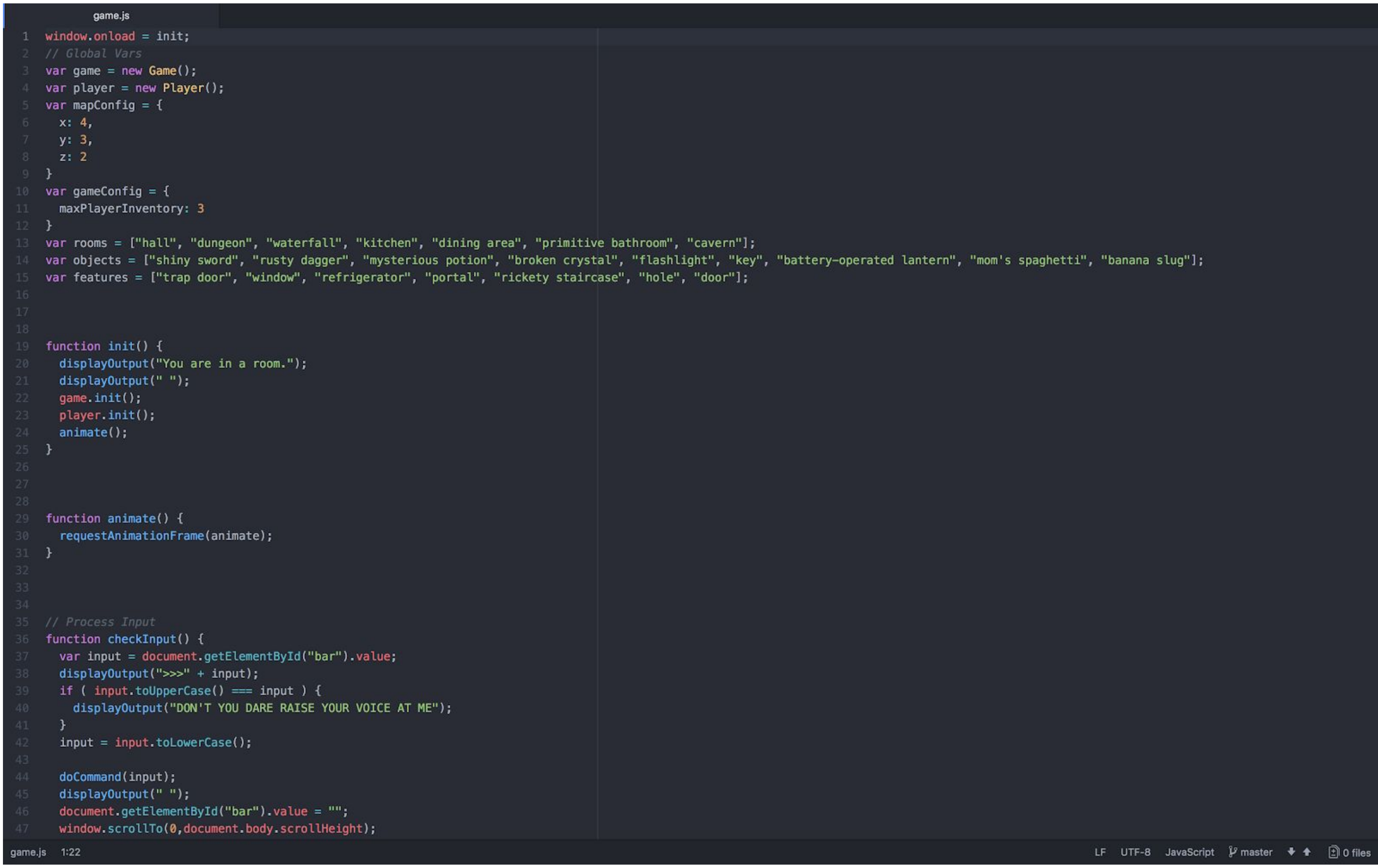
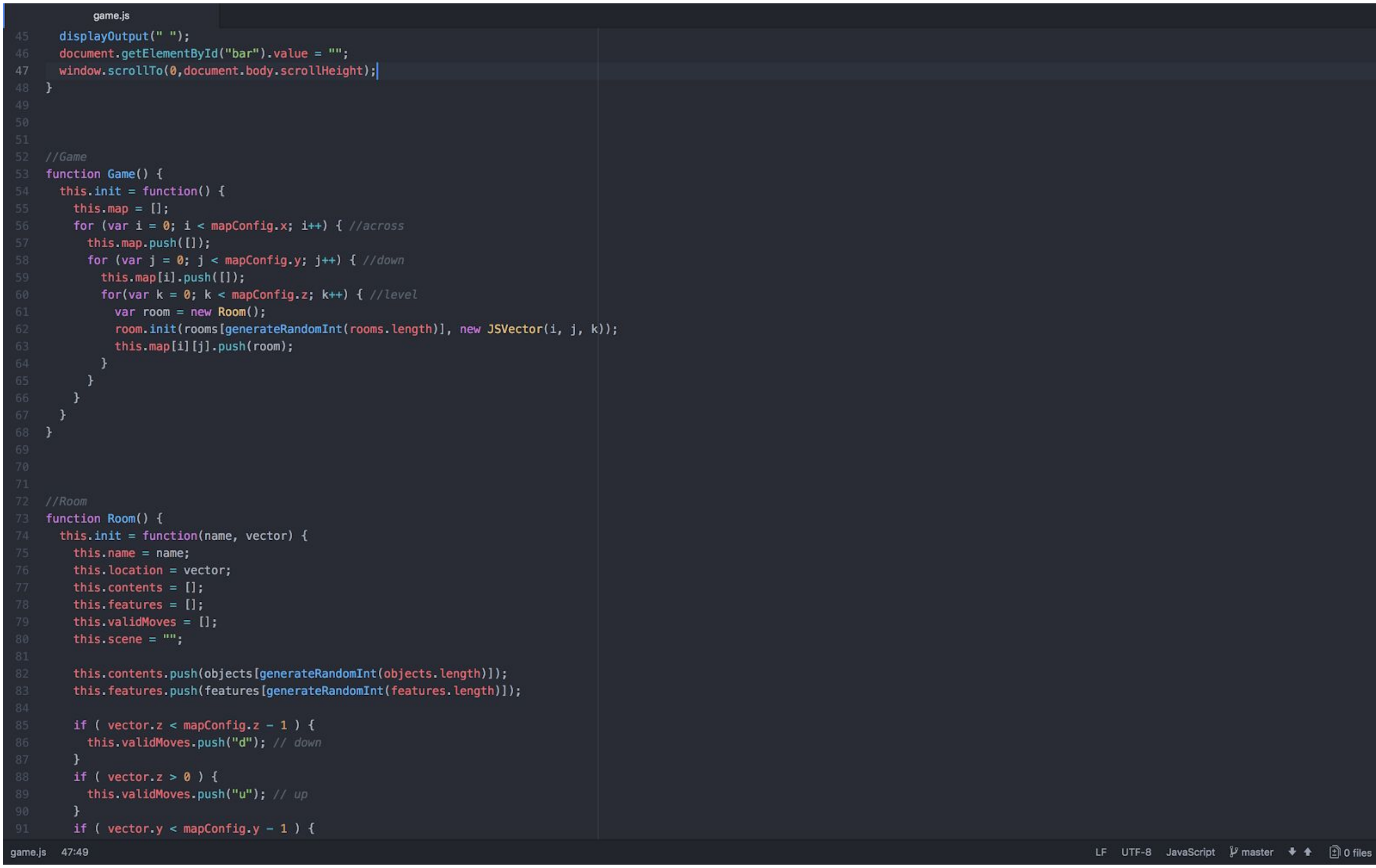
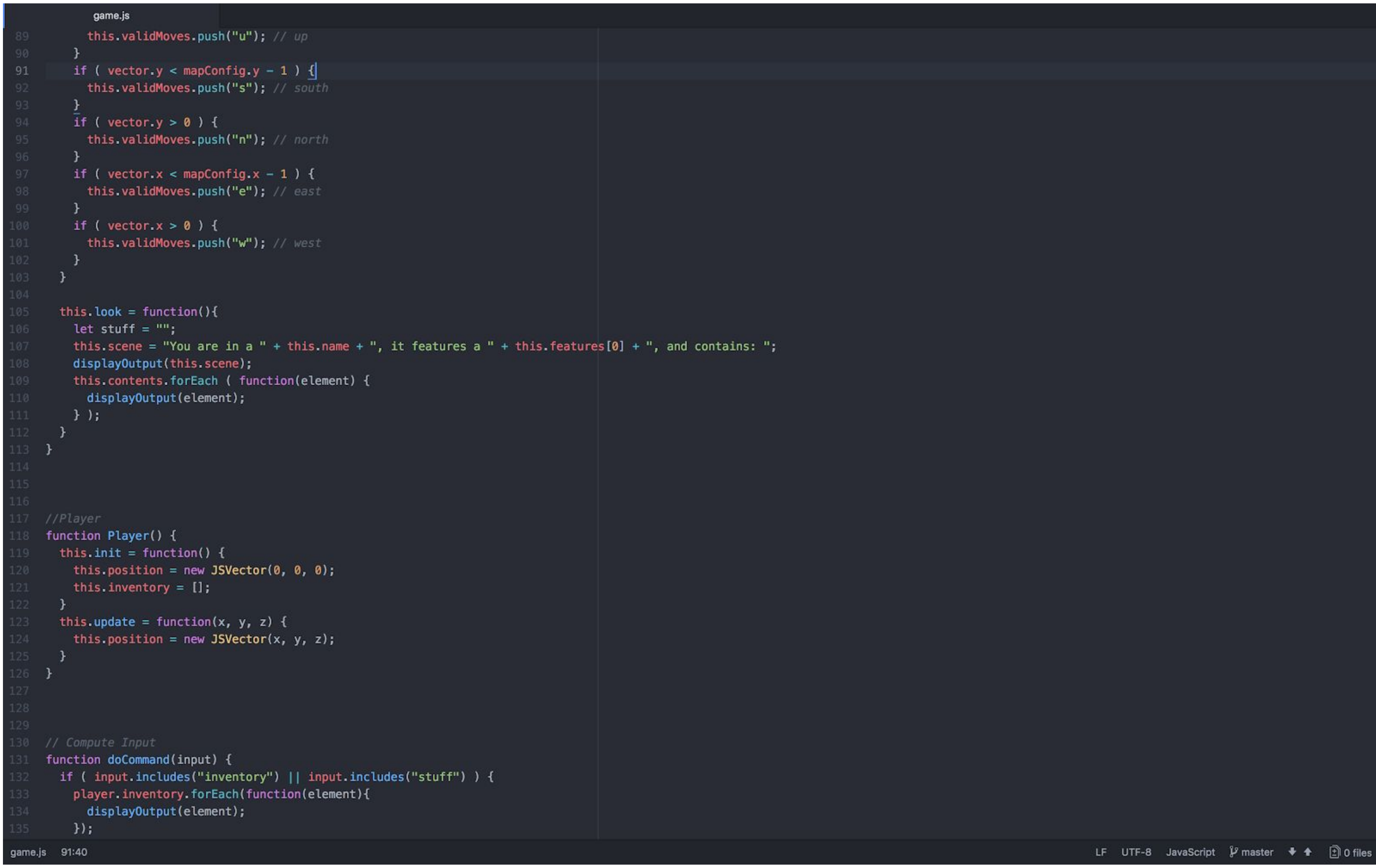


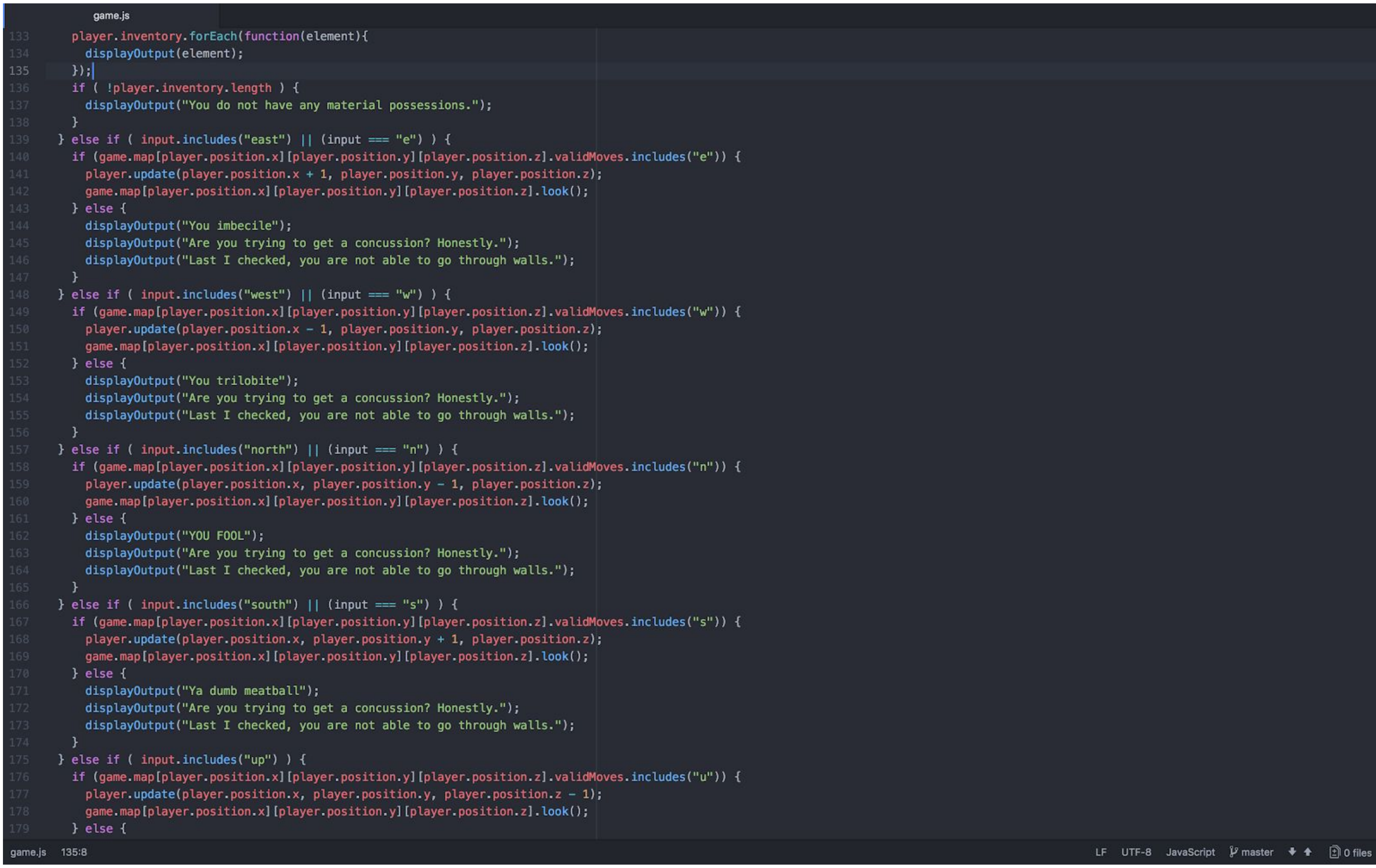
```
window.onload = init;
// Global Vars
var game = new Game();
var player = new Player();
var mapConfig = {
  x: 4,
  y: 3,
  z: 2
}
var gameConfig = {
  maxPlayerInventory: 3
}
var rooms = ["hall", "dungeon", "waterfall", "kitchen", "dining area", "primitive bathroom", "cavern"];
var objects = ["shiny sword", "rusty dagger", "mysterious potion", "broken crystal", "flashlight", "key", "battery-operated lantern", "mom's spaghetti", "banana slug"];
var features = ["trap door", "window", "refrigerator", "portal", "rickety staircase", "hole", "door"];



function init() {
  displayOutput("You are in a room.");
  displayOutput(" ");
  game.init();
  player.init();
  animate();
}



function animate() {
  requestAnimationFrame(animate);
}



// Process Input
function checkInput() {
  var input = document.getElementById("bar").value;
  displayOutput(">>>" + input);
  if ( input.toUpperCase() === input ) {
    displayOutput("DON'T YOU DARE RAISE YOUR VOICE AT ME");
  }
  input = input.toLowerCase();

  doCommand(input);
  displayOutput(" ");
  document.getElementById("bar").value = "";
  window.scrollTo(0,document.body.scrollHeight);
```



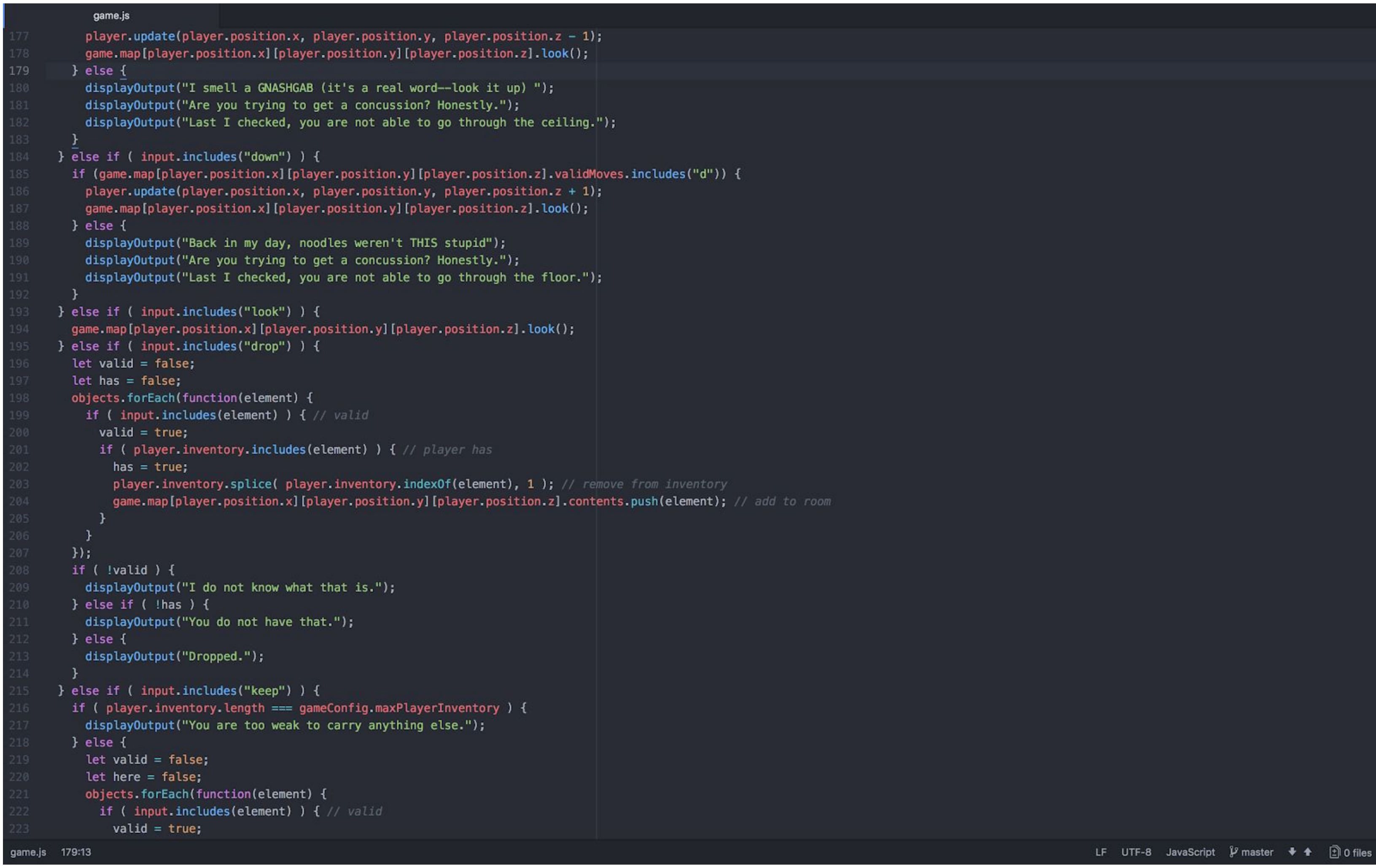
```
  displayOutput(" ");
  document.getElementById("bar").value = "";
