JW Player Quick Start Guide

Getting Started

Embedding the JW Player on your website is a simple, 3-step process:

1. Upload the *jwplayer.js* and *player.swf* files from the download ZIP to your server. All other files in the download (documentation, source code, etc) are optional.
2. Include the *jwplayer.js* somewhere in the head of your webpage:

```
<script type="text/javascript" src="/jwplayer/jwplayer.js"></script>
```

3. Call the player setup somewhere in the body of your website. Here's a basic example:

```
<div id="container">Loading the player ...</div>

<script type="text/javascript">
    jwplayer("container").setup({
        flashplayer: "/jwplayer/player.swf",
        file: "/uploads/video.mp4",
        height: 270,
        width: 480
    });
</script>
```

When the page is loading, the JW Player is automatically instantiated on top of the * *. By default, the player is rendered in Flash. If Flash is not supported (e.g. on an iPad), the player is rendered in HTML5.

The *flashplayer* option (to tell the JavaScript where the SWF resides) is just one of many configuration options available for configuring the JW Player.

Here's another setup example, this time using a *<video>* tag instead of a generic div:

```
<video
    src="/videos/video.mp4"
    height="270"
    id="container"
    poster="/thumbs/image.jpg"
    width="480">
</video>

<script type="text/javascript">
    jwplayer("container").setup({
        flashplayer: "/jwplayer/player.swf"
    });
</script>
```

In this case, the JW Player is actually inspecting <video> tag and loading its attributes as configuration options. It's a useful shortcut for setting up a basic player.

Quick Embed

If you've uploaded your *player.swf* and *jwplayer.js* files to a folder called "jwplayer" in the root of your website, you can embed the player by using two simple lines of HTML:

```
<script type="text/javascript" src="/jwplayer/jwplayer.js"></script>
<video class="jwplayer" src="/uploads/video.mp4"
poster="/uploads/image.jpg"></video>
```

That's it! As long as you have everything in the right place, all <video> tags on your page whose class is **jwplayer** will be replaced on your page by the JW Player.

Setup Syntax

Let's take a closer look at the syntax of the *setup()* call. It has the following structure:

```
jwplayer(container).setup({options});
```

In this block, the *container* is the DOM element(*<video>* or * *, *<p>*, etc.) you want to load the JW Player into. If the element is a*<video>* tag, the attributes of that tag (e.g. the *width* and *src*) are loaded into the player.
The *options* are the list of configuration options for the player. The configuration options guide contains the full overview. Here's an example with several of options:

```
<div id="container">Loading the player ...</div>

<script type="text/javascript">
    jwplayer("container").setup({
        autostart: true,
        controlbar: "none",
        file: "/videos/video.mp4",
        duration: 57,
        flashplayer: "/jwplayer/player.swf",
        volume: 80,
        width: 720
    });
</script>
```

Though generally a flat list, there are a couple of options that can be inserted as structured blocks inside the setup method. Each of these blocks allow for quick but powerful setups:

- **playlist**: allows inline setup of a full playlist, including metadata.
- **levels**: allows inline setup of multiple quality levels of a video, for bitrate switching purposes.
- **plugins**: allows inline setup of JW Player plugins, including their configuration options.
- **events**: allows inline setup of JavaScripts for player events, e.g. when you want to do something when the player starts.
- **modes**: allows inline setup of a custom mode fallback, e.g. HTML5 first, fallback to Flash

The sections below explain them in detail.

Skins

The JW Player has a wide variety of skins that can be used to modify the look and feel of the player. They can be downloaded from our AddOns Library.

To embed a JW Player 5 skin, simply place the ZIP file on your web server and add the *skin* property to your embed code:

```
<div id="container">Loading the player ...</div>

<script type="text/javascript">
    jwplayer("container").setup({
        flashplayer: "/jwplayer/player.swf",
        file: "/uploads/video.mp4",
        height: 270,
        width: 480,
        skin: "/skins/modieus/modieus.zip"
    });
</script>
```

Note: If you're configuring the Embedder to run primarily in HTML5 mode using the **modes** block, you'll need to take the additional step of unzipping the skin ZIP and uploading its contents to your web server in the same location as the ZIP file itself. Your skin's folder structure would look something like this:

```
 /skins/modieus/modieus.zip
 /skins/modieus/modieus.xml
 /skins/modieus/controlbar/
 /skins/modieus/playlist/
 etc.
```

Playlists

In previous Flash-only versions of the JW Player, loading a playlist in the JW Player was only available by using an XML playlist format like RSS or ATOM. With the JW Embedder, it is possible to load a full playlist into the player using the **playlist** object block.

