**ES5 "Class" Recap**

Since ES6 classes are just a mirage and hide the fact that prototypal inheritance is actually going on under the hood, let's quickly look at how to create a "class" with ES5 code:

**function** **Plane**(numEngines) {

**this**.numEngines = numEngines;

**this**.enginesActive = false;

}

*// methods "inherited" by all instances*

Plane.prototype.startEngines = **function** () {

console.log('starting engines...');

**this**.enginesActive = true;

};

**var** richardsPlane = **new** Plane(1);

richardsPlane.startEngines();

**var** jamesPlane = **new** Plane(4);

jamesPlane.startEngines();

In the code above, the Plane function is a *constructor function* that will create new Plane objects. The data for a specific Plane object is passed to the Plane function and is set on the object. Methods that are "inherited" by each Plane object are placed on the Plane.prototype object. Then richardsPlane is created with one engine while jamesPlane is created with 4 engines. Both objects, however, use the same startEngines method to activate their respective engines.

Things to note:

* the constructor function is called with the new keyword
* the constructor function, by convention, starts with a capital letter
* the constructor function controls the setting of data on the objects that will be created
* "inherited" methods are placed on the constructor function's prototype object

Keep these in mind as we look at how ES6 classes work because, remember, ES6 classes set up all of this for you under the hood.

**ES6 Classes**

Here's what that same Plane class would look like if it were written using the new class syntax:

**class** **Plane** {

constructor(numEngines) {

**this**.numEngines = numEngines;

**this**.enginesActive = false;

}

startEngines() {

console.log('starting engines…');

**this**.enginesActive = true;

}

}