<https://www.youtube.com/watch?v=tAgVINdc_o0>

https://www.youtube.com/watch?v=LVVgJGvq\_go

Типы данных (typeof возвращает строку)

Примитивы: [Boolean, String, Number, Null, Undefined, Symbol]

Ссылки (объекты): [Object, Array, Function, RegExp, Date, Error]

typeof null => object

typeof function () {} => function

new Boolean(false) => создаст объект соответственно всегда true

Boolean(false) => будет false так как просто преобразует

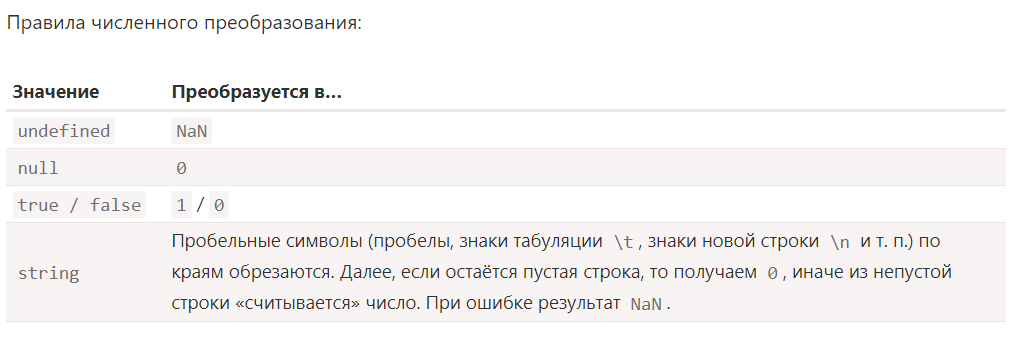
Преобразование типов

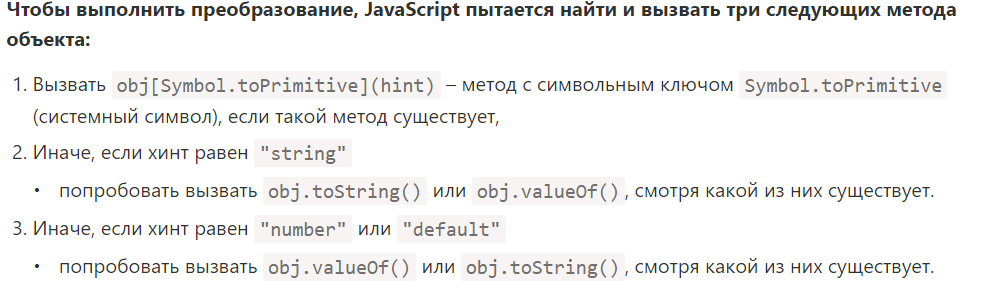
1. ToBoolen (!!, Boolean(), ?:, if, for, while, do/while, &&, ||)
2. ToString (String(), object[value], ${value}, alert)
3. ToNumber (+, Number(), ==, !=, <=, >=, <, >)
   1. True/false to 1/0
   2. Строку в число
   3. Для объектов вызовет ToPrimitive
4. ToPrimitive (если вернули объект – ошибка)
   1. valueOf
   2. toString

Любой объект всегда true

Все остальное тоже true кроме 7 (‘’, false, null, undefined, -0, 0, 0n, NaN)

**Если хотя бы один операнд является строкой, то второй будет также преобразован в строку.**







Область видимости (это способ, где искать переменную)

1. блочная видимость
2. функциональная область видимости

поднятие переменных:

value // undefined

var value = 42

value //42

foo // ref error

var value // игнорирование

замыкание – это функция + ее область видимости

контекст или this – способ связать функцию с объектом

1. по-умолчанию globalWindow or undefined (use strict)
2. неявное
3. явное (call, apply, bind)
4. связывание new (создается новый объект, выставляется prototype, новый объект === this)

