Inheritance in OOP

Let's say in our game we have players, ~~they~~ there could be different types of players in the same game. Let's say our game has a hero who can have weapons like gun, knife etc and the villian who can destroy things around with a special power.

If we wanted to design this game, both hero and villian would have name, battery and charge function. But only hero has a weapon and villian has a method that destroys things.

How do we solve this?

Option 1: create 2 different classes each for hero and villian. But we would be repeating code, since they both have something in common. This is where inheritance comes into picture

* We first create a (common) base class that has the common functionality for all players.

```
class Player {
    constructor(name, battery) {
        this.name = name;
        this.battery = battery;
    }
    charge() {
        this.battery = 100;
    }
}
```

* Then we create a class for hero/villian that inherits all that Player class has and any new functionaly we need to add

```
class Hero extends Player {
    constructor(name, battery, weapon){
        super();
        this.weapon = weapon;
    }
}
```

* extends is a keyword that helps us inherit from ~~for~~ a base class (also called super class) ~~now Hero class~~

> const hero = new Hero('John', 80, 'gun')

Now a new hero is created with a weapon gun which a general Player doesn't have.

Also hero has access to data and methods defined in Player class

> hero.charge();

↳ hero has access to charge method

* We are not instantiating Player anywhere, we're directly instantiating Hero class, but Hero extends Player, so we need to pass the parameters (name, battery) required by Player class in order ~~to~~ for it to work.

So what about super()? Why is that required?

In order for a class (that extends from another class) to have access to the this keyword, we need to first always call super() because ~~super actu~~ this

* actually comes from super class

So if we try to do

this.weapon = weapon before calling super() it will give us an error that this is not defined.

And what does this contain after we call super?

> this

{name: 'John', battery: 80}

and by doing this.weapon = 'gun' we adding to the this coming from super

> this.weapon = 'gun'

> this

{name: 'John', batt-ery: 80, weapon: 'gun'}