  window.scrollTo(0,document.body.scrollHeight);
}



//Game
function Game() {
  this.init = function() {
    this.map = [];
    for (var i = 0; i < mapConfig.x; i++) { //across
      this.map.push([]);
      for (var j = 0; j < mapConfig.y; j++) { //down
        this.map[i].push([]);
        for(var k = 0; k < mapConfig.z; k++) { //level
          var room = new Room();
          room.init(rooms[generateRandomInt(rooms.length)], new JSVector(i, j, k));
          this.map[i][j].push(room);
        }
      }
    }
  }
}



//Room
function Room() {
  this.init = function(name, vector) {
    this.name = name;
    this.location = vector;
    this.contents = [];
    this.features = [];
    this.validMoves = [];
    this.scene = "";

    this.contents.push(objects[generateRandomInt(objects.length)]);
    this.features.push(features[generateRandomInt(features.length)]);

    if ( vector.z < mapConfig.z - 1 ) {
      this.validMoves.push("d"); // down
    }
    if ( vector.z > 0 ) {
      this.validMoves.push("u"); // up
    }
    if ( vector.y < mapConfig.y - 1 ) {
```



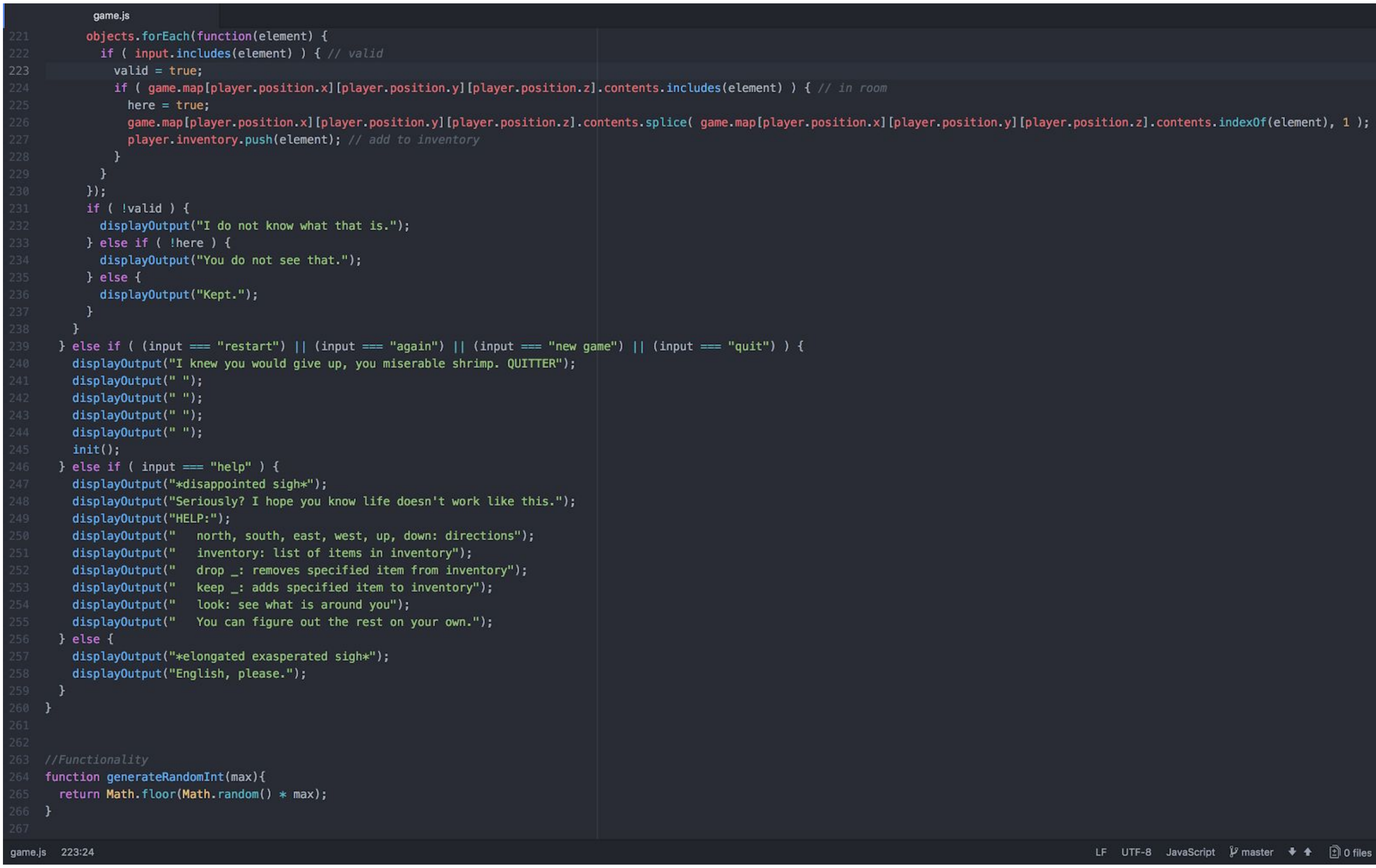
```
      this.validMoves.push("u"); // up
    }
    if ( vector.y < mapConfig.y - 1 ) {
      this.validMoves.push("s"); // south
    }
    if ( vector.y > 0 ) {
      this.validMoves.push("n"); // north
    }
    if ( vector.x < mapConfig.x - 1 ) {
      this.validMoves.push("e"); // east
    }
    if ( vector.x > 0 ) {
      this.validMoves.push("w"); // west
