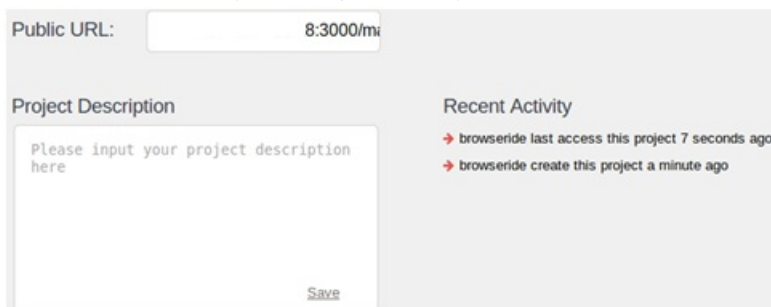


Figure 15: Information of project
Start editing project

Start editing by Browser IDE via click the "Start Editing".



Figure 16: Start Editing

Delete project

You can delete the project through clicking the “Delete” button which in the far right side of the dashboard.

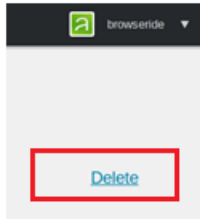


Figure 17: Delete

Get demonstration of project

Browser IDE also provides the demonstration projects since it’s not easy to develop the Smart TV application as a freshman. You will know how to manage the focus of user input, showing and hiding each DIVs and event handlers via reading the source code of demonstration projects and take it as a reference. Click the “My Projects” and look at the right side of the list.

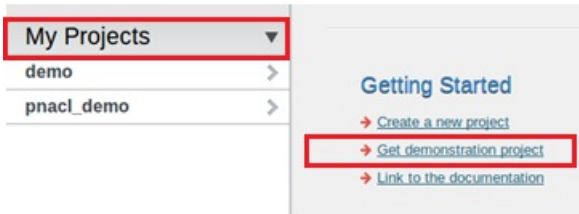


Figure 18: Demonstration project

User could choose the project type among the projects encountered.

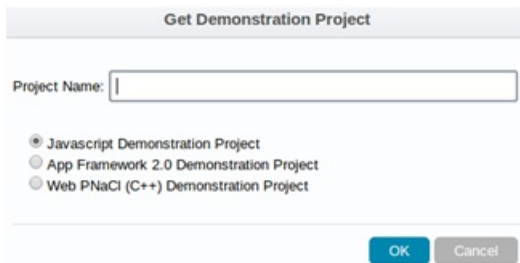


Figure 19: Choose the type of demonstration project

Browser IDE Menu Bar

File

The **File** menu can operate file and folder, include **New File**, **New Folder**, **Save**, **Save As**, **Save All**, **Open**, **Close**, **Close All**, **Quit Project**.

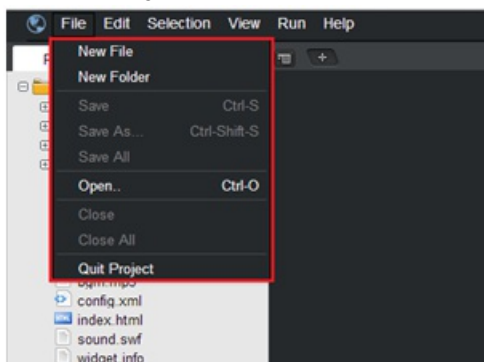


Figure 20: File

The **New File** can create a new file.

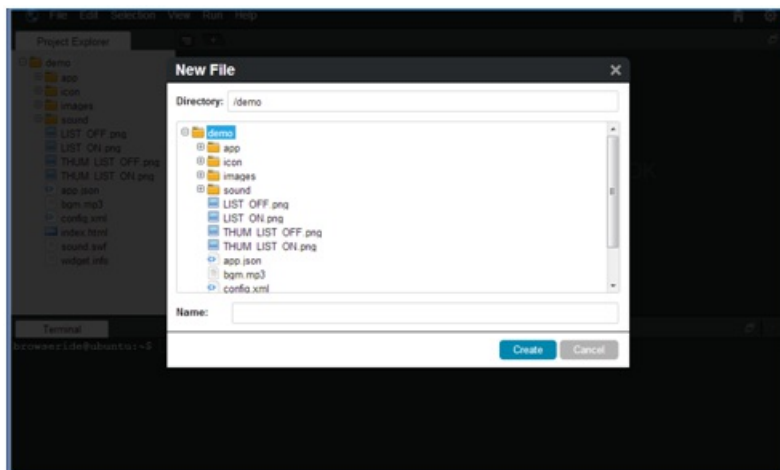


Figure 21: New File

The **New Folder** can create a new file.

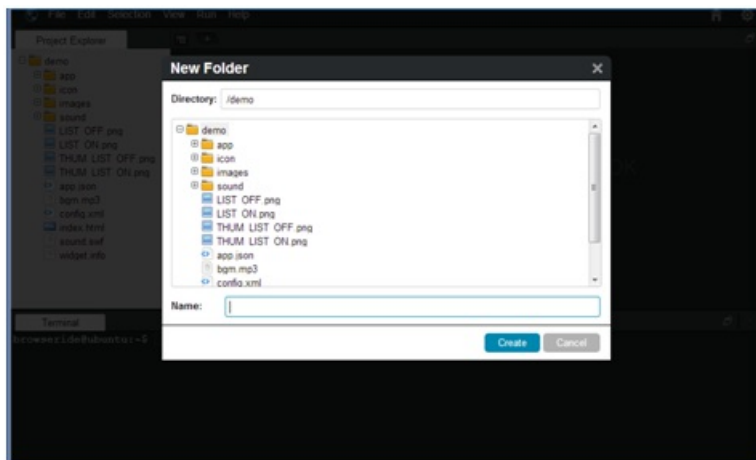


Figure 22: New Folder

The **Open File** can open a file in the source editor:

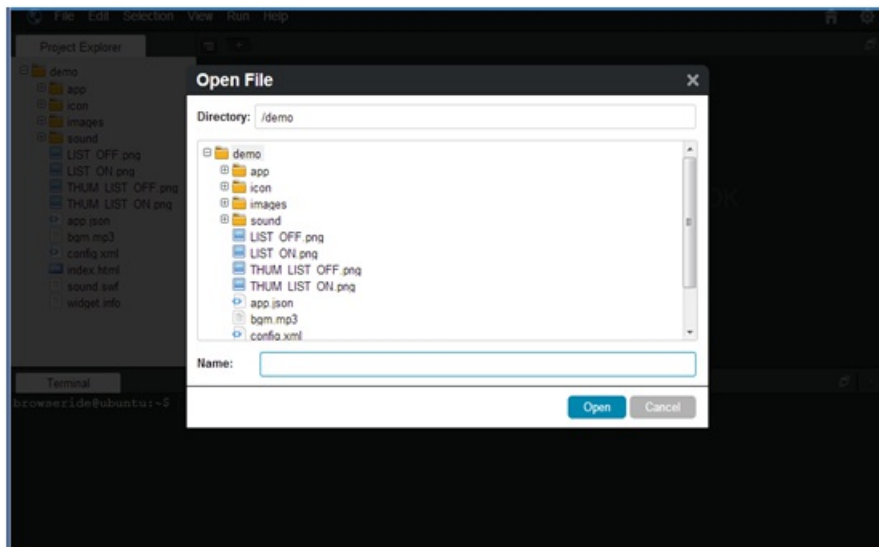


Figure 23: Open File

The **Save File** can save a file:

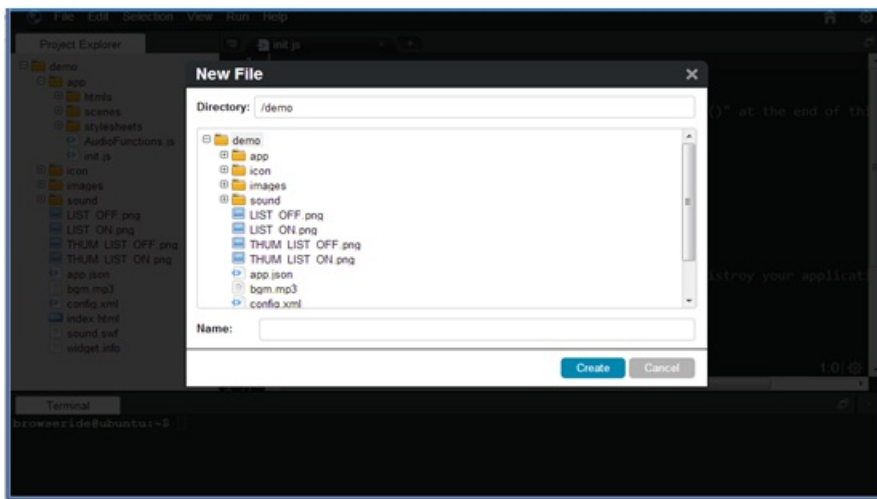


Figure 24: Save File

Edit

The **Edit** submenu in menu bar contains all the commands corresponding to editing. Most of the commands are used to manipulate Source Editor. For example, **redo/undo** can, **copy**, **cut**, **paste**, **delete**, these are regular operations.

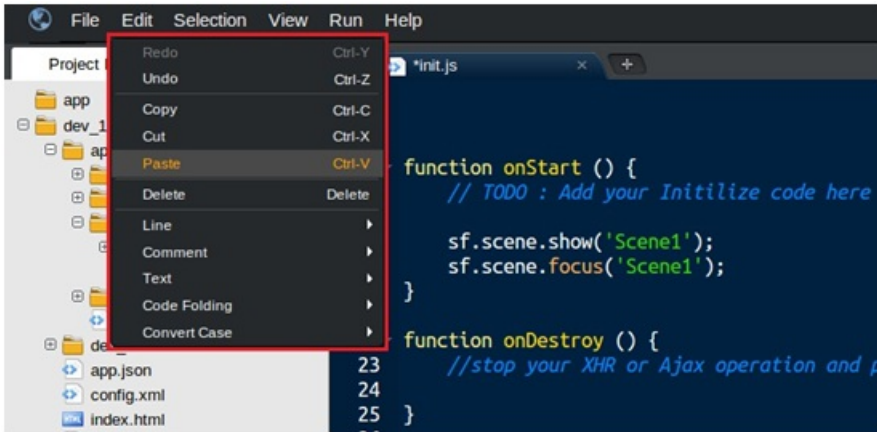


Figure 25: Edit Menu

The **Line** submenu provides line editing commands:

1. **indent** and **outdent**;
2. **move line up** and **move line down** Cut the whole line and paste upward or downward;
3. **copy lines up** and **copy lines down** Copy the whole line and paste upward or downward;
4. **remove line**, **remove to line end** and **remove to line start**, **split line** Break current line into two lines at the cursor.

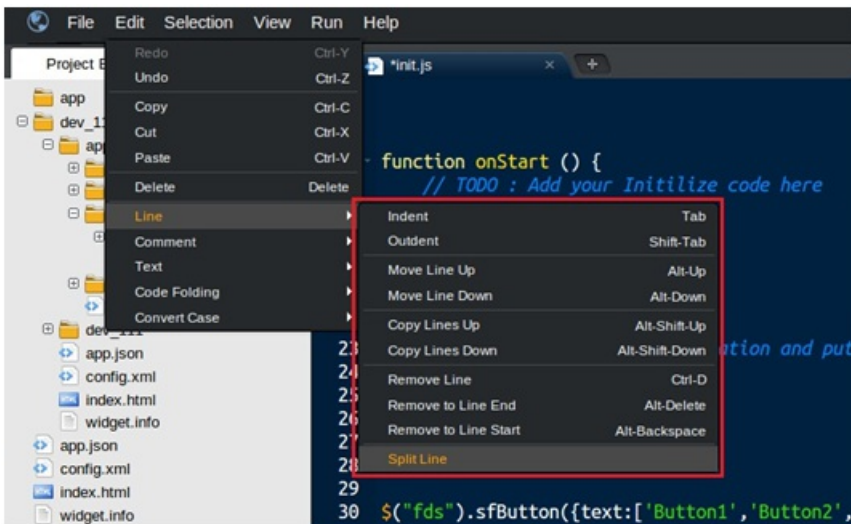


Figure 26: Line submenu in Edit Menu

Other options are used to assist coding. **Comment** commands provide two style of **comment** and **uncomment**: Convert selected text into commented or uncommented.

