ReadSpeaker
The Voice of the Web!

Dolderseweg 2a, 3712 BP Huis ter Heide
T: 030-692 44 90, F: 030-692 45 99
E: info@readspeaker.com
www.readspeaker.com

READSPEAKER ENTERPRISE HIGHLIGHTING 2.5

Advanced Skinning Guide

Introduction

The graphical user interface of ReadSpeaker Enterprise Highlighting is built with standard web technologies, Hypertext Markup Language (HTML), Cascading Style Sheets (CSS) and JavaScript. Therefore, it is very simple to create a customized version of the player, for example to make it match your organization's graphical profile.

This guide is intended for web developers, and assumes that you have previous experience of the aforementioned web technologies.

Getting started

This is a simple step-by-step guide that explains the overall process of creating and implementing a new skin. In the chapters that follow we will explain each step more in-depth.

1. Setup the folder structure

This is the recommended way of structuring your skin files. You are not required to follow it, but doing so will make it easier to follow this guide.

- Create a new folder that will contain your skin files, i.e. JavaScript, CSS as well as images or any other skin-related files. Use only alphanumeric characters and capitalize the initial letter of each new word. Omit spaces. Ex. `MyNewReadSpeakerSkin`
- Inside the new folder, create two empty text files:
  - A CSS file. Example: `MyNewReadSpeakerSkin.css`
  - A JavaScript file. Example: `MyNewReadSpeakerSkin.js`

2. Add reference to your skin files

Add links to your skin files from the page/template where you want use ReadSpeaker. These references should go in the HEAD section of the HTML page. If you have made inline configurations using the rsConf object, it is important that you include the references AFTER the inline configuration.

Dolderseweg 2a, 3712 BP Huis ter Heide
T: 030-692 44 90, F: 030-692 45 99
E: info@readspeaker.com
www.readspeaker.com

Example:

```
<head>
...
<script type="text/javascript">
<!--
// Inline ReadSpeaker configuration
window.rsConf = {
...
}
//-->
</script>
<link rel="stylesheet" type="text/css"
href="MyNewReadSpeakerSkin/MyNewReadSpeakerSkin.css" />
<script type="text/javascript"
src="MyNewReadSpeakerSkin/MyNewReadSpeakerSkin.js"></script>
...
</head>
```

Note that the code might have to look different depending on which document type you are using. All examples in this document assume that XHTML 1.0 Strict is being used.

3. Add the CSS class to the configuration

Your JavaScript skin file should set the name of the CSS class that your skin is built on. The JavaScript skin file should contain at least the following code:

```
if (! window.rsConf) { window.rsConf = {}; }
if (! window.rsConf.ui) { window.rsConf.ui = {}; }
window.rsConf.ui.rsbtnClass = 'MyNewReadSpeakerSkin';
```

The first two lines are there to make sure that the configuration object is set up correctly. The third line tells ReadSpeaker which class to look for when rendering the skin.

4. Change the listen button's code

The listen button's code contains several CSS classes. If you want to use a custom skin, you need to replace the class `rsbtn` with whatever name your skin uses as its base class. Based on the example in step 3, your class name should be: `MyNewReadSpeakerSkin`.

Example:

```
<div id="readspeaker_button" class="rsbtn rs_skip rs_preserve">
    <a class="rsbtn_play"
    ...
    </a>
</div>
```

Dolderseweg 2a, 3712 BP Huis ter Heide
T: 030-692 44 90, F: 030-692 45 99
E: info@readspeaker.com
www.readspeaker.com

Should then be:

```
<div id="readspeaker_button" class="MyNewReadSpeakerSkin rs_skip
rs_preserve">
    <a class="rsbtn_play"
    ...
    </a>
</div>
```

The Anatomy of the Player

The player mainly consists of two parts:

1. The Listen button. This is the link that activates the speech. It is visible when the page is first loaded.
2. The Expanding area also referred to as the player area. This is where the playback controls are located.

