



JumpSlide

Retrieve Project Files

1. Open the **Terminal** application.

2. Type (this is all one line):

```
bash < <(curl -s http://stem2015.gomagames.com/setup)
```

3. Hit **Enter**. Lines should appear in Terminal ending with “all done”

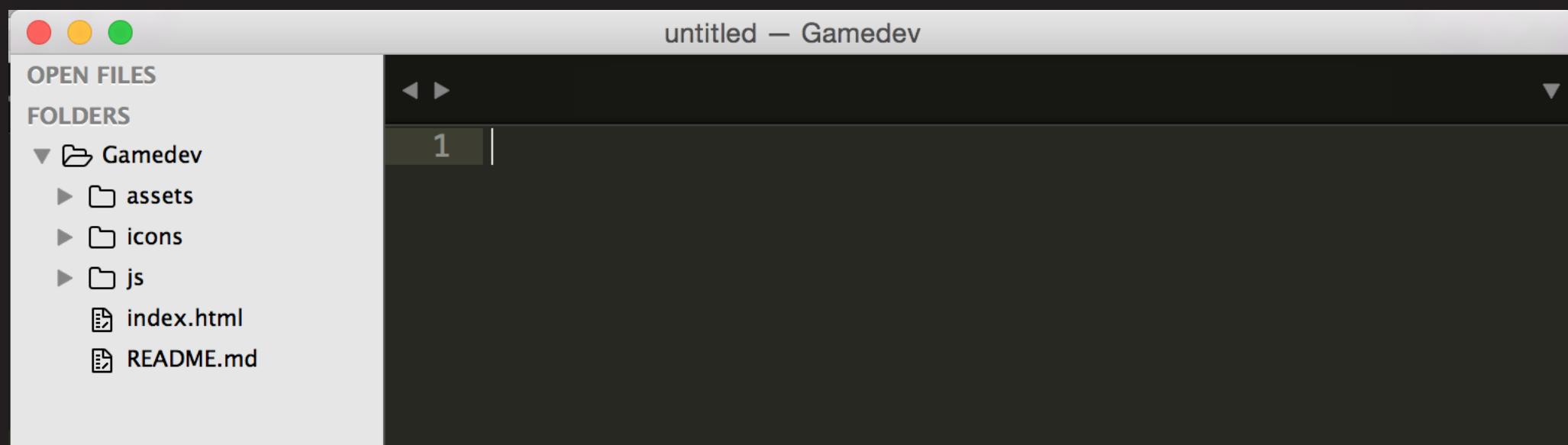
```
creating: /Users/kelli/Desktop/JumpSlide-master/js/
inflating: /Users/kelli/Desktop/JumpSlide-master/js/JumpSlide.js
inflating: /Users/kelli/Desktop/JumpSlide-master/js/game.js
inflating: /Users/kelli/Desktop/JumpSlide-master/js/howler.min.js
inflating: /Users/kelli/Desktop/JumpSlide-master/js/pixi.js
installing deploy command
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
           %         %         Dload  Upload   Total   Spent    Left   Speed
100  769 100  769    0     0  1176      0 --:--:-- --:--:-- --:--:-- 1175
setting up ssh keys
all done
Pentacorn:~ kelli$
```

4. You should have the **GameDev** project folder to your Desktop.

Workspace Setup

Open the GameDev 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**.



/GameDev

/GameDev | 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.

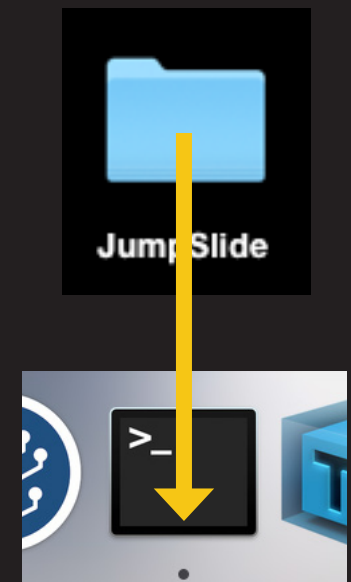
Start a Server

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, 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 command:

http://localhost:8000

3. The port is either 8000 or 8080. Just make sure that your port matches the port listed in the Terminal window.
4. If you did it correctly, you should see a black background



```
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 ...
```

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

Change the Game Name and add yourself as the Author.

