Модуль - JavaScript

11/15 занятие

Объекты в JS

Объект это вид собственного(уникального) типа данных, предназначенный для хранения разнородной информации, как единое целое

Поля(свойства, параметры) - Статические данные, например числа, строки или другие объекты(вложенные)

Методы - функции связанные с этим объектом и имеющие доступ к его контексту

Значения объекта передаются по ссылке, а не по значению, как у примитивных типов

Копирование объекта

Object.assign(**{}**, **post**)

Где **первый** аргумент, это *целевой объект* - куда копируем, а **второй** - откуда

Возвращается *целевой объект*

Spread оператор (...) - расширяет(дополняет) новый объект, свойствами другого объекта, перенося их по значению

const obj = { a : 2 }

{...obj}

Задача 1

Сделать глубокую(полную) копию объекта

```
const post = {
        id: 1,
        text: "My first post",
        author: {
                id: 2,
                name: "John Doe"
        },
        comments: [
                {
                        id: 3,
                        text: "And first comment"
                 },
                {
                        id: 5,
                        text: "Second comment"
                }
         ]
}
```

< JS >

this - контекст объекта

this внутри методов ссылается на сам объект

```
const carObject = {
        color: 'red',
        complect: {
                airbag: true,
                transmission: 'automatic'
        },
        getCarTransmission: function () {
                return this.complect.transmission === 'automatic' ? 'автомат' : 'ручная'
        }
}
```

< JS >

this в стрелочных функциях

this внутри методов ссылается на сам объект

```
const carObject = {
        color: 'red',
        complect: {
                airbag: true,
                transmission: 'automatic'
        },
        getCarTransmission: () => {
                return this.complect.transmission  // Ошибка this ссылается на window
}
```

< JS >

Object - конструктор объектов

new Object() - создает новый экземпляр по “схеме” родительского Object

Object содержит полезные методы и каждый экземпляр наследует их

В главный прототип можно добавить свои методы, записав их

```
Object.prototype.myMethod = function () {
    console.log(this.id)
}
```

Стили программирования

Какие бывают стили:

процедурный - код последовательный, функция выполняет много задач

функциональный - код делится на атомарные функции (1ф = 1 действие)

объектно-ориентированный - данные скрываются в одном большом классе (объекте) и вычисления происходят внутри него, наружу смотрят методы для записи и получения данных(геттеры и сеттеры)

ООП

Инкапсуляция - сокрытие всей логики и данных внутри объекта или класса

Наследование - передача данных внутри Класса или объекта к методам

Полиморфизм - возможность использовать родительские методы расширенных объектов, не описывая их

ООП на основе прототипов

Классы в JavaScript были добавлены в ES6 и представляют визуальное оформление конструктора, основанного под капотом на работе с прототипами

```
class Car {
	constructor (f) {
		this.fuel = f
		console.log('Машина создана. Бак заправлен на ' + this.fuel)
	}

	reFuel (q) {
		let diff  = (this.fuel + q) - 100
		this.fuel = (this.fuel + q) <= 100 ? (this.fuel + q) : 100
		return (this.fuel + q) > 100 ? diff : 0
	}
}

let renoLogan = new Car(71)
```

Задача 2

- Дополнить класс **Car** методом **go** который будет имитировать движение машины, принимать аргумент distance(расстояние, которое проехал автомобиль) и сжигать топливо в пропорции 13% на 100км

- Добавить в конструктор расход топлива(в %) на 100км, который будет передаваться при создании нового экземпляра

- Добавить метод(геттер), показывающий текущий запас топлива

Вызывается:

```
let mercedesBenz  = new Car(71, 15)
mercedesBenz.go(199)
mercedesBenz.getFuelLevel()
```

Расширение классов

```
class Truck extends Car {
	constructor (f, load ) {
		super(f)
		this.load = load || 0
	}

	toLoad (w) {
		let diff  = (this.load + w) - 4000
		this.load = (this.load + w) <= 4000 ? (this.load + w) : 4000
		return (this.load + w) > 4000 ? diff : 0
	}
}

let kamaz = new Truck (71, 3500)
```

перерыв

Конец занятия