**Правила JavaScript this**

- this всегда ссылается на объект

- this ссылается на объект, вызывающий функцию

- в глобальном контексте this содержит ссылку на объект окна (window), либо содержит undefined в строгом режиме (strict mode)

this - это ключевое слово, которое ссылается на тот или иной объект в зависимости от того, где оно записано или как была вызвана функция, где this присутствует.

глобальном контексте, за пределами функций this равен объекту window в независимости от того, какой режим разработки установлен строгий 'use strict' или нестрогий.

**this в функциях**

Если this записывать в отдельной функции, то его значение не всегда равно window. В нестрогом режиме всё без сюрпризов – window, в строгом –undefined

Ключевое слово this всегда динамично и указывает на тот объект в контексте которого было вызвано. В примерах выше контекст глобальный оттуда мы и видим window. Кстати, если this записать в функцию, которая лежит внутри другой функции, результат будет таким же.

**Методы объекта и this**

**Стрелочные функции**

Стрелочная функция не имеют собственного контекста и если в таком случае прописать this то ссылаться ключевое слово будет на объект ближайший по иерархии. Такая особенность удобна, когда мы хотим передать в функцию внешний контекст.

**Функция-конструктор**

Функция-конструктор позволяет создавать новые объекты по шаблону на основе других данных. При вызове функции конструктора this принимает значение вновь созданного объекта, который наполняется вследствие выполнения тела функции.

function Student(name, course) {

this.name = name

this.course = course

this.school = 'LearnJS'

}

let student1 = new Student('Вера', 'JavaScript Basic')

console.log(student1)

**Методы call() и apply()**

call() и apply() позволяют явно настроить контекст выполнения функции, результатом работы методов будет выполнение этой функции. Методы идентичны между собой, единственное их различие, это формат в котором записываются аргументы, для call() через запятую, для appy() в массиве.

let student1 = {name: 'Степан'}

let student2 = {name: 'Юлия'}

let sayName = function(course) {

console.log(`${this.name} изучает ${course}`)

}

sayName.call(student1, 'JS') // Степан

sayName.apply(student2, ['HTML + CSS']) // Юлия

В первом случае мы определили контекст выполнения функции sayName относительно объекта student1, во втором относительно student2 и получили в console Степан и Юлия соответственно.

**Метод bind()**

bind() – это встроенный метод, который также даёт возможность зафиксировать контекст выполнения, чтобы заранее быть уверенным, какое значение будет у this. В отличие от call() и apply() метод bind() возвращает новую функцию, а не вызывает изначальную.

let student1 = {name: 'Степан'}

let student2 = {name: 'Юлия'}

let sayName = function() {

console.log(this.name)

}

sayName.bind(student1)() // Степан

sayName.bind(student2)() // Юлия

**Итого**

1. Ключевое слово this позволяет работать с контекстом, получая доступ до объектов, а также имеет в арсенале методы, для того, чтобы менять этот контекст.

2. this не является фиксированным и определяется во время выполнения кода. Таким образом, пока функция не вызвана this не имеет значения.

3. Стрелочные функции не имеют собственного контекста и связываются с ближайшим по иерархии.

4. bind(), call() и apply() - методы позволяющие управлять контекстом.

5. При вызове функции-конструктора this принимает значение вновь созданного объекта.