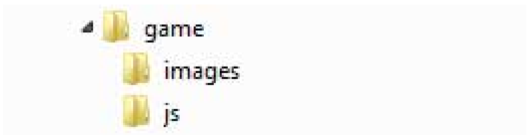
The Code Hub Gothenburg

GAME TUTORIAL: COIN HERO

Create folder structure

1. Create a folder called **game** on your hard drive
2. In the game folder create two folders: **js** and **images**



3. Put the images in the image folder. We are going to use three images: background, a hero and a coin.

Setup game project in sublime text

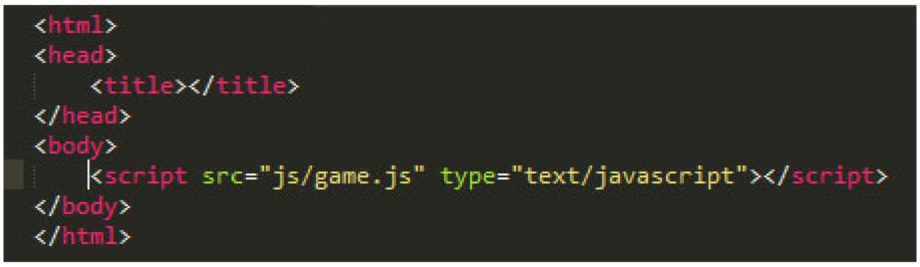
1. Start "Sublime Text"
2. Choose file and **Open folder**
3. Choose the **game** folder that you created.
4. Right click on game folder on the left side and choose **New file**
5. **Save** the file the name of the file is: **index.html**
6. Right click on the js folder and create a **New file**
7. Save it and name the file **game.js**

Setup index.html file

1. Open up index.html if isn't open, double click on the index.html
2. Click on the **Tools** menu and choose **snippets…**
3. Choose **snippet: html** and should get the html-page to start with.
4. Between the body tags, add tag

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

5. Now we have a foundation to work with.



```
<html>
<head>
    <title></title>
</head>
<body>
    <script src="js/game.js" type="text/javascript"></script>
</body>
</html>
```

Start coding the game and make the canvas for the game

1. Open up the game.js
2. Begin by creating two variables canvas and ctx like this:

```
var canvas = document.createElement("canvas");
var ctx = canvas.getContext("2d");
```

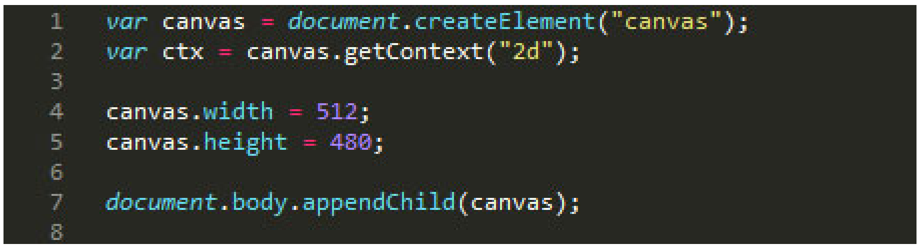
3. First the canvas needs width and height, we'll go for 512*480 in this tutorial.

```
canvas.width = 512;
canvas.height = 480;
```

4. Now we have done a canvas tag in javascript, let's to add it in the index.html

```
document.body.appendChild(canvas);
```

Document is the index.html, body is the tag that we should append the canvas.



```
var canvas = document.createElement("canvas");
var ctx = canvas.getContext("2d");

canvas.width = 512;
canvas.height = 480;

document.body.appendChild(canvas);

```

Adding the images

1. Starting with the background image, we need to create two variables. First we need a Boolean to check when the image has been loaded

```
var bgReady = false;
```

