Homework Turnin

Name:	Stephen Hung
Email:	hungs3@uw.edu
Student ID:	1536327
Section:	AD
Course:	CSE 154 16au
Assignment:	hw4
Receipt ID:	e7dc57ffd04105cb25c58f7dca39e153

Replacing prior submission from Sat 2016/10/29 10:46pm.

```
no compiler configured for web
```

Turnin Successful!

The following file(s) were received:

fifteen.js (7095 bytes)

```
/*
    Stephen Hung
    CSE 154 AD
    The javascript code for fifteen.html.
    This javascript code creates tiles for the game, Fiften Puzzles
    and creates the logic for the user to play the game. Tiles that
    are movable are highlighted in red and have a pointer cursor over
    them. Once the user wins, the page will display they they won.
    The extra feature I added was the end-of-game notification which
    notifies the user if they finish the puzzle.
*/
(function(){
    "use strict";

    // variable to hold the coordinates for the empty box.
    var puzzleModule = {
        emptyLeft: 300+"px",
        emptyTop: 300+"px"
    };

    // Once the window loads, handle event handlers and create
    // 15 tiles in a particular order & position.
    window.onload = function(){
        var puzzleArea = document.getElementById('puzzlearea');
        var counter = 0;
        var shuffleButton = document.getElementById('shufflebutton');
        shuffleButton.onclick = shuffleTiles;

        for(var i = 0; i < 15; i++){
            var tile = document.createElement('div');

            tile.className = "tile";
            tile.id = parseInt(i + 1);

            // Sets each tile to a specific part of the background image.
            var backgroundX = (i * -100) + "px" ;
            var backgroundY = (Math.floor(i / 4) * -100) + "px";

            tile.style.backgroundPosition = backgroundX + " " + backgroundY;
            tile.innerText = parseInt(i + 1);
            if(counter === 4){
                counter = 0;
            }
            tile.style.left = 100 * counter + "px";
            tile.style.top = Math.floor(i / 4 ) * 100 + "px";
```

```
            puzzleArea.appendChild(tile);
            tile.onclick = tileClick;
            tile.onmouseenter = tileHover;
            tile.onmouseleave = tileLeave;
            counter++;
        }
    };

    // Shuffles the tiles 1000 times by swapping random tiles with the empty spot.
    function shuffleTiles(){
        removeResultDiv();
        for(var i = 0; i < 1000; i++){
            var neighbors = [];
            var emptyX = parseInt(puzzleModule.emptyLeft);
            var emptyY = parseInt(puzzleModule.emptyTop);
            var tileArray = document.querySelectorAll('.tile');
            for(var j = 0; j < tileArray.length; j++){
                // for every neighbor, if it can move, add it to an array.
                var tile = tileArray[j];
                if (canMove(tile.style.left,tile.style.top)){
                    neighbors.push(tile);
                }
            }
            // gets a random index for the array and swaps tiles with it.
            var randomIndex = parseInt(Math.random()*neighbors.length);
            switchTiles(neighbors[randomIndex]);
        }
    }

    // Removes the 'You Win!' notification.
    function removeResultDiv(){
        if( document.getElementById('result') ){
            document.body.removeChild(document.getElementById('result'));
        }
    }

    // Switches the tile passed in with the empty spot.
    function switchTiles(tile){
        var tempLeft = tile.style.left;
        var tempTop = tile.style.top;

        tile.style.left = puzzleModule.emptyLeft;
        tile.style.top = puzzleModule.emptyTop;

        puzzleModule.emptyLeft = tempLeft;
        puzzleModule.emptyTop = tempTop;
    }

    // Event handler for the tile being clicked.
    function tileClick(){
        if(canMove(this.style.left,this.style.top)){
            switchTiles(this);
        }
        if(checkFinish()){
            removeResultDiv();
            var resultDiv = document.createElement('div');
            resultDiv.id = "result";
            resultDiv.innerText = "You Win!"
            resultDiv.className = "winner";