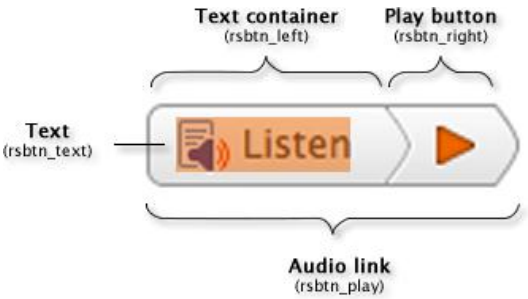
The Listen button

This is the markup for the listen button (shortened):

```
<div id="AUniqueID" class="rsbtn rs_skip rs_preserve">
  <a class="rsbtn_play" accesskey="L" title="Title" href="...">
    <span class="rsbtn_left rsimg rspart">
      <span class="rsbtn_text">
        <span>Listen</span>
      </span>
    </span>
    <span class="rsbtn_right rsimg rsplay rspart"></span>
  </a>
</div>
```

Our recommendation is that the markup for the listen button remains unchanged, except for the rsbtn class of the outer DIV.

This is how the listen button in the default skin maps against the code above:



Dolderseweg 2a, 3712 BP Huis ter Heide
T: 030-692 44 90, F: 030-692 45 99
E: info@readspeaker.com
www.readspeaker.com

The expanding area

When the service has been started the markup for the expanding area is automatically injected and the code then looks like this (shortened):

```
<div id="AUniqueID" class="rs_skip rsbtn rs-no-touch rsexpanded rsplaying">
  <a class="rsbtn_play" accesskey="L" title="Title" href="...">
    <span class="rsbtn_left rsimg rspart">
      <span class="rsbtn_text"><span>Listen</span></span>
    </span>
    <span class="rsbtn_right rsimg rsplay rspart"></span>
  </a>
  <span class="rsbtn_exp rsimg rspart">
    <a class="rsbtn_pause rsimg rspart" href="...">
      <span class="rsbtn_btnlabel">Play</span>
    </a>
    <a class="rsbtn_stop rsimg rspart" href="...">
      <span class="rsbtn_btnlabel">Stop</span>
    </a>
    <span class="rsbtn_player rsimg rspart">
      <span class="rsbtn_progress_container rsimg rsplaypart">
        <span class="rsbtn_progress_played rsimg" ></span>
        <span class="rsbtn_progress_handle rsimg">
          <a href="..." class="keyLink"> </a>
        </span>
      </span>
      <a class="rsbtn_volume rsimg rsplaypart" href="...">
        <span class="rsbtn_btnlabel">Volume</span>
      </a>
      <span class="rsbtn_volume_container rsimg">
        <span class="rsbtn_volume_slider">
          <span class="rsbtn_volume_handle rsimg">
            <a href="..." class="keyLink"> </a>
          </span>
        </span>
      </span>
      <a class="rsbtn_settings rsimg rsplaypart" href="...">
        <span class="rsbtn_btnlabel">Settings</span>
      </a>
      <a class="rsbtn_dl rsimg rsplaypart" href="...">
        <span class="rsbtn_btnlabel">No Sound?</span>
      </a>
    </span>
    <a class="rsbtn_closer rsimg rspart" href="...">
      <span class="rsbtn_btnlabel">Close</span>
    </a>
    <span class="rsbtn_powered rsimg">
      <span class="rsbtn_btnlabel">Speech enabled by
        <a href="...">ReadSpeaker</a>
      </span>
    </span>
  </span>
</div>
```

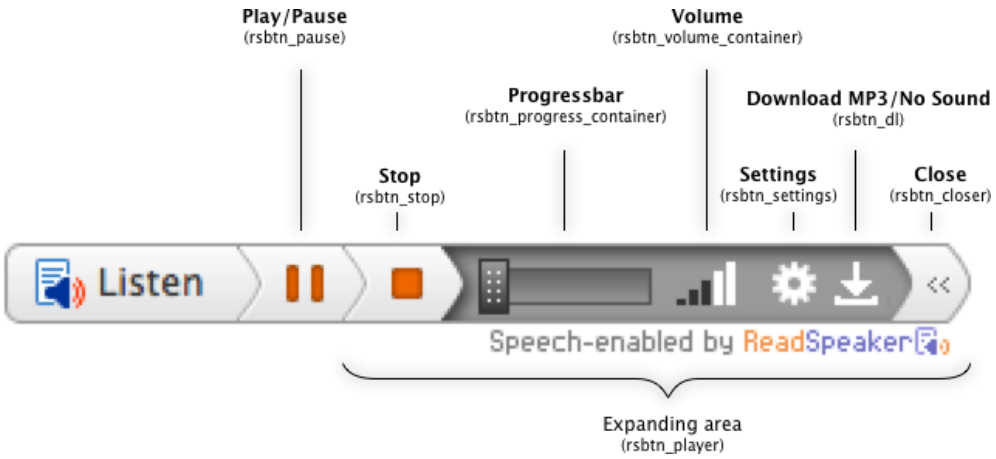
Dolderseweg 2a, 3712 BP Huis ter Heide
T: 030-692 44 90, F: 030-692 45 99
E: info@readspeaker.com
www.readspeaker.com

The green area represents the classes assigned to the outer container. The yellow section is the code that is inserted when player is expanded.

