



JumpSlide



JumpSlide

Level Design

Level Design

Using the paper prototype tools (stem.gomagames.com/jumpslide), construct a paper version of the game level that you would want to build.

Platforms

- floating platforms
- floors
- tall walls
- pits, traps
- tunnels
- low hanging walls

Coins

- when collected, these add to your game score

Goal Flags

- touching a flag ends the game
- flags can be used as a final goal to reach or a diversion to avoid



JumpSlide

Setup (Mac)

Retrieve Project Files (Mac Only*)

1. Open the **Terminal** application.
(Applications > Utilities > Terminal)

2. Type this in the **Terminal**.
cd Desktop

3. Hit **Enter**. Terminal should reply with the word “Desktop” in the line.

4. Type this (all one line)*
**mkdir JumpSlide && curl -L
https://github.com/GomaGames/JumpSlide/tarball/master | tar
-xzv -C JumpSlide --strip-components=1**

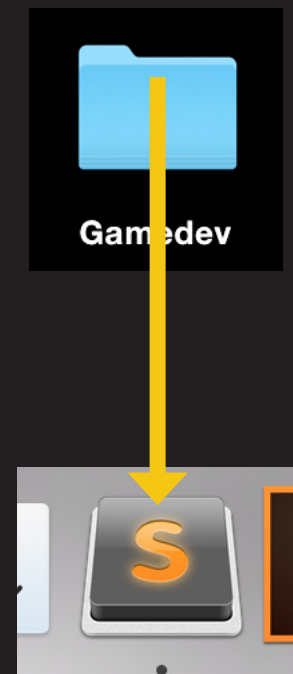
3. Hit **Enter**. You should have the **JumpSlide** project folder on your Desktop.

*PCs and Macs can also download project files directly from stem.gomagames.com/jumpslide

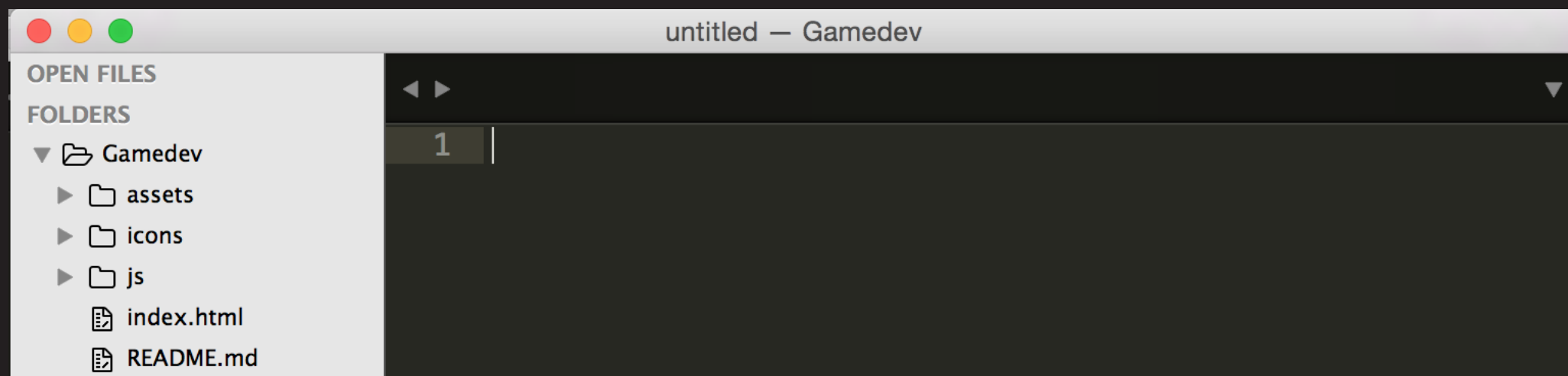
Workspace Setup (Mac Only*)

Open the **JumpSlide** 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.

game.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

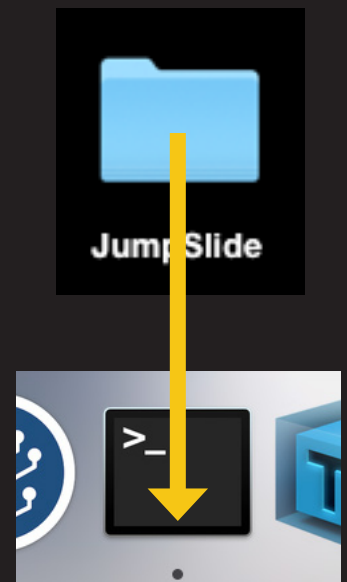
Start a Server

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:



```
JumpSlide — Python — 80x24
Last login: Thu Apr 16 18:38:58 on ttys001
Pentacorn:~ kelli$ cd Desktop/JumpSlide
Pentacorn:JumpSlide kelli$ python -m SimpleHTTPServer
Serving HTTP on 0.0.0.0 port 8000 ...
```


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**



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 it will show along with hints to the line number of the error.**

Add Yourself as the Game Author

1. Click on **index.html**
2. At the top of the file, you will see:
`<!-- AUTHOR:Your Name -->`
3. Change “Your Name” to your actual name. It should look something like this:
`<!-- AUTHOR:Kelli Borgonia -->`
4. Save.