Secondly we'll need a variable for the image

```
var bgImage = new Image();
```

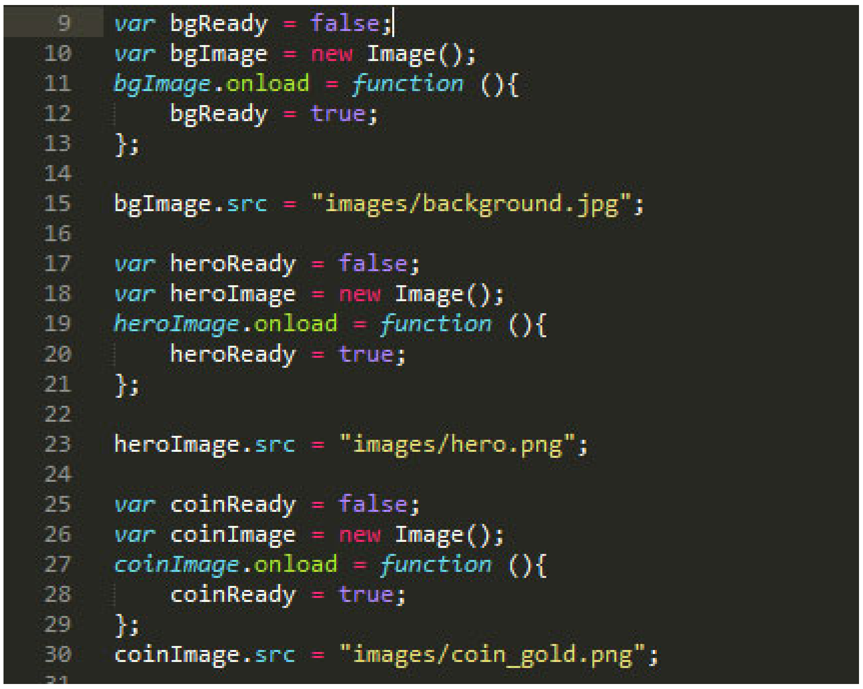
2. Time to load the image! We'll use the image variable

```
bgImage.onload = function (){
        bgReady = true;
};
```

3. At last we need to put in the source of the image like this

```
bgImage.src = "images/background.jpg";
```

4. Repeat steps 1-3 for hero.png and coin_gold.png



```
var bgReady = false;
var bgImage = new Image();
bgImage.onload = function (){
    bgReady = true;
};

bgImage.src = "images/background.jpg";

var heroReady = false;
var heroImage = new Image();
heroImage.onload = function (){
    heroReady = true;
};

heroImage.src = "images/hero.png";

var coinReady = false;
var coinImage = new Image();
coinImage.onload = function (){
    coinReady = true;
};
coinImage.src = "images/coin_gold.png";
```

Next step is to do a render function that add the picture on the canvas

Render image on the canvas

1. Begin to code a function named render like this:

```
var render = function (){
};
```

2. Between the curly brackets, add the code to check if the image is loaded using the bgReady variable. If it's true, then drawImage on the canvas.

```
if(bgReady){
        ctx.drawImage(bgImage,0,0);
    }
```

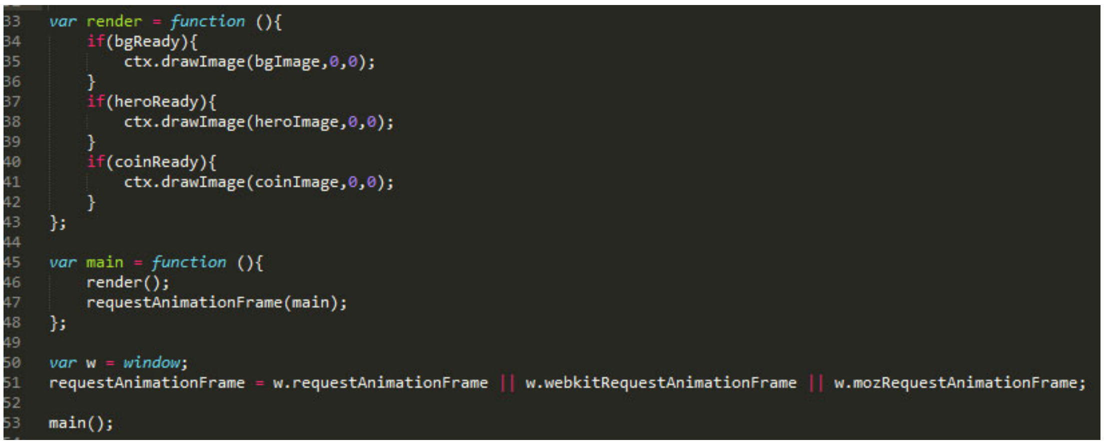
3. As you see we use the ctx variable to draw the image on the canvas. Repeat step 2 for heroImage and coinImage.