Here is an example in which a playlist of three items is loaded into the player. Each item contains a **duration** hint, the **file** location and the location of a poster **image**.

```
<div id="container">Loading the player...</div>

<script type="text/javascript">
    jwplayer("container").setup({
        flashplayer: "/jwplayer/player.swf",
        playlist: [
            { duration: 32, file: "/uploads/video.mp4", image:
"/uploads/video.jpg" },
            { duration: 124, file: "/uploads/bbb.mp4", image:
"/uploads/bbb.jpg" },
            { duration: 542, file: "/uploads/ed.mp4", image:
"/uploads/ed.jpg" }
        ],
        "playlist.position": "right",
        "playlist.size": 360,
        height: 270,
        width: 720
    });
</script>
```

Note: The *playlist.position* and *playlist.size* options control the visible playlist inside the Flash player. To date, the HTML5 player doesn't support a visible playlist yet (though it can manage a playlist of videos).

A playlist can contain 1 to many videos. For each entry, the following properties are supported:

- **file**: (required, unless you have *levels*, see below). The location of the media. If this property is not set, the playlist item will be skipped.
- **image**: location of the poster image, displayed before the video starts, after it finishes, and as part of the graphical playlist.

- **duration**: duration of the video, in seconds. The player uses this to display the duration in the controlbar, and in the graphical playlist.
- **start**: starting point inside the video. When a user plays this entry, the video won't start at the beginning, but at the offset specified here.
- **title**: title of the video, displayed in the graphical playlist.
- **description**: description of the video, displayed in the graphical playlist.
- **streamer**: streaming application to use for the video. This is only used for RTMP or HTTP streaming.
- **provider**: specific media playback API (e.g. *http*, *rtmp* or *youtube*) to use for playback of this playlist entry.
- **levels**: a nested object block, with multiple quality levels of the video. See the *levels* section for more info.

Levels

The **levels** object block allows you to load multiple quality levels of a video into the player. The multiple levels are used by the Flash player for RTMP or HTTP bitrate switching. Bitrate switching is a mechanism where the player automatically shows the best possible video quality to each viewer.

Here's an example setup, using RTMP bitrate switching (also called *dynamic streaming*). Note the additional *streamer* option, which tells the player the location of the RTMP server:

```
<div id="container">Loading the player...</div>

<script type="text/javascript">
    jwplayer("container").setup({
        flashplayer: "/jwplayer/player.swf",
            height: 270,
            width: 480,
            image: "/thumbs/video.jpg",
            levels: [
                { bitrate: 300, file: "videos/video_300k.mp4", width: 320 },
                { bitrate: 600, file: "videos/video_600k.mp4", width: 480 },
                { bitrate: 900, file: "videos/video_900k.mp4", width: 720 }
            ],
            provider: "rtmp",
            streamer: "rtmp://rtmp.example.com/application/"
    });
</script>
```

Here is another example setup, this time using HTTP bitrate switching. The HTTP switching is enabled by setting the *provider*option to *http*:

```
<div id="container">Loading the player...</div>

<script type="text/javascript">
    jwplayer("container").setup({
        flashplayer: "/jwplayer/player.swf",
        height: 270,
        width: 480,
        image: "/thumbs/video.jpg",
        levels: [
            { bitrate: 300, file: "http://example.com/videos/video_300k.mp4",
width: 320 },
            { bitrate: 600, file: "http://example.com/videos/video_600k.mp4",
width: 480 },
            { bitrate: 900, file: "http://example.com/videos/video_900k.mp4",
width: 720 }
        ],
        provider: "http",
        "http.startparam":"starttime"
    });
</script>
```

Using Levels in HTML5 Mode

You can also use the *levels* block in HTML5 mode to specify alternate video formats for support in various browsers. If the viewer's browser doesn't support the first item in *levels*, the player will try to play the second item, and so on. Here's an example using *levels* which specifies multiple encodings for the same file:

```
<div id="container">Loading the player...</div>

<script type="text/javascript">
    jwplayer("container").setup({
        flashplayer: "/jwplayer/player.swf",
        height: 270,
        width: 480,
        image: "/thumbs/video.jpg",
        levels: [
            { file: "/videos/video.mp4" },  // H.264 version
            { file: "/videos/video.webm" }, // WebM version
            { file: "/videos/video.ogv" }   // Ogg Theroa version
        ]
    });
</script>
```

