Области Видимостей (Scope)

Имеется такой код:

let i = 12;

let func = function () {

let i = 24;

console.log ('func: ' + i);

let innerFunc = function() {

let i = 36;

console.log ('innerFunc: ' + i);

}

console.log ('после innerFunc: ' + i);

innerFunc ();

}

console.log ('перед: ' + i);

func ();

console.log ('после: ' + i);

*Область видимости* – это блок кода, в котором переменная может использоваться. Области делятся на локальные, которые определены в какой-либо функции и глобальные, определенные не в функции.

Т.е. i в 1 строке кода – это глобальная переменная, а все остальные объявления локальны.

У локальной переменной приоритет выше. Но не надо путать объявление и изменение, если в третий строке убрать let, то изменится глобальное значение. Получается надо всегда указывать let, только если не в глобальной области.

Вложенные функции создают *цепные области видимости*, разберем на примере поиска I во вложенной функции:

1. Посмотреть на локальную область функции innerFunc
2. Если не нашел, то пойти на верх, а на верху функция func
3. Посмотреть в локальную область функции func
4. Если не нашел, то пойти на верх
5. … до тех-пор пока не дойдет до глобальной области
6. Если нет в глобальной области, то ReferenceError

Замыкание (closure)

Замыкание – это функция, которая содержит локальные переменные и возвращаемую вложенную функцию, которая работает с этими локальными переменными.

Можно сказать, что замыкание позволяет оградить локальные переменные от внешних переменных. Уже ранее о нем упоминалось в примере с кэшированием. Так что можно сказать замыкания часто используют в качестве обвертки.

Пример:

function counter(initial=1) {

let i = initial - 1;

return function () {

i += 1;

console.log (i);

}

}

i = 123;

c = counter ();

c();

с();

с();

с();

с();

Несмотря на то, что перед вызовом была объявлена такая же переменная, она не повлияла на counter(). Это пример без особой трудности преобразовать в обвертку.

Все это работает из-за *лексического окружения* (LexicalEnvironment) – это внутренний объект функции. Он представляет две сущности *глобальную* и *локальную* область видимости.

Получается во вложенной функции глобальная область – это локальная область внешней функции. Это немного исковерканная трактовка, но она дает понимание замыкания. Другие примеры:

Считает сумму двух чисел, такой пример можно применить, когда делаешь калькулятор:

function sum(x) {

return function (y) {

let r = x + y;

console.log (`${x} + ${y}:= ${r}`);

return r;

}

}

sum(3)(5);

Постоянно прописывать новые фильтры, меняя, к примеру, только границы неудобно. Благодаря замыканию можно сделать фабрику этих фильтров.

let inBeetwen = (x, y) => num => num >= x && num <= y;

let a = [1,1,1,3,4,5,6,8].filter(inBeetwen(2, 5));

console.log (a);

Обобщим применение замыканий:

* обвертки для функции
* реализации всяких фильтров и т.п.