Введение в Javascript. Введение в ООП

Frontend. Урок 21

Вопросы на повторение

1. Что такое итерация?
2. Какие типы циклов мы прошли на прошлом уроке?
3. Что такое метод в программировании?

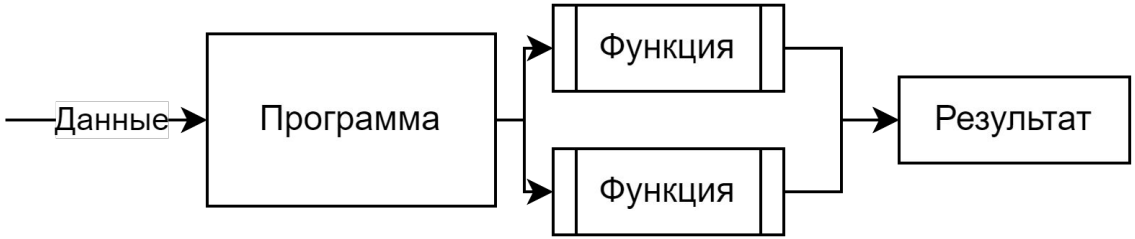
Постановка проблемы

В современных языках программирования наш код можно сделать более понятным для человека, создавать переменные такого типа, чтобы по её “характеристикам” было понятно, для чего она нужна и где мы её можем переиспользовать. Также иногда нам могут пригодиться кастомные типы переменных. Об этом мы поговорим в рамках занятий.

Введение в ООП

До сегодняшнего дня мы использовали процедурный подход для создания наших программ.

- Программа получает какие-либо данные
- В программе выполняется ряд определенных процедур(функций)
- На выходе мы получаем какой-либо результат



Функции

```
const a = 3 //стороны треугольника
const b = 4
const c = 2
function findTriangleArea(a, b, c){ //объявляем функцию с
тремя параметрами a, b, c
    const p = (a + b + c) / 2.0;
    const S = Math.sqrt(p * (p - a) * (p - b) * (p - c));
    return S.toFixed(2) //возвращаем результат выполнения
фунции с двумя цифрами после запятой
}
console.log(findTriangleArea(a, b, c))//выводим в консоль
результат выполнения функции
```

Объекты

```
let triangle = { // создаем объект треугольника
     a: 3, // свойства объекта - стороны треугольника
     b: 4,
     c: 2,
     findArea: function(){ // функция внутри объекта
называется методом. Данный метод считает площадь
      const p = (this.a + this.b + this.c) / 2.0 // this -
это обращение к текущему объекту
      //т.е. this.a - свойство "a" ЭТОГО объекта
      const S = Math.sqrt(p * (p - this.a) * (p - this.b) *
(p - this.c))
      return S.toFixed(2)
    }
}
console.log(triangle.findArea())
```

Классы

Когда нам необходимо создать много одинаковых объектов по одному “шаблону” для этого мы можем использовать класс.

```
class Triangle{ //объявляем новый класс Triangle - имена
классов с ЗАГЛАВНОЙ буквы
     constructor(a,b,c){ //метод конструктор - метод, который
принимает данные и создаем объект по эти данных
      this.a = a //this.a = a - у объекта, который мы создаем
в поле "a" задаем параметр "a" конструктора
      this.b = b
      this.c = c
    }
  findArea(){ //метод объекта класса для нахождения площади
    const p = (this.a + this.b + this.c) / 2.0
    const S = Math.sqrt(p * (p - this.a) * (p - this.b) * (p
- this.c))
    return S.toFixed(2)
  }
}
```

Создание объекта класса

```
let triangleA = new Triangle(2,4,3) //создаем объект класса Triangle
// new - говорит нам о том, что мы создаем НОВЫЙ объект класса
//дальше Triangle(2,4,3) - вызываем метод constructor - дефолтный метод для создания и передаем стороны треугольника
let triangleB = new Triangle(4,8,6) //еще один новый треугольник
let triangleC = new Triangle(12,12,12) //еще один новый треугольник
console.log(triangleA.findArea())
console.log(triangleB.findArea())
console.log(triangleC.findArea())
```

Теперь мы можем создавать сколько угодно разных треугольников и у всех них будут одинаковые методы для расчетов.

