Retro Gaming Dialog Box Library

User Manual

Peter J. Walsh

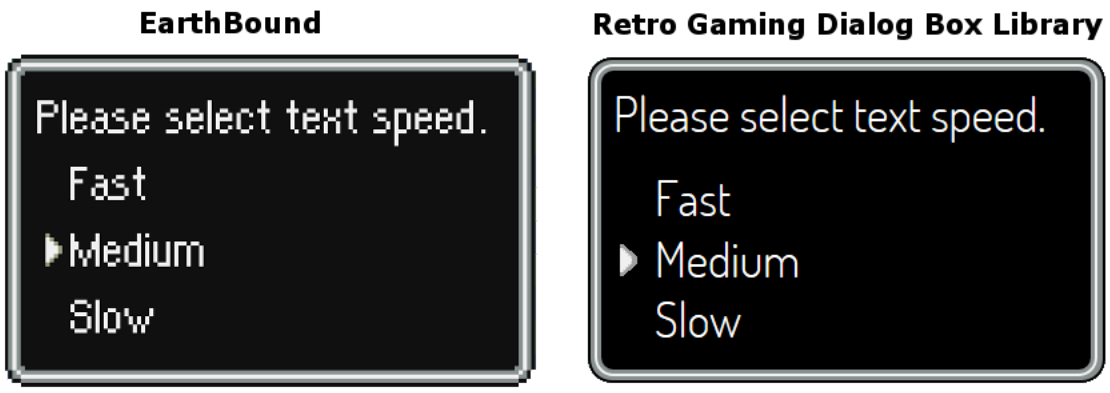
March 16, 2012

Contents

1 Overview

1.1 Purpose

The purpose of this library is to automate the creation and maintenance of the HTML and CSS required for the rendering of UI elements styled in a manner similar to the 1995 game EarthBound. In addition to this the library encapsulates all the necessary functionality for maintaining the state of these UI elements.



Please select text speed.

Please select text speed.

Figure 1: Comparison

1.2 Usage

In order to use this library you must:

1. Include a link to a stylesheet declaring the Dosis fontface.[1] Dosis is available from Google Fonts, so you can simply include the following in your HTML documents head tag:

```
<link href='http://fonts.googleapis.com/css?family=Dosis
    :300,700' rel='stylesheet' type='text/css'>
```

[1]Dosis was the most EarthBound-like, readily available font I could find and was created by Pablo Impallari. A lot of spacing calculations are based on the font height, so substituting another font will require some tweaking.

2. Include the Retro Gaming Dialog Box JavaScript library.

```
<script src="RetroUI.js"></script>
```

3. Initialize a UI box object, and append it to the document

```
var objDisplayBox = new DisplayBox('myDisplayBox', 8, 4,
    'plain', 'My Display Box');
document.getElementsByTagName('body')[0].appendChild(
    objDisplayBox.GetRootElement());
```

4. Add a stylesheet to the end of your documents body tag to position the dialog box UI elements. This stylesheet must be placed in the body because the objects themselves will append their own stylesheets to the head element at runtime and override your styles.

1.3 Example

For an interactive demonstration of this library please visit:

http://pw624.cnawebdev.org/earthboundUiMockup/

Here you will find the new game user flow from EarthBound recreated using all the classes from this library.

1.4 Credits

The Retro Gaming UI Library and accompanying documentation were written by Peter J. Walsh on March 17, 2013.

The Dosis typeface that this library was designed around was created by Pablo Impallari[2] and is available through Google Web Fonts.

The design of the user interface elements were inspired by the designs of Nintendo and HAL Laboratories EarthBound.

1.5 Legalities

EarthBound and all related materials are copyright 1994, 1995 Nintendo and HAL Laboratories. No harm is meant in this re-implementation, only flattery.

