Colossal Adventure (Console RPG)

20 JULY 2015 on [Programming Principles (/tag/programming-principles/)](https://coursework.vschool.io/tag/programming-principles/), [Level 2 (/tag/level-2/)](https://coursework.vschool.io/tag/level-2/), [Project (/tag/project/)](https://coursework.vschool.io/tag/project/)

The year is 1985. Your job is to build a text-based (console) RPG game.

The point of this exercise is to better master Javascript functions, loops, conditionals, and some data types.

Project Requirements:

1. Console must **greet** player with a **fun message**
2. Console must **ask for the player's name** and **store it**
3. Walking:

The console will ask the user to enter a **"w" to walk**

Every time the player walks, a **random algorithm** will be run that determines if a wild enemy has appeared (**A 1/3 or 1/4 chance of being attacked**)

1. If **a wild enemy** appears:

The **enemy is random** (can be chosen out of a **minimum of 3 different enemy names**)

The user can decide to **attack** or run

If ***attacking***, you will choose a **random attack power** between a **min and max**

If ***running***, you will choose a random number between 1 and 2 - meaning a 50% chance of escaping

After the player attacks or runs the enemy attacks back for a **random damage amount**

If the player kills the enemy you can **give the Player some HP** and a **special item that is stored in the inventory**

If the enemy kills the player the console prints an **cool death message** and the game ends

**1. Inventory**

When the player kills enemies, they are **awarded with items**

If the user enters **'Print'** in the console, the console will print the **players name, HP, and each item in their inventory**

Passing Criteria: Demo to the Instructor/TA & Code Review

An instructor or TA will look through your code with you and make sure you have a solid grasp on using functions, loops, and conditionals in JavaScript, as well as to verify that you're following JavaScript code standards and good general programming practices.

This project passes off the following levels of the Skills Tree:

Programming Principles, Level 2

const readline = require('readline-sync')

​

let isAlive = true;

let hasWon = false;

​

​

function Enemy (name, hp, attack){

this.name = name;

this.hp = hp;

this.attack = attack;

}

​

let greed = new Enemy('Greed', 50, 10)

let sloth = new Enemy('Sloth', 75, 5000)

let wrath = new Enemy('Wrath', 1, 10000)

​

const enemies = [greed, sloth, wrath]

​

​

function Person (name, hp = 100){

this.name = name;

this.hp = hp;

this.attack = 30;

}

​

let name = readline.question('What is your name? ')

​

const playerUno = new Person(name)

​

console.log(`welcome to the game ${name}`)

​

while(isAlive && !hasWon){

let question = readline.keyIn('Would you like to walk? [y] to walk, [n] to end game ', {limit: 'yn'})

​

if(question === 'y'){

walk()

}else if (question === 'n'){

isAlive = false

}

​

}

​

​

function walk (){

let random = Math.floor(Math.random()\*3)

if(random > 0){

console.log('You just walked')

}else {

encounter()

}

}

​

function encounter(){

let enemy = enemies[Math.floor(Math.random()\*enemies.length)]

console.log(`You have encountered ${enemy.name}`)

let action = readline.keyIn('Would you like to run [r], or attack [a]? ', {limit: 'ra'})

if(action === 'r'){

console.log('you ran away')

}else {

attack(enemy)

}

}

​

function attack (enemy){

while(player1.hp > 0 && enemy.hp > 0){

let enemyAttack = generateAttack(enemy)

let heroAttack = generateAttack(player1)

player1.hp -= enemyAttack;

console.log(`${enemy.name} attacked you!, your hp is now ${player1.hp}`);

enemy.hp -= heroAttack;

console.log(`You hit ${enemy.name}! ${enemy.name} now has an hp of ${enemy.hp}`);

}

if(player1.hp <= 0){

console.log('Game over, you dead')

isAlive = false;

}else {

console.log(`You killed ${enemy.name}`)

enemies.splice(enemies.indexOf(enemy), 1)

if(enemies.length === 0){

console.log('You won the game')

hasWon = true

}

}

}

​

function generateAttack (player){

return Math.floor(Math.random() \* player.attack)

}

const readline = require('readline-sync')

let isAlive = true;

let hasWon = false;

function Enemy (name, hp, attack){

this.name = name;

this.hp = hp;

this.attack = attack;

}

let terminator = new Enemy('Terminator', 50, 10)

let robocop = new Enemy('Robocop', 75, 5000)

let cyborg = new Enemy('Cyborg', 1, 10000)

const enemies = [terminator, robocop, cyborg]

function Person (name, hp = 100){

this.name = name;

this.hp = hp;

this.attack = 30;