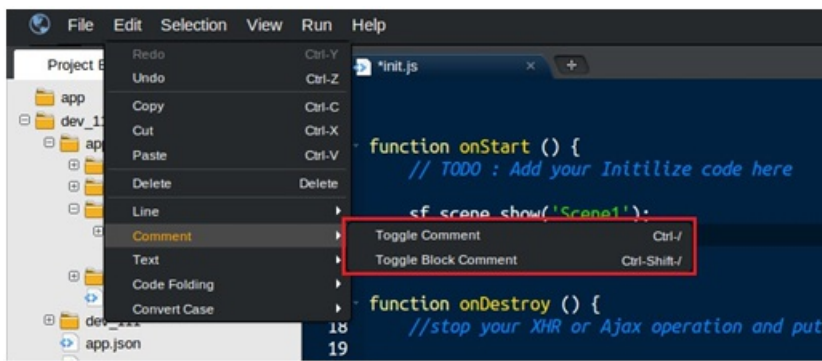


Figure 27: Comment submenu in Edit Menu

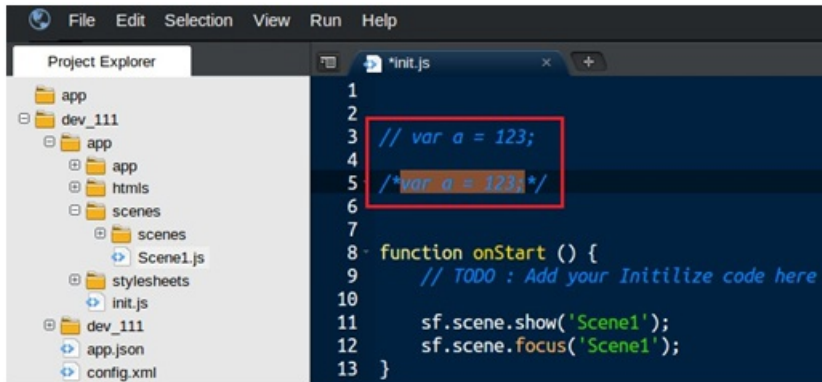


Figure 28: Comment submenu in Edit Menu

Developer can use **Remove Word Left/Right** commands to delete text word by word and use **Increase/Decrease Number at Cursor** to make small adjustment for the number at caret.

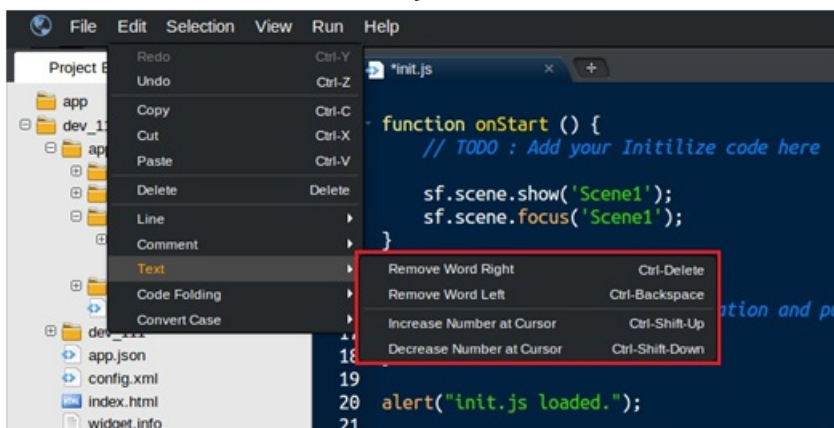


Figure 29: Text submenu in Edit Menu

For **Fold/Unfold** and **Fold/Unfold All**, these commands are used to collapse or expand code block in Source Editor.

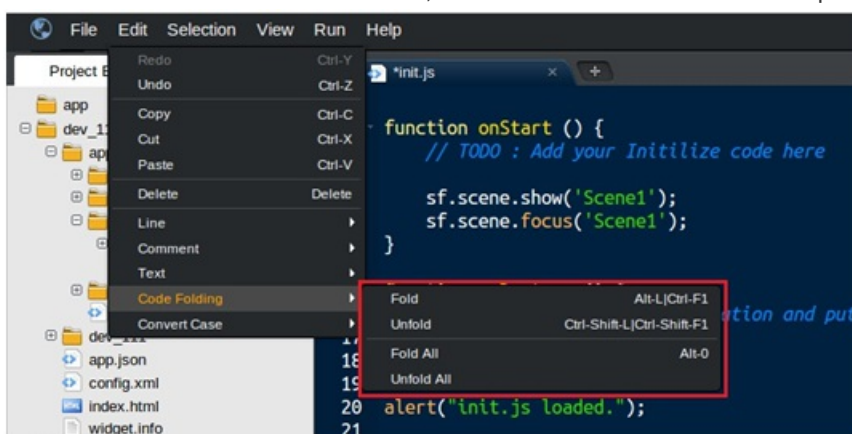


Figure 30: Folding submenu in Edit Menu

```

3 // var a = 123;
4
5 /*var a = 123;*/
6
7
8 function onStart () { }
9
10 function onDestroy () { }
11
12
13 alert("init.js loaded.");
14
15
16 $("fds").sfButton({text:['Button1','Button2','Button3']});
17
18
19
20
21
22
23
24

```

Figure 31: Folding code

Upper/Lower Case can transform the selected text to the upper or lower case.

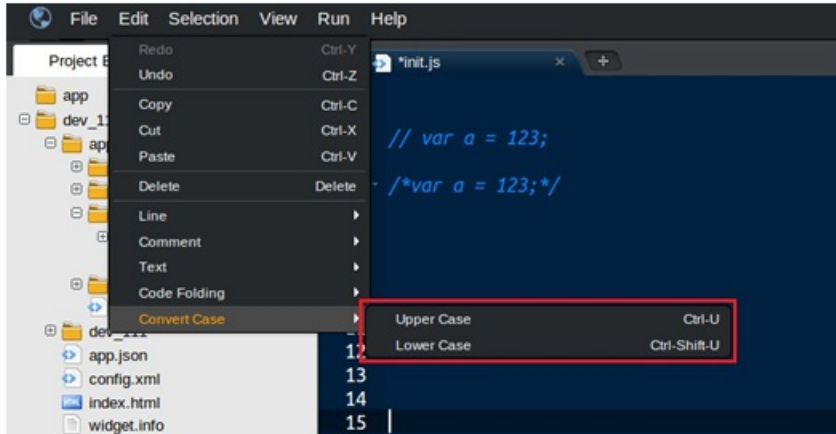


Figure 32: Convert case

Selection

The **Selection** submenu in main menu bar contains all the selection related commands. All these commands are also used to manipulate selections in Source Editor.

Select All will make all the text in Source Editor selected.

Select Word Left/Right commands can select text word by word.

Select to Line Start/End commands will select text from current caret to line start/end.

Select to Document Start/End commands will select text from current caret to Start/End of the file.

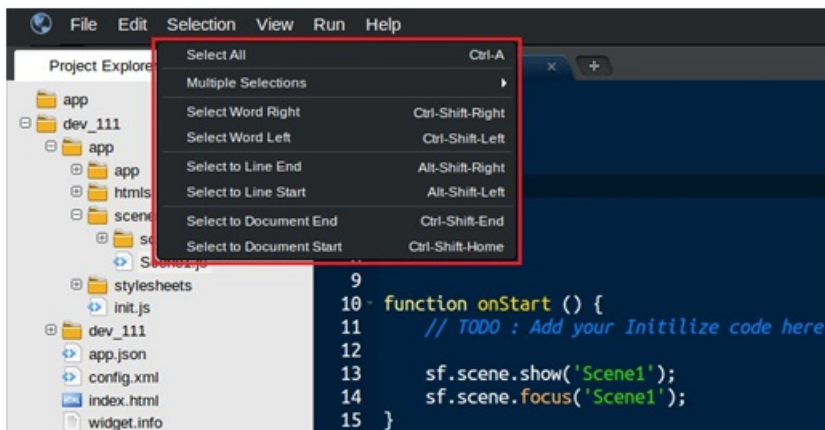


Figure 33: Selection Menu

In **Multiple Selections** submenu, there are six commands:

Add Cursor Up/Down which adds one more caret in Source Editor at the line above or below;

Move Active Cursor Up/Down which moves the shining cursor line up/down;

Add Next/Previous Selection Match which adds one more caret at the selection matches search string.

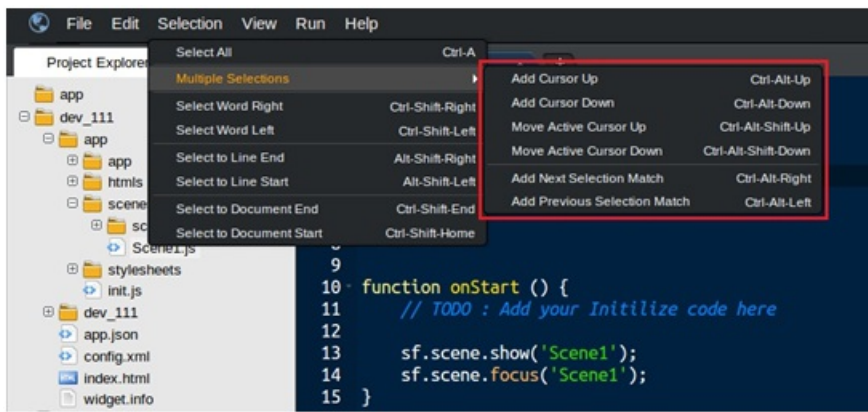


Figure 34: Multiple Selections submenu in Selection Menu
View

The view menu provides some useful functions for user to operate or set the IDE conveniently. When user doesn't open a text editor, then some functions in the view menu are not available. As shown in the figure below:

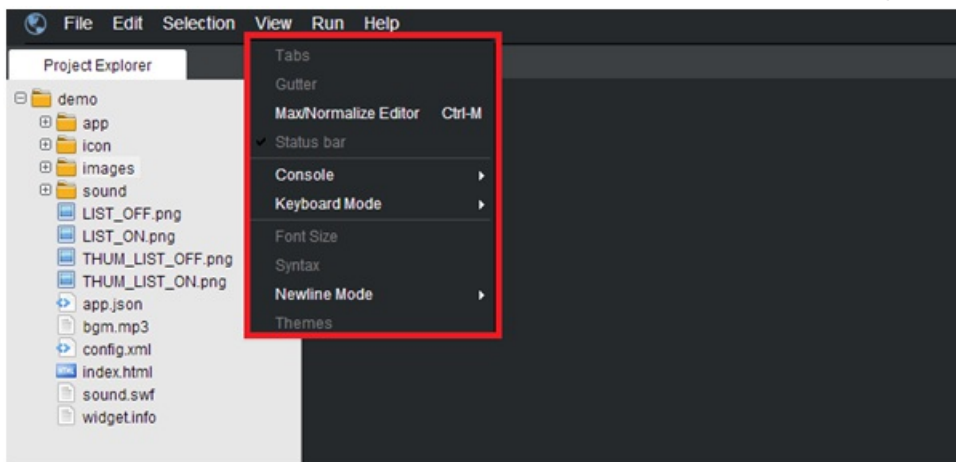


Figure 35: View Menu With disabled items

And if the user has opened a text editor, then the disabled functions in the view menu are available.

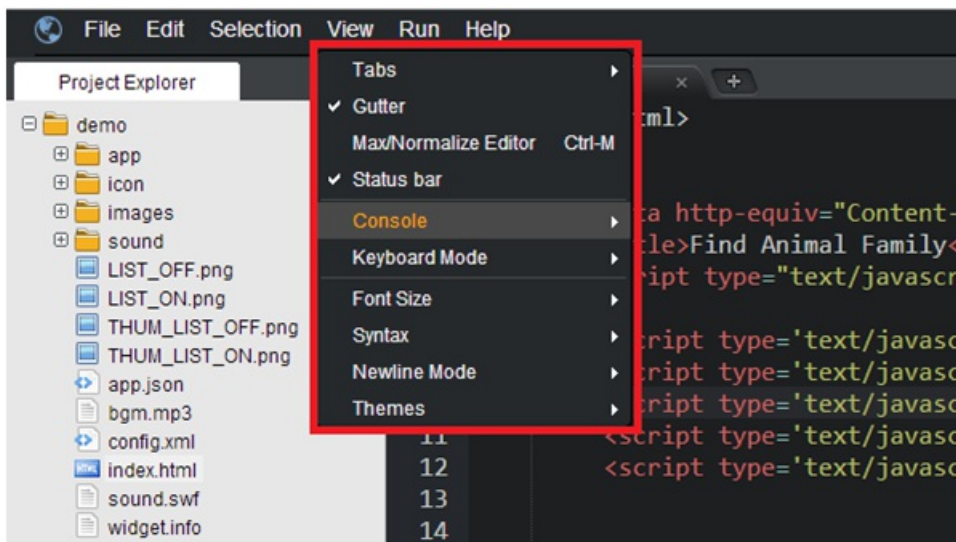


Figure 36: View Menu with enable items

1. **Tabs** menu provides user to operate opened files in editor, user can use **Close Tab** menu to close current editing file, and use **Close All Tabs** to close all opened files in editor, and use **Close Others** to close files opened in editor except current editing.