[2]https://plus.google.com/114391601624281927771/posts

2 Public Interface

2.1 Overview

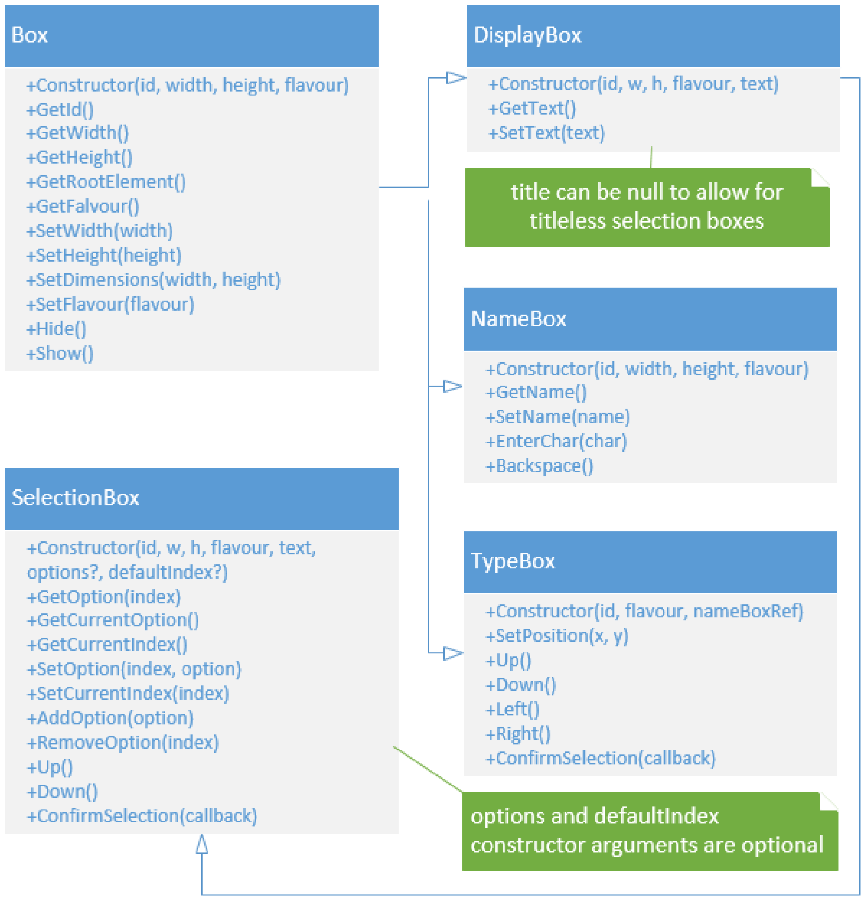

Box
+Constructor(id, width, height, flavour)
+GetId()
+GetWidth()
+GetHeight()
+GetRootElement()
+GetFalvour()
+SetWidth(width)
+SetHeight(height)
+SetDimensions(width, height)
+SetFlavour(flavour)
+Hide()
+Show()
DisplayBox
+Constructor(id, w, h, flavour, text)
+GetText()
+SetText(text)
title can be null to allow for titleless selection boxes
NameBox
+Constructor(id, width, height, flavour)
+GetName()
+SetName(name)
+EnterChar(char)
+Backspace()
SelectionBox
+Constructor(id, w, h, flavour, text, options?, defaultIndex?)
+GetOption(index)
+GetCurrentOption()
+GetCurrentIndex()
+SetOption(index, option)
+SetCurrentIndex(index)
+AddOption(option)
+RemoveOption(index)
+Up()
+Down()
+ConfirmSelection(callback)
TypeBox
+Constructor(id, flavour, nameBoxRef)
+SetPosition(x, y)
+Up()
+Down()
+Left()
+Right()
+ConfirmSelection(callback)
options and defaultIndex constructor arguments are optional

Figure 2: UML Representation of the libraries classes and public methods

2.2 Box Class

The Box class serves as the base class for all the other box classes. It will instantiate a box with no contents, and has no public methods for populating a box with content. Therefore you will probably not want to directly instantiate objects of this type, instead you will want to extend this class if you wish to create your own dialog box types.

NOTE: All dimensions (widths, heights) are expressed in ems.

The constructor for this class looks like:

```
Box(id, width, height, flavour)

//EXAMPLE INSTANTIATION
var objBox = new Box('myBox', 10, 5, 'plain');
```

The Box class exposes the following public methods:

2.2.1 Getters

GetId()
Returns the HTML element ID of this box object.

GetWidth()
Returns the width of this box object.

GetHeight()
Return the height of this box object.

GetRootElement()
Returns the root HTML element of this box object. You must append this element to your document in order to see/interact with any box objects you instantiate.

GetFlavour()
Returns the flavour of this box object. Can be any of the following string values: plain, mint, strawberry, banana, or peanut.

2.2.2 Setters

SetWidth(width)
Sets the width of this box object, then resizes the border and content child elements as necessary to maintain a proper border. Expects a single argument representing the new width of the container.

SetHeight(height)
Sets the height of this box object, then resizes the border and content child elements as necessary to maintain a proper border. Expects a single argument representing the new height of the container.

SetDimensions(width, height)
Sets the width and height of this box object, then resizes the border and content child elements as necessary to maintain a proper border. Expects two arguments representing the new width and height of the container respectively.

SetFlavour(flavour)
Sets the favour of this box object and updates its style as necessary. Expects a single string argument representing the new box flavour. Valid flavours are: plain, mint, strawberry, banana, and peanut.

2.2.3 Actions

Hide()
Removes this box from the document flow by setting its CSS display property to none.

