1. Для практикума из занятия 7 продумать, где можно применить замыкания.

Мне не удалось нормально выполнить это задание. «придумать» не вышло, я пошла читать гугл. Хабр-статья привела несколько примеров и объяснила примерно то же, что вот, очень удобно изолировать переменные. Это звучит прекрасно, но непонятно. Про lexical scope тоже понятно, но куда это применять – снова нет идей. Статьи на <https://developer.mozilla.org/ru/docs/Web/JavaScript/Closures> и <https://www.w3schools.com/js/js_function_closures.asp> круто объясняют, что это вот очень удобно для «эмуляции privat-методов, как если бы это работало в Java и прочих языках», но базовое представление о private-методах не очень дает понятие, почему и какие задачи в js надо реализовывать замыканиями.

Отдельно прекрасная статья <https://medium.com/javascript-scene/master-the-javascript-interview-what-is-a-closure-b2f0d2152b36> тоже возвращает нас к тезису «такое крутое свойство, вот, переменные замыкает на себя, но зачем применять, нуууу… зато это частый вопрос на собеседованиях». Возможно применение замыканий – собеседования? Если серьезно, от чтения стопки источников у меня появился вопрос, собственно, зачем и почему в описываемых примерах таки нужно это замыкание и почему все же нельзя реализовать те же задачи иным способом? Потому что вроде бы можно, то есть замыканиями мы обходим какой-то фундаментальный недостаток js, не позволяющий без них сделать…. А ЧТО сделать – остается непонятным. =( Извините за дурацкий ответ =(

1. Не выполняя кода, ответить, что выведет браузер и почему:

Не выполняя кода фокус не удался, но я так понимаю, задачки в том числе для того, чтобы разобраться, да и ресурса с полноценным решением все равно не нашлось (хотя нашелся с задачами), так что честно думала и тестировала по всякому сама.

if (!("a" in window)) {

var a = 1;

}

alert(a);

это было интересное исследование, спасибо за задачку. Получим Алерт с содержимым undefined, потому что глобальная переменная -> вроде как объявлена в том числе за пределами if, но по умолчанию не определена. А значит при поиске ее в window, находится, а значит в условие возвращает true и содержимое if не выполняется, а значит присвоения 1 не случается. Она так и остаётся undefined

var b = function a(x) {

x && a(--x);

};

alert(a);

Это было сложно, тут по идее сломано в 2 местах. Мы из функции ничего не передаём, нет ни return, ничего. Потом мы просим а, которое само по себе е переменная, ничего. Поэтому получаем ошибку и undefined. Если вписать retern и запросить например b(натуральное число) - рекурсивная функция дойдет до 0, вернёт левое значение аргумента и остановится. Впрочем, если спросить отрицательное - уйдет в бесконечную рекурсию

function a(x) {

return x \* 2;

}

var a;

alert(a);

Получаем тело функции, мы запихнули функцию в переменную, а вызываем переменную без скобок и аргумента.

function b(x, y, a) {

arguments[2] = 10;

alert(a);

}

b(1, 2, 3);

Получим 10, снаружи приходит аргумент 2, но внутри он по индексу переназначается в 10 и выводится. Это наконец было легко

function a() {

alert(this);

}

a.call(null);

Оо, это было отличное задание. Методом call мы передаём объектив функцию, но null на самом деле не объект, а значит в функцию приходит ничего, а this от отсутствия - object window