The classes that need to remain unchanged have been marked in bold. The elements that have these classes will represent buttons and other functionality in the player and will automatically be assigned the necessary event-handlers.

In addition to the reserved classes mentioned above, there is also `.rsbtn_current_time` and `.rsbtn_total_time`. Any elements that have these classes will be updated with the player's current time and the total time that has been buffered so far. The updates happen every 0.5 seconds, while the audio is playing. The format of the time labels will be `hh:mm:ss`, e.g. `01:35:02`.

Here is an overview of the classes and their respective functions in the default skin:



The code for the expanding area can be customized using the public configuration API. The path should be `rsConf.ui.player`. See the next chapter for more information on how to change this code.

Player CSS Classes

The outer container will be assigned different classes (the green area in the code block above) depending on the current state of the player. These classes can be used to change the appearance of the player using CSS.

- `rsexpanded` – This is added when the player is expanded, ie the user clicks on the listen button. Tip: Use this class to determine whether the expanding area, `rsbtn_exp`, should be visible or not.
- `rsplaying` – The player is currently playing audio.
- `rspaused` – The player has been paused.

Dolderseweg 2a, 3712 BP Huis ter Heide
T: 030-692 44 90, F: 030-692 45 99
E: info@readspeaker.com
www.readspeaker.com

- `rsstopped` – The player was previously playing or paused, but has now been stopped.
- `rs-no-touch` – This class is added to all browsers except on iOS devices. Tip: Use this class to disable :hover pseudo-classes on iOS to avoid multiple clicks to activate player buttons.
- `rsfloating` – This class is assigned to the outer container when the expanding area is not adjacent to the listen button. This happens when using non-standard implementation code for the listen button, e.g. when using Highlighting 1.x implementation code with Highlighting 2.5.
- `rsimg` – This class is added to all elements that are expected to be visible. Tip: The existence of this class facilitates the use of CSS sprites.

Note: The same classes will be used for the popup player.

The Javascript file

As mentioned earlier, the JavaScript file needs these two of lines of code to work:

```
if (! window.rsConf) { window.rsConf = {}; }
if (! window.rsConf.ui) { window.rsConf.ui = {}; }
```

These two lines make sure that the configuration object exists before we start to make changes to it. Therefore they must be placed at the very beginning of the file.

Note: You will have to add a conditional statement for every sub-section of the rsConf object that you will be changing. This is to prevent that any previous configurations get overwritten.

Example:
To change `rsConf.general.usePost`, you need to add:

```
if (! window.rsConf.general) { window.rsConf.general = {}; }
```

After this declaration you can change any setting in the general object in this way:

```
window.rsConf.general.usePost = true;
```

Apart from the graphical user interface, any configurable aspect of Enterprise Highlighting can be changed using this JavaScript file. See the ReadSpeaker Highlighting Configuration API reference for more information on how to do this.

Dolderseweg 2a, 3712 BP Huis ter Heide
T: 030-692 44 90, F: 030-692 45 99
E: info@readspeaker.com
www.readspeaker.com

Try to always update the lowest level of the configuration structure. It is better to override multiple settings inside the same element multiple times rather than updating the entire parent object at once.

Example:

```
window.rsConf.ui.player = [ … ];
window.rsConf.ui.popupplayer = [ … ];
```

is better than:

```
window.rsConf.ui = {
  player: [],
  popupplayer: []
}
```

This is because additional properties might be added to the `ui` object in the future, and overriding the entire `ui` object might delete the added properties.

Note that since the skin's JavaScript file should be referenced after any inline configurations, the changes made in it will override their counterparts in the inline configuration.