Инкапсуляция

```
    class User{
        name //мы можем объявить параметры сразу
        password
        id
        role

        constructor(name, password, id, role) {
            this.name = name
            this.password = password
            this.id = id
            this.role = role
        }
    }
    //Создали пользователя
    //Date.now() возвращает текущее время в
миллисекундах, с помощью этого метода
    //очень удобно генерировать длинные рандомные числа
    let user1 = new User("Alex", "qwerty123", Date.now(),
"admin")
    console.log(user1)
    //А теперь мы можем из любой части кода изменить его
пароль
    user1.password = "newpassword"
    console.log(user1)
```

Инкапсуляция

Нужно защитить пароль от такого простого изменения!
Для решения этой проблемы мы используем инкапсуляцию
Инкапсуляция - это упаковка данных и функций в один компонент (например, класс) и последующий контроль доступа к этому компоненту, создавая тем самым "чёрный ящик" или **КАПСУЛУ** из объекта.
Давайте скроем поля пользователя, сделать это можно следующим образом:

Инкапсуляция

```
    class User {
        #name; //хештег делает поле приватным - доступным
только внутри класса
        #password;
        #id;
        #role;

        constructor(name,password,id, role) {
            this.#name = name
            this.#password = password
            this.#id = id
            this.#role = role
        }

    }
    let user1 = new User("Alex", "qwerty123", Date.now(),
"admin")
    console.log(user1)
```

Геттеры и сеттеры

Теперь поля скрыты, но, например, нам нужно считать имя и id пользователя. Для приватных полей сущест
специальные методы:

- геттеры - получают значение поля
- сеттеры - меняют значения поля

```
class User {
    #name;
    #password;
    #id;
    #role;
    get name(){ //get - метка геттера name - название
геттера, тело геттера должно возвращать
        //соответствующее значение
        return this.#name
    }
    get id(){
        return this.#id
    }
    constructor(name,password,id, role) {
        this.#name = name
        this.#password = password
        this.#id = id
        this.#role = role
    }

}
let user1 = new User("Alex", "qwerty123", Date.now(),
"admin")
console.log(user1.name, user1.id) //обращаемся к
геттерам класса
```

Сеттеры

С помощью сеттера можно изменять разрешенные поля. В данном случае это имя пользователя

```
class User {
    #name;
    #password;
    #id;
    #role;
    get name(){
        return this.#name
    }
    get id(){
        return this.#id
    }

    set name(newName){ //set - метка сеттера, name - его название, параметр newName - обновленное имя
        this.#name = newName
        //сеттер не должен ничего возвращать
    }
    constructor(name,password,id, role) {
        this.#name = name
        this.#password = password
        this.#id = id
        this.#role = role
    }

}
let user1 = new User("Alex", "qwerty123", Date.now(), "admin")
console.log(user1.name, user1.id) //обращаемся к геттерам класса
user1.name = "Max"
console.log(user1.name, user1.id) //обращаемся к геттерам класса
```

Вопросы на закрепление

1) Что такое класс и объект?
2) Для чего нужна инкапсуляция?

Практика А

Задача: Прямоугольник

Создайте класс прямоугольника с конструктором и функцией подсчета площади.

Практика B

Задача: Работа с массивом

- Доработайте класс пользователя так, чтобы только у пользователя с ролью “admin” была возможность изменить пароль
  - Для этого создайте специальный метод

Практика С

Задача: реализация смены паролья

- Добавить метод, который позволит любому пользователю поменять пароль по алгоритму:
  - Введите текущий пароль
    - Если пароль верный идем далее
    - Если нет, то выводим сообщение об ошибке
  - Введите новый пароль
    - Новый пароль не должен совпадать со старым -> сообщение об ошибке
    - Повторите новый пароль
  - Пароли должны совападать -> сообщение об ошибке