установка [[Prototype]]

const prototype = { question: 42 }

const object = {}

object.\_\_proto\_\_ = prototype

или

Object.setPrototypeOf(object, prototype)

Или

Object.create(object, prototype)

object.questin // 42

const promise = new Promise((resolve, reject) => undefined.toString())

promise.catch(console.error).then(() => log(is Ok))

catch также как then возвращает promise

свойства объекта

[[Class]]

[[Prototype]] – выставляется в момент вызова new

[[Extensible]]

\_\_proto\_\_ - нестандартное свойство для доступа к [[prototype]] 45мин

Защищенный конструктор – 48мин

<https://www.youtube.com/watch?v=0vs6WkNyzec&t=778s>

<https://www.youtube.com/watch?v=b55hiUlhAzI&t=1773s>

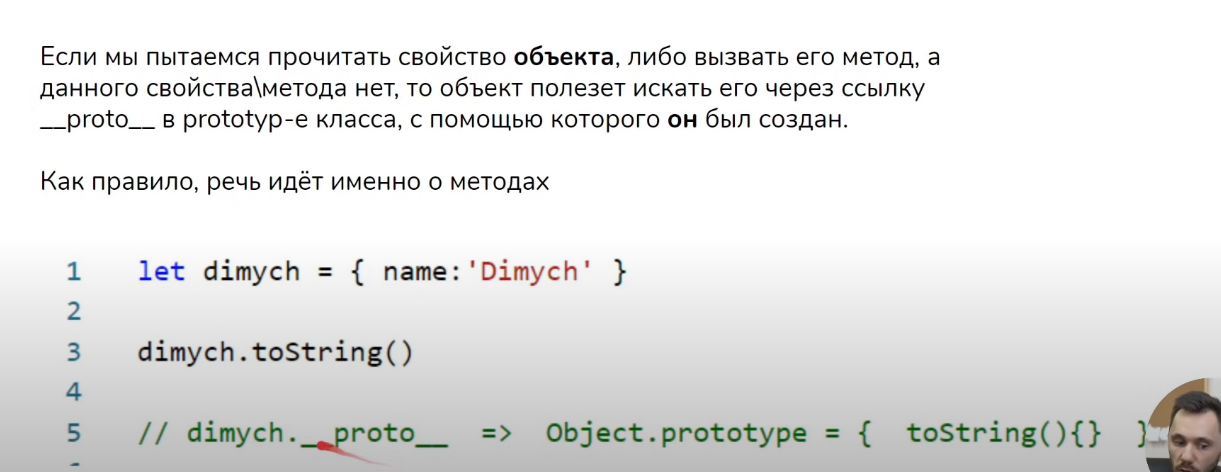
\_\_proto\_\_ и prototype – это свойство объекта

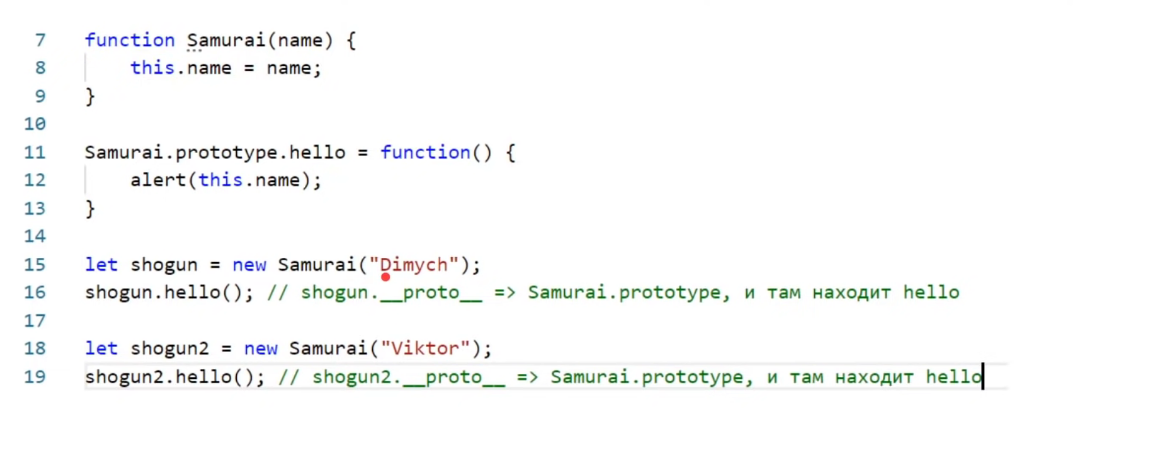
Любое св-во создано с помощью класса (упаковка) и у любого объекта есть \_\_proto\_\_

