Что бы я посоветовал человеку оказавшемуся на моем месте?

Какие цели я преследую.

Нет четкой цели..

Создавать реальность, а не быть частью ее окружения.

Дерево, что растет вниз, растет и в верх…

Задачи

1. Вы не знаете JS, отработать третью главу.
2. Видео на ноутбуке.
3. Определить цели JS. Краткосрочные и долгосрочные.
4. Законспектировать идеи для проектов.
5. Вопрос по поводу присвоенных значений свойствам объектов + видео с ютуба.
6. Потренировать функции с различными методами.
7. Использование нескольких методов к строкам, когда они используются и вообще где?
8. Заново проделать работу по addEventListener.
9. Выучить пропись стилей через JS.
10. Отрепетировать forEach вариант с массивами через addEventListener
11. Округление чисел toFixed, отдельно записать все подобное методы.
12. Законспектировать, что в JS работаем не через textContent, а через value.
13. Отработать все вкладки в хроме.
14. Теория графов.

Вы не знаете JS

1. Чем String отличается от new String?
2. Объектная обертка примитивных значений, для чего это используется?
3. Instanceof.

Законспектировать

addEventListener – функция в которую передаются параметры событий.

1-й параметр событие которое должно происходить.

2-й параметр функция описывающая поведение при данном событии.

Объект event содержит данные о произошедшем событии. ВНИМАНИЕ!!! Он передается как аргумент callbackfunction в addEventListener.

Метод copywithin

Предназначен для изменения массива. Подставляет вместо уже имеющихся элементов, указанные элементы из этого же массива. Имеет 3 аргумента в форме цифр:

- Первый указывает индекс элемента вместо которого надо начать замену элементов;

- Второй указывает индекс элемента с которого следует начать копировать;

- Третий указывает индекс элемента которым следует закончить копирование(если указан диапазон например 5-6, то будет скопирован всего один элемент, не более);

let arr = [1,2,3,4,5,6,7,8,9,0];

arr.copyWithin(2,6,8)

console.log(arr);

<https://www.youtube.com/watch?utm_campaign=Frontend%2BWeekly&utm_medium=email&utm_source=Frontend_Weekly_228&v=V8ce-QBxD5E&ab_channel=dcode>

Метод reduce

Метод применяющий функцию к массиву. Он перебирает массив и сохраняет промежуточный результат. Reduce принимает параметр-функцию.

let arr = [3,4,5,6,7,8,9];

let a = arr.reduce(function(accumulator, currentValue, index){

console.log(' accumulator = ' + accumulator);

console.log(' currentValue = ' + currentValue);

return accumulator + currentValue;

},

arr[2]

);

Аргументы reduce делятся на 2 части:

- параметры call-back функции, включающие в себя 3 параметра - (accumulator, currentValue, item);

- 2-й аргумент arr[2] – указывает чему может быть равен accumulator, любое число, можно даже массив засунуть, например [0, a[0]], НО стоит помнить, что тогда значение accumulator == [0, a[0]];

accumulator – Основной аргумент с которым мы работаем!!!!!!! До старта reduce оно равно нулевому элементу массива. Accumulator = a[0];

И результат всех наш вычислений, наш return он перезаписывается в accumulator в следующем цикле итераций. У него уже будет измененное значение.

currentValue – второй элемент массива, его текущее значение. На следующем круге итераций, это будет третий элемент массива и т.д.

Index – номер элемента в массиве.

.reduceRight – делает реверс обработки массива, индекс не меняется.

Остаточные параметры в функциях.

Ссылочные типы и типы значения, рассказывается в контексте C#

Переменные типа Integer и Massive имеют разное поведение.

Стек – структура данных работающая по принципу первый вошел, последний вышел. Если глубже, стек это контексты методов.

Контекст методов – область памяти содержащая значения всех локальных переменных объявленных в данном методе.

Переменная – именованная область памяти.

Каждый раз, когда какой-либо метод начинает свою работу, в стеке резервируется определенное количество памяти.

Выделение памяти под число в контексте происходит, не в момент объявления переменной, а еще до того, как выполняется метод. Компилятор во время компиляции анализирует и выделяет необходимое количество памяти на переменные.

Мы не можем остановиться на объявлении переменной дебагером, потому что объявление, не делает ничего. Дебагер останавливается только на действиях, а объявление просто объявляет.

Объявление – инструкция к компилятору, что в контексте данного метода нужно отвести место на локальные переменные.

Есть стек, в котором хранятся контексты методов, в котором хранятся все локальные переменные всех методов, после того, как программа отработает, он будет снят со стека. Все локальные переменные больше не нужны, стек освобождается.