    }
  }

  this.look = function(){
    let stuff = "";
    this.scene = "You are in a " + this.name + ", it features a " + this.features[0] + ", and contains: ";
    displayOutput(this.scene);
    this.contents.forEach ( function(element) {
      displayOutput(element);
    } );
  }
}



//Player
function Player() {
  this.init = function() {
    this.position = new JSVector(0, 0, 0);
    this.inventory = [];
  }
  this.update = function(x, y, z) {
    this.position = new JSVector(x, y, z);
  }
}



// Compute Input
function doCommand(input) {
  if ( input.includes("inventory") || input.includes("stuff") ) {
    player.inventory.forEach(function(element){
      displayOutput(element);
    });
```



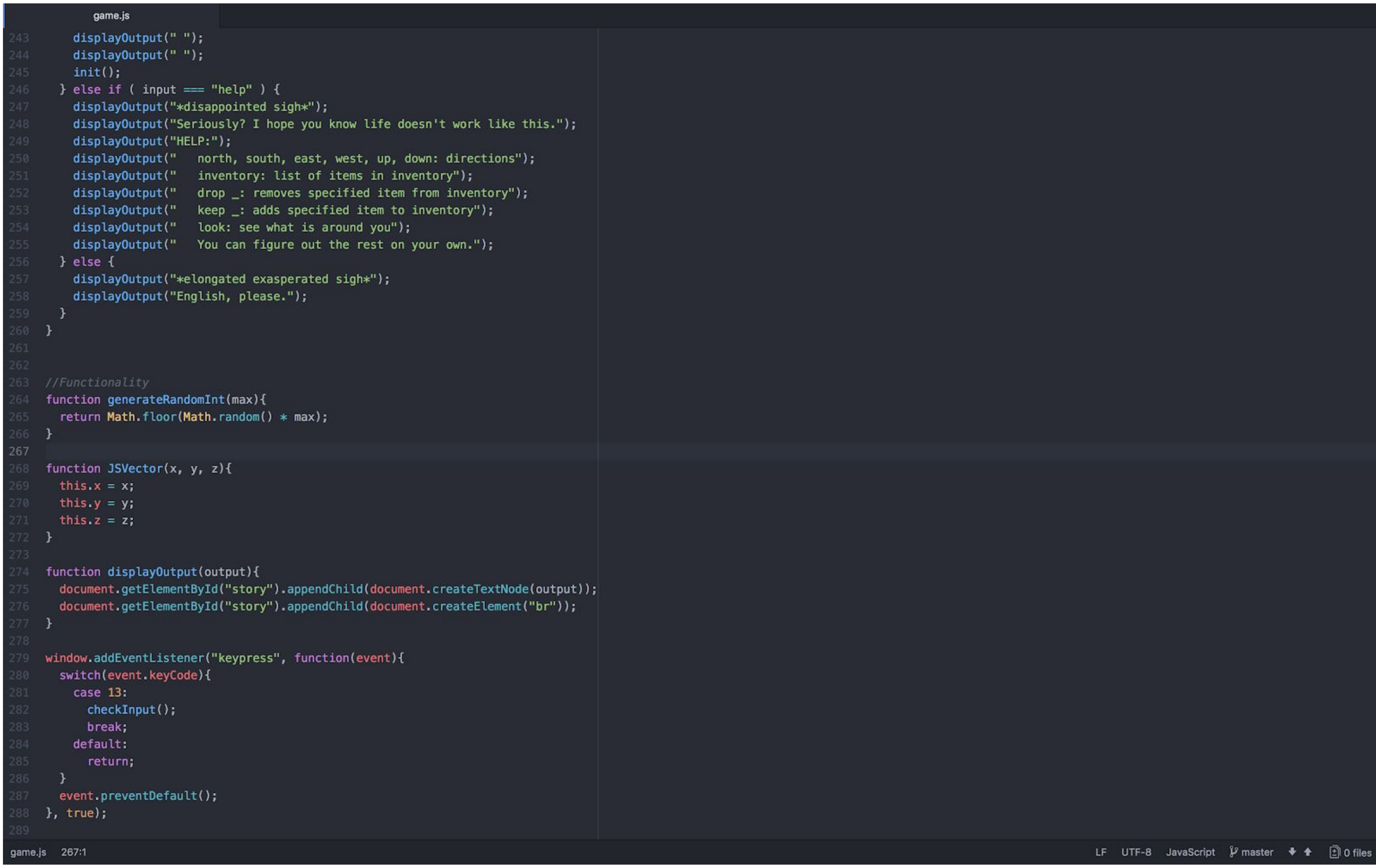
```
    player.inventory.forEach(function(element){
      displayOutput(element);
    });
    if ( !player.inventory.length ) {
      displayOutput("You do not have any material possessions.");
    }
  } else if ( input.includes("east") || (input === "e") ) {
    if (game.map[player.position.x][player.position.y][player.position.z].validMoves.includes("e")) {
      player.update(player.position.x + 1, player.position.y, player.position.z);
      game.map[player.position.x][player.position.y][player.position.z].look();
    } else {
      displayOutput("You imbecile");
      displayOutput("Are you trying to get a concussion? Honestly.");
      displayOutput("Last I checked, you are not able to go through walls.");