4. To see the images on the canvas we need to do a function named main that draw the image that calls every frame to rewrite the canvas. And I put the render function in

```
var main = function (){
    render();
};
```

5. To call the main we can use a function named requestAnimationFrame and put in the main. If this going to work in different browser you have to change global variable for the requestAnimationFrame as well.

```
var main = function (){
        render();
        requestAnimationFrame(main);
};

var w = window;
requestAnimationFrame = w.requestAnimationFrame ||
w.webkitRequestAnimationFrame || w.mozRequestAnimationFrame;
main();
```



```
var render = function (){
    if(bgReady){
        ctx.drawImage(bgImage,0,0);
    }
    if(heroReady){
        ctx.drawImage(heroImage,0,0);
    }
    if(coinReady){
        ctx.drawImage(coinImage,0,0);
    }
};

var main = function (){
    render();
    requestAnimationFrame(main);
};

var w = window;
requestAnimationFrame = w.requestAnimationFrame || w.webkitRequestAnimationFrame || w.mozRequestAnimationFrame;

main();
```

Now we can test if everything works right! Click on the index.html, select open containing folder and double click index.html. Now you should see the three images on the stage.

Fix hero and coin to show only one frame of the sprite sheet.

1. Go to the render function change ctx.drawImage(heroImage,0,0); to ctx.drawImage(heroImage, startPoint X, startPointY, frame width, frame Height, x, y, width, height); look like this when put in some number.

   ```
   ctx.drawImage(heroImage,0, 0 ,32, 32, 512/2, 480-50, 32,32);
   ```

2. And for the coin:

   ```
   ctx.drawImage(coinImage,0, 0 ,32, 32, 0, 0, 32,32);
   ```

If you want, you can experiments with the numbers and see what happen with the images:
http://www.w3schools.com/tags/canvas_drawimage.asp

Adding some dynamic numbers for the characters

Time to add dynamic variable by doing a variable containing an object that has its own variables! We want to keep the render and main function at the bottom of the script. So implement this variable after the coinImage.src = "images/coin_gold.png";

1. The hero object needs three variables for now, we'll add more later on. She needs a speed set to 256, x set to 0 and y set to 0.

```
var hero = {
       speed : 256,
       x : 0,
       y : 0
};
```

2. Now make a coin variable contain x set to 0 and y set to 0.

```
var coin = {
       x : 0,
       y : 0
};
```

3. Next step is to add start values for the characters by doing a reset function. Use the hero object and the coin object to hold the values.

```
var reset = function(){
    hero.x = canvas.width / 2;
    hero.y = canvas.height / 2;

    coin.x = 32 + (Math.random() * (canvas.width - 64));
    coin.y = 32 + (Math.random() * (canvas.height - 64));
};
```

