Listing 6-3. Setting Up the Asset Loader (common.js)

```
var loader = {
    loaded: true,
    loadedCount: 0, // Assets that have been loaded so far
    totalCount: 0, // Total number of assets that need loading

    init: function() {
        // Check for sound support
        var mp3Support, oggSupport;
        var audio = document.createElement("audio");

        if (audio.canPlayType) {
            // Currently canPlayType() returns: "", "maybe", or "probably"
            mp3Support = "" !== audio.canPlayType("audio/mpeg");
            oggSupport = "" !== audio.canPlayType("audio/ogg; codecs=\"vorbis\"");
        } else {
            // The audio tag is not supported
            mp3Support = false;
            oggSupport = false;
        }

        // Check for ogg, then mp3, and finally set soundFileExtn to undefined
        loader.soundFileExtn = oggSupport ? ".ogg" : mp3Support ? ".mp3" : undefined;
    },

    loadImage: function(url) {
        this.loaded = false;
        this.totalCount++;

        game.showScreen("loadingscreen");

        var image = new Image();

        image.addEventListener("load", loader.itemLoaded, false);
        image.src = url;

        return image;
    },

    soundFileExtn: ".ogg",

    loadSound: function(url) {
        this.loaded = false;
        this.totalCount++;

        game.showScreen("loadingscreen");

        var audio = new Audio();
```

```
        audio.addEventListener("canplaythrough", loader.itemLoaded, false);
        audio.src = url + loader.soundFileExtn;

        return audio;
    },

    itemLoaded: function(ev) {
        // Stop listening for event type (load or canplaythrough) for this item now that it
        has been loaded
        ev.target.removeEventListener(ev.type, loader.itemLoaded, false);

        loader.loadedCount++;

        document.getElementById("loadingmessage").innerHTML = "Loaded " + loader.loadedCount +
        " of " + loader.totalCount;

        if (loader.loadedCount === loader.totalCount) {
            // Loader has loaded completely
            // Reset and clear the loader
            loader.loaded = true;
            loader.loadedCount = 0;
            loader.totalCount = 0;

            // Hide the loading screen
            game.hideScreen("loadingscreen");

            //and call the loader.onload method if it exists
            if (loader.onload) {
                loader.onload();
                loader.onload = undefined;
            }
        }
    }
};
```

Next, we will define our game object inside game.js, as shown in Listing 6-4.

Listing 6-4. Defining the game Object (game.js)

```
var game = {

    // Start initializing objects, preloading assets, and display start screen
    init: function() {
        // Initialize game objects
        loader.init();

        // Display the main game menu
        game.hideScreens();
        game.showScreen("gamestartscreen");
    },
```

```
    hideScreens: function() {
        var screens = document.getElementsByClassName("gamelayer");

        // Iterate through all the game layers and set their display to none
        for (let i = screens.length - 1; i >= 0; i--) {
            let screen = screens[i];

            screen.style.display = "none";
        }
    },

    hideScreen: function(id) {
        var screen = document.getElementById(id);

        screen.style.display = "none";
    },

    showScreen: function(id) {
        var screen = document.getElementById(id);

        screen.style.display = "block";
    },

    scale: 1,
    resize: function() {

        var maxWidth = window.innerWidth;
        var maxHeight = window.innerHeight;

        var scale = Math.min(maxWidth / 640, maxHeight / 480);

        var gameContainer = document.getElementById("gamecontainer");

        gameContainer.style.transform = "translate(-50%, -50%) " + "scale(" + scale + ")";

        game.scale = scale;

        // What is the maximum width we can set based on the current scale
        // Clamp the value between 640 and 1024
        var width = Math.max(640, Math.min(1024, maxWidth / scale ));

        // Apply this new width to game container and game canvas
        gameContainer.style.width = width + "px";
    },
};

/* Set up initial window event listeners */

// Initialize and resize the game once page has fully loaded
window.addEventListener("load", function() {
    game.resize();
    game.init();
}, false);
```

```
// Resize the game any time the window is resized
window.addEventListener("resize", function() {
    game.resize();
});
```

In this code, we create a game object with an `init()` method that first initializes our asset loader and then uses the `hideScreens()` and `showScreen()` methods to display the game start screen.

We also define a `resize()` method just like we did in our previous game. The method calculates the scale and maximum possible width for our game and sets the `gamecontainer` style accordingly.

Finally, we add event listeners to the `window` object to call these methods. We call `game.resize()` and `game.init()` once the window has loaded completely. We also call `game.resize()` whenever the window is resized.

Next, we need to append the CSS for the game starting screen and loading screen in `styles.css`, as shown in Listing 6-5.

Listing 6-5. Style for the Game Starting Screen and Loading Screen (styles.css)

```
/* Game Starting Menu Screen */

.game-title {
    position: absolute;

    top: 5%;
    right: 5%;
    text-align: right;
    width: 100%;

    font-family: "Courier New", Courier, monospace;
    font-size: 90px;
    line-height: 80px;

    color: white;
    text-shadow: -2px 0 purple, 0 2px purple, 2px 0 purple, 0 -2px purple;
}

.game-option {
    position: relative;
    top: 65%;
    left: 10%;
    display: block;

    font-family: "Courier New", Courier, monospace;
    font-size: 48px;
    color: white;
    text-shadow: -2px 0 purple, 0 2px purple, 2px 0 purple, 0 -2px purple;

    cursor: pointer;
}
```

```
.game-option:hover {
    color: yellow;
}

/* Loading Screen */

#loadingscreen {
    background: rgba(100, 100, 100, 0.7);
}

#loadingmessage {
    position: relative;
    top: 400px;
    text-align: center;

    height: 48px;
    color: white;
    background: url("images/loader.gif") no-repeat center;
    font: 12px Arial;
}
```

When we open the game in the browser, we should see the starting screen with the main menu, as shown in Figure 6-2.

Figure 6-2. *The starting screen with the main menu*

You will notice that the game automatically scales and widens to fit the available screen area. The extra hidden portion of the splash screen image also automatically becomes visible as necessary.

The menu currently offers options for Campaign, which is our story-based single-player mode, and Multiplayer, which is our player-versus-player mode. You may have noticed in Listing 6-1 that the `onclick` handlers for these two options call the `singleplayer.start()` and `multiplayer.start()` methods, respectively. Right now, clicking the Campaign option won't do anything since we haven't yet implemented the `singleplayer.start()` method to start the single-player levels.

Before we can do so, however, we need to create our first single-player level.