**Class is just a function**

Just to prove that there isn't anything special about class, check out this code:

**class** **Plane** {

constructor(numEngines) {

**this**.numEngines = numEngines;

**this**.enginesActive = false;

}

startEngines() {

console.log('starting engines…');

**this**.enginesActive = true;

}

}

**typeof** Plane; *// function*

***Returns:****function*

That's right—it's just a function! There isn't even a new type added to JavaScript.

***⚠️ Where Are All The Commas? ⚠️***

*Did you notice that there aren't any commas between the method definitions in the Class? Commas are not used to separate properties or methods in a Class. If you add them, you'll get a SyntaxError of unexpected token ,*

**QUIZ QUESTION**

Take a look at the following code:

**class** **Animal** {

constructor(name = 'Sprinkles', energy = 100) {

**this**.name = name;

**this**.energy = energy;

}

eat(food) {

**this**.energy += food / 3;

}

}

Which of the following are true?

* The eat() method ends up on Animal.prototype.
* 

typeof Animal === 'class'

* typeof Animal === 'function'

SUBMIT

**Static methods**

To add a static method, the keyword static is placed in front of the method name. Look at the badWeather() method in the code below.

**class** **Plane** {

constructor(numEngines) {

**this**.numEngines = numEngines;

**this**.enginesActive = false;

}

static badWeather(planes) {

**for** (plane **of** planes) {

plane.enginesActive = false;

}

}

startEngines() {

console.log('starting engines…');

**this**.enginesActive = true;

}

}

See how badWeather() has the word static in front of it while startEngines() doesn't? That makes badWeather() a method that's accessed directly on the Plane class, so you can call it like this:

Plane.badWeather([plane1, plane2, plane3]);

***NOTE:****A little hazy on how constructor functions, class methods, or prototypal inheritance works? We've got a course on it! Check out*[*Object Oriented JavaScript*](https://www.udacity.com/course/object-oriented-javascript--ud015)*.*

**Benefits of classes**

1. Less setup
   * There's a lot less code that you need to write to create a function
2. Clearly defined constructor function
   * Inside the class definition, you can clearly specify the constructor function.
3. Everything's contained
   * All code that's needed for the class is contained in the class declaration. Instead of having the constructor function in one place, then adding methods to the prototype one-by-one, you can do everything all at once!

**Things to look out for when using classes**

1. class is not magic
   * The class keyword brings with it a lot of mental constructs from other, class-based languages. It doesn't magically add this functionality to JavaScript classes.
2. class is a mirage over prototypal inheritance
   * We've said this many times before, but under the hood, a JavaScript class just uses prototypal inheritance.
3. Using classes requires the use of new
   * When creating a new instance of a JavaScript class, the new keyword must be used

For example,

**class** **Toy** {

...

}

**const** myToy1 = Toy(); *// throws an error*

***Uncaught TypeError:****Class constructor Toy cannot be invoked without 'new'*

but, adding the new keyword fixes the problem

**const** myToy2 = **new** Toy(); *// this works!*