1. Click on **index.html**
2. Find **AUTHOR:Your Name** and change “Your Name” to your actual name.
3. Find **<title>JumpSlide</title>** and change the title to your own game name.
4. Save.
5. Switch to your **Google Chrome** browser that contains your game.
6. Refresh the page by hitting **Command-R**
7. You won't see anything on the page, however the title of the game in the Google Chrome tab should be updated to the game name you chose.

Start Coding!

Follow the slides in this tutorial. After every slide, be sure to test if the code works! All edits are done to the, **game.js** file unless you are super-advanced and you want to experiment with JumpSlide.js.

To begin coding, click **game.js** in the Sublime Text sidebar.

Start the Game!

In the Game.init block, Start the Game. We will be added more stuff to this block. The JumpSlide function call should always be at the END of the block.

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

Test in Chrome!

Plan Out the Game Level

The next step requires some planning the old fashioned way (on paper). Since the game uses an inverted-y cartesian coordinate system on a **1024 x 768** area, the placement of objects on the game sceen might seem “opposite” (a smaller Y-coordinate means an object will appear higher on the screen and a larger Y-coordinate means an object will appear lower on the screen. It is a good idea to plan out the game level you want to create using graph paper and Excel or Google Sheets.

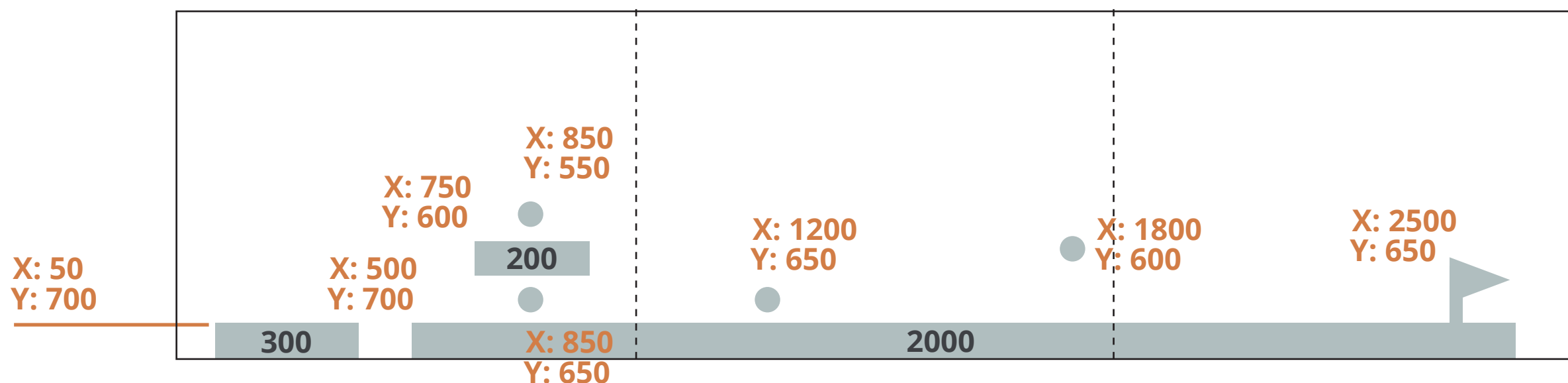
Plan Out the Game Level

Here's a Demo Game Level Plan.



Plan Out the Game Level

Here's a Demo Game Level Plan. Note that position is based on the upper left corner of the object. Yes, this requires **math**. Though don't worry about super exact numbers since you might change things when you test the game later.