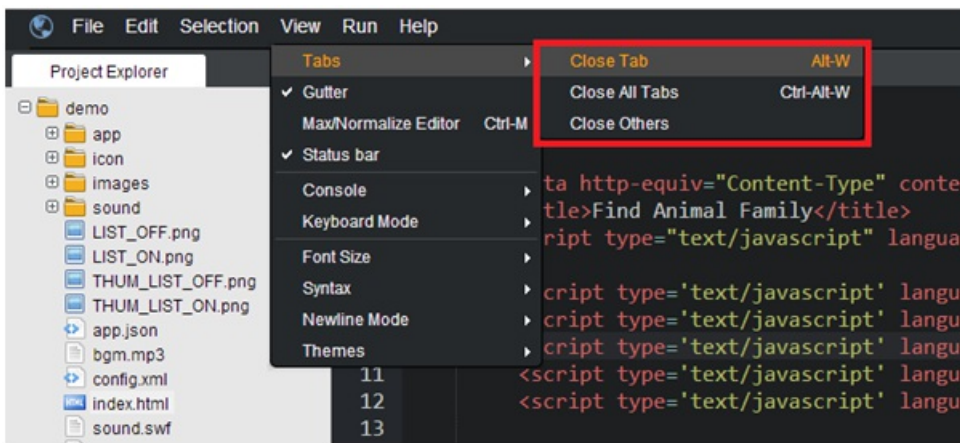


Figure 37: Tabs Menu and with three items

- When user click tab button shown as below picture, it will show menus, tabs button also provides user to operate opened files conveniently as **Tabs** menu, but it has a different function, it can show all opened files in editor, if user opened too many files to watch them, then this function provides a good way to watch them.

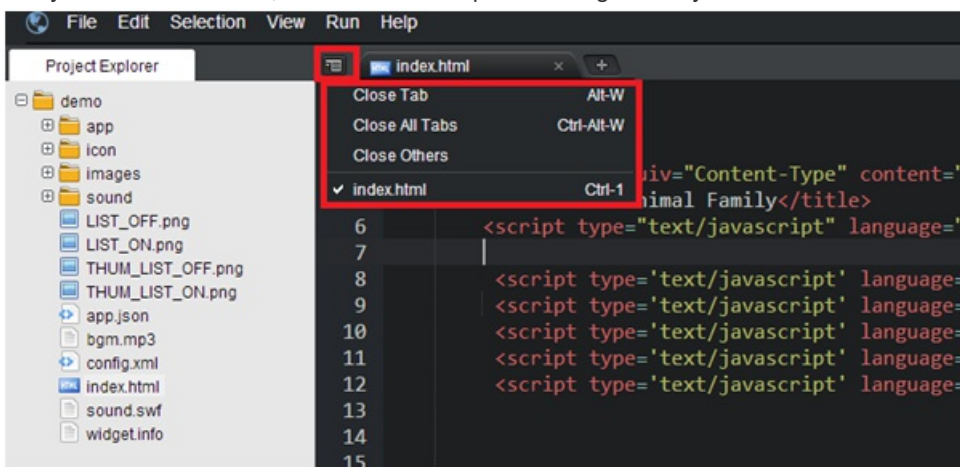


Figure 38: Tabs button with opened files.

- When user put the cursor on the tab page of opened files, and right click, it will show Tabs context menu, it also provides user to operate opened files conveniently as **Tabs** menu, it also has more functions than Tabs menu, it has **Close Tabs to the Right** and **Close Tabs to the Left** two items, with this two functions, user can close every direction of files opened in the editor.

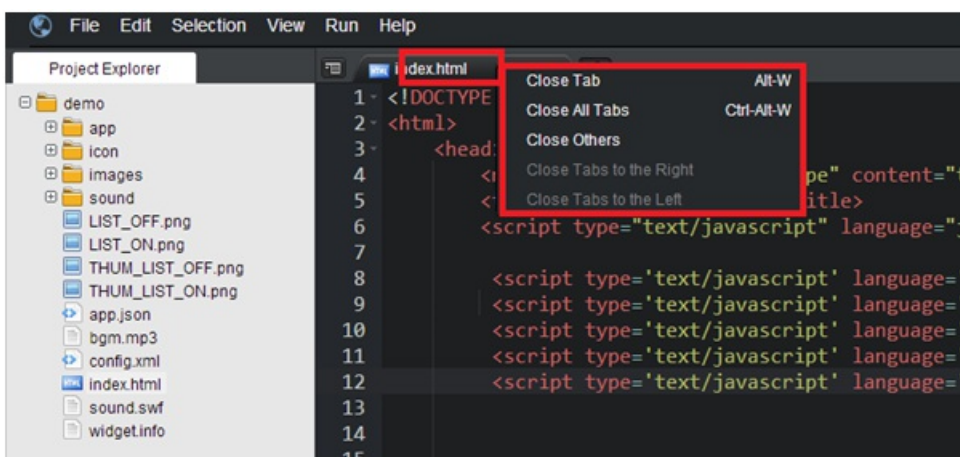


Figure 39: Context menu of tab page with close menus.

- Gutter** menu provides user to set the left side of editor's gutter bar to show or hide, if it hides, user can't see the line number and fold widgets of editor.

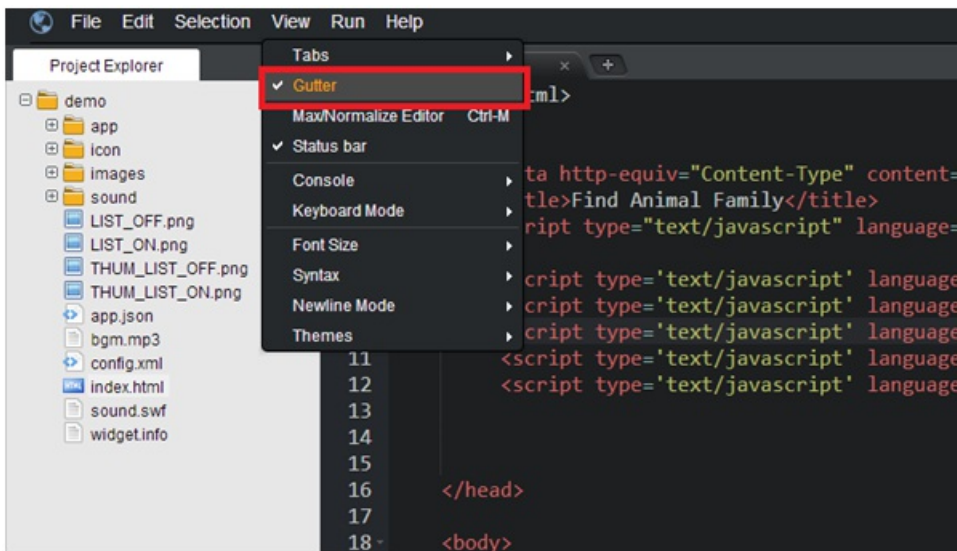


Figure 40: Gutter with gutter in the left side
When user clicks the gutter to hide the gutter, it hides.

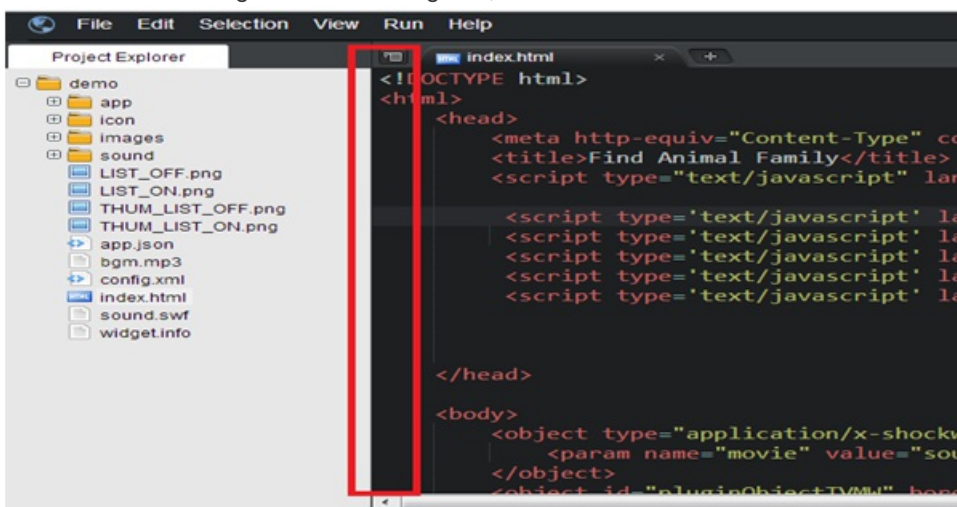


Figure 41: without gutter of editor

5. **Max/Normalize Editor** provides user to set the editor max or normal , it's useful for user to edit when user feel the editor is so small, it makes the editor full screen of browser.

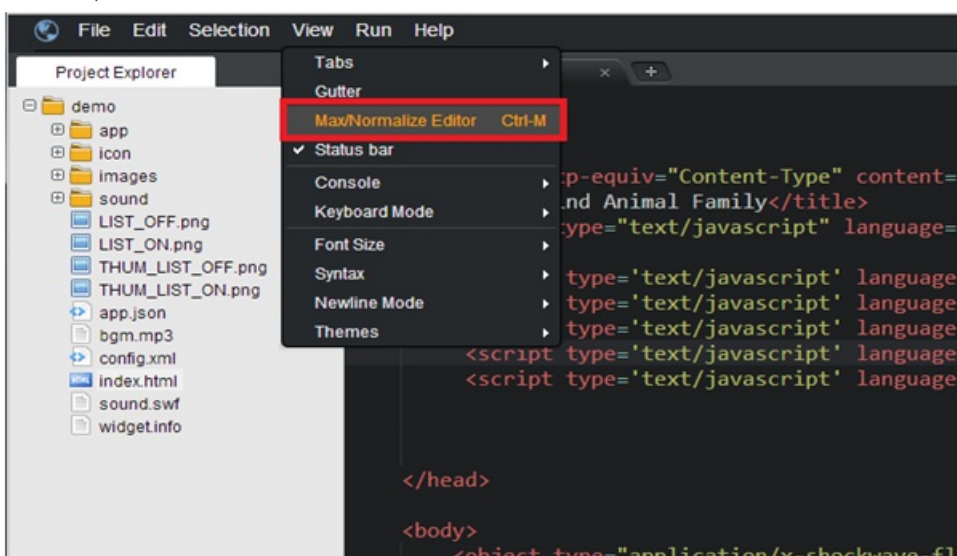
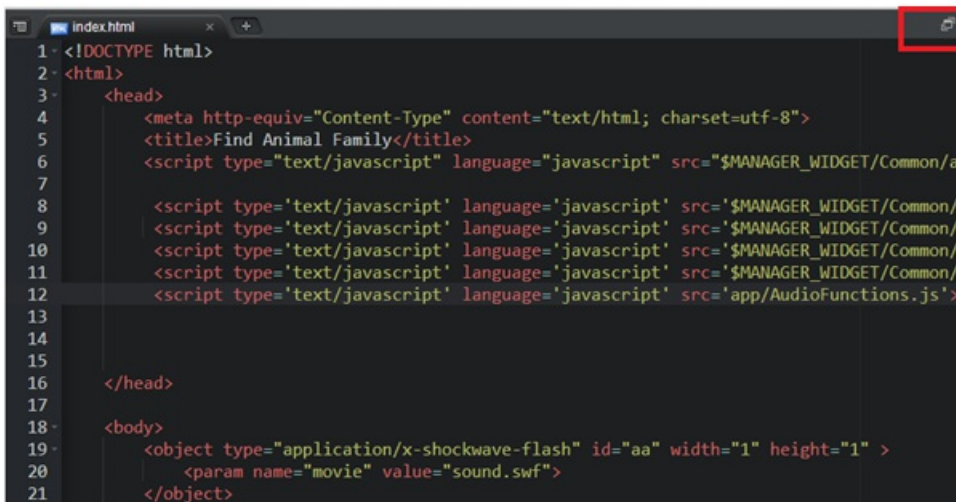


Figure 42: Max/Normalize Editor with normalized editor

When user clicks the Max/Normalize Editor menu, the editor will be full screen, if user clicks the menu again, then the editor would be normalized, and user can click the button in the right side of editor or use hotkey of Ctrl-M to make the editor normalize.



```
1- <!DOCTYPE html>
2- <html>
3-   <head>
4-     <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
5-     <title>Find Animal Family</title>
6-     <script type="text/javascript" language="javascript" src="$MANAGER_WIDGET/Common/a
7-
8-     <script type='text/javascript' language='javascript' src='$MANAGER_WIDGET/Common/A
9-     <script type='text/javascript' language='javascript' src='$MANAGER_WIDGET/Common/A
10-    <script type='text/javascript' language='javascript' src='$MANAGER_WIDGET/Common/A
11-    <script type='text/javascript' language='javascript' src='$MANAGER_WIDGET/Common/D
12-    <script type='text/javascript' language='javascript' src='app/AudioFunctions.js'></
13-
14-
15-
16-   </head>
17-
18-   <body>
19-     <object type="application/x-shockwave-flash" id="aa" width="1" height="1" >
20-       <param name="movie" value="sound.swf">
21-     </object>
```

Figure 43: Full screen of editor

6. **Status bar** menu provides user to set the editor's status, such as **Show Gutter**, **Highlight Selected Word**, they are all useful for user editing the file.