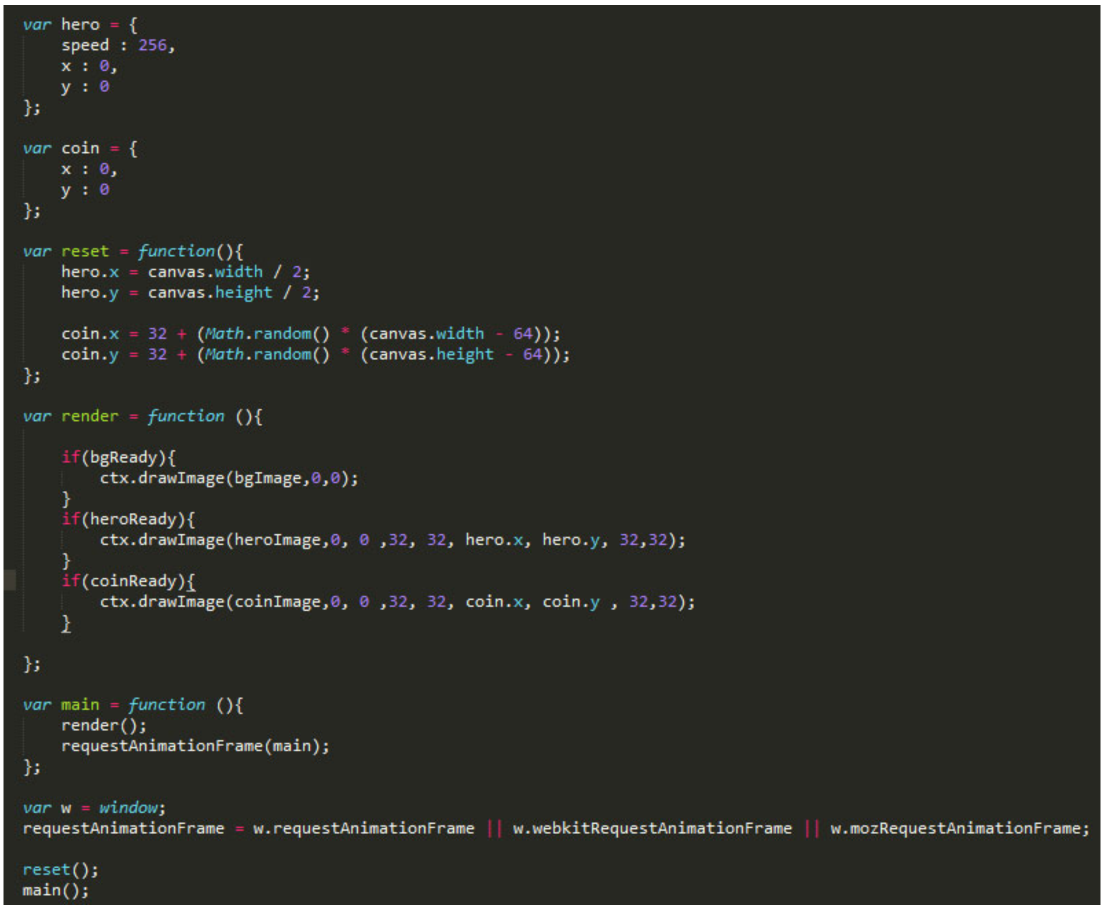
   This place the hero in the middle on the background, use the canvas height and width parameter and divide with 2. For the coin we use a random number and use canvas height and width, so the random number will not be too big.

4. Call the reset function by putting reset(); before main();

5. Now we'll put the hero.x and hero.y on the draw image. And the same goes for the coin.

```
if(heroReady){
       ctx.drawImage(heroImage,0, 0 ,32, 32, hero.x, hero.y, 32,32);
    }
    if(coinReady){
       ctx.drawImage(coinImage,0, 0 ,32, 32, coin.x, coin.y , 32,32);
}
```

Save and test the game. Test to reload the game and see the coin switch place



```
var hero = {
    speed : 256,
    x : 0,
    y : 0
};

var coin = {
    x : 0,
    y : 0
};

var reset = function(){
    hero.x = canvas.width / 2;
    hero.y = canvas.height / 2;

    coin.x = 32 + (Math.random() * (canvas.width - 64));
    coin.y = 32 + (Math.random() * (canvas.height - 64));
};

var render = function (){

    if(bgReady){
        ctx.drawImage(bgImage,0,0);
    }
    if(heroReady){
        ctx.drawImage(heroImage,0, 0 ,32, 32, hero.x, hero.y, 32,32);
    }
    if(coinReady){
        ctx.drawImage(coinImage,0, 0 ,32, 32, coin.x, coin.y , 32,32);
    }

};

var main = function (){
    render();
    requestAnimationFrame(main);
};

var w = window;
requestAnimationFrame = w.requestAnimationFrame || w.webkitRequestAnimationFrame || w.mozRequestAnimationFrame;

reset();
main();
```

Moving the hero

1. Add listener to check if keys are pressed (keydown) or released (keyup). Make two variables; one containing an empty object (keysDown) and one containing a Boolean (isKeyPressed).

```
var keysDown = {};
var isKeyPressed = false;
```