    }
  } else if ( input.includes("west") || (input === "w") ) {
    if (game.map[player.position.x][player.position.y][player.position.z].validMoves.includes("w")) {
      player.update(player.position.x - 1, player.position.y, player.position.z);
      game.map[player.position.x][player.position.y][player.position.z].look();
    } else {
      displayOutput("You trilobite");
      displayOutput("Are you trying to get a concussion? Honestly.");
      displayOutput("Last I checked, you are not able to go through walls.");
    }
  } else if ( input.includes("north") || (input === "n") ) {
    if (game.map[player.position.x][player.position.y][player.position.z].validMoves.includes("n")) {
      player.update(player.position.x, player.position.y - 1, player.position.z);
      game.map[player.position.x][player.position.y][player.position.z].look();
    } else {
      displayOutput("YOU FOOL");
      displayOutput("Are you trying to get a concussion? Honestly.");
      displayOutput("Last I checked, you are not able to go through walls.");
    }
  } else if ( input.includes("south") || (input === "s") ) {
    if (game.map[player.position.x][player.position.y][player.position.z].validMoves.includes("s")) {
      player.update(player.position.x, player.position.y + 1, player.position.z);
      game.map[player.position.x][player.position.y][player.position.z].look();
    } else {
      displayOutput("Ya dumb meatball");
      displayOutput("Are you trying to get a concussion? Honestly.");
      displayOutput("Last I checked, you are not able to go through walls.");
    }
  } else if ( input.includes("up") ) {
    if (game.map[player.position.x][player.position.y][player.position.z].validMoves.includes("u")) {
      player.update(player.position.x, player.position.y, player.position.z - 1);
      game.map[player.position.x][player.position.y][player.position.z].look();
    } else {
```



```
      player.update(player.position.x, player.position.y, player.position.z - 1);
      game.map[player.position.x][player.position.y][player.position.z].look();
    } else {
      displayOutput("I smell a GNASHGAB (it's a real word--look it up) ");
      displayOutput("Are you trying to get a concussion? Honestly.");
      displayOutput("Last I checked, you are not able to go through the ceiling.");
    }
  } else if ( input.includes("down") ) {
    if (game.map[player.position.x][player.position.y][player.position.z].validMoves.includes("d")) {
      player.update(player.position.x, player.position.y, player.position.z + 1);
      game.map[player.position.x][player.position.y][player.position.z].look();
    } else {
      displayOutput("Back in my day, noodles weren't THIS stupid");
      displayOutput("Are you trying to get a concussion? Honestly.");
      displayOutput("Last I checked, you are not able to go through the floor.");