Standard Floor Height: 68px
Y-Coordinate Floor Position: 700px

Floating Platform Height: 45px
Floating Platform Width: 200px

Add Some Platforms

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

```
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!

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!

Making the Player Run - Part 1 of 3

In the Game.loop block, add the following **code** to start making the player run. To make it look like the player is running, we 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!

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!

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!

Making the Player Stop Sliding

Find the `GAME.touch_end` block. Add `code` to make the player stop sliding when you lift your finger off the mouse or iPad:

```
GAME.touch_end = function ( point ) {  
  
    JumpSlide.player.stop_sliding();  
  
}
```

Test in Chrome!

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();
}
```

```
});
```

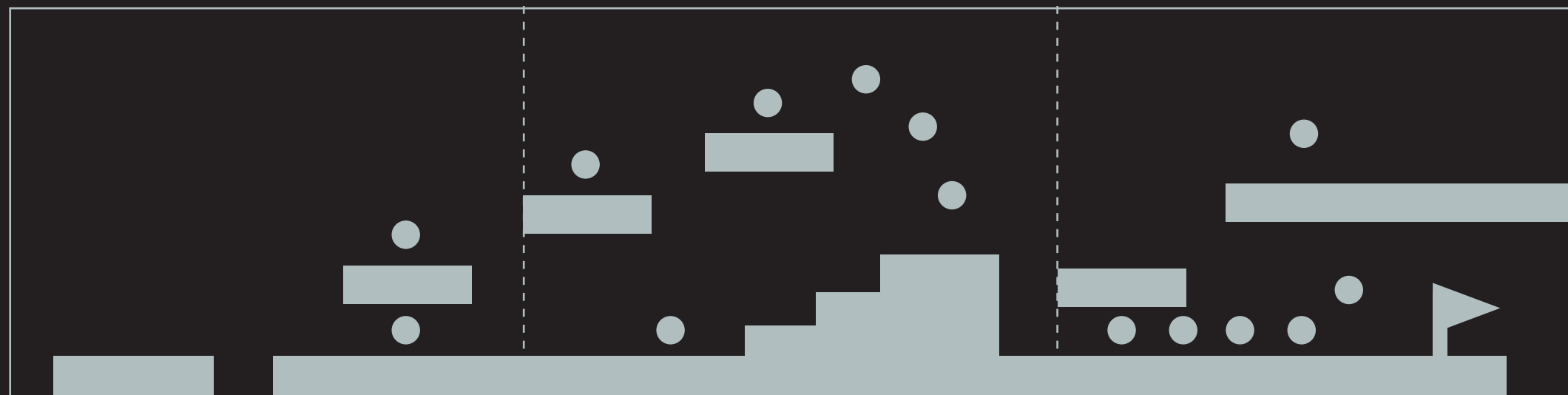
Test in Chrome!

Build a Longer Game Level

Go back to the `GAME.init` block and add more walls, floors, and coins to make a full game level. Use Graph Paper to plan out the level. It is much easier to be creative and plan it out on paper than while you are programming.

Refer to the README for tips on how to use the game api.

Be sure to adjust the speed settings **early in your planning stages**. If you adjust them later you will find that speed greatly affects how far apart objects are to the character that is jumping and sliding.



Experimentation

If you want to be experimental, you can swap out the graphics & sound. The original game graphics and mockups can be downloaded at **stem.gomagames.com**. All game art and sfx are from **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...

When you are ready, move on to **Game Deployment**.

Deployment

When you are ready to Publish your game:

1. Double click the **deploy.commad** script on your desktop.
2. If it asks, type **yes** and hit **Enter**.
3. Navigate to <http://stem2015.gomagames.com/> in your browser or on an iPad.
Your game is now published online.
4. To play the game on iPad, open the game in Safari, click the **Bookmark Icon** and **Save to your Homescreen**.

BUG! Unfortunately, the sound doesn't work on iPad when you Save to Homescreen :(

Note that authentic iOS app deployment is a much longer process. This quick deployment method simulates publishing a mobile app without the lengthy deployment process.