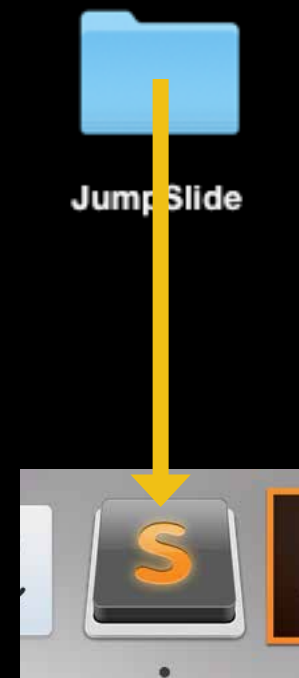# JumpSlide

## Setup (Mac)

# Retrieve Project Files ( Mac Only* )

1.  **Download the JumpSlide Project Starter files from**
    `stem.gomagames.com/jumpslide`

2.  **Unzip the project files. Move the unzipped folder to your Desktop.**

# Workspace Setup ( Mac Only* )

Open the **JumpSlide-master** directory in Sublime Text. Be sure to open the whole folder, not just a single file. To Open the entire folder...

1.  **Open the Sublime Text application.**

2.  **Drag your project folder onto the Sublime Text icon in the dock.**

3.  **Your entire folder should open with a sidebar that shows all the project files. If you don't see a sidebar with files, go to View > Sidebar > Show Sidebar.**

*PCs and Macs, you can also open SublimeText and open the project from the Menu.
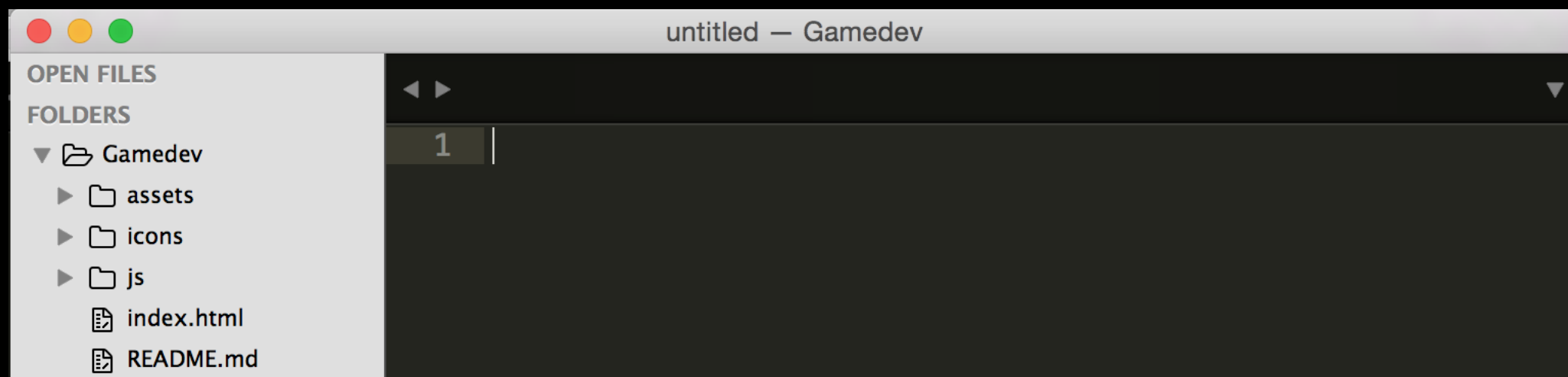
**/JumpSlide**  |  This is your main project directory.

  **index.html**  |  This is the file your game loads into.

  **/js**  |  This directory holds all the javascript files.

    **level0.js**  |  This is the main file you will be editing.

    **JumpSlide.js**  |  This is the where the game logic is stored.

    **pixi.js**  |  This is the Game Engine. Leave it alone!

    **howler.min.js**  |  This is the Sound Effects Engine. Leave it alone!

  **/assets**  |  This directory holds all the graphics and sound effects.

  **/icons**  |  This directory holds icons (for Apple iPad & iPhone).

  **README.md**  |  This is documentation. It is easier to browse at
                    https://github.com/gomagames/jumpslide