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 **game.js** file.

To begin programming, click the **/js** directory and then the **game.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) {  
  
    // 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) {  
  
    /* 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);

}
```

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);  
  
}
```

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:

```
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:

```
// 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:

```
// 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();  
}
```

don't hit enter.
this is a
continuous line



```
// 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

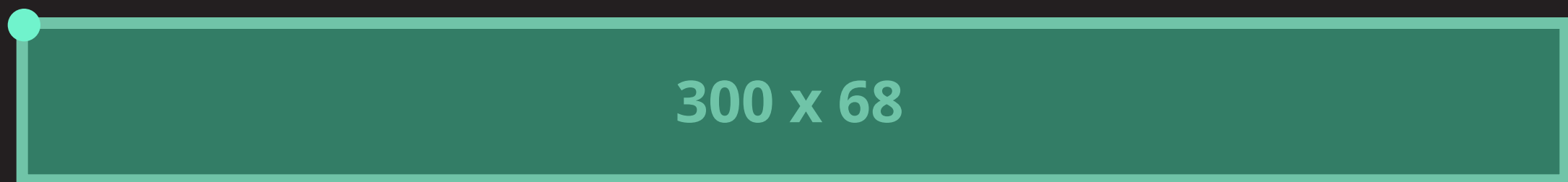
YOUR Game Level

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)



Placing a Platform with Code:

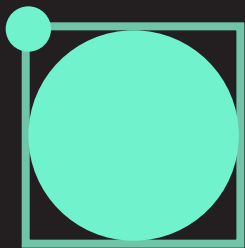
	x	y	width	height
JumpSlide.addPlatform	(50,	700,	300,	68);

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:

```
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:

```
JumpSlide.addGoal(50, 700);
```

Your Game Level

All the code for your level goes in the GAME.init block. You need to delete all the sample level code, and replace it with code for your level.

```
GAME.init = function (JumpSlide) {  
  
    /* Your Level */  
  
    // Start the Game  
    JumpSlide.start();  
  
}
```

Level Design to Code (Advanced)

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;
```

```
// Demo Game Level
```

```
JumpSlide.addPlatform(50, floor_y, 500, floor_height);
```

```
JumpSlide.addPlatform(650, floor_y, 500, floor_height);
```

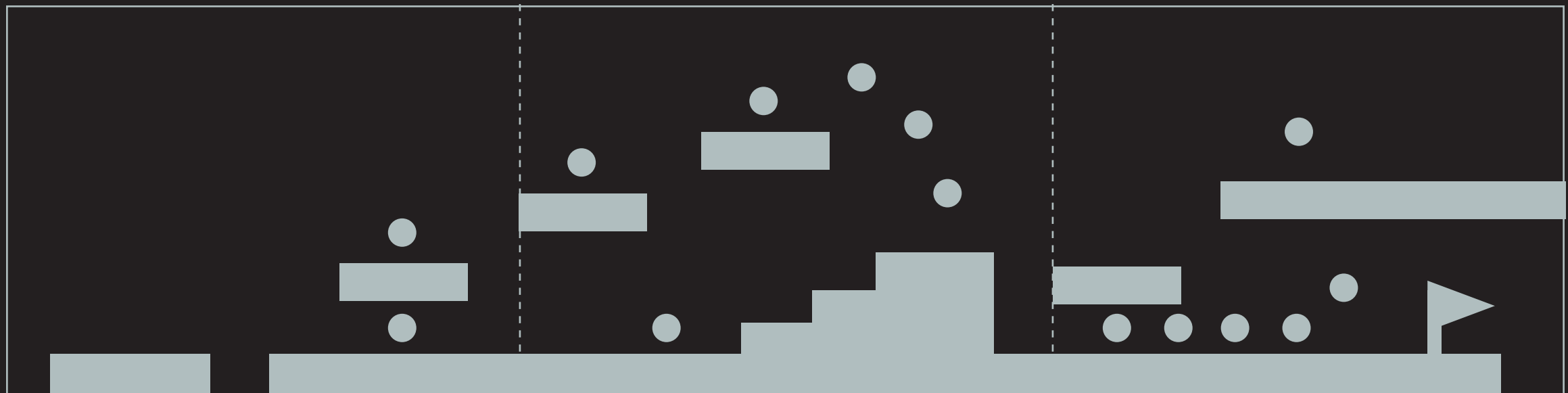
```
JumpSlide.addPlatform(1200, 600, platform_width, platform_height);
```


Level Design (Advanced)

You can also refer to the API guide for how to use all the game settings and functions:

<https://github.com/GomaGames/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.



Experimentation

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 <http://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!