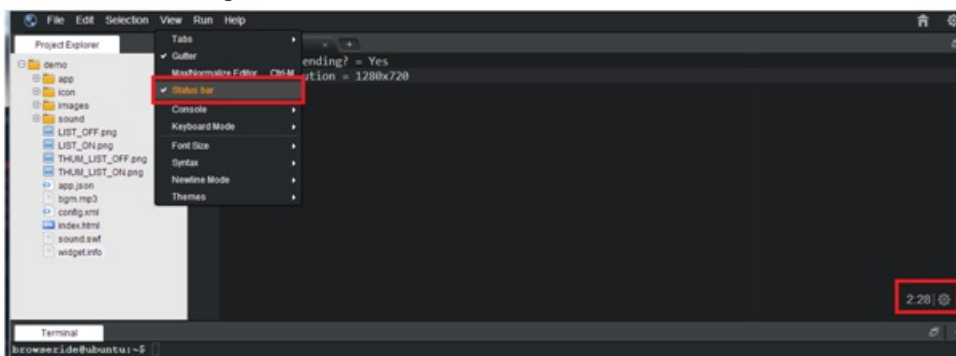


Figure 44: Status bar menu with status bar in editor

When user clicks the **Status bar** menu, then the status bar will hide, if click again, it will show.

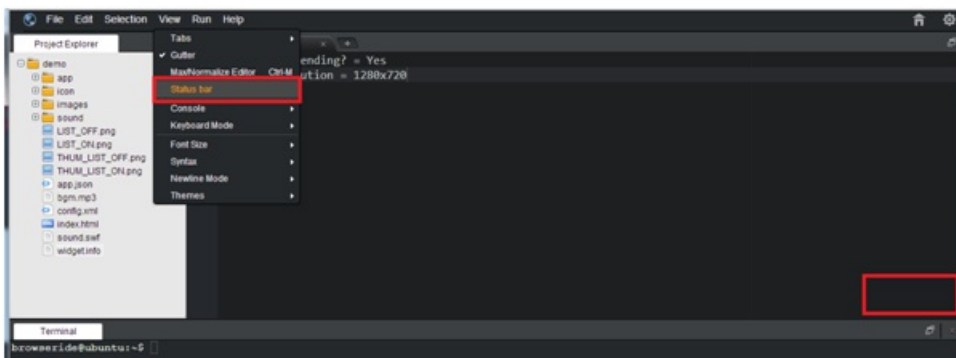


Figure 45: Status bar menu without status bar in editor

7. **Console** menu provides user to operate console of Browser IDE, user can click **Show/Hide Console** or press hotkey of F6 to make the console show or hide, and user also can click **Max/Normalize Console** to make the console view full screen of browser.

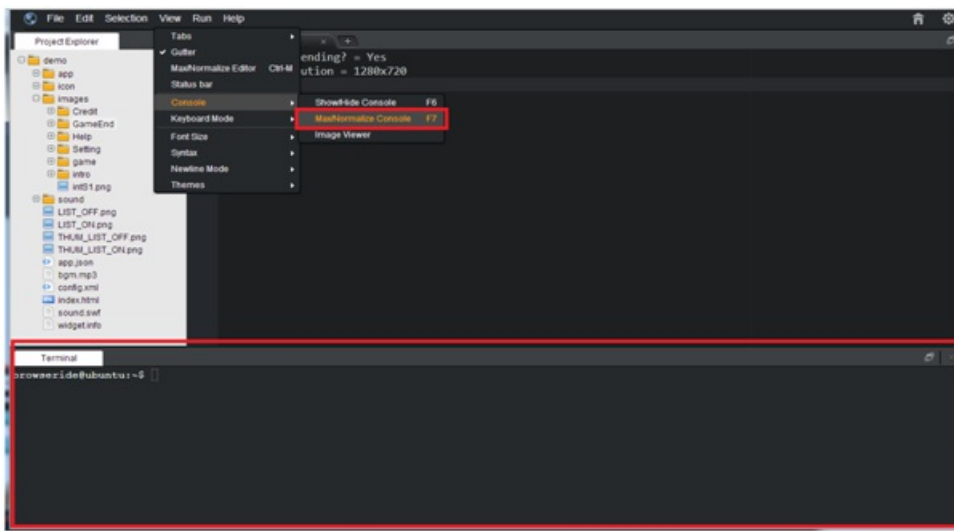


Figure 46: Console view in normal size

When user clicks **Max/Normalize Console**, the console will full screen, it's useful for user to type in commands with terminal.

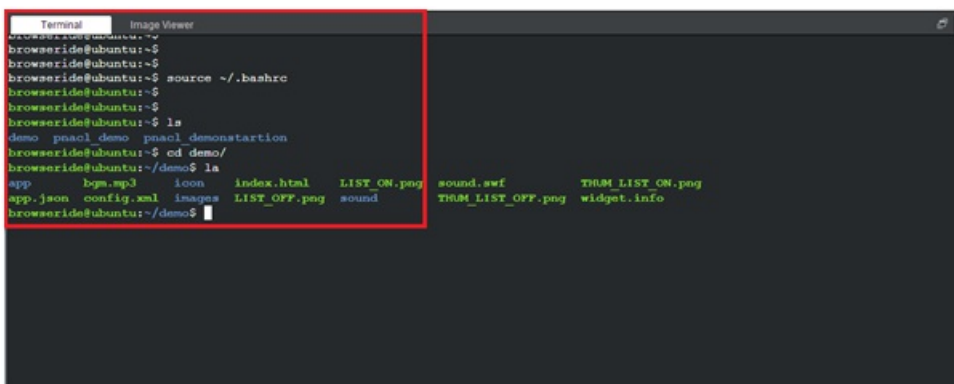


Figure 47: Full screen console view

When user clicks the **Image Viewer** of **Console**, then a view will be added in the console view, user can use **Image Viewer** to watch a lot of images together.

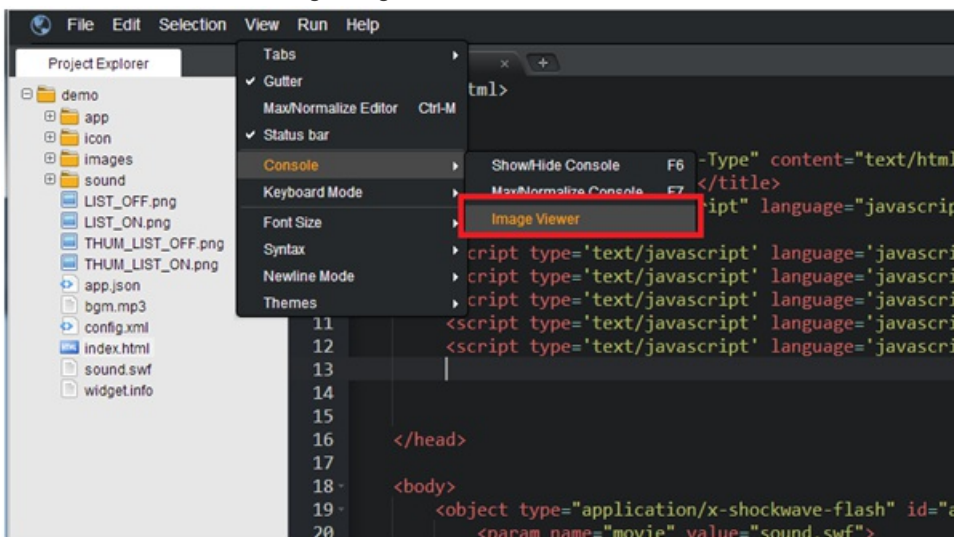


Figure 48: Image Viewer Menus in Console Menu

If user has too many images in **Project Explorer**, user can right click the images fold and click **Show Images**, then the images will show in the image viewer.

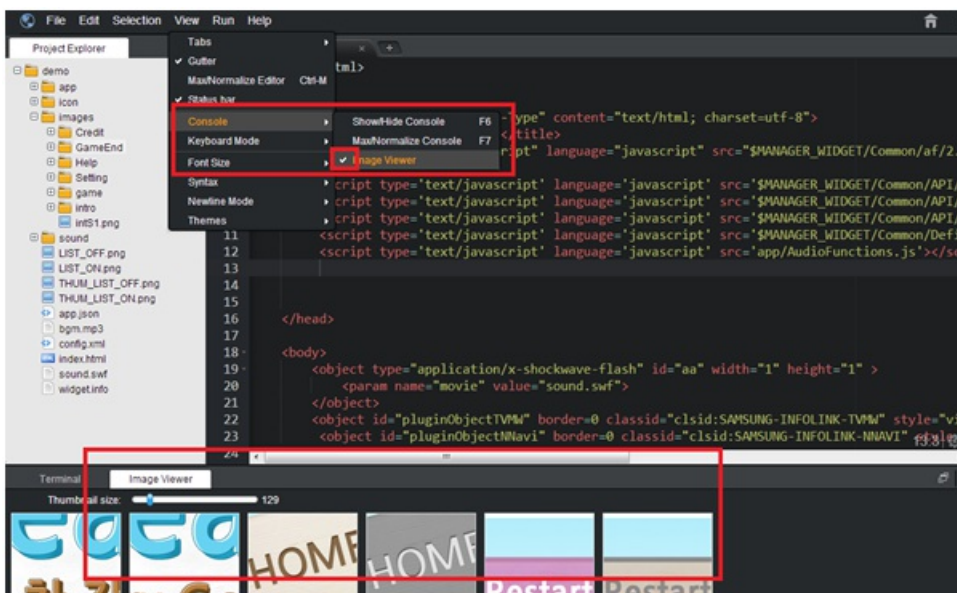


Figure 49: Image Viewer with images

8. **Keyboard Mode** menu provides user to edit the file uses Vim or Emacs editor; it's convenient for Linux user.

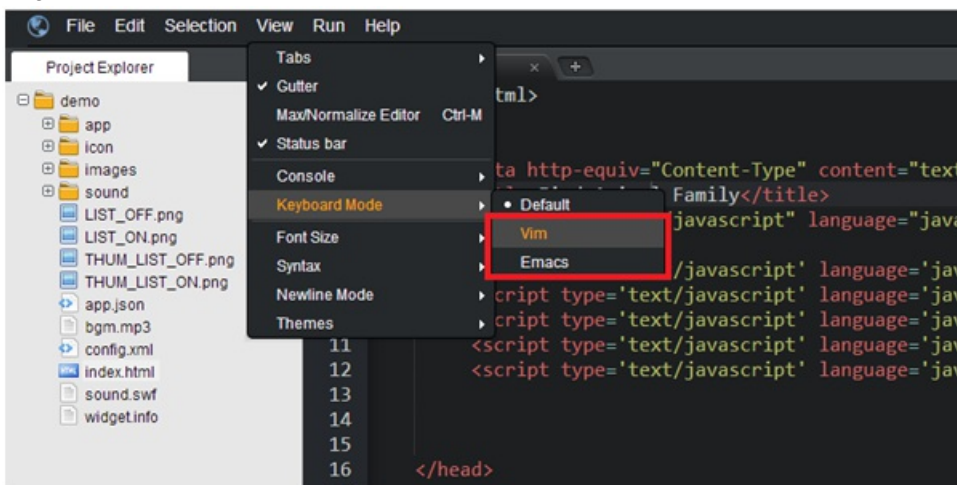


Figure 50: Keyboard mode with vim or emacs

9. **Font Size** menu provides user to change the font size of content in the editor; user can increase or decrease the font size through menu or hotkey.