In terms of the user interface and skinning, these properties are particularly relevant:

- `rs.Conf.ui.progressHandleClass` – Defines the CSS class/es that will be added to the progress slider's drag handle element. Multiple classes should be separated by a space. If not set, it will default to horizontal.
- `rsConf.ui.progressDir` – This sets the direction in which the progress slider will be rendered. 'v' for vertical and 'h' for horizontal.
- `rsConf.ui.volumeHandleClass` – This is the CSS class/es that will be added to the volume drag handle element.
- `rsConf.ui.volumeDir` – The direction of the volume slider. 'v' for vertical and 'h' for horizontal. If omitted, it will default to vertical.
- `rsConf.ui.player` – This is an array that defines the markup of the player's expanding area.
- `rsConf.ui.popupbutton` – This is an array that defines the appearance of the popup button, that is displayed when the user selects text.
- `rsConf.ui.popupplayer` – An array that defines the player area of the popup player.

Dolderseweg 2a, 3712 BP Huis ter Heide
T: 030-692 44 90, F: 030-692 45 99
E: info@readspeaker.com
www.readspeaker.com

The CSS file

The CSS file will look very different from one skin to another and there are not many limitations. However, there is one thing that is important to remember:

All CSS definitions must start with the name of the skin's base class. In the example above (Gettings Started, section 3) we used `MyNewReadSpeakerSkin` as the base class.

Example:

```
.MyNewReadSpeakerSkin .rsbtn_exp .rsbtn_pause { … }
.MyNewReadSpeakerSkin .rsbtn_exp .rsbtn_stop { … }
```

Sample Skin Walkthrough

In this section, we will go through the code of the sample skin which is included with this documentation. The sample skin looks something like this (at about 200% zoom):

All necessary files can be found in the folder named `ReadSpeakerSampleSkin`:

- ReadSpeakerSampleSkin.css – The style definitions for this sample skin.
- ReadSpeakerSampleSkin.js – The skin JavaScript file.
- ReadSpeakerSampleSkin.png – The CSS sprite image we will be using for this sample skin.

1. The JavaScript file

We have only included the absolutely necessary data in the JavaScript file to keep it as small and readable as possible.

```
if (! window.rsConf) { window.rsConf = {}; }
if (! window.rsConf.ui) { window.rsConf.ui = {}; }
```

These two lines create the required objects, if they do not already exist.

```
window.rsConf.ui.rsbtnClass = 'rsbtn_sampleskin';
```

Dolderseweg 2a, 3712 BP Huis ter Heide
T: 030-692 44 90, F: 030-692 45 99
E: info@readspeaker.com
www.readspeaker.com

This line tells ReadSpeaker what the current skin is called, and which style definitions should be used.

```
window.rsConf.ui.player = [
  '<span class="rsbtn_box">',
  '  <a href="javascript:void(0);" class="rsbtn_pause rsimg rspart
rsbutton" title="Pause">',
  '    <span class="rsbtn_btnlabel">Pause</span>',
  '  </a>',
  '  <a href="javascript:void(0);" class="rsbtn_stop rsimg rspart
rsbutton" title="Stop">',
  '    <span class="rsbtn_btnlabel">Stop</span>',
  '  </a>',
  '  <span class="rsbtn_progress_container rspart">',
  '    <span class="rsbtn_progress_played"></span>',
  '  </span>',
  '  <a href="javascript:void(0);" class="rsbtn_volume rsimg rspart
rsbutton" title="Volume">',
  '    <span class="rsbtn_btnlabel">Volume</span>',
  '  </a>',
  '  <span class="rsbtn_volume_container">',
  '    <span class="rsbtn_volume_slider"></span>',
  '  </span>',
  '  <a href="javascript:void(0);" class="rsbtn_settings rsimg rspart
rsbutton" title="Settings">',
  '    <span class="rsbtn_btnlabel">Settings</span>',
  '  </a>',
  '  <a href="javascript:void(0);" class="rsbtn_closer rsimg rspart
rsbutton" title="Close player">',
  '    <span class="rsbtn_btnlabel">Close</span>',
  '  </a>',
  '  <span class="rsdefloat"></span>',
  '</span>'
];
```

This is the markup for the player. It is a simplified version of the default player. The classes that are automatically assigned functionality by ReadSpeaker have been marked in bold.

In theory, you can omit any of the buttons as well as the outer slider containers from the markup. In practice, of course it does not make much sense to create a player without buttons. However, omitting certain buttons or the sliders might be a way to create a very easy-to-use player.

The order in which the buttons and sliders appear is not important. You can also add additional elements to the markup to achieve the structure you want.