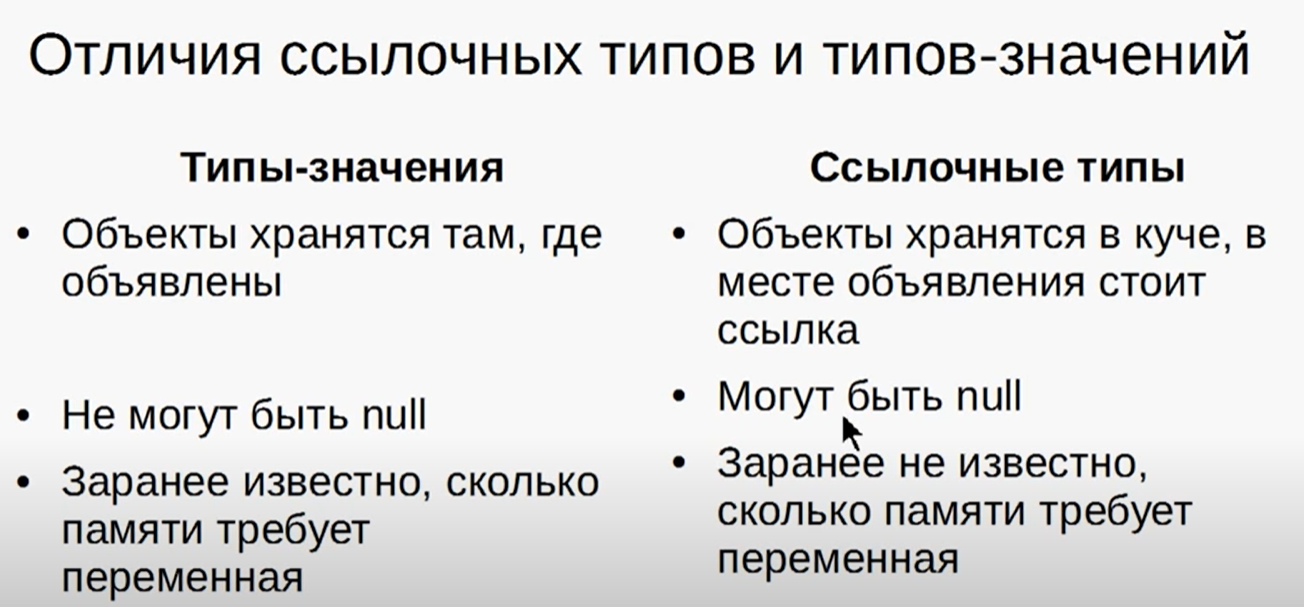
В отличии от локальных, глобальные методы хранятся не в стеке, они хранятся в памяти(“куча – как назвал ее автор”).

С обычными переменными все довольно тривиально, под них резервируется стандартное количество памяти, однако с массивами все сложнее, т.к. мы не можем знать конечную длину массива заранее, оно определяется в момент работы функции - "онлайн". Мы не можем хранить эту память в контексте этой функции, т.к. неизвестно сколько ей нужно места, мы не можем хранить ее в стеке.

Поэтому в контексте функции, массивы и объекты являются ссылками. Ссылки это адрес памяти, показывающая, где искать соответствующий массив, она создается по запросу. Пока массив не инициализирован, ссылка ведет в никуда. И ссылка = null. Несмотря на то, что массив или объект может быть объявлен в функции, место под него резервируется не в стеке, а в куче, которой достаточно для хранения длины массива.

После того, как наша локальная функция выполнилась, стек очищается, соответственно теряется ссылка на наш метод в куче, и дальше, с ним невозможно работать, поэтому он помечается как свободный, чтобы освободить место.

Со строками ситуация аналогичная, как с ссылками, т.к. мы не знаем, конечную длину строк, поэтому на нее идет ссылочный тип данных.



<https://www.youtube.com/watch?v=OMxsc6Csj4k&feature=emb_title&ab_channel=BasicProgramming>

let arr = [3,4,5,6,7,8,9];

let a = arr.reduce(function(prevValue, currentValue){

console.log('prevValue = ' + prevValue);

console.log(' currentValue = ' + currentValue);

console.log(prevValue + currentValue);

return;

}

);

Почему в данном случае выдает Nan?

СОБЫТИЯ

Объект event содержит данные о произошедшем событии. ВНИМАНИЕ!!! Он передается как аргумент callbackfunction в addEventListener.

Его свойства вызываются внутри функции. И если мы хотим управлять самими действиями события, то объект event всегда следует передавать callback функции.

btn[1].addEventListener("mouseenter",function(event){

let target = event.target;

target.style.backgroundColor = "red";

});

//event.type - свойство объекта event(тип события)

//event.target - цель события

Всплытие событий – обработчик срабатывает на самом вложенном элементе, а после, на объемлющих.

Однако иногда нам требуется отменять стандартное поведение событий. Например запретить выделять текст двойным щелчком, или переходить по ссылке.

let link = document.getElementsByTagName("a");

link[0].addEventListener('click', function(event){

event.preventDefault();

})

Мобильные события

Все мобильные события начинаются со слова touch

touchstart — аналог клика

touchmove — перемещение

touchend — конец соприкосновения с сенсором

touchenter — палец намеренно заходит на какой-либо элемент

touchleave — палец вышел из зоны элемента

touchcancel — точка соприкосновения больше не регистрируется внутри браузера

Действовать аккуратно, т.к. могут быть конфликты между обработчиками событий click и touchstart.