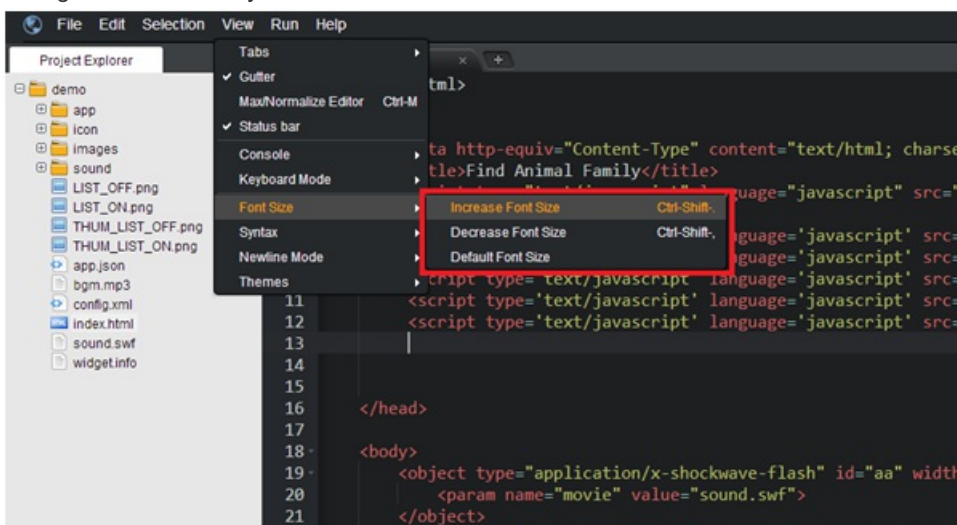


Figure 51: Change font size of editor

10. **Syntax** menu provides user to set or get syntax of current file opened in the editor, and the code in the editor will be highlight according to its syntax.

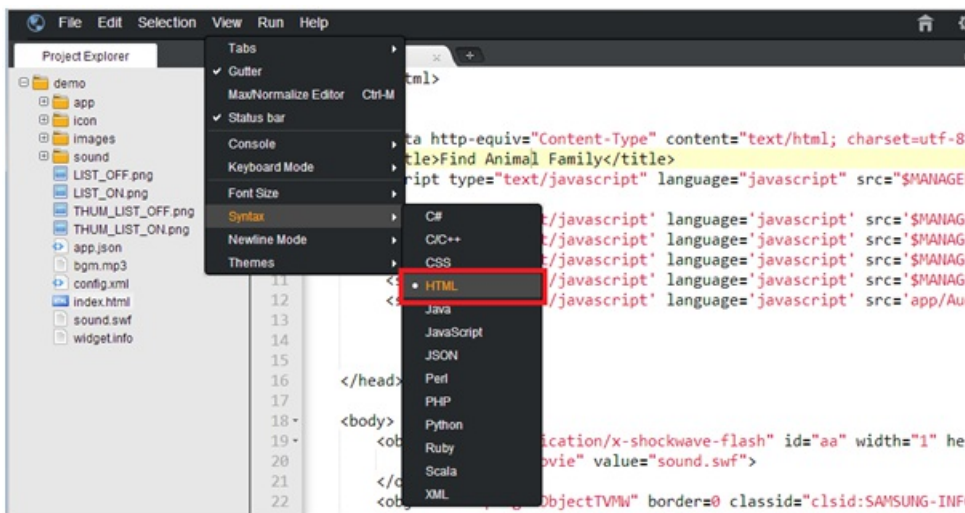


Figure 52: Syntax highlight

11. **Newline Mode** menu provides user to set the editor mode of windows or Linux, the difference between windows and Linux is that windows OS contains CRLF (Carriage-Return Line-Feed), but Linux doesn't have CR, so different mode has different effects.

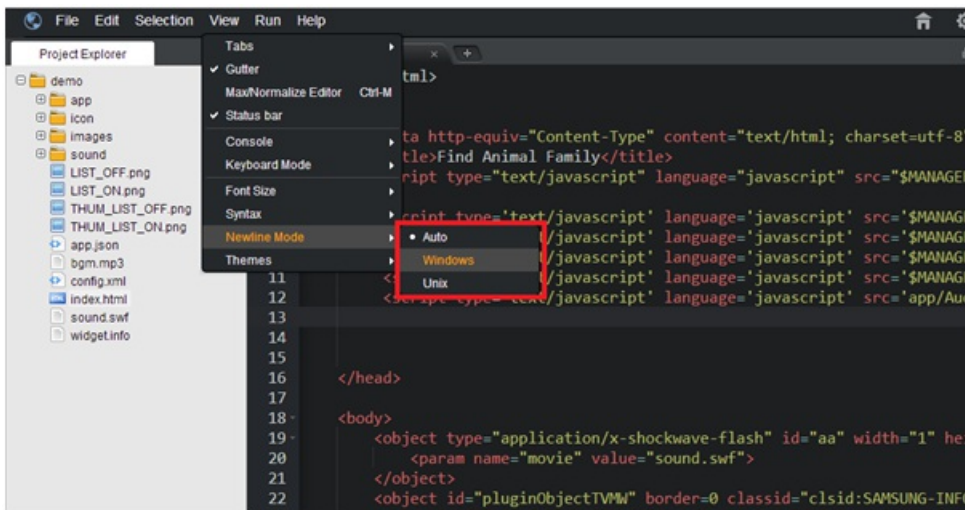


Figure 53: Newline Mode of code setting

12. **Themes** menu provides user to set the editor's theme, user can define his own style whatever he wants according to his habits, and the default theme of editor is Tomorrow Night, user can choose a theme he likes.

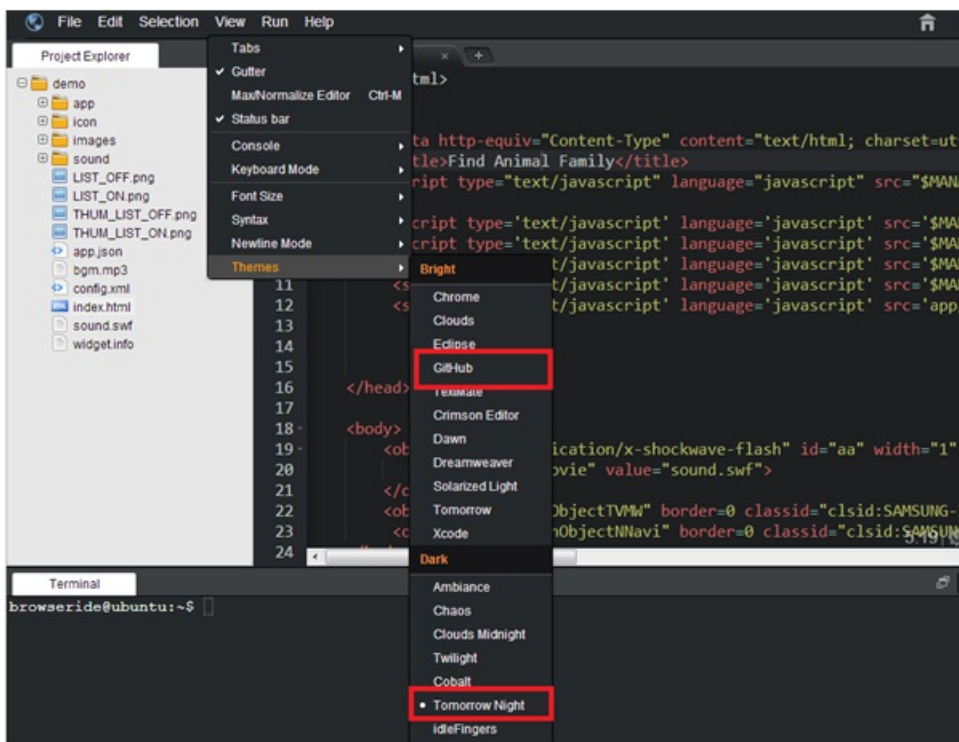


Figure 54: Themes menu with default theme

When the user set the theme to Github, then the theme of editor will be change to the style of Github.

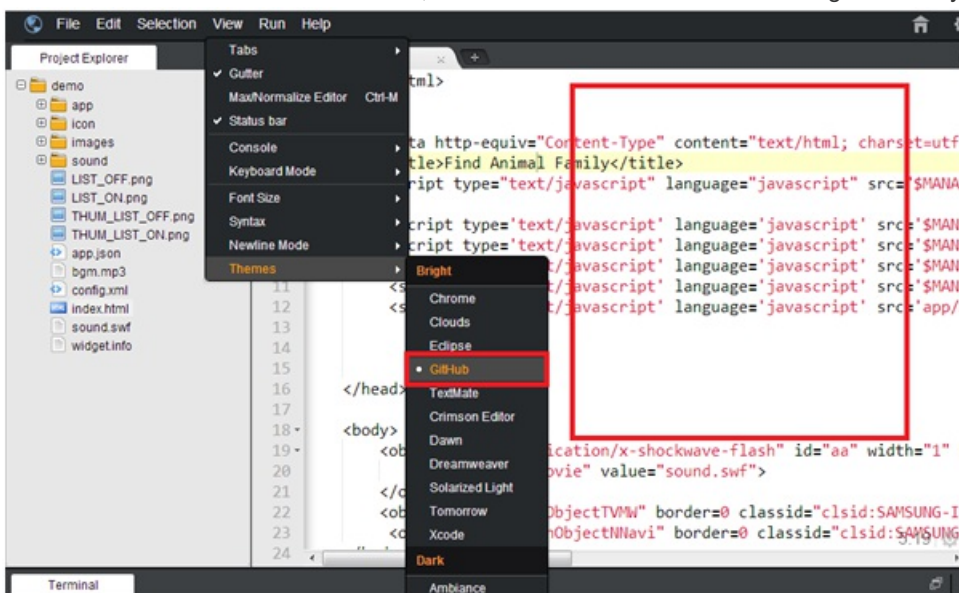


Figure 55: Set theme of editor

Run

Run menu provides user to run his project through Samsung Web Emulator, when user clicks **Run Project**, it will open a new window and run emulator.

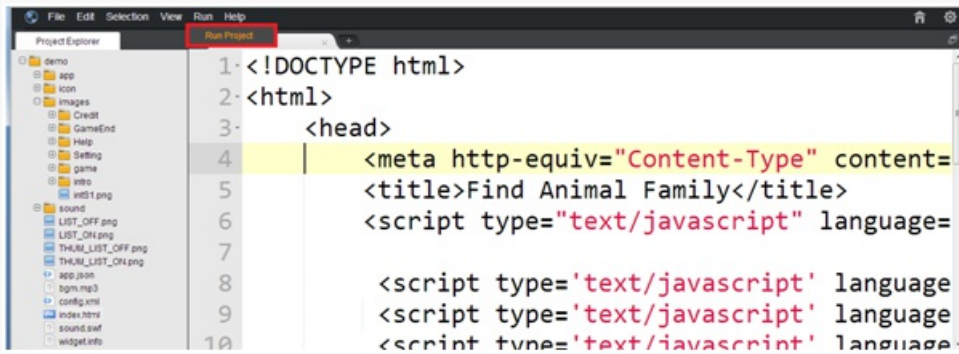


Figure 56: Run project with emulator

If running successful, the emulator page will be open, and run user's application.

