**Код прозрачного кэширования функции**

*function* slow(*x*) {

    // здесь могут быть ресурсоёмкие вычисления

    alert(`Called with ${*x*}`);

    return *x*;

  }

*function* cachingDecorator(*func*) {

    //Создали коллекцию

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

    return *function*(*x*) {

      if (cache.has(*x*)) {    // если кеш содержит такой x,

        return cache.get(*x*); // читаем из него результат

      }

*let* result = func(*x*); // иначе, вызываем функцию

      cache.set(*x*, result); // и кешируем (запоминаем) результат

      return result;

    };

  }

  //Передали функцию slow в cachingDecorator

  slow = cachingDecorator(slow);

alert( slow(1) ); // slow(1) кешируем

alert( "Again: " + slow(1) ); // возвращаем из кеша

alert( slow(2) ); // slow(2) кешируем

alert( "Again: " + slow(2) ); // возвращаем из кеша \*/

cachingDecorator – это декоратор, специальная функция, которая принимает другую функцию и изменяет её поведение.

Объяснение



На 162 строке описана функция slow, которая выполняет математические вычисление над числом x, которое содержится в параметре и возвращает преобразованное число х.

На 168 строке описана функция chachingDecorator, которая выполняет кэширование.

На 185 строке функции slow присвоили функцию chachingDecorator(slow), которая в качестве аргумента принимает функцию slow. Функция chachingDecorator создает коллекцию и выводит новую функцию. Как раз функция на 172 строке, которая содержится в return и попадет в slow на 185 строке. Этот участок кода помечен красным маркером.

На 187 строке вызываем slow из 185 строки и передаем ему аргумент 1. Этот аргумент попадает в х на 172 строке. Далее проводится проверка если в коллекции cache есть переменная х (has), то получаем эту переменную из коллекции (get) и передаем в return. И значение из return выводится через alert в 187 строке. При первом вызове функции эта переменная по любому будет отсутствовать в коллекции cache/

Если такое переменной нет, значит создаем переменную result и помещаем в нее вызов функции slow, которую передали в качестве аргументу функции chachingDecorator на 185 строке с аргументом Х. По итогу работы функции func(x) выполнится математическая операция над Х и все попадет в return. По итогу в result окажется преобразованное число Х.

На 179 строке запишем свойство х со значением result в коллекцию cache и выведем результат в return на 187 строке.

Теперь при повторном вызове функции данные не будут повторно высчитываться, а будут браться из коллекции cache, т.к. будет выполнятся условие на 173 строке.