Object Oriented Programming

In this lesson, we will take a detailed look at the Character Class and discuss some of its important methods.

Now back to our RPG, which is still pretty boring. What does it lack? Monsters and fights, of course!

Following is how a fight will be handled. If attacked, a character sees their life points decrease from the strength of the attacker. If its health value falls below zero, the character is considered dead and cannot attack anymore. Its vanquisher receives a fixed number of 10 experience points.

First, let's add the capability for our characters to fight one another. Since it's a shared ability, we define it as a method named `attack()` in the `Character` class.

```
class Character {
  constructor(name, health, strength) {
    this.name = name;
    this.health = health;
    this.strength = strength;
    this.xp = 0; // XP is always zero for new characters
  }
  // Attack a target
  attack(target) {
    if (this.health > 0) {
      const damage = this.strength;
      console.log(
        `${this.name} attacks ${target.name} and causes ${damage} damage points`
      );
      target.health -= damage;
      if (target.health > 0) {
        console.log(`${target.name} has ${target.health} health points left`);
      } else {
        target.health = 0;
        const bonusXP = 10;
        console.log(
          `${this
            .name} eliminated ${target.name} and wins ${bonusXP} experience points`
        );
        this.xp += bonusXP;
      }
    } else {
      console.log(`${this.name} can't attack (they've been eliminated)`);
    }
```

```
  }
  // Return the character description
  describe() {
    return `${this.name} has ${this.health} health points, ${this
      .strength} as strength and ${this.xp} XP points`;
  }
}
```

Now we can introduce a monster in the game and make it fight our players. Here's the rest of the final code of our RPG.

```
const aurora = new Character("Aurora", 150, 25);
const glacius = new Character("Glacius", 130, 30);

console.log("Welcome to the adventure! Here are our heroes:");
console.log(aurora.describe());
console.log(glacius.describe());

const monster = new Character("Spike", 40, 20);
console.log("A wild monster has appeared: it's named " + monster.name);

monster.attack(aurora);
monster.attack(glacius);
aurora.attack(monster);
glacius.attack(monster);

console.log(aurora.describe());
console.log(glacius.describe());
```

The previous program is a short example of *Object-Oriented Programming* (in short: OOP), a programming paradigm (a programming style) based on objects containing both data and behavior.