Main Concept:

A method can be extracted from an object into a separated variable

const alone = myObj.myMethod

When the method is called alone:

alone() ----> FUNCTION INVOCATION

detached from the original object, you might think that ‘this’ is the object myObj on which the method was defined.

Correctly if the method is called without an object, then a function invocation happens, where ‘this’ is the *global object window* or *undefined* in strict mode

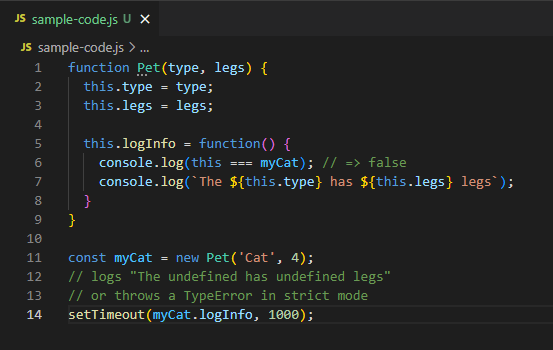
Fixing this misconception using bound function

* .bind(obj) method

const alone = myObj.myMethod.bind(myObj)

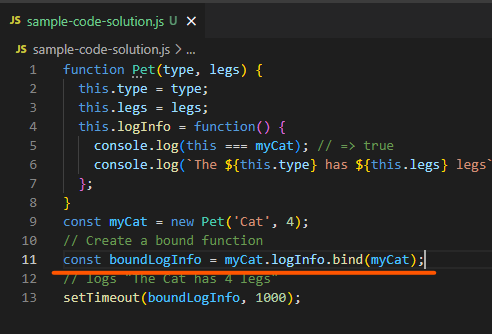
* ‘this’ is now equal to myObj

Sample Code inside a constructor:

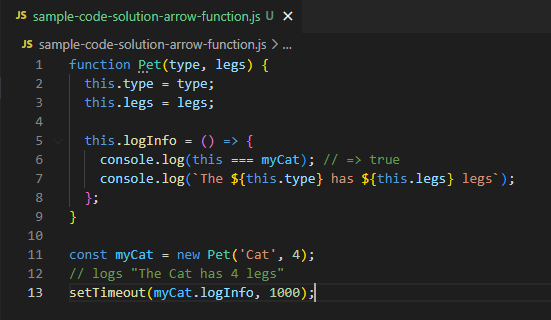


.logInfo is treated as a separate VARIABLE that contains the function(). Thus calling this.logInfo invokes a FUNCTION invocation which in turn return ‘this’ as *undefined* (in strict-mode) or *global object* (non-strict)

Again this can be solved by bounding the obj



Another solution is ARROW function



Calling this.logInfo will call the anonymous method thus performing a METHOD invocation

ES2015 class

Solution: use arrow function