Please note that if you want to use the sliders, all elements marked in bold must be included, not just the outer container. For the progress container this means that the

Dolderseweg 2a, 3712 BP Huis ter Heide
T: 030-692 44 90, F: 030-692 45 99
E: info@readspeaker.com
www.readspeaker.com

markup must include both the `.rsbtn_progress_container` and the `.rsbtn_progress_played` elements.

In the sample skin we have omitted the `rsConf.ui.popupbutton` and `rsConf.ui.popupplayer` configuration settings. This means that we will use the default ones instead.

2. The Sample CSS File

Let's go through the CSS file section by section. Explanations and comments follow after each code block.

The container

```
.rsbtn_sampleskin {
    position: relative;
    font-family: Arial, Helvetica, sans-serif;
    font-size: 14px;
    line-height: 1.3em;
    float: left;
    border: 1px solid #ccc;
}
```

We start off by defining the layout of the div container that hosts the listen button as well as the player. The `position` directive makes sure that all child elements are positioned relative to this container. The `border` needs to be defined here to ensure it surrounds both the player and the listen button.

General classes

```
.rsbtn_sampleskin a, .rsbtn_sampleskin span {
    position: relative;
    display: block;
    text-decoration: none;
}
```

Here we tell the browser to treat `a` and `span` tags as block element, even though they are normally inline elements. The main difference between block and inline elements is that block elements occupy the entire width of their parent container, while inline elements can be placed on the same line. We prefer block style elements in the player because it is easier to control their position and dimensions.

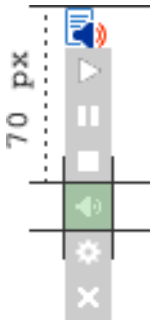
Dolderseweg 2a, 3712 BP Huis ter Heide
T: 030-692 44 90, F: 030-692 45 99
E: info@readspeaker.com
www.readspeaker.com

```
.rsbtn_sampleskin .rsimg {
    background: transparent url(ReadSpeakerSampleSkin.png) no-repeat
scroll 0 0;
}
```

We are going to use a CSS sprite for all bitmap graphics. With the definition above we make sure that all elements that have the `rsimg` class will use the sprite as a background image.

The sprite image looks like this (background color has been added for visibility):

Using the `background-position` directive we can select certain slices of the image by changing the offset. This example shows how the volume icon is selected by setting the vertical offset to - 70 pixels:



In the above example the position was defined: `background-position: 0 -70px`

```
.rsbtn_sampleskin .rspart {
    float: left;
    margin-left: 2px;
}
```

Since all span tags have been defined as block level elements they would normally occupy the entire width of the parent element. So in order to have them line up horizontally, we need to make them float. We use the class `.rspart` to control which elements should float.

Dolderseweg 2a, 3712 BP Huis ter Heide
T: 030-692 44 90, F: 030-692 45 99
E: info@readspeaker.com
www.readspeaker.com

```
.rsbtn_sampleskin .rsdefloat {
    clear: both;
}
```

This class is used to make sure that the elements that come after the floated elements start on a separate line.

```
.rsbtn_sampleskin .rspart.rsbutton {
    background-color: #999;
    width: 18px;
    height: 18px;
    -moz-transition: background-color .2s ease;
    -webkit-transition: background-color .2s ease;
    -o-transition: background-color .2s ease;
    transition: background-color .2s ease;
}
```

This class defines all buttons in our sample player. We want to make sure that they use the same dimensions and background color. The transitions are there to make the rollover effect a little smoother.

```
.rsbtn_sampleskin .rspart.rsbutton:hover {
    background-color: #aaa;
}
```

We add a very subtle color effect when the mouse pointer hovers the buttons. Thanks to the transition directives in the `.rsbutton` definition, this effect is appears animated.

```
.rsbtn_sampleskin .rspart.rsbutton:active {
    background-color: #888;
}
```

In the same way as above, using a pseudo selector (in this case `:active`), we create a color effect for when the buttons are pressed.

```
.rsbtn_sampleskin .rspart .rsbtn_btnlabel {
    display: none;
}
```

In our graphical player we do not want the text labels to be visible. However, by including them in the markup and hiding them with CSS we make sure that they are still accessible to text-based browsers and assistive technology.