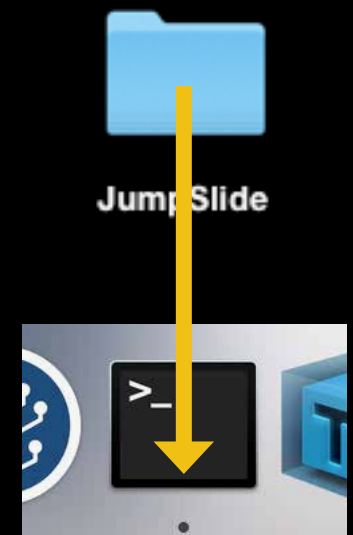  **JumpSlide-Slideshow.pdf**  |  Step-by-Step Instructions

# Start a Server

The files require a local server in order to run the game. To start a server:

1. **Drag your project folder onto the Terminal App in the dock.**
   **This will navigate directly to the folder containing your game files.**

2. **Type the following command:**

   `python -m SimpleHTTPServer`

3. **Hit Enter. If you did it correctly, after a few seconds, you should see something similar to this:**

# View the Game in Chrome

To view the game in Google Chrome:

1.  **Open a new Google Chrome browser.**

2.  **Type the following url in the address bar:**

    `http://localhost:8000`

3.  **You should see a black background.**

# General Programming Workflow

Whenever you add a chunk of code, test it immediately!

1.  **Save the code you typed.**

2.  **Switch to your Google Chrome browser that contains your game.**

3.  **Refresh the page by hitting Command-R**

tab has an x if
saved.

game.js

game.js

Is something broken? If something is broken, Inspect the Error.

1.  **In Chrome, right-click (Control-Click) the game page and choose Inspect Element from the dropdown.**

2.  **Click Console in the nav. If there any errors in your code if will show along with hints to the line number of the error.**

# Start Coding!

Follow the slides in this tutorial. After every slide, be sure to test if the code works!
All coding is done in the **level0.js** file.


To begin programming, click the **/js** directory and then the **level0.js** file in Sublime Text.
Study the structure of the file so you have a general idea of where you will be adding code.

# Start the Game!

In the Game.init block, add the following code that starts the game. Soon, we will be adding more code in this block. The **JumpSlide.start();** function call should always be at the END of the block. All new code in the Game.init block should be above this code.

```
GAME.init = function (JumpSlide) {
    localStorage.coins = "0";

    // Start the Game
    JumpSlide.start();

}
```

Test in Chrome!

# Create a Sample Game Level

We will start by adding a demo game level. Later, replace the demo level with your own level. In the `Game.init` block, add some floors and platforms to the map. Always add visible game objects ABOVE the JumpSlide.start(); code.

```
GAME.init = function (JumpSlide) {

    localStorage.coins = "0";


    /* Demo Level */

    // Floors
    JumpSlide.addPlatform(50, 700, 300, 68);
    JumpSlide.addPlatform(480, 700, 2000, 68);


    // Short Platform
    JumpSlide.addPlatform(750, 580, 200, 45);


    // Start the Game
    JumpSlide.start();
```

Test in Chrome!

# Add Some Coins

In the Game.init block, add some demo coins to the map:

```
// Floors
JumpSlide.addPlatform(50, 700, 300, 68);
JumpSlide.addPlatform(480, 700, 2000, 68);

// Short Platform
JumpSlide.addPlatform(750, 580, 200, 45);

// Coins
JumpSlide.addCoin(850, 517);
JumpSlide.addCoin(850, 660);
JumpSlide.addCoin(1200, 637);
JumpSlide.addCoin(1630, 517);

// Start the Game
```

Test in Chrome!
You won't see all the coins yet because the game doesn't scroll yet!

# Add a Goal

In the Game.init block, add a Goal to the map:

```
JumpSlide.addCoin(850, 660);
JumpSlide.addCoin(1200, 637);
JumpSlide.addCoin(1630, 517);

// Goal
JumpSlide.addGoal(2100, 665);

/* End Demo Level */


// Start the Game
JumpSlide.start();

}
```

Test in Chrome!
You  won't see the goal yet because the game doesn't scroll yet!

# Making the Player Run - Part 1 of 3

In the `Game.loop` block, add the following code to start making the player move. To make it look like the player is running, we actually need to move all the other game elements to the **left** to simulate the player moving **right**. First, we move the platforms left:

```javascript
GAME.loop = function (JumpSlide) {


  // make the player run right
  // by looping through each platform
  JumpSlide.forEachPlatform(function(platform) {
    // translate it's x position
    platform.position.x -= JumpSlide.SETTINGS.run_speed;
  });


}
```

Test in Chrome!