Plugins

Plugins can be used to stack functionality on top of the JW Player. A wide array of plugins is available in our library, for example for viral sharing, analytics or advertisements.
Here is an example setup using both the HD plugin and the Google Analytics Pro plugin:

```
<div id="container">Loading the player...</div>

<script type="text/javascript">
    jwplayer("container").setup({
        flashplayer: "/jwplayer/player.swf",
        file: "/videos/video.mp4",
        height: 270,
        plugins: {
            hd: { file: "/videos/video_high.mp4", fullscreen: true },
            gapro: { accountid: "UKsi93X94O-24" }
        },
        image: "/thumbs/video.jpg",
        width: 480
    });
</script>
```

Here is another example, using the sharing plugin. In this example, plugin parameters are also included in the playlist block:

```
<div id="container">Loading the player...</div>

<script type="text/javascript">
    jwplayer("container").setup({
        flashplayer: "/jwplayer/player.swf",
        playlist: [
            { file: "/videos/bunny.mp4", "sharing.link":
"http://bigbuckbunny.org" },
            { file: "/videos/ed.mp4", "sharing.link":
"http://orange.blender.org" }
        ],
        plugins: {
            sharing: { link: true }
        },
        height: 270,
        width: 720
    });
</script>
```

Events

The **events** block allows you to respond on player events in JavaScript. It's a short, powerful way to add player - pager interactivity. Here is a swift example:

```
<div id="container">Loading the player ...</div>

<script type="text/javascript">
    jwplayer("container").setup({
        flashplayer: "/jwplayer/player.swf",
        file: "/videos/video.mp4",
        height: 270,
        width: 480,
        events: {
            onComplete: function() { alert("the video is finished!"); }
        }
    });
</script>
```

Here is another example, with two event handlers. Note the *onReady()* handler autostarts the player using the ***this*** statement and the *onVolume()* handler is processing an event property:

```
<div id="container">Loading the player ...</div>

<script type="text/javascript">
    jwplayer("container").setup({
        flashplayer: "/jwplayer/player.swf",
        file: "/videos/video.mp4",
        height: 270,
        width: 480,
        events: {
            onReady: function() { this.play(); },
            onVolume: function(event) { alert("the new volume is
"+event.volume); }
        }
    });
</script>
```

See the API reference for a full overview of all events and their properties.

Modes

The **modes** option block can be used to customize the order in which the JW Player uses the different browser technologies for rendering the player. By default, the JW Player uses this order:

1. The **Flash** plugin.

2. The **HTML5** <video> tag.
3. A **Download** link to the original file.

Using the **modes** block, it is possible to specify that the Embedder try the HTML5 player first:

```
<div id="container">Loading the player ...</div>

<script type="text/javascript">
    jwplayer("container").setup({
        file: "/videos/video.mp4",
        height: 270,
        width: 480,
        modes: [
            { type: "html5" },
            { type: "flash", src: "/jwplayer/player.swf" },
            { type: "download" }
        ]

    });
</script>
```

Mode-specific Configuration (Version 5.5+)

It is possible to specify alternate player configurations for each mode. These will override the default configuration settings if the player is embedded in a specific mode. Here's an example of an RTMP stream with a progressive-download fallback for HTML5:

```
<div id="container">Loading the player ...</div>

<script type="text/javascript">
    jwplayer("container").setup({
        height: 270,
        width: 480,
        image: "http://server.com/images/thumbnail.jpg",
        modes: [
            { type: "flash",
              src: "/jwplayer/player.swf",
                config: {
                  file: "video.mp4",
                  streamer: "rtmp://rtmp.server.com/videos",
                  provider: "rtmp"
                }
            },
            { type: "html5",
                config: {
                  file: "http://server.com/videos/video.mp4"
                }
            },
            { type: "download" }
        ]
    });
</script>
```

Player Removal

In addition to setting up a player, the JW Player embed script contains a function to unload a player. It's very simple:

```
jwplayer("container").remove();
```

This formal **remove()** function will make sure the player stops its streams, the DOM is re-set to its original state and all event listeners are cleaned up.