**Classes**

1. **Class**

ES6 provides a way to create objects using the ‘class’ keyword. The class syntax replaces the constructor function creation 🡪 Only syntatic sugar

**class <ClassName>{**

**constructor (<para>, <para>…){**

this.<property> = para;

**}**

<method> = function() {

//statement with <para> or this.<property>,….;

}

}

We can then create Object instances using the **new** keyword as constructor function

**const <instanceName> = new <ClassName> (para1, para2, …)**

🡪 In general, class syntax is a more readable way of declaring constructor functions.

*\*We can also create a Class generator function, e.g., called makeClass(), that takes no argument and return the Class itself.*

|  |
| --- |
| *function makeClass () {*  *class ReturnClass {*  *constructor (color) {*  *this.color = colorr;*  *}*  *}*  *return ReturnClass;*  *}*  *const Vegetable = makeClass();*  *const carrot = new Vegetable(‘orange’); // ‘orange’ passed into the constructor function of Vegetable class*  *console.log (carrot.color); // orange* |

1. **Inheritance**

When multiple classes share properties or methods, they become candidates for *inheritance.* With inheritance, we can create a parent class, containing properties and methods that are shared by child classes

Text

Description automatically generated with medium confidence

- First, we create the parent class.

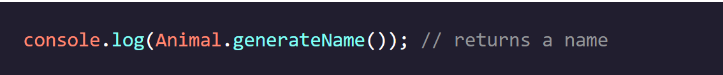
- Then, we can extend the defined properties and methods to subclass, using the class…extends… keywords.

class Cat extends Animal {  
   constructor(name, usesLitter) {  
     super(name);  
     this.\_usesLitter = usesLitter;  
   }  
}

* The extends keyword makes the methods of the Animal class available inside the Cat class.
* The constructor, called when you create a new Cat object, accepts two arguments, name and usesLitter.
* The super keyword calls the constructor of the parent class. In this case, super(name) passes the name argument of the Cat class to the constructor of the Animal class. When the Animal constructor runs, it sets this.\_name = name; for new Cat instances.
* \_usesLitter is a new property that is unique to the Cat class, so we set it in the Cat constructor.

1. **Static Methods**

Sometimes, we might want to create methods that can only be accessed via the classes themselves, NOT the instances. To do this, we can add the keyword **static** before a method while defining the class.

Text

Description automatically generatedText

Description automatically generated

\*\*Static methods are often utility functions, such as functions to create or clone objects, whereas static properties are useful for caches, fixed-configuration, or any other data you don't need to be replicated across instances.