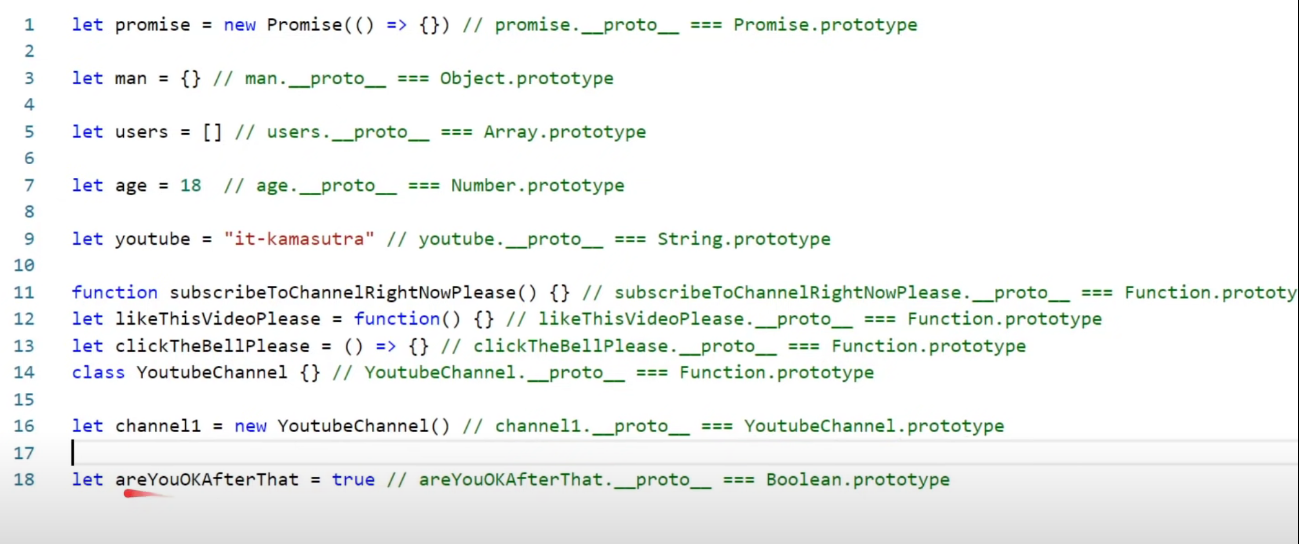
У любого объекта есть \_\_proto\_\_ (**proto** ссылается на **prototype** класса с помощью которого этот объект был создан)

Prototype – есть только у class или function () {} (не () => {}) это независимый объект и он не равен другому

let foo = new Foo(‘foo’) – у foo нет foo.prototype тк это объект. Это объект, не класс









События:

DOMContentLoaded - происходит, когда весь HTML был полностью загружен и пройден парсером, не дожидаясь окончания загрузки таблиц стилей, изображений и фреймов. Значительно отличающееся от него событие load используется для отслеживания только полностью загруженной страницы.

Ждет загрузки и исполнение (<script defer src="…"> и <script type="module">).

Не ждет окончания загрузки других вещей, таких как изображения, стилей, субфреймы и асинхронные скрипты.