Show()
Adds this box to the document flow by setting its CSS display property to block.

2.3 DisplayBox Class

The DisplayBox class extends the base Box class by adding a heading. You should use this type of box if you want to display simple information.

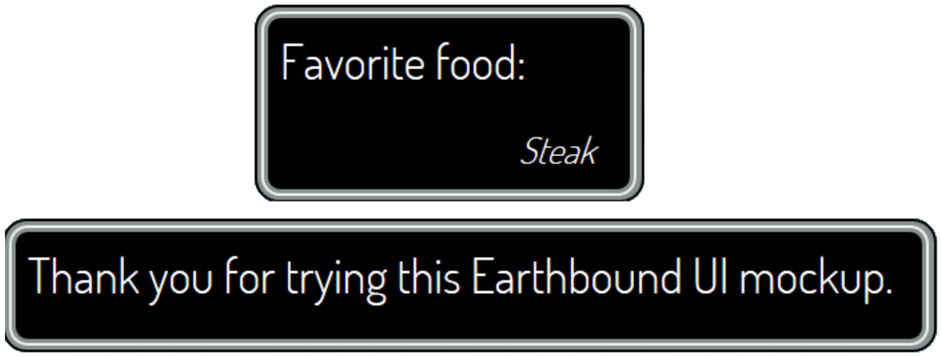


Figure 3: Display Box Examples

The constructor for this class looks like:

```
DisplayBox(id, width, height, flavour, text)

//EXAMPLE INSTANTIATION
var objDisplayBox = new DisplayBox('myDisplayBox', 'plain', 'My
    Display Box');
```

The DisplayBox class exposes the following public methods:

2.3.1 Getters

GetText()
: Returns the display text of this display box.

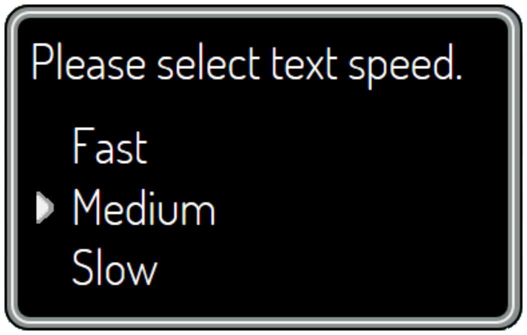
2.3.2 Setters

SetText()
: Sets the display text of this display box. Expects a single string argument representing the new display text.

2.4 SelectionBox Class

The SelectionBox class extends the DisplayBox class by adding a cursor image and unordered list of options to the box markup. This class also provides the necessary methods to update and maintain the state of these options.

You should use this type of box if you want to display a list of choices to the user, and capture their response.


Please select text speed.

Fast

▶ Medium

Slow

Figure 4: Selection Box Example

The constructor for this class looks like:

```
SelectionBox(id, width, height, flavour, text?, options?,
    defaultIndex?)

//EXAMPLE INSTANTIATION
var SelectionBox = new SelectionBox('mySelBox', 10, 20, 'plain',
     'Choose One:', ['Choice 1', 'Choice 2', 'Choice 3'], 1);
```

The text, options and defaultIndex arguments are optional. Options is a string array representing the choices displayed to the user. Default index determines which item initially has focus, and is zero-based.

The SelectionBox class exposes the following public methods:

2.4.1 Getters

GetOption(index)
Returns a value from the options array. This array represents the list item elements in the unordered list. In other words, the options you are displaying to your user. Takes a single argument, the index to return from the options array and returns a string.

GetCurrentOption()
Returns the value of the option the user is currently focused on. Takes no arguments and returns a string.

GetCurrentIndex()
Returns an index to the options array representing the choice the user is currently focused on. Takes no arguments and returns an integer.

2.4.2 Setters

SetOption(index, option)
Sets a new value for the specified index of the options array and rebuilds the unordered list markup to reflect its new contents. Takes two arguments, an integer representing the index for the options array, and a string representing the value you want displayed to the user.

SetCurrentIndex(index)
Sets the index of the option that has focus and updates the markup for the cursor to reflect this new value. Takes one argument, an integer representing the index of the item to set focus on.

2.4.3 Actions

AddOption(option)
Adds a new element to the options array and rebuilds the unordered list markup to refelct its new contents. Takes a single argument, a string representing what you want displayed to the user.

RemoveOption(index)
Removes an option from the options array at the specified index. Takes a single argument, the index of the option to be removed.

Up()
Sets focus to the option above the current. If the current option is the top option, focus will wrap around and be set to the bottom option. You would normally map some form of input to this function.

Down()
Sets focus to the option below the current. If the current option is the bottom option, focus will wrap around and be set to the top option. You would normally map some form of input to this function.