    }
  } else if ( input.includes("look") ) {
    game.map[player.position.x][player.position.y][player.position.z].look();
  } else if ( input.includes("drop") ) {
    let valid = false;
    let has = false;
    objects.forEach(function(element) {
      if ( input.includes(element) ) { // valid
        valid = true;
        if ( player.inventory.includes(element) ) { // player has
          has = true;
          player.inventory.splice( player.inventory.indexOf(element), 1 ); // remove from inventory
          game.map[player.position.x][player.position.y][player.position.z].contents.push(element); // add to room
        }
      }
    });
    if ( !valid ) {
      displayOutput("I do not know what that is.");
    } else if ( !has ) {
      displayOutput("You do not have that.");
    } else {
      displayOutput("Dropped.");
    }
  } else if ( input.includes("keep") ) {
    if ( player.inventory.length === gameConfig.maxPlayerInventory ) {
      displayOutput("You are too weak to carry anything else.");
    } else {
      let valid = false;
      let here = false;
      objects.forEach(function(element) {
        if ( input.includes(element) ) { // valid
          valid = true;
```



```
      objects.forEach(function(element) {
        if ( input.includes(element) ) { // valid
          valid = true;
          if ( game.map[player.position.x][player.position.y][player.position.z].contents.includes(element) ) { // in room