Dolderseweg 2a, 3712 BP Huis ter Heide
T: 030-692 44 90, F: 030-692 45 99
E: info@readspeaker.com
www.readspeaker.com

```
.rsbtn_sampleskin .rsbtn_box {
    margin: 1px;
}
```

The `.rsbtn_box` element has been added to wrap all elements in the player so that we can add a margin between the elements and the containing element. The reason we use margin instead padding in the containing element is because we want to avoid problems caused by discrepancies between different box models.

Playback control

```
.rsbtn_sampleskin .rsbtn_play {
    float: left;
    margin: 1px 0;
}

.rsbtn_sampleskin .rsbtn_left .rsbtn_text {
    background: transparent url(ReadSpeakerSampleSkin.png) no-repeat
scroll 0 0;
    padding-left: 20px;
}
```

This defines the listen button and the listen icon. Since this is part of the implementation code that sits in the actual webpage we choose to not add the `.rsimg` class, but instead we redefine the background image. Our recommendation is to not change the markup of the listen button part, to prevent potential problems with future updates.

```
.rsbtn_sampleskin .rsbtn_exp.rsimg.rspart {
    background: none;
    float: left;
    display: none;
}

.rsbtn_sampleskin.rsexpanded .rsbtn_exp.rsimg {
    display: block;
}
```

The `.rsbtn_exp` container holds all elements in the expanded player area. The `.rsexpanded` class is automatically added to the outer container when the player opens. We use this class to determine whether or not the expanded area should be visible.

Dolderseweg 2a, 3712 BP Huis ter Heide
T: 030-692 44 90, F: 030-692 45 99
E: info@readspeaker.com
www.readspeaker.com

```
.rsbtn_sampleskin .rsbtn_pause {
    background-position: 0 -34px;
}

.rsbtn_sampleskin.rspaused .rsbtn_pause,
.rsbtn_sampleskin.rsstopped .rsbtn_pause {
    background-position: 0 -16px;
}
```

This is the definition of the Play/Pause button (.rsbtn_pause). The first rule makes sure that the background image is correct. Then, we change the background image to display a play symbol when the player is in the paused (.rspaused) or stopped (.rsstopped) state, which we can see in the second rule. When the player is expanded, the play/pause toggle event will automatically be assigned to .rsbtn_pause.

```
.rsbtn_sampleskin .rsbtn_stop {
    background-position: 0 -52px;
}
```

This is the stop button (.rsbtn_stop). We make sure that the background image is correctly positioned. ReadSpeaker will assign the stop event to this element.

Progress slider

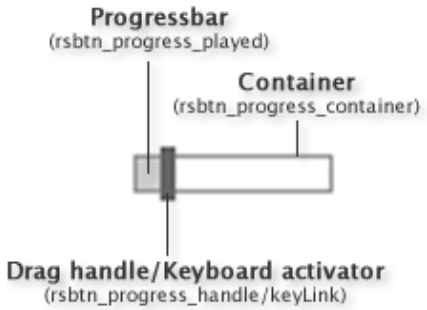
```
.rsbtn_sampleskin .rsbtn_progress_container {
    border: 1px solid #999;
    width: 50px;
    height: 8px;
    margin-top: 4px;
    margin-left: 4px;
    background-image: none;
}

.rsbtn_sampleskin .rsbtn_progress_handle.rsimg {
    position: absolute;
    background-color: #666;
    top: -3px;
    left: -2px;
    width: 4px;
    height: 14px;
    background-image: none;
}

.rsbtn_sampleskin .rsbtn_progress_played {
    position: absolute;
    height: 100%;
    background-color: #ccc;
    background-image: none;
}
```

Dolderseweg 2a, 3712 BP Huis ter Heide
T: 030-692 44 90, F: 030-692 45 99
E: info@readspeaker.com
www.readspeaker.com

The progress slider consists of four elements:



- The container, `.rsbtn_progress_container`. This is typically the area to which the drag handle is constrained.
- The drag handle, `.rsbtn_progress_handle`. This is a draggable handle that can be used to skip to different parts of the audio file. The drag handle's CSS-class can be changed using the JavaScript skin file:
  ```
  window.rsConf.ui.progressHandleClass = 'AClassName';
  ```
- The keyboard activator, `.keyLink`. This is an anchor tag which is used for activating the drag handle via the keyboard. It is usually not visible.
- The progressbar, `.rsbtn_progress_played`. This element can be used to visualize the current progress. Its width (expressed in percent) will be automatically updated every 500 milliseconds to reflect the current position in the audio stream.

