React favours composition over inheritance

In this lesson, we study why React programmers prefer to use class composition over class inheritance.

Why not inheritance?

Imagine you are making a video game and need two characters: one that can shoot lasers and another that can cast spells. Both characters have a name and health.

You solve this problem neatly with inheritance. As per the following, a class structure with one parent class, the `character`, and two child classes called `caster` and `shooter` can be created.

```
class Character {
  constructor(name){
    this.name = name;
    this.health = 100;
  }
}

class Shooter extends Character{
  constructor(name){
    super(name);
  }

  shoot(){
    console.log(`${this.name}: prepare to die!`);
    this.health--;
  }

   getHealth(){
     console.log(this.health);
   }

}


class Caster extends Character{
  constructor(name){
    super(name);
  }

  cast(){
    console.log(`${this.name}: Avada Kedavra!`);
    this.health--;
  }
```

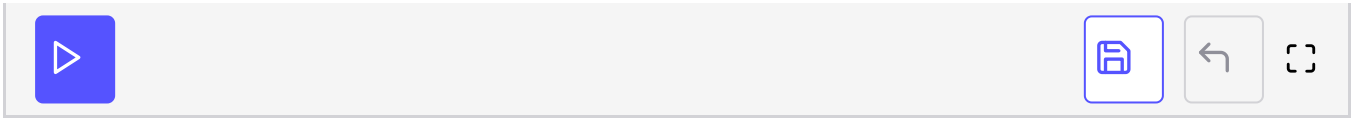
```
  }

   getHealth(){
     console.log(this.health);
   }

}

Dumbledore = new Caster("Albus Percival Wulfric Brian Dumbledore")
Dumbledore.cast();

DarthVader = new Shooter("Anakin Skywalker");
DarthVader.shoot();
```



Suppose you launched your game and it was a big hit. But now, your game's buyers demand another character that can both shoot lasers and cast spells.

There are two things you can do:

1. Make the `shoot()` and `cast()` functions a part of the parent `Character` class and have the `ShooterFighter` class inherit it. The only problem with this is that the `shooter` and `caster` would inherit `cast()` and `shoot()` which they do not need. This problem is called the Gorilla/Banana problem, i.e., you need a banana but get a gorilla and the entire forest with it because of the class inheritance structure.
2. Make copies of the `shoot()` and `cast()` functions and add them as local methods to the 'ShooterFighter' class but this means that you are copying code - which is bad practice as it defeats the purpose of using classes. This problem is called the duplication by necessity problem.

Composition

So, to get around these issues, we can use composition. *Object composition*, in programming, is just classes containing instances of other classes as attributes to implement the desired functionality instead of inheriting classes. How that is done in React and JavaScript is somewhat advanced, and is beyond the scope of this course.

However, you can check out my course The Road to Learn React to study it. Essentially, inheritance follows the **IS-A** principle - which means that child classes *are* parent classes. So in our example, the `Caster` IS-A `Character`. In

contrast, composition uses the **HAS-A** principle - so a class 'HAS-A' certain

attribute or functionality. In our example, the 'ShooterFighter' class 'HAS-A' `shoot()` function and 'HAS-A' `cast()` function.

Advantages of composition over inheritance

- Class taxonomy does not have to be defined in advance. This makes the code dynamic and adaptable to change.
- Since lesser changes will have to be made in the code, it will also introduce fewer bugs.
- The code is more reusable

Thus, it is evident that using composition over inheritance has serious advantages. Actually, the developers of Facebook, who also developed React and use it *to* develop Facebook (phew!), claim that there haven't been any use cases where they would recommend creating inheritance hierarchies!