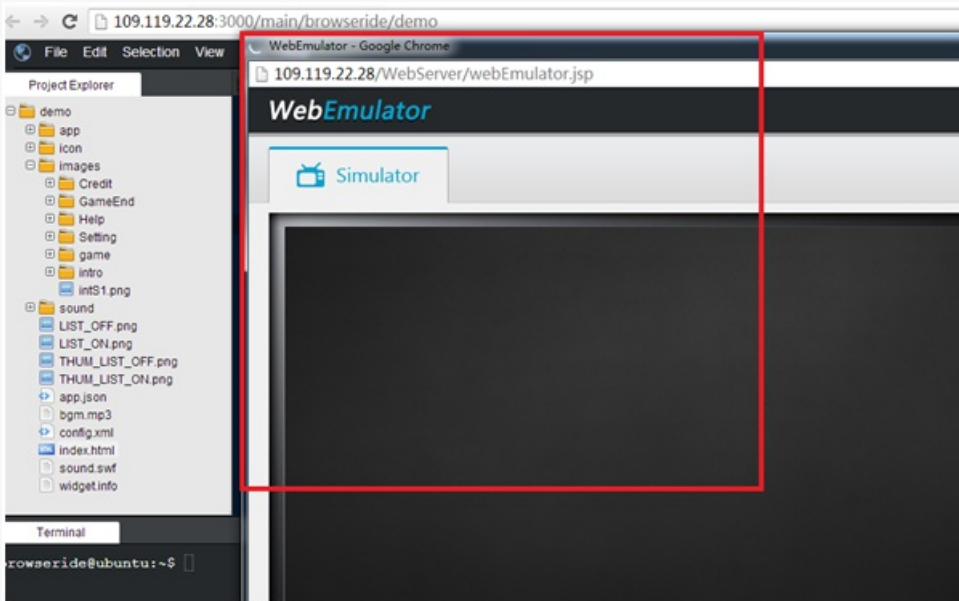


Figure 57: Emulator page

Help

Help menu contains some helpful functions for user, include **Hot key**, **Help Document**, **About Samsung Smart TV SDK** three menus, user can use these functions to understand how to use Samsung Smart TV SDK.

1. **Hot key** menu provides all hot keys exist in Samsung Smart TV SDK, if the users enjoy using keyboard rather than mouse, this will be helpful.

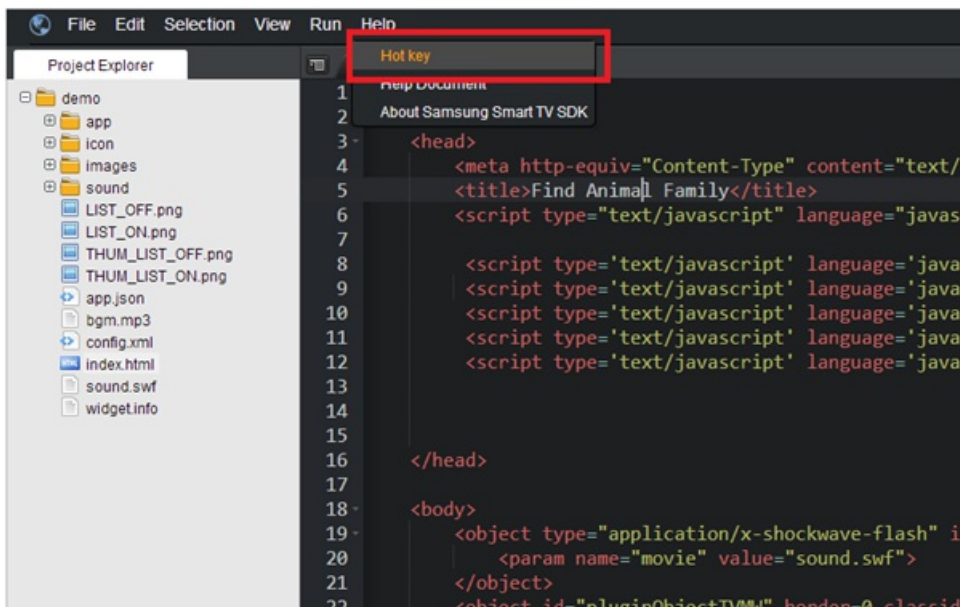


Figure 58: Hotkey menu

When user click the **Hot key** to show the hot key view, user will see all hot keys, user can query hot keys by Filter bindings input box.

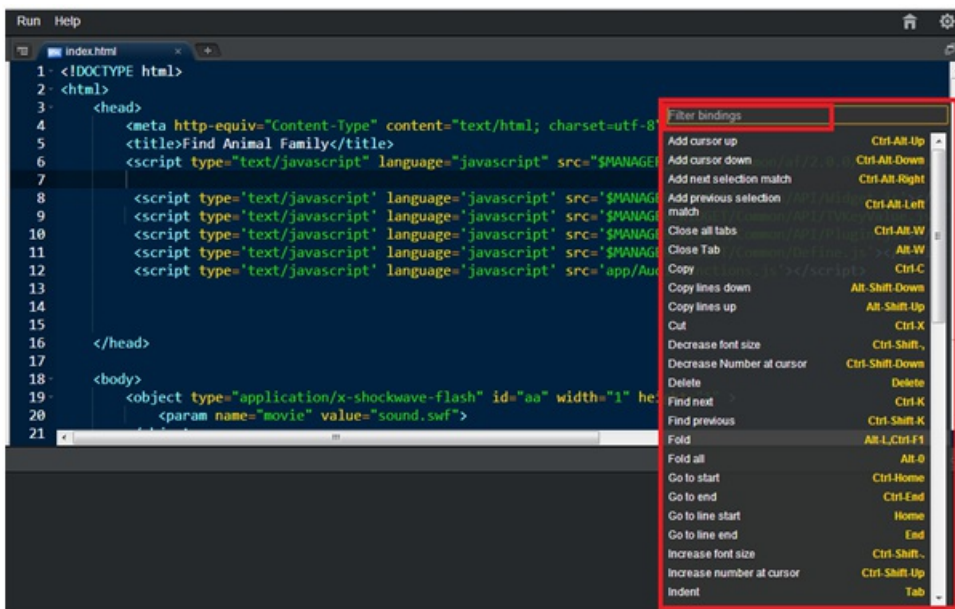


Figure 59: Hot key views

2. **Help Document** provides user to understand the develop guide of Samsung Smart TV SDK, with this user can easy to code Samsung Smart TV Applications.

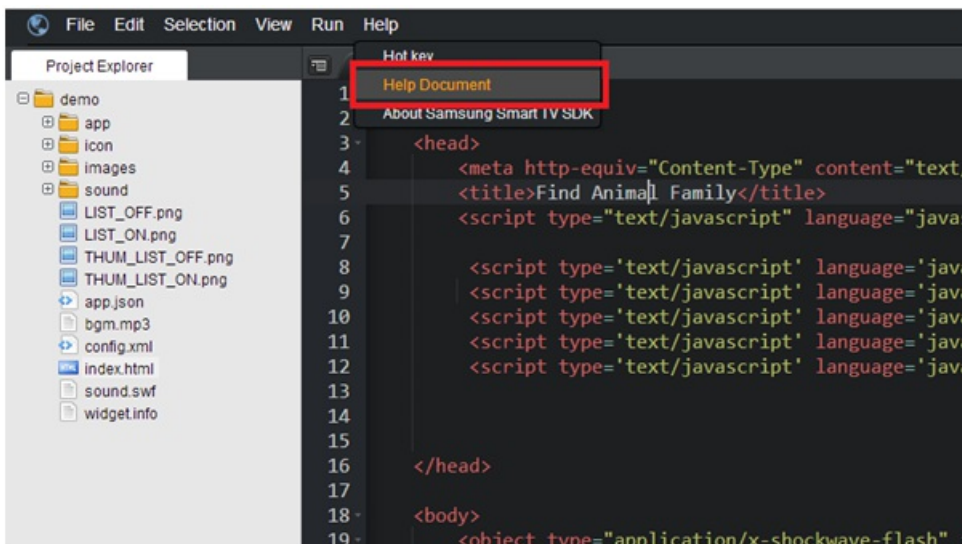


Figure 60: Help document menu

When user clicks the **Help Document**, it will jump to Samsung D FORUM.

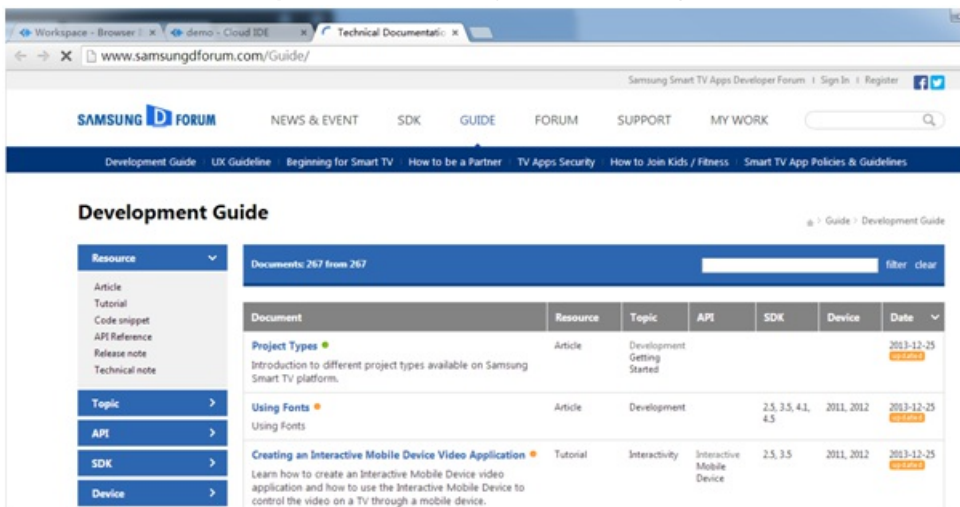


Figure 61: Samsung D FORUM Website

3. **About Samsung Smart TV SDK** menu provides user to know the copy right of Samsung Smart TV SDK.

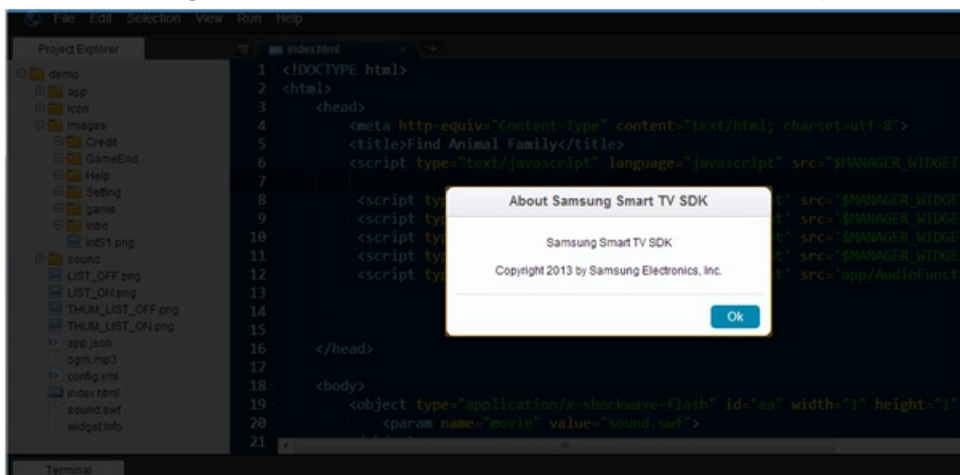


Figure 62: About Samsung Smart TV SDK

Source Editor Syntax Highlight

Source Editor can show source code in different color corresponding to syntax. The syntax coloring is automatically rendered when source code file loaded or edited. No need of user interactive. By using syntax highlight, developer can easily find syntax errors and identify symbols. The style of coloring can be changed by selecting item in main menu. Source Editor provides over 20 themes of coloring, both bright and dark.


```

3
4 var int = 123;
5
6 function onStart () {
7   // TODO : Add your Initilize code here
8
9   sf.scene.show('Scene1');
10  sf.scene.focus('Scene1');
11 }
12

```

Figure 63: Syntax Highlight

AutoComplete

Auto-completion is a useful function for developer. It helps developer coding by prompting a name list. Select one, then the entire name insert. To show the name list, users can simply press Alt+/, In Source Editor, auto-completion can be divided into several types:

Local:

All the items in list which is labeled by “local” come from the local source code file. All the user-defined names (including variable names, function names, class names) and all the names appeared in the local source code file are labeled by “local”.

```

1 alert('SceneScene1.js loaded');
2
3 function SceneScene1() {
4
5 };
6
7 SceneScene1 local
8 function local function ()
9 prototype local );
10 loaded local d only once
11 initialize local ls and style
12 js local loaded before
13 alert local
14 this local sfLabel({
15   text: 'Welcome to Browser SDK!'
16 });
17

```

Figure 64: Auto-complete

Device API:

The names defined by Samsung Smart TV also can be shown in the name list. All the names labeled by “device API” are provided by Samsung Smart TV platform. These names stand for those global objects dealing with TV device and it’s API, including their parameters and methods.

```

1
2
3 playc
4 playcontrol device API
5
6 function onStart () {
7   // TODO : Add your Initilize code here
8   // NOTE : In order to start your app, call
9

```

```

1
2
3 playcontrol.
4 check3DModeEnable method
5 get3DEffectMode method
6 function on get3DModeSupportList method
7 // TODO MODE_3D_DEVICE_NOT_CONNECTED param
8 // NOTE MODE_3D_EFFECT_CHECKER_80 param
9 MODE_3D_EFFECT_FRAME_SEQUENCE param
10 MODE_3D_EFFECT_FROM_2D_TO_3D param
11 sf.scene MODE_3D_EFFECT_LINE_BY_LINE param
12 sf.scene.focus('Scene1');
13

```

Figure 65: Auto completion of device API **Figure 66:** Auto completion of device API

Apps Framework:

Source Editor auto-completion also provides Samsung Apps Framework 2.0 related objects and operations. Supporting Visual Components and their operations, scene management API, core API, environment API etc.