2. Make the listener for the key down.

```
this.addEventListener("keydown",function(e){
    keysDown[e.keyCode] = true;
    isKeyPressed = true;
},false);
```

3. Make the listener for the key up.

```
this.addEventListener("keyup",function(e){
    delete keysDown[e.keyCode];
    isKeyPressed = false;
},false);
```

4. Make an update function to check for keypress and begin to move the hero. And we need to send in a parameter in function called modifier

    ```
    var update = function (modifier){
    };
    ```

5. In the update function check the key code for up, down, left and right arrow button.

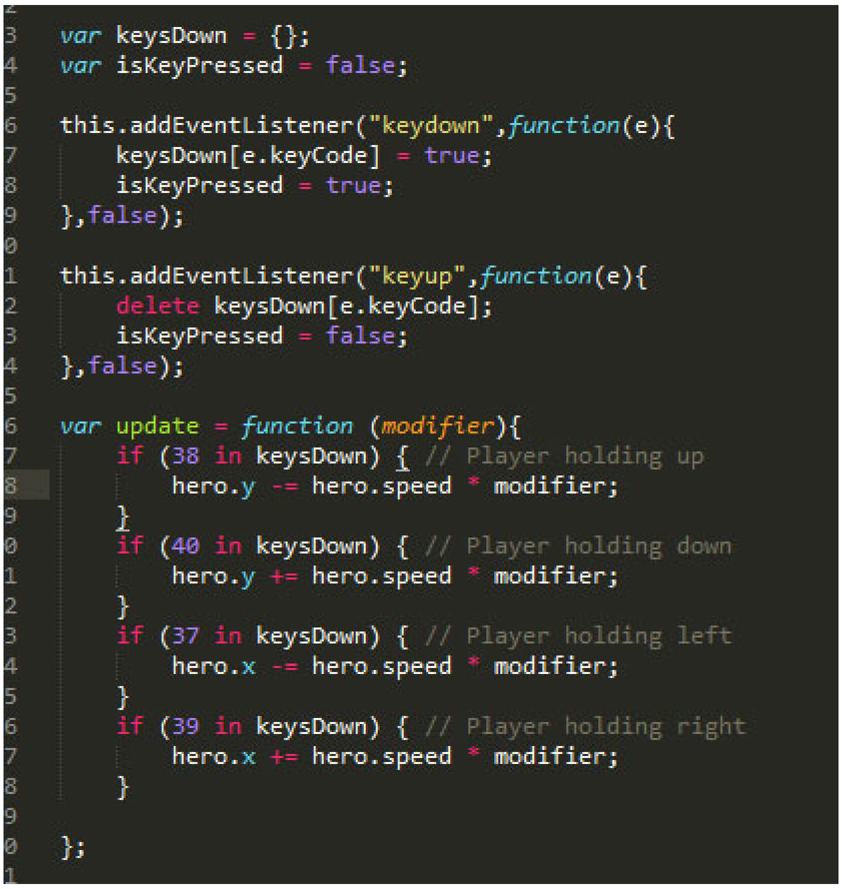
    ```
    if (38 in keysDown) {
            hero.y -= hero.speed * modifier;
        }
        if (40 in keysDown) {
            hero.y += hero.speed * modifier;
        }
        if (37 in keysDown) {
            hero.x -= hero.speed * modifier;
        }
        if (39 in keysDown) {
            hero.x += hero.speed * modifier;
        }
    ```

    As you see I change the hero.x and hero.y depending on what key's pressed

6. Next step is to call the update(); in the main function so that it happens every frame, and we need to make a calculation for the modifier as well. Add the changes in the code marked with red to your existing code:

    ```
    var main = function (){
        var now = Date.now();
        var delta = now - then;
        update(delta/ 1000);

        render();
        then = now;

           requestAnimationFrame(main);
    };

    var w = window;
    requestAnimationFrame = w.requestAnimationFrame ||
    w.webkitRequestAnimationFrame || w.mozRequestAnimationFrame;

    var then = Date.now();
    reset();
    main();
    ```

Save and test the game. Press the keys on the keypad and the hero moves.



```
var keysDown = {};
var isKeyPressed = false;

this.addEventListener("keydown",function(e){
    keysDown[e.keyCode] = true;
    isKeyPressed = true;
},false);

this.addEventListener("keyup",function(e){
    delete keysDown[e.keyCode];
    isKeyPressed = false;
},false);

var update = function (modifier){
    if (38 in keysDown) { // Player holding up
        hero.y -= hero.speed * modifier;
    }
    if (40 in keysDown) { // Player holding down
        hero.y += hero.speed * modifier;
    }
    if (37 in keysDown) { // Player holding left
        hero.x -= hero.speed * modifier;
    }
    if (39 in keysDown) { // Player holding right
        hero.x += hero.speed * modifier;
    }

};
```

Catching the coin

1. Make a variable that keep track how many coins gets caught:

   ```
   var coinsCaught = 0;
   ```

2. In the update function we add a condition to state that if the hero is in the coin area, then coin should get caught. Add one for the coin caught and reset() the game.

   ```
       if (
           hero.x <= (coin.x + 32)
           && coin.x <= (hero.x + 32)
           && hero.y <= (coin.y + 32)
           && coin.y <= (hero.y + 32)
       ) {
           ++coinsCaught;
           reset();
       }
   ```

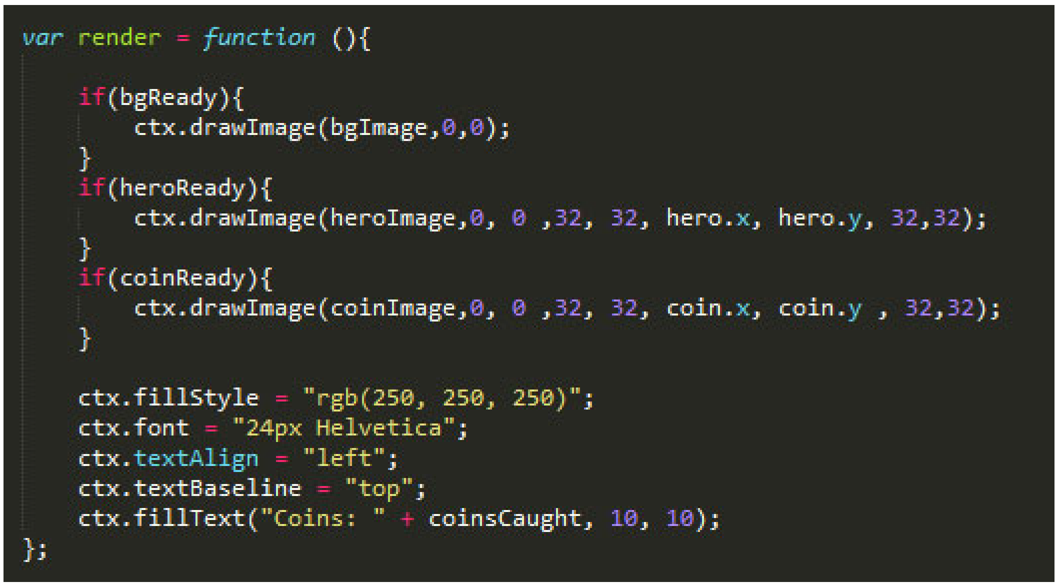
Save and test the game. Catch the coin.

Add text to canvas to display amount of coins caught

1. That code we add at the render function and use the ctx variable.

```
ctx.fillStyle = "rgb(250, 250, 250)";
    ctx.font = "24px Helvetica";
    ctx.textAlign = "left";
    ctx.textBaseline = "top";
    ctx.fillText("Coins collected: " + coinsCaught, 10, 10);
```

As you see, we use the coinsCaught variable to fill the text.



```
var render = function (){

    if(bgReady){
        ctx.drawImage(bgImage,0,0);
    }
    if(heroReady){
        ctx.drawImage(heroImage,0, 0 ,32, 32, hero.x, hero.y, 32,32);
    }
    if(coinReady){
        ctx.drawImage(coinImage,0, 0 ,32, 32, coin.x, coin.y , 32,32);
    }

    ctx.fillStyle = "rgb(250, 250, 250)";
    ctx.font = "24px Helvetica";
    ctx.textAlign = "left";
    ctx.textBaseline = "top";
    ctx.fillText("Coins: " + coinsCaught, 10, 10);
};
```

