Snake Game Steps

Script

Let's create a snake game using HTML and JavaScript. We will use an HTML element called canvas to create a spot for our game. We are going to give the canvas and id of game with a width and height of 400 pixels.

Now we will create a script element where we will put all our JavaScript.

```
<canvas id="game" width="400" height="400"></canvas>
<script>
```

Now we are going to create a javascript function that will run every time our window opens. This function starts by creating a variable that will refer to the canvas element in our HTML file. We use the getElementById, to get the canvas, that we gave the ID to game.

CTX is a context that gives us some rules for drawing on our canvas. The getContext(2d) functions. Now we can use CTX

Next we are going to add

```
window.onload=function() {
    canvas=document.getElementById("game");
    ctx=canvas.getContext("2d");
    document.addEventListener("keydown", keyPush);
    setInterval(game_function,1000/15);
}
```

```
pos_x = 10; // x position -- starts at 10
pos_y = 10; // y position -- starts at 10

grid_size = 20; // grid size
tile_count = 20; // tile count

x_vel = 0; // x velocity
y_vel = 0; // y velocity

apple_x = 15; // apple x position
apple_y = 15; // apple y position

trail = []; // array to hold tail positions
tail_length = 5; // set the initial tail length
```

```
function game_function() {

    // move position based on velocity
    pos_x = pos_x + x_vel;
    pos_y = pos_y + y_vel;

    // if the x position is less than 0, send the snake to the other side
    if(pos_x < 0) {
        pos_x = tile_count - 1;
    }

    // if the x position is greater than the tile count, send the snake to 0
    if(pos_x > tile_count - 1) {
        pos_x = 0;
    }

    // if the y position is less than 0, send the snake to the other side
    if(pos_y < 0) {
        pos_y = tile_count - 1;
    }

    // if the y position is greater than the tile count, send the snake to the other
```

```
side
    if(pos_y > tile_count - 1) {
        pos_y = 0;
    }

    // color in the canvas
    ctx.fillStyle = "black";
    ctx.fillRect(0, 0, canvas.width, canvas.height);
```

Now we will create the snake motion by moving the trail. We are going to use the trail array, which stores the x and y values for each segment of the snake. The basic movement strategy is to add the new position to one end of the trail array, then remove the old last position from the other end of the trail array. The first array method we are going to use is the push method. This will take the current position of the head of the snake and push it onto the end of the trail array. Pushing the new position onto the end of the snake means we need to remove the last position, the one we are leaving, from the other end of the array. We do that with the shift method. We put the shift inside a while loop so that the snake trail array can be changed by changing the tail_length variable.

```
    // create the trail array. This is an array of x and y values. We already used
them above when we called trail[i].x and trail[i].y

    trail.push({x:pos_x, y:pos_y});
    while(trail.length > tail_length){
        trail.shift(); // this is how our snake moves
    }
```

In order to view the page as it is right now we have to put in the keypush function -- it won't do anything yet.

```
function keyPush(event) {

}
```

Let's view what happens when we load the page as it is now.

We can go ahead and write the keypush function now. We will need to look up key codes for the arrow keys. Her we can see that the key codes for the arrow keys are 37, 38, 39, and 40.

We are going to use the switch statement to tell our program what to do when it gets keyboard input. Each case will correspond to one of either up, down, left or right arrows.

The case for each one will change the velocity of the snake. The velocity of the snake tells us which way the snake will move. The x positions get smaller moving to the left of the screen and the y positions get smaller moving towards the top of the screen.

- When the snake goes right, its x velocity is positive one and its y velocity is zero
- When the snake goes left, its x velocity is negative one and its y velocity is zero
- When the snake goes up, its x velocity is zero and its y velocity is negative one

- When the snake goes down, its x velocity is zero and its y velocity is positive one

```
function keyPush(event) {
    // get arrow keys using their codes: 37, 38, 39, 40
    switch(event.keyCode) {

        case 37:
            x_vel = -1;
            y_vel = 0;
            break;

        case 38:
            x_vel = 0;
            y_vel = -1;
            break;

        case 39:
            x_vel = 1;
            y_vel = 0;
            break;

        case 40:
            x_vel = 0;
            y_vel = 1;
            break;
    }
}

</script>
```

```
    // color the snake
    ctx.fillStyle = "lime";
    for(var i = 0; i < trail.length; i++) {
        ctx.fillRect(trail[i].x * grid_size,
            trail[i].y * grid_size, grid_size - 2,
            grid_size - 2)

        // check to see if snake hit its own tail
        if(trail[i].x == pos_x && trail [i].y == pos_y) {
            // restart at original tail length of 5
            tail_length = 5;
        }
    }
```

```
    // snake eats an apple
    if (apple_x == pos_x && apple_y == pos_y) {
        // increase the length of the snake by 1
        tail_length++;

        //put a new apple in a random spot
        apple_x = Math.floor(Math.random()*tile_count);
        apple_y = Math.floor(Math.random()*tile_count)
    }

    // color in the apple
    ctx.fillStyle="red";
    ctx.fillRect(apple_x * grid_size, apple_y * grid_size, grid_size - 2, grid_size
- 2)
}
```

```
function keyPush(event) {
    // get arrow keys using their codes: 37, 38, 39, 40
    switch(event.keyCode) {

        case 37:
            x_vel = -1;
            y_vel = 0;
            break;

        case 38:
            x_vel = 0;
            y_vel = -1;
            break;

        case 39:
            x_vel = 1;
            y_vel = 0;
            break;

        case 40:
            x_vel = 0;
            y_vel = 1;
            break;
    }
}

</script>
```