# Making the Player Run - Part 2 of 3

Then, we move the Coins left:

```javascript
// make the player run right
// by looping through each platform
JumpSlide.forEachPlatform(function(platform) {
  // translate it's x position
  platform.position.x -= JumpSlide.SETTINGS.run_speed;
});

// loop through each coin
JumpSlide.forEachCoin(function(coin) {
  // translate it's x position
  coin.position.x -= JumpSlide.SETTINGS.run_speed;
});

}
```

Test in Chrome!

# Making the Player Run - Part 3 of 3

Finally, we make the goal move left:

```javascript
// make the player run right
// loop through each coin
JumpSlide.forEachCoin(function(coin) {
  // translate it's x position
  coin.position.x -= JumpSlide.SETTINGS.run_speed;
});

// loop through each goal
JumpSlide.forEachGoal(function(goal) {
  // translate it's x position
  goal.position.x -= JumpSlide.SETTINGS.run_speed;
});

}
```

Test in Chrome!
You actually cannot get to the goal yet since the character cannot jump and he runs into a platform.

# Making the Player Jump

Find the **GAME.tap** block. Add code to make the player jump:

```
GAME.tap = function ( point ) {

  // check if player touches the top part of the screen
  if( point.y < JumpSlide.SETTINGS.controls.up ){

    // make player jump
    JumpSlide.player.jump();
  }
}
```

Test in Chrome!
Now if you click above the center of the screen, the character jumps.

# Making the Player Slide

Find the `GAME.touch_start` block. Add code to make the player slide:

```
GAME.touch_start = function ( point ) {

  // check if player touches bottom part of screen
  if( point.y > JumpSlide.SETTINGS.controls.down ){


    // make player start sliding
    JumpSlide.player.slide();
  }
}
```

Test in Chrome!
Now if you click below the center of the screen, the character crouches down.

# Making the Player Stop Sliding

Find the `GAME.touch_end` block. Add code to make the player stop sliding when you release the mouse button after initiating a slide action.

```
GAME.touch_end = function ( point ) {

    JumpSlide.player.stop_sliding();

}
```

Test in Chrome!
When the character is sliding, he now stands up again when you release the mouse button.

# Add the Ability to Collect Coins

Back in the `GAME.loop` block, find the section starting with `// loop through each coin` and add the following code after the line ending with `.run_speed;`

```
// loop through each coin
JumpSlide.forEachCoin(function(coin) {
    // translate it's x position
    coin.position.x -= JumpSlide.SETTINGS.run_speed;

    // check if player is touching a coin
    if(JumpSlide.player.check_collision(coin)){

        // collect the coin to score
        JumpSlide.collectCoin(coin);
    }
});
```

Test in Chrome!

# Add Collect Coin SFX

After the `JumpSlide.collectCoin(coin);` line, add the following code to add the coin sfx:

```
    // check if player is touching a coin

    if(JumpSlide.player.check_collision(coin)){


        // collect the coin to score

        JumpSlide.collectCoin(coin);


        // play coin sound effects

        JumpSlide.sfx.coin.play();
    }
});
```

Test in Chrome!

# Add the Victory Condition

In the `GAME.loop()` block, find the block starting with `// loop through each goal` and add the following code after the line ending with `.run_speed;`

```
    // loop through each goal
  JumpSlide.forEachGoal(function(goal) {

      // translate it's x position
      goal.position.x -= JumpSlide.SETTINGS.run_speed;

      // victory condition
      if(JumpSlide.player.check_collision(goal)){
          JumpSlide.game_win();
      }

  });
```

Test in Chrome!

# Add the Lose Condition

In the `GAME.loop()` block, add code after the victory condition:

```
    goal.position.x -= JumpSlide.SETTINGS.run_speed;


    // victory condition
    if(JumpSlide.player.check_collision(goal)){
    JumpSlide.game_win();
    }



    // check if player falls, then lose game
    if( JumpSlide.player.position.y >= JumpSlide.SETTINGS.ipad_dimensions.height ){
        JumpSlide.game_lose();
    }


});
```
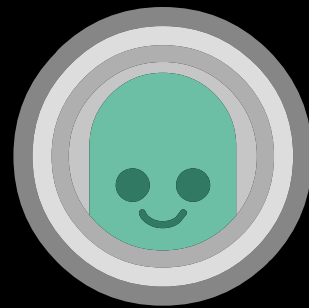
Test in Chrome!
You might not see anything change, however this adds logic that causes the player to lose if the character falls off the screen.