Use the sprite sheet on the hero when moving

1. Now we'll add some more variables on the hero object which we will call spriteX and spriteY and set to 0.

```
var hero = {
      speed : 256,
      spriteX : 0,
      spriteY : 0,
       x : 0,
       y : 0
};
```

2. Create a new variable called spriteSize and set it to 32. Because one frame in the sprite sheet is width 32 and height 32.

```
var spriteSize = 32;
```

3. Go to the update function and add the code marked in red to the existing code for the spriteY depending on the key press

```
if (38 in keysDown) {
        hero.y -= hero.speed * modifier;
        hero.spriteY = 3;
    }
    if (40 in keysDown) {
        hero.y += hero.speed * modifier;
        hero.spriteY = 0;
    }
    if (37 in keysDown) {
        hero.x -= hero.speed * modifier;
        hero.spriteY = 1;
    }
    if (39 in keysDown) {
        hero.x += hero.speed * modifier;
        hero.spriteY = 2;
    }
```

4. Now we need to add the variables to the existing ctx.drawImage in render function like this (marked in red):

```
ctx.drawImage(heroImage,0, spriteSize * hero.spriteY ,32, 32, hero.x,
hero.y, 32,32);
```

Save and test the game now, see how it uses the other frames of the sprite sheet when pressing the buttons.

5. I want to animate the hero as well to change the spriteX coordinate. I need a spriteXcount variable that can change every frame so I add that to the code.

```
var spriteXcount = 0;
```

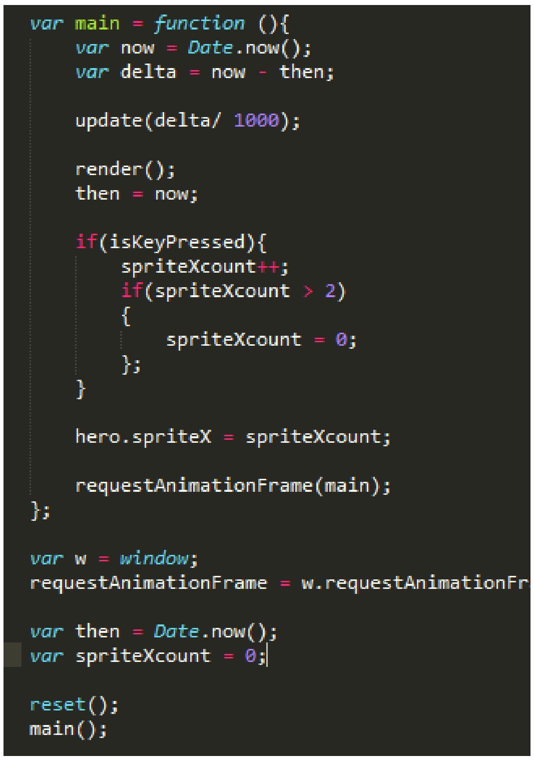
6. Count up should be in the main function and it should only count if the key is pressed.

```
if(isKeyPressed){
        spriteXcount++;
        if(spriteXcount > 2)
        {
            spriteXcount = 0;
        };
    }

    hero.spriteX = spriteXcount;
```

7. ctx.drawImage needs this information too, so put hero.spriteX there as well.

```
ctx.drawImage(heroImage,spriteSize * hero.spriteX, spriteSize *
hero.spriteY ,32, 32, hero.x, hero.y, 32,32);
```

Save and test the game.



```
var main = function (){
    var now = Date.now();
    var delta = now - then;

    update(delta/ 1000);

    render();
    then = now;

    if(isKeyPressed){
        spriteXcount++;
        if(spriteXcount > 2)
        {
            spriteXcount = 0;
        };
    }

    hero.spriteX = spriteXcount;

    requestAnimationFrame(main);
};

var w = window;
requestAnimationFrame = w.requestAnimationFr

var then = Date.now();
var spriteXcount = 0;

reset();
main();
```