}

let name = readline.question('What is your name? ')

const player1 = new Person(name)

console.log(`welcome to the game ${name}`)

while(isAlive && !hasWon){

let question = readline.keyIn('Would you like to walk? [y] to walk, [n] to end game ', {limit: 'yn'})

if(question === 'y'){

walk()

}else if (question === 'n'){

isAlive = false

}

}

function walk (){

let random = Math.floor(Math.random()\*3)

if(random > 0){

console.log('You just walked')

}else {

encounter()

}

}

function encounter(){

let enemy = enemies[Math.floor(Math.random()\*enemies.length)]

console.log(`You have encountered ${enemy.name}`)

let action = readline.keyIn('Would you like to run [r], or attack [a]? ', {limit: 'ra'})

if(action === 'r'){

console.log('you ran away')

}else {

attack(enemy)

}

}

function attack (enemy){

while(player1.hp > 0 && enemy.hp > 0){

let enemyAttack = generateAttack(enemy)

let heroAttack = generateAttack(player1)

player1.hp -= enemyAttack;

console.log(`${enemy.name} attacked you!, your hp is now ${player1.hp}`);

enemy.hp -= heroAttack;

console.log(`You hit ${enemy.name}! ${enemy.name} now has an hp of ${enemy.hp}`);

}

if(player1.hp <= 0){

console.log('Game over, you dead')

isAlive = false;

}else {

console.log(`You killed ${enemy.name}`)

enemies.splice(enemies.indexOf(enemy), 1)

if(enemies.length === 0){

console.log('You won the game')

hasWon = true

}

}

}

function generateAttack (player){

return Math.floor(Math.random() \* player.attack)

}

Awards

|  |
| --- |
| class Status { |
|  |  |  | constructor ({character}) { |
|  |  |  | this.\_character = character |
|  |  |  | } |
|  |  |  | get (name) { |
|  |  |  | let status = {} |
|  |  |  | this.\_addToStatus(status, this.\_character.equipment.getModifiers()) |
|  |  |  | this.\_addToStatus(status, this.\_character.characteristics.getCharacteristics()) |
|  |  |  | this.\_addToStatus(status, this.\_character.role.getCharacteristics()) |
|  |  |  | this.\_addCurrencies(status) |
|  |  |  |  |
|  |  |  | status.experience = this.\_character.experience.getExperience() |
|  |  |  | status.level = this.\_character.experience.computeLevel() |
|  |  |  |  |
|  |  |  | return name ? status[name] || 0 : status |
|  |  |  | } |
|  |  |  | \_addCurrencies (status) { |
|  |  |  | let currencies = this.\_character.bank.getCurrencies() |
|  |  |  | Object.keys(currencies).forEach(key => { |
|  |  |  | status[key] = currencies[key].getValue() |
|  |  |  | }) |
|  |  |  | } |
|  |  |  | \_addToStatus (status, characteristics) { |
|  |  |  | characteristics.forEach(characteristic => { |
|  |  |  | if (!characteristic || !characteristic.getName || !characteristic.getValue) return |

|  |
| --- |
| class Status { |
|  | constructor ({character}) { |
|  | this.\_character = character |
|  | } |
|  | get (name) { |
|  | let status = {} |
|  | this.\_addToStatus(status, this.\_character.equipment.getModifiers()) |
|  | this.\_addToStatus(status, this.\_character.characteristics.getCharacteristics()) |
|  | this.\_addToStatus(status, this.\_character.role.getCharacteristics()) |
|  | this.\_addCurrencies(status) |
|  |  |
|  | status.experience = this.\_character.experience.getExperience() |
|  | status.level = this.\_character.experience.computeLevel() |
|  |  |
|  | return name ? status[name] || 0 : status |
|  | } |
|  | \_addCurrencies (status) { |
|  | let currencies = this.\_character.bank.getCurrencies() |
|  | Object.keys(currencies).forEach(key => { |
|  | status[key] = currencies[key].getValue() |
|  | }) |
|  | } |
|  | \_addToStatus (status, characteristics) { |
|  | characteristics.forEach(characteristic => { |
|  | if (!characteristic || !characteristic.getName || !characteristic.getValue) return |
|  | if (status[characteristic.getName()]) { |
|  | status[characteristic.getName()] += characteristic.getValue() |
|  | } else { |
|  | status[characteristic.getName()] = characteristic.getValue() |
|  | } |
|  | }) |
|  | return status |
|  | } |
|  | } |
|  |  |
|  | export default Status |