# JumpSlide

## YOUR Game Level

# Platforms

The game level you constructed earlier needs to be translated to a set of instructions written in code. The function call for placing a platform on the screen follows the format below.

A Platform looks like a rectangle in your diagram:

**( 50, 700 )**

| A | 300 x 68 |
|---|----------|

Placing a Platform with Code:
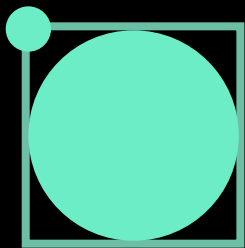
```
// Platform A
JumpSlide.addPlatform(50, 700, 300, 68);
                      x    y    width  height
```

# Coins

The function call for placing a coin on the screen follows the format below. Coins are always 48 x 48, so they don't need dimensions.

A Coin looks like a square in your diagram:

**( 50, 700 )**
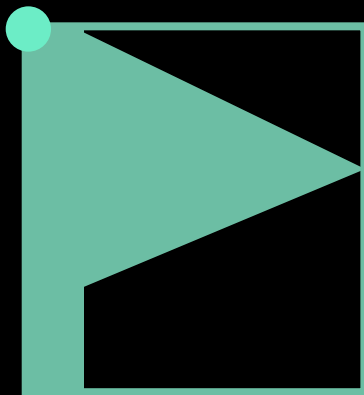
Placing a Coin with Code:

```
// Coin
JumpSlide.addCoin(50, 700);
              x       y
```

# Goals

The function call for placing a goal (flag) on the screen follows the format below. Goals are always 68 x 71, so they don't need dimensions.

A Goal looks like a small rectangle in your diagram:

**( 50, 700 )**

Placing a Goal with Code:

```
// Goal
JumpSlide.addGoal(50, 700);
              x       y
```

# Program Your Game Level

All the code for your level goes in the GAME.init block.

First, delete all the demo level code from */* Demo Level *//* to */* End Demo Level *//*, and replace it with code for your level.
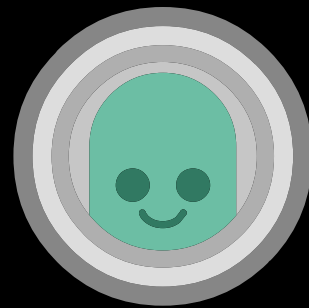
```
GAME.init = function (JumpSlide) {

    localStorage.coins = "0";


    /* Level 0  */




    /* End Level 0 */

    // Start the Game
    JumpSlide.start();
```

# JumpSlide

## Advanced Stuff

# Extra 1: Add Variables to Your Level

If you want to re-use some values throughout your game level, you can create some settings for your objects. Inside the GAME.init block, at the TOP you can define your own variables, then throughout your game level you can reuse those variables.

```
// Settings
var platform_width = 200;
var platform_height = 50;
var floor_height = 68;
var floor_y = 700;

// Level 0 Variables
JumpSlide.addPlatform(50, floor_y, 500, floor_height);
JumpSlide.addPlatform(650, floor_y, 500, floor_height);
JumpSlide.addPlatform(1200, 600, platform_width, platform_height);
```

# Extra 2: Modify JumpSlide.js

Refer to the API guide for how to use all the game settings and functions:

