***Задание 1***

***Для предыдущих заданий с корзиной каталогом товаром, продумать, где можно применить bind и замыкания.***

***Замыкания.***

Затрудняюсь с разбором данной функции, наверно необходимо больше практики и времени для изучения.  
Если я правильно понимаю, в том участке кода

var card = document.getElementById("card");

  var list = document.createElement("div");

  var Image = document.createElement("img");

card, list, image и т.д — это примеры замыканий. Эти функции делят одно определение тела функции, но при этом они сохраняют различные окружения. Окружение функции card хранит сами карточки товаров, в то время как в окружении list — это блок с картой, а image содержит картинку товара и т.д по коду.Тут функция, возвращаемая function getCard(), использует переменные card/ list и т.д, которые сохраняют нужное значение между ее вызовами (вместо того, чтобы сразу прекратить своё существование с возвратом getCard).

Благодаря замыканию возвращаемая функция «запоминает» параметры, переданные функции создающей, что часто необходимо.  
 Похожая ситуация возникает, когда мы внутреннюю функцию не возвращаем, а вешаем на какое-либо событие — поскольку событие возникает уже после того, как исполнилась функция, замыкание опять же помогает не потерять переданные при создании обработчика данные.

Метод, привязывающий функцию к определённому контексту (т.е. объекту, на который в ней будет указывать слово this) – у меня нет.

В принципе, при создании корзины покупок, содержащей список добавленных продуктов и количество каждого продукта. Включающей метод, вычисляющий общую цену элементов корзины применить callback-замыкания.

***Метод bind()***

**bind()** применяется для каррирования, то есть преобразования функции со многими аргументами в набор функций с одним аргументом. This в программе с корзиной я не применяла, поэтому функции у меня нет.

Если бы мне потребовалось явно задать объект, ссылка на который будет зна

чением this, при вызове функции, тогда бы я воспользовалась методом bind.

***Задание 2***

***Не выполняя кода, ответить, что выведет браузер и почему:***

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

var a = 1;

}

alert(a);

**Ответом будет undefined, т.к атрибута «a» в window нет, и после ifему там неоткуда появится.**

var b = function a(x) {

x && a(--x);

};

alert(a);

**b = function a() { по имени «a» функцию видно только здесь }. Выше мы её не определили, поэтому «a» здесь undefined.**

function a(x) {

return x \* 2;

}

var a;

alert(a);

**Думаю, что var не переинициализирует переменную, поэтому**

**undefined**

function b(x, y, a) {

arguments[2] = 10;

alert(a);

}

**Выводом будет – undefined, т.к a и arguments[2] — это ссылка на одно и то же значение. Иначе мы попадаем на память.**

b(1, 2, 3);

function a() {

alert(this);

}

a.call(null);

**[object Window], в консоли выдает undefined.**

**Здесь this в глобальной области указывает на window. Когда мы вызываем call() у функции с параметром null, то это равнозначно вызову call() без аргументов. А без аргументов вызываемая функция не привязана ни к какому объекту (привязана лишь к «объекту по умолчанию» — window).**