Код программы

*let* worker = {

    someMethod() {

      return 1;

    },

    slow(*x*) {

      alert("Called with " + *x*);

      return *x* \* this.someMethod(); // (\*)

    }

  };

*function* cachingDecorator(*func*) {

*let* cache = new *Map*();

    return *function*(*x*) {

      if (cache.has(*x*)) {

        return cache.get(*x*);

      }

      //В данном случае вызывается просто функция slow из параметра func. Но эта функция не понимает откуда брать this, ведь она вызвана не из объекта. Поэтому используем call для привязки к контексту. В this попадет объект worker, т.к. на 189 строке функция вызывается из объекта worker.

*let* result = *func*.call(this, *x*); // теперь 'this' передаётся правильно

      cache.set(*x*, result);

      return result;

    };

  }

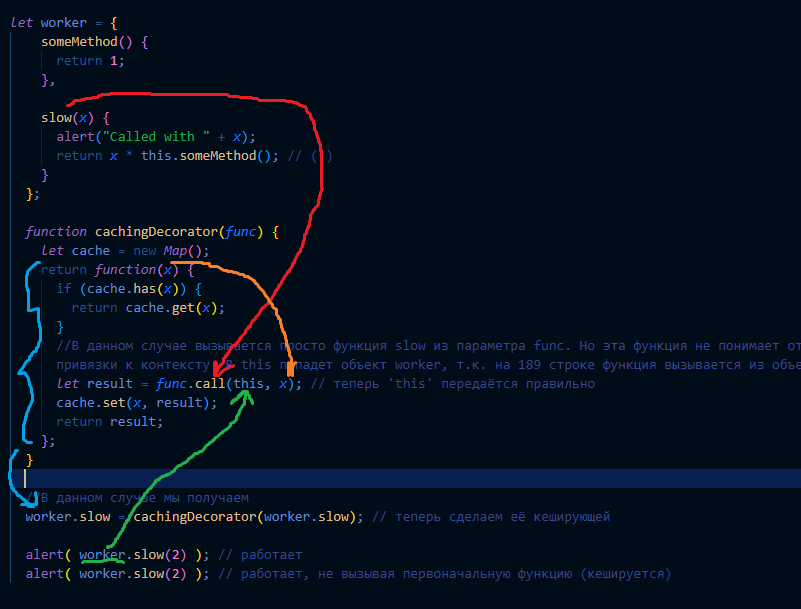
  //В данном случае мы получаем

  worker.slow = cachingDecorator(worker.slow); // теперь сделаем её кеширующей

  alert( worker.slow(2) ); // работает

  alert( worker.slow(2) ); // работает, не вызывая первоначальную функцию (кешируется)

Объяснение



Чтобы всё было понятно, давайте посмотрим глубже, как передаётся this:

1. После *декорации* worker.slow становится обёрткой function (x) { ... }.
2. Так что при выполнении worker.slow(2) обёртка получает 2 в качестве аргумента и this=worker (так как это объект перед точкой).
3. Внутри обёртки, если результат ещё не кеширован, func.call(this, x) передаёт текущий this (=worker) и текущий аргумент (=2) в оригинальную функцию.