            document.body.insertBefore(resultDiv,document.getElementById('output'));
        }else{
            removeResultDiv();
        }
    }

    // Checks if the game is over.
    function checkFinish(){
        var solvedArray = [];
        for(var i = 1; i < 14; i+=4){
            solvedArray.push(nextTo(i));
        }
        if(solvedArray.includes(false)){
            return false;
        }
        return true;
    }

    // Checks if the id passed in is next to the empty spot.
    // It does so by checking to make sure each row is located correctly.
    function nextTo(leadingId){
```

```
        var firstTile = document.getElementById(parseInt(leadingId));
        var iterator = 3;
        // checks if the first tile is located at the right spot.
        if(!correctInitialSpot(firstTile,leadingId)){
            return false;
        }
        if(leadingId === 13){
            iterator = 2;
        }

        // Checks to see if tiles are next to each other in the correct order.
        for( var i = 0; i < iterator; i++){
            firstTile = document.getElementById(parseInt(leadingId));
            var secondTile = document.getElementById(parseInt(leadingId + 1));
            var firstX = parseInt(firstTile.style.left);
            var firstY = parseInt(firstTile.style.top);
            var secondX = parseInt(secondTile.style.left);
            var secondY = parseInt(secondTile.style.top);
            if( !(firstY === secondY) || !(firstX === (secondX - 100) ) ){
                return false;
            }
            leadingId++;
        }
        return true;
    }

    // Checks to make sure the first column of tiles are in their
    // correct spots.
    function correctInitialSpot(firstTile, leadingId){
        var loc = 0;
        if(leadingId === 5){
            loc = 100;
        }else if(leadingId === 9){
            loc = 200;
        }else if(leadingId === 13){
            loc = 300;
        }

        // Checks if the tile is in the correct location.
        if(parseInt(firstTile.style.top) !== loc &&
            parseInt(firstTile.style.left === 0) ){
            return false;
        }
        return true;
    }

    // Event handler for a neighboring tile being hovered over.
    // Turns the tile's border & text red.
    function tileHover(){
        if(canMove(this.style.left,this.style.top)){
            this.classList.add('hover');
        }
    }

    // Event handler for a cursor leaving a neighboring tile.
    // Makes the tile return to default style.
    function tileLeave(){
        this.classList.remove('hover');
    }

    // If the passed in coordinates are next to the empty tile.
    function canMove(curLeft, curTop){
        var curX = parseInt(curLeft);
        var curY = parseInt(curTop);
        var emptyX = parseInt(puzzleModule.emptyLeft);
        var emptyY = parseInt(puzzleModule.emptyTop);
        // Checks if the coordinates are to the N, S, E, or W of the empty tile.
        if (curX === emptyX) {
            if( (curY === (emptyY - 100)) || (curY === (emptyY + 100)) ){
                return true;
            }
        }else if(curY === emptyY) {
            if( (curX === (emptyX - 100)) || (curX === (emptyX + 100)) ){
                return true;
            }
        }
        return false;
    }

})();
```

fifteen.css (723 bytes)

```
/*
    Stephen Hung
    CSE 154 AD
    This CSS page is the style sheet for fifteen.html.
    It's main use is getting the page to look nice and
    styling the tiles.
*/
body{

    font-family: cursive;
    font-size: 14pt;
}

body, .tile {
    text-align: center;
}

#puzzlearea{
    height: 400px;
    margin: 0 auto;
    position: relative;
    text-align: left;
    width: 400px;
}

.tile {
    border: 5px solid black;
    background-image: url('background.jpg');
    display: inline;
    font-size: 40pt;
    height: 90px;
    position: absolute;
    width: 90px;
}

.hover{
    border-color: red;
    color: red;
    cursor: pointer;
}

.winner {
    color: blue;
    font-size: 40pt;
    font-family: monospace;
}
```

background.jpg (94100 bytes)

(binary file)