Coin sprite sheet animation

1. Add variable spriteX in the coin object.

```
var coin = {
    x : 0,
    y : 0,
    spriteX : 0
};
```

2. Make the coinXCount variable to count it up in the main function and a frame variable that also count up

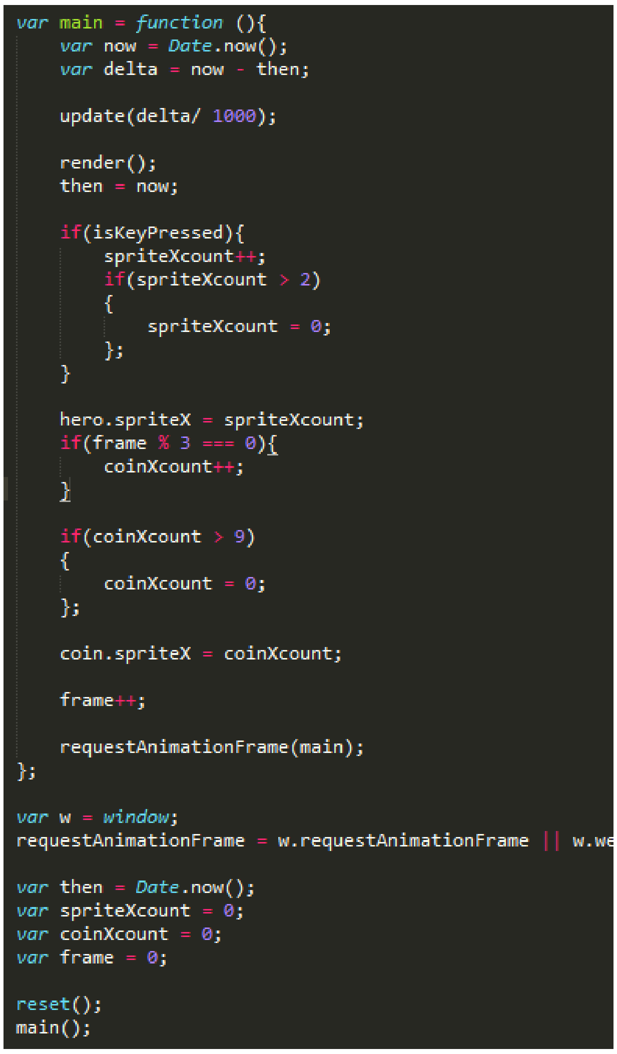
```
var coinXcount = 0;
var frame = 0;
```

3. In the main function put this code in:

```
hero.spriteX = spriteXcount;
    if(frame % 3 === 0){
          coinXcount++;
    }
    if(coinXcount > 9)
    {
        coinXcount = 0;
   };

    coin.spriteX = coinXcount;
    frame++;
```

4. Last, but not least, this is to fix the ctx.drawImage in render function

```
ctx.drawImage(coinImage,spriteSize * coin.spriteX, 0 ,32, 32,
coin.x, coin.y , 32,32);
```



```
var main = function (){
    var now = Date.now();
    var delta = now - then;

    update(delta/ 1000);

    render();
    then = now;

    if(isKeyPressed){
        spriteXcount++;
        if(spriteXcount > 2)
        {
            spriteXcount = 0;
        };
    }

    hero.spriteX = spriteXcount;
    if(frame % 3 === 0){
        coinXcount++;
    }

    if(coinXcount > 9)
    {
        coinXcount = 0;
    };

    coin.spriteX = coinXcount;

    frame++;

    requestAnimationFrame(main);
};

var w = window;
requestAnimationFrame = w.requestAnimationFrame || w.we

var then = Date.now();
var spriteXcount = 0;
var coinXcount = 0;
var frame = 0;

reset();
main();
```

Save and test the game.

That’s the end of this tutorial!

If you want, you can evolve your game by putting in a timer, fixing so that the hero can’t go outside the background or whatever you can think of.

If you want to read more about sprite sheets, see:
http://www.williammalone.com/articles/create-html5-canvas-javascript-sprite-animation/