            here = true;
            game.map[player.position.x][player.position.y][player.position.z].contents.splice( game.map[player.position.x][player.position.y][player.position.z].contents.indexOf(element), 1 );
            player.inventory.push(element); // add to inventory
          }
        }
      });
      if ( !valid ) {
        displayOutput("I do not know what that is.");
      } else if ( !here ) {
        displayOutput("You do not see that.");
      } else {
        displayOutput("Kept.");
      }
    }
  } else if ( (input === "restart") || (input === "again") || (input === "new game") || (input === "quit") ) {
    displayOutput("I knew you would give up, you miserable shrimp. QUITTER");
    displayOutput(" ");
    displayOutput(" ");
    displayOutput(" ");
    displayOutput(" ");
    init();
  } else if ( input === "help" ) {
    displayOutput("*disappointed sigh*");
    displayOutput("Seriously? I hope you know life doesn't work like this.");
    displayOutput("HELP:");
    displayOutput("   north, south, east, west, up, down: directions");
    displayOutput("   inventory: list of items in inventory");
    displayOutput("   drop _: removes specified item from inventory");
    displayOutput("   keep _: adds specified item to inventory");
    displayOutput("   look: see what is around you");
    displayOutput("   You can figure out the rest on your own.");
  } else {
    displayOutput("*elongated exasperated sigh*");