**stem.gomagames.com/jumpslide**

If you want to adjust the speed settings, you need to do so **early in your planning stages**. If you adjust speed settings after you design a level, you will find that speed greatly affects how far apart objects are to the character that is moving.
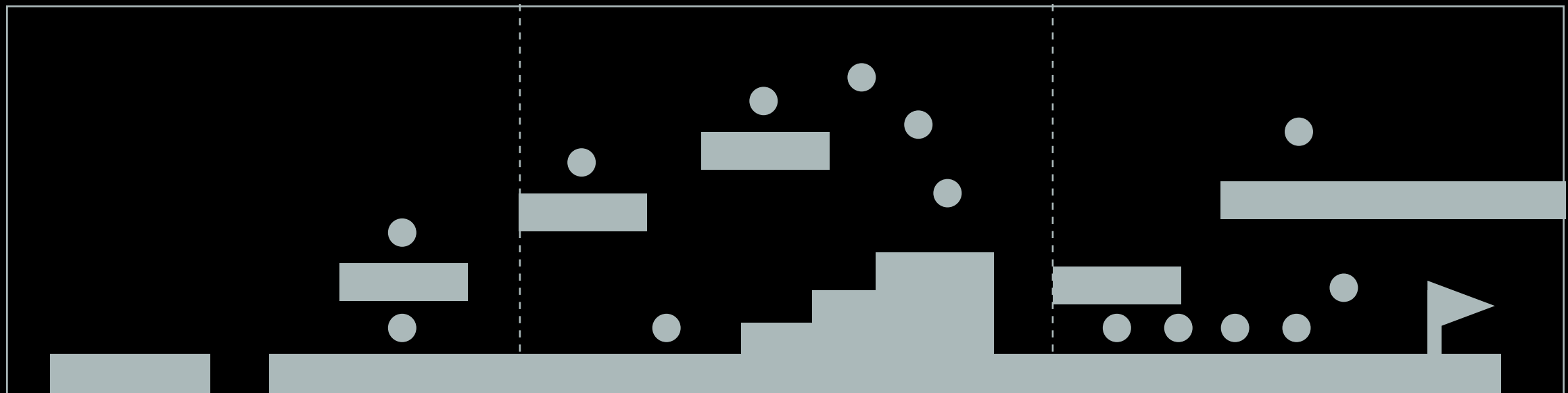
# Extra 3: Swap Out Assets

If you want to be experimental, you can swap out the graphics & sound. The entire set of game graphics and Adobe Illustrator mockups can be downloaded at **stem.gomagames.com/jumpslide**.

All the game art and sfx were found on **opengameart.org**.

If you want to be really adventurous, you can modify the code in JumpSlide.js. Careful! Be sure you know what you are doing in here and make a lot of backups!
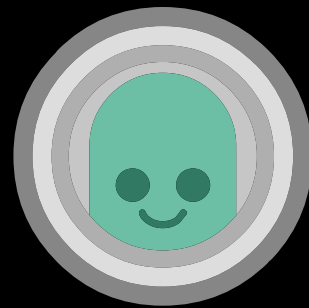
# JumpSlide

## Extra 4: Adding More Levels

# Modify level0.js

Go to the end of the level0.js file, and find the **GAME.win** block of code. **Change the counter from "0" to "1".** This number tells what level to load when the player "wins" the current level. Since we are on level0.js, the next level is level1.js, thus the counter "1".

```
GAME.win = function (JumpSlide) {
    // set sessionStorage counter ++
    // this allows you to add more than one game level
    localStorage.counter = "1";
}
```

# Set up level1.js

1. Select all the code on the **level0.js file**. Copy the code.

2. Make a new file, Paste the code.

3. Save the file in the js folder and name it **level1.js**

4. Go to the very top of the level1.js file, and **change the counter from "0" to "1"**

```
if(localStorage.counter === "1"){
```

# Set the Next Level

Read this carefully! Go to the end of the level1.js file, and find the **GAME.win** block of code.

--------------------------------------------------------------------------------

If you have made another level after level1.js, **change the counter from "1" to "2"**

```
GAME.win = function (JumpSlide) {
    // advance to this level when you win
    localStorage.counter = "2";
}
```

--------------------------------------------------------------------------------

OR If there is NOT another level after level1.js, then change the entire contents of the win function to:

```
GAME.win = function (JumpSlide) {
    localStorage.counter = "0";
    alert('You Win! Total score was '+ localStorage.coins);
}
```

# index.html

1. Open index.html

2. Add a script line to load level1.js.

```
<script src="js/pixi.js"></script>
<script src="js/howler.min.js"></script>
<script src="js/level0.js"></script>
<script src="js/level1.js"></script>
<script src="js/JumpSlide.js"></script>
```

# Turn on the Lose Game Alert

In the JumpSlide.js file, find line 490, where it has an alert for losing the game commented out. UNcomment the alert so there are no slashes in front of it:

```
    GAME.lose( JumpSlide );

    JumpSlide.sfx.death.play();

    alert('You Lose! Score: '+ localStorage.coins);

}
```