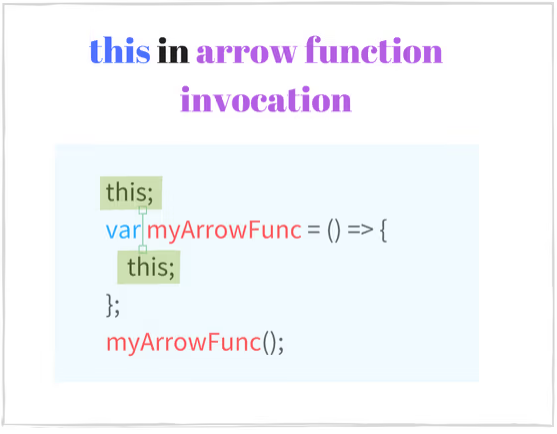
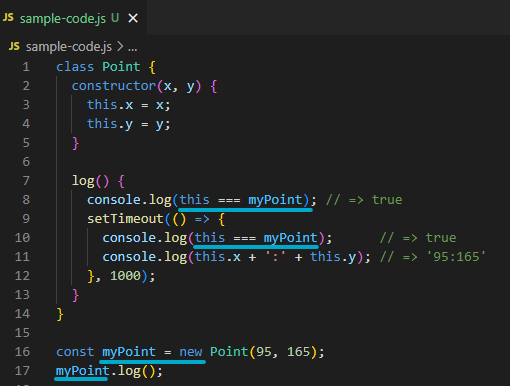
Main Concept:

‘this’ is the enclosing context *where the arrow function is defined*

The arrow function doesn't create its own execution context but takes this from the *outer function where it is defined*. In other words, the arrow function resolves ‘this’ lexically.



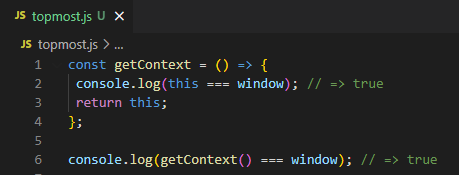
Example:



A regular function in this example would create its own context (window or undefined in strict mode). So to make the same code work correctly with a function expression it's necessary to manually bind the context: setTimeout(function() {...}.bind(this)). This is verbose, and using an arrow function is a cleaner and shorter solution.

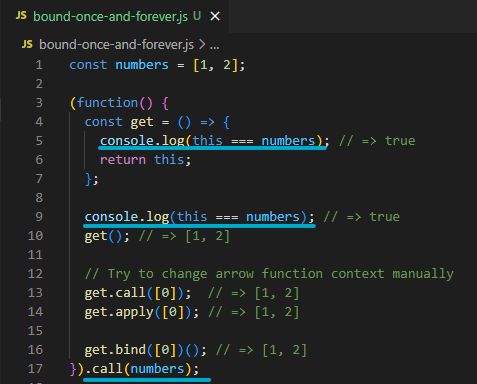
Top-most scope arrow function

* Context is always the *global object* (window in a browser)



Bound with the lexical ‘this’ ONCE AND FOREVER

* Cannot be modified even context modification methods (.call, .apply and rebinding)



Anywhere in the scope, ‘this’ will always be numbers

No matter how the arrow function get() is called, it always keeps the lexical context numbers.

Important note:

An arrow function CANNOT BE USED AS A CONSTRUCTOR

* + Invoking it as a constructor new get() throws an error: TypeError: get is not a constructor.