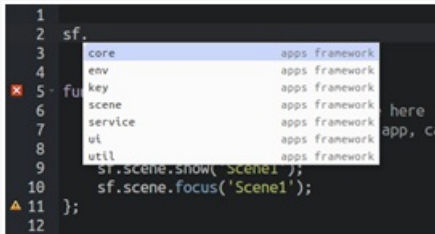


Figure 67: Auto completion of Samsung Apps Framework 2.0

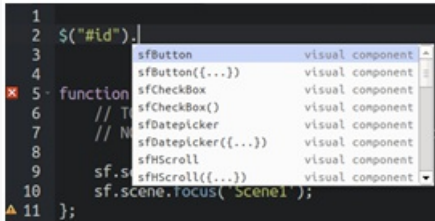


Figure 68: Auto completion of Samsung Apps Framework 2.0

Others:

Other types of auto-completion are JavaScript related, including keywords, global objects, DOM objects and methods. These names are frequently used in JavaScript coding.

Snippets

Code snippet provides developers a lot of code segments for rapid coding. It can generate commonly used code segments and insert into editor, such as “Do-While”, “For-Loop” etc. Developer can modify the generated snippets as they want. To use code snippets, developer can type a part of the keyword and then press “TAB”, or another way, using auto-completion, press “Alt+”, select snippet item from completion list.

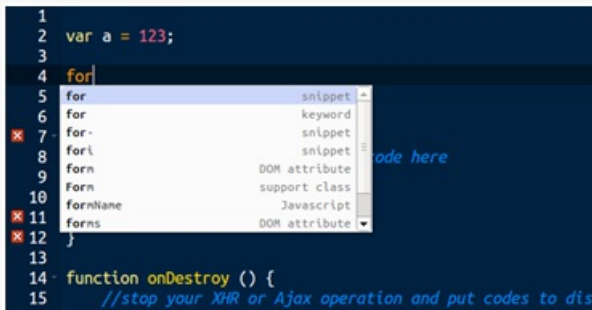


Figure 69: Snippets function

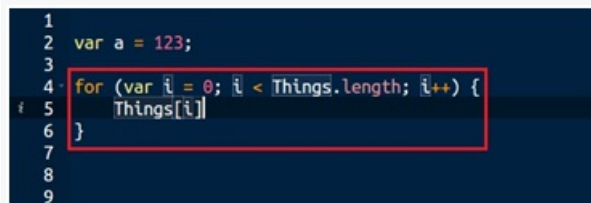


Figure 70: Snippets function

Multi-lines Edit

Multi-line Editor is useful for multi-line edit, such as block edit, user can select multi-line and there are many cursors in the editor, user can operate multi-line code and operate them at the same time.

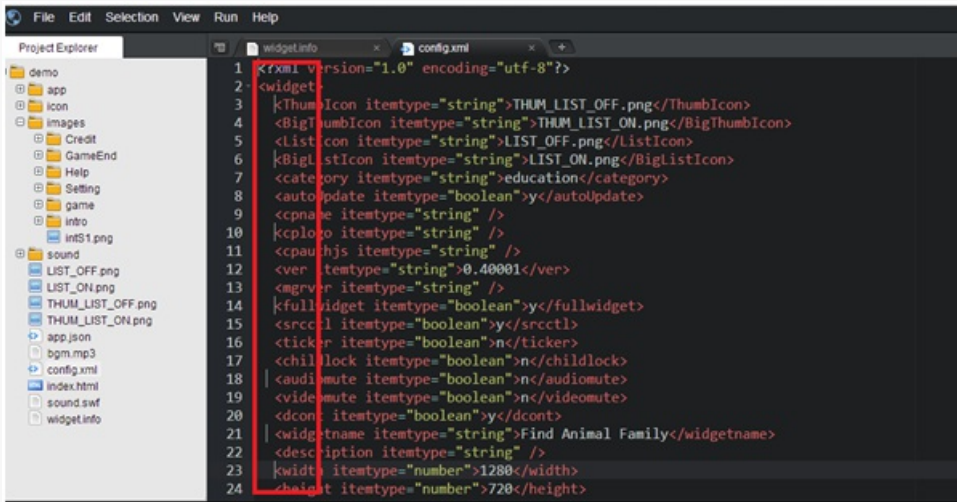


Figure 71: Multi-line edit with cursor focused

When user select multi-line and type in “hello, world”, the editor will appear multi of “hello, world”.

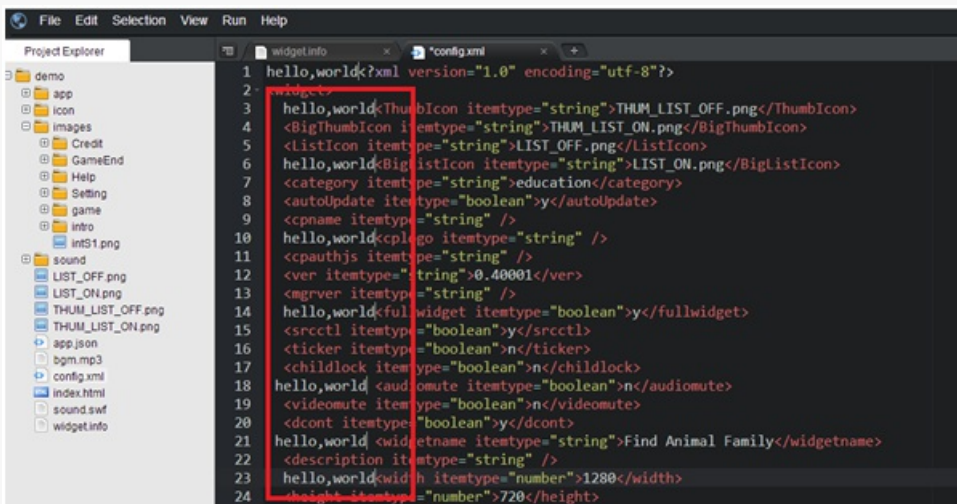


Figure 72: Multi-line editor with multi-input values

Context menu

Context menu provides user to operate editor conveniently, such as the other editors, user can use it to copy, paste, and cut and delete code from editor.

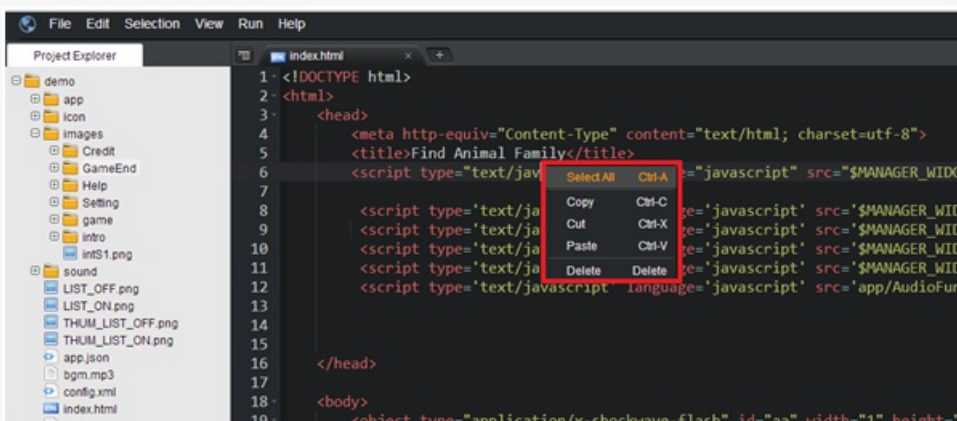


Figure 73: Context menu of editor

Settings

Settings menu layout in the right-bottom of editor, it's useful for user to set the global attributes of editor, such as **Show Gutter**, **Highlight Selected Word** and so on, next time user open the editor, the setting is still available.

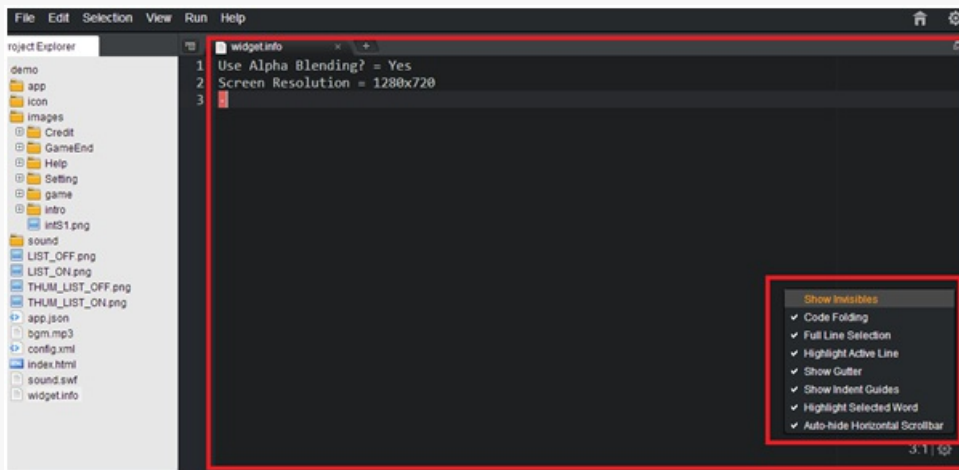


Figure 74: Source Editor Settings

Project Explorer Context menu

The context menu of project can operate file and folder include Open File, New File, New Folder, Copy, Paste, Delete, Rename, Property as follow figure:

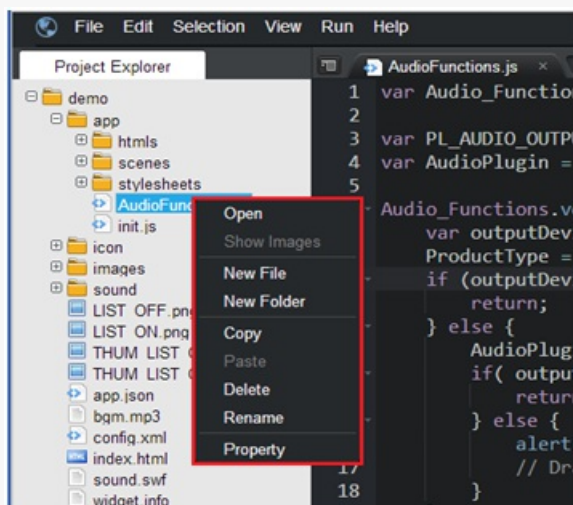


Figure 75: context menu-project

The open, new file, new folder as well as file menu.

User can copy, paste, delete and rename file in the project explorer.