Note! Both the drag handle and the progressbar need to be positioned absolutely in order for them to work.

When the drag handle (`.rsbtn_progress_handle`) is controlled by the keyboard, an additional class, `.rskeycontrolled` will be added to it. You can redefine this class in order to change the appearance of the drag handle to indicate that it is currently keyboard-controlled.

In the same way, the class `.dragged` will be added to the drag handle when it is being dragged, either with the mouse or the keyboard.

Dolderseweg 2a, 3712 BP Huis ter Heide
T: 030-692 44 90, F: 030-692 45 99
E: info@readspeaker.com
www.readspeaker.com

Volume

```
.rsbtn_sampleskin .rsbtn_volume {
    margin-left: 4px;
    background-position: 0 -70px;
}
```

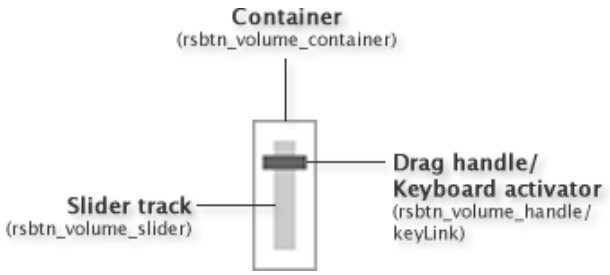
We set the volume button's background position and add a little margin to make it symmetrical to the other buttons. Clicking this element will toggle the visibility of the volume slider container.

Volume slider

```
.rsbtn_sampleskin .rsbtn_volume_container {
    position: absolute;
    display: none;
    top: 100%;
    width: 16px;
    height: 40px;
    border: 1px solid #999;
    background: #fff;
}

.rsbtn_sampleskin .rsbtn_volume_slider {
    width: 6px;
    height: 30px;
    margin: 5px;
    background: #ccc;
}

.rsbtn_sampleskin .rsbtn_volume_handle.rsimg {
    position: absolute;
    top: -2px;
    left: -3px;
    width: 12px;
    height: 4px;
    background: #666;
}
```



The volume slider's structure is similar to that of the progress slider, although the volume slider renders vertically by default. You need to take the direction into

Dolderseweg 2a, 3712 BP Huis ter Heide
T: 030-692 44 90, F: 030-692 45 99
E: info@readspeaker.com
www.readspeaker.com

consideration when defining the volume slider's styles.

You can change the direction of the volume slider in the skin JavaScript file:

```
window.rsConf.ui.volumeDir = 'v|h';
```

'v' for vertical and 'h' for horizontal.

We hide the container, because it will be made visible when the user clicks the volume button.

We give the drag handle negative top and left values because we want the center of the handle to start at the top of the slider track. The ReadSpeaker user interface controller will take this into account when rendering the slider component.

Note: Just like for the progress slider the drag handle must be positioned absolutely in order to work.

The `.rskeycontrolled` and `.dragged` classes will be added to the drag handle (`.rsbtn_volume_handle`) in the same way as for the progress slider. See the Progress Slider section for more information about this.

Settings Button

```
.rsbtn_sampleskin .rsbtn_settings {
    background-position: 0 -88px;
}
```

This button opens the settings dialog. All we do in the CSS is to change the background position so that the right icon is displayed.

Download Button

In this sample skin we have chosen to not include the download button. If you want to include it in your skin, you can define it using this selector:

```
.rsbtn_sampleskin .rsbtn_dl.rsimg { … }
```

Dolderseweg 2a, 3712 BP Huis ter Heide
T: 030-692 44 90, F: 030-692 45 99
E: info@readspeaker.com
www.readspeaker.com

Close Button

```
.rsbtn_sampleskin .rsbtn_closer {
    margin-left: 4px;
    background-position: 0 -106px;
}
```

We add a little extra left margin and set the background position. The close player event will be automatically assigned to this element.

```
.rsbtn_sampleskin.rspopup {
    position: absolute;
    background: #fff;
    border: 1px solid #555;
    box-shadow: 0 0 5px #777;
    display: none;
}
```

Finally, we add a rule that controls the popup button/popup player. The position statement is important in order to not disturb the document flow when the popup button is displayed. The box shadow is there to make the popup stand out from the rest of the page and make it more visible.