ConfirmSelection(callback?)
Confirms the currently selected option. This will disable and hide the cursor, toggle a purple background colour on for the selected option, and call an optional callback function if provided. The callback function will be called with a single argument, the value of the option selected.

2.5 NameBox Class

The NameBox class[3] displays up to five characters, and can receive as input single characters. Characters yet to be entered will be displayed as a dash. The users current position will be displayed as a circle.

This class can recieve its input from an object of type TypeBox.

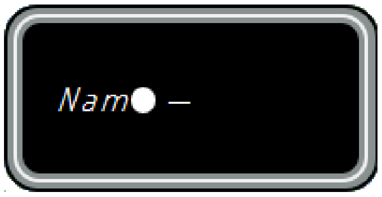


Figure 5: An example of a NameBox

The constructor for this class looks like:

```
NameBox(id, width, height, flavour)

//EXAMPLE INSTANTIATION
var objNameBox = new NameBox('myNameBox', 8, 4, 'plain');
```

The NameBox class exposes the following public methods:

2.5.1 Getters

GetPlayerName()
Returns the value of the users current input as a string.

2.5.2 Setters

SetName(name)
Sets the value of the name currently being entered. Takes a single argument of type string. This value will be truncated if it is above 5 characters in length.

[3]so named because of the functionality of the UI element it mimics

2.5.3 Actions

EnterChar(char)
Appends a single character to the users current input. Accepts a single argument of type string. This value will be truncated if it is above 1 character in length.

Backspace()
Removes the last entered character from the users input.

2.6 TypeBox Class

The TypeBox class displays a keypad like entry box to the user, and exposes the necessary public methods to maintain and update the state of this box as well as capture user input.

Objects of this type have a fixed width and height, and as such do not require these options in their constructor.

The output of an object of this type is meant to be fed into an object of type NameBox. As such the constructor requires a reference to an object of type NameBox.

The constructor for this class looks like:

```
TypeBox(id, flavour, nameBoxReference)

//EXAMPLE INSTANTIATION
var objTypeBox = new TypeBox('myTypeBox', 'plain', objNameBox);
```

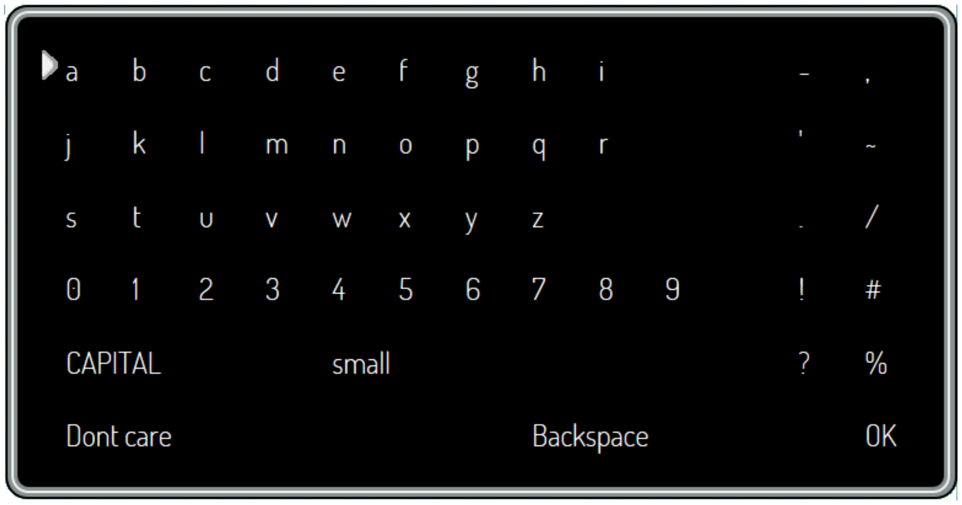


Figure 6: An example of a TypeBox

The TypeBox class exposes the following public methods:

2.6.1 Setters

SetPosition(x, y)
Moves the cursor to the item identified by the x, y pair given. Takes two arguments, both integers, specifying the x and y coordinates respectively.

2.6.2 Actions

Up()
Moves the cursor to the position above its current. If the cursor is currently in the top row it will wrap around to the bottom.

Down()
Moves the cursor to the position below its current. If the cursor is currently in the bottom row it will wrap around to the top.

Left()
Moves the cursor to the position to the left of its current. If the cursor is in the left-most column it will wrap around to the right-most.

Right()
Moves the cursor to the position to the right of its current. If the cursor is in the right-most column it will wrap around to the lest-most.

ConfirmSelection(callback?)
Confirms the current selection. This will check to see if the option selected was CAPITAL or small and change the case of the letters displayed accordingly. It will also call a callback function if specified, and pass it a single argument representing the option selected.