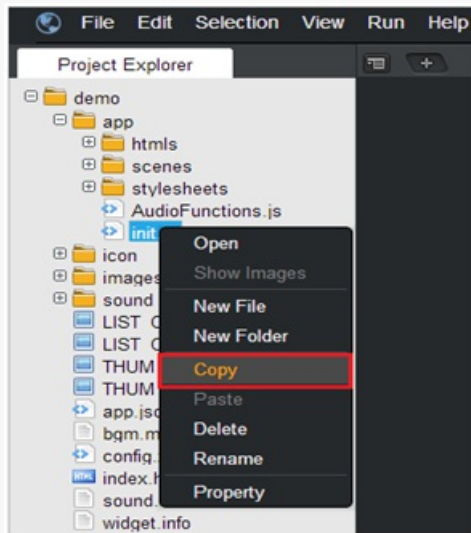


Figure 76: copy context menu

The “Property” can display file information. As follow figure:

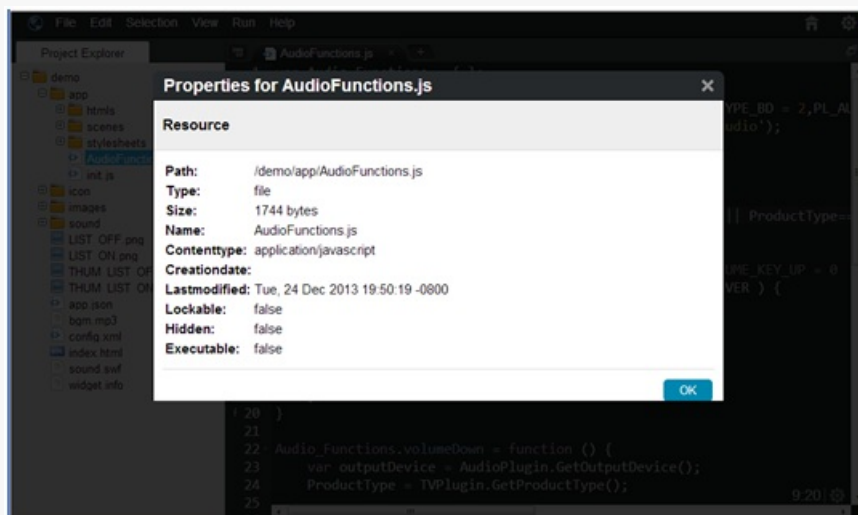


Figure 77: context menu-property

Image Viewer

The image viewer tab in the console shows project images whenever a user click “Show Images” menu. As follow figure:

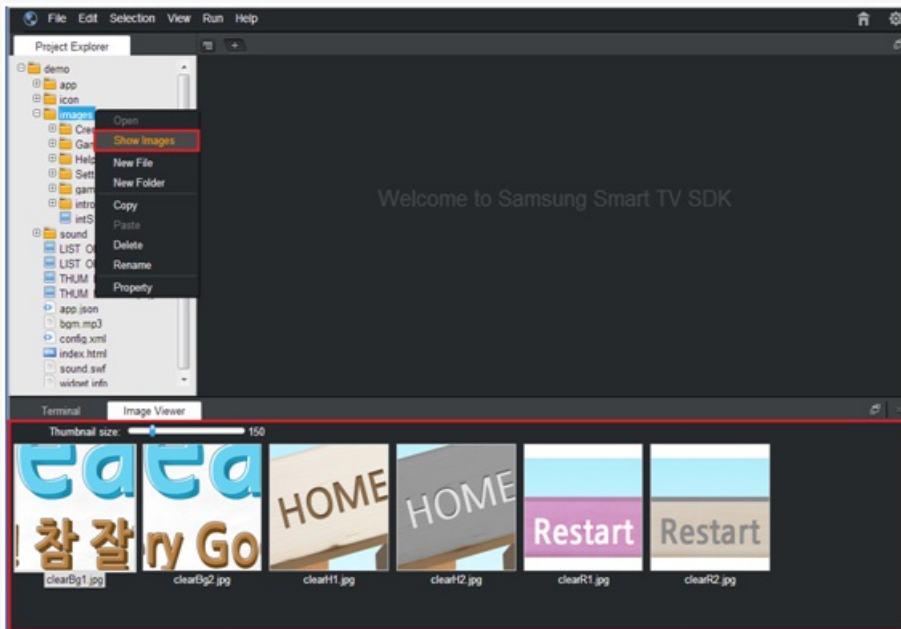


Figure 78: dashboard-project

Terminal

The terminal provides user with a TTY interface for direct access to the browser IDE host. It supports shells in UNIX.

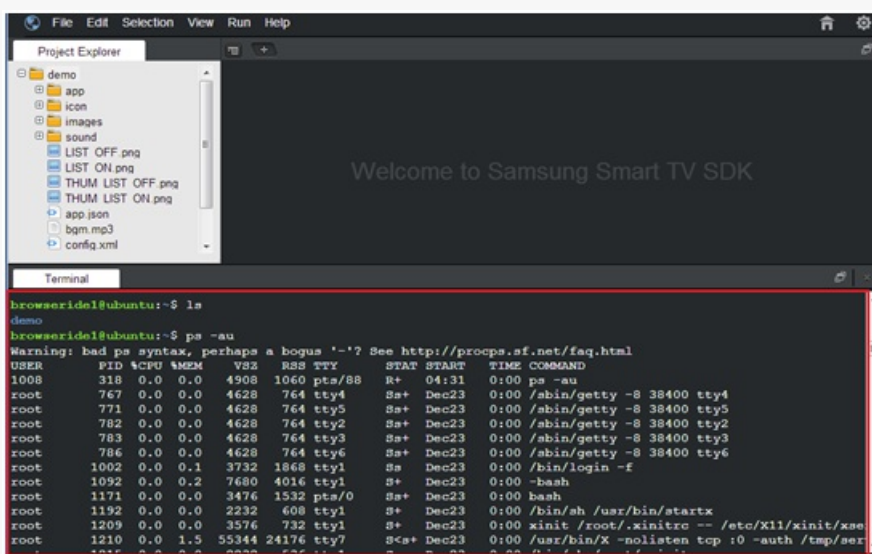


Figure 79: Shell 1

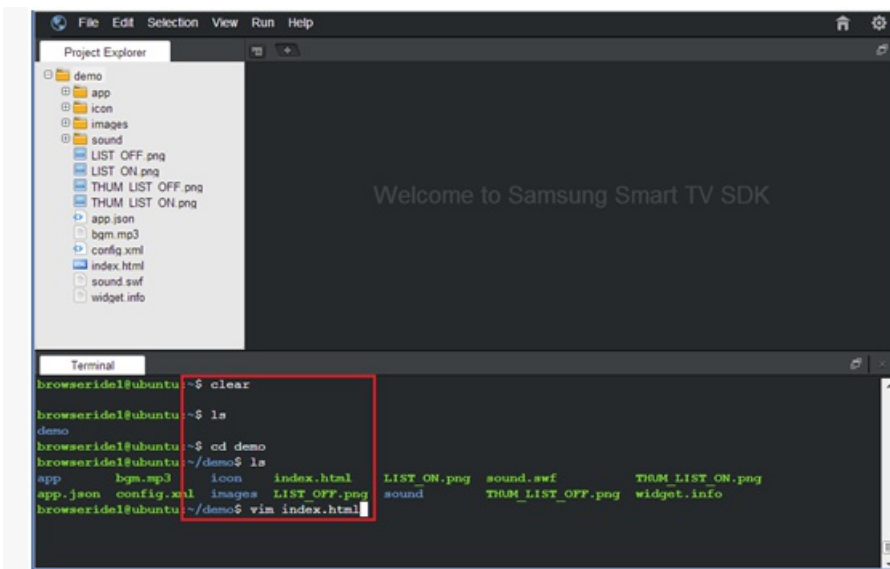


Figure 80: Shell 2

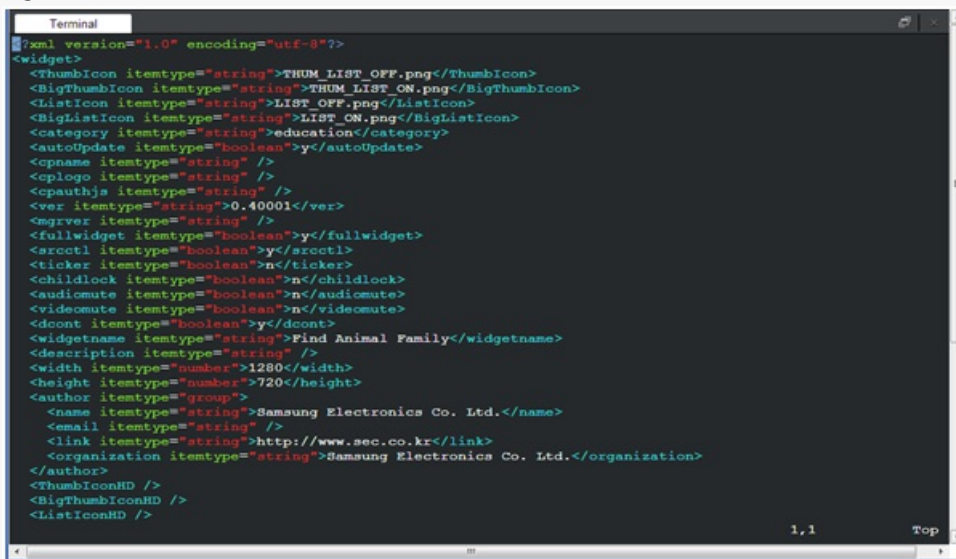


Figure 81: terminal vim

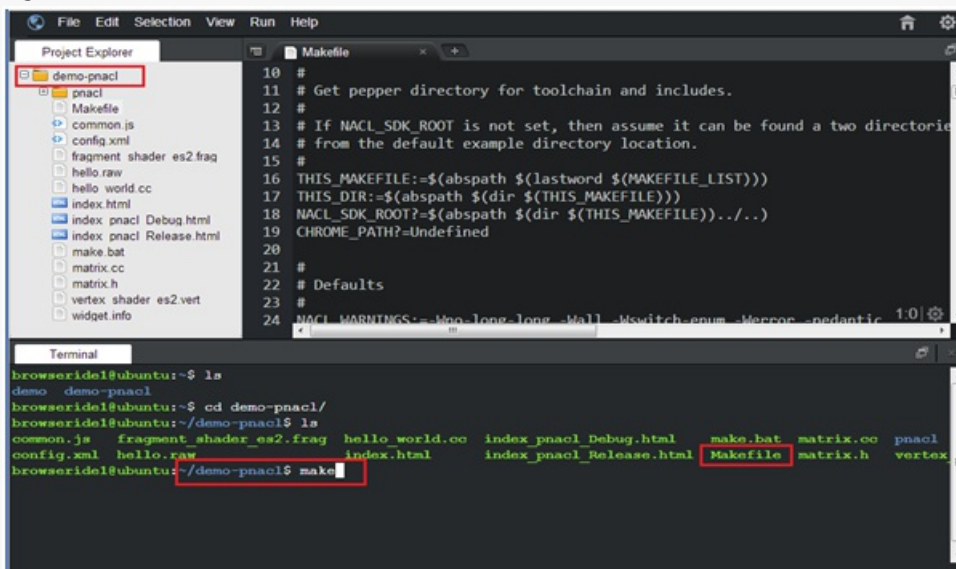


Figure 82: Make PNACI

Summary

No.	Functions	Local Browser IDE	Native IDE	Remark

No.	Functions					Local Browser IDE	Native IDE	Remark	
1	Project		Basic Project / App Framework 2.0 Project				O	O	
			JavaScript Project				O	O	
			PNaCL Module (C++) Project				O	O	
2	Source Editor	Syntax Highlight	HTML	JavaScript	CSS	C / C++			
			V				O	O	
				V			O	O	
					V		O	O	
						V	O	O	
		Auto complete	V				O	O	
				V			O	O	
					V		O	O	
						V	X	O	
		Snippets	V				O	O	
				V			O	O	
					V		O	O	
						V	O	O	
3	Debug		V				O	O	See DOM via web inspector
				V			X	O	Browser IDE: JS debug not supported yet
					V		O	O	
						V	X	O	
4	Terminal		Windows / Linux / Mac OS X				Bash (as like gnome - terminal)	Window Windows shell Linux / Mac OS X: Linux shell	
5	Visualization Tools		Visual Editor				X	O	
			Visual Kit				X	O	
6	Restrictions		Runtime Environment				1. Virtual Box 2. Browser (Chrome)	1. JRE 2. Virtual Box 3. CDT 4. Python 5. PNaCL 6. Tool chain 7. Eclipse	
			Memory		Recommended (Server)		>= 2G	>= 2G	
					Least		>= 1G	>= 1G	
			Operating System		Recommended		Windows / Linux / Mac OS	Windows 7 / Ubuntu / Mac OS	

No.	Functions			Local Browser IDE	Native IDE	Remark
			Normal	Windows / Linux / Mac OS	1. Windows XP/7/8 2. Linux 3. Mac OS	