    displayOutput("English, please.");
  }
}


//Functionality
function generateRandomInt(max){
  return Math.floor(Math.random() * max);
}

```



```
    displayOutput(" ");
    displayOutput(" ");
    init();
  } else if ( input === "help" ) {
    displayOutput("*disappointed sigh*");
    displayOutput("Seriously? I hope you know life doesn't work like this.");
    displayOutput("HELP:");
    displayOutput("   north, south, east, west, up, down: directions");
    displayOutput("   inventory: list of items in inventory");
    displayOutput("   drop _: removes specified item from inventory");
    displayOutput("   keep _: adds specified item to inventory");
    displayOutput("   look: see what is around you");
    displayOutput("   You can figure out the rest on your own.");
  } else {
    displayOutput("*elongated exasperated sigh*");
    displayOutput("English, please.");
  }
}


//Functionality
function generateRandomInt(max){
  return Math.floor(Math.random() * max);
}

function JSVector(x, y, z){
  this.x = x;
  this.y = y;
  this.z = z;
}

function displayOutput(output){
  document.getElementById("story").appendChild(document.createTextNode(output));
  document.getElementById("story").appendChild(document.createElement("br"));
}

window.addEventListener("keypress", function(event){
  switch(event.keyCode){
    case 13:
      checkInput();
      break;
    default:
      return;
  }
  event.preventDefault();
}, true);
```