OOP (class, inheritance)



НАШИ ПРАВИЛА

Включенная камера

Вопросы по поднятой руке

Не перебиваем друг друга

Все вопросы, не связанные с тематикой курса (орг-вопросы и т. д.), должны быть направлены куратору

Подготовьте свое рабочее окружение для возможной демонстрации экрана (закройте лишние соцсети и прочие приложения)



Изучить class, inheritance

ПЛАН ЗАНЯТИЯ

- class
- methods
- inheritance
- private fields*

Что такое класс (как вы его знаете по джаве)?

Класс

Классы в JavaScript представляют собой шаблоны для создания объектов. Они предоставляют удобный способ определения объектов с общими свойствами и методами.

Класс

С выходом ECMAScript 6 появился целый набор ключевых слов, реализующих классы. Они могут показаться знакомыми людям, изучавшим языки, основанные на классах, но есть существенные отличия. **JavaScript был и остаётся прототипно-ориентированным языком**.

Новые ключевые слова: "class", "constructor", "static", "extends" и "super".



```
// Объявили класс Wizard
class Wizard {
constructor(name, house) {
   this.name = name;
   this.house = house;
 introduce() {
   console.log(`I am ${this.name} from ${this.house} house.`);
// Создание экземпляра класса
const harry = new Wizard('Harry Potter', 'Gryffindor');
harry.introduce(); // "I am Harry Potter from Gryffindor house."
```

EXTENDS

Наследование позволяет создавать новые классы, используя свойства и методы существующего класса.

Мы создали класс DarkWizard на основе класс Wizard

```
class DarkWizard extends Wizard {
 constructor(name, house, darkPower) {
   super(name, house);
   this.darkPower = darkPower;
useDarkPower() {
   console.log(`${this.name} uses dark power: ${this.darkPower}`);
const voldemort = new DarkWizard('Lord Voldemort', 'Slytherin',
'Avada Kedavra');
voldemort.introduce(); // "I am Lord Voldemort from Slytherin
house."
voldemort.useDarkPower(); // "Lord Voldemort uses dark power:
Avada Kedavra"
```

Приватные поля

Приватные поля и методы могут быть созданы с использованием предлагаемого синтаксиса `#`.

```
class Wizard {
 #privateField;
 constructor(name, house) {
   this.name = name;
   this.house = house;
   this.#privateField = 'Secret';
 #privateMethod() {
   console.log('This is a private method.');
 revealSecret() {
   console.log(`My secret is ${this.#privateField}.`);
   this.#privateMethod();
```

GETTER, SETTER

Геттеры используются для получения значения свойства, а сеттеры - для установки его значения.

```
class Wizard {
 constructor(name, house) {
   this. #name = name; // Приватное поле
   this.house = house;
 get name() {
   return this. #name;
 set name(newName) {
   this.#name = newName;
const ron = new Wizard('Ron Weasley', 'Gryffindor');
console.log(ron.name); // "Ron Weasley"
ron.name = 'Ronald Weasley';
console.log(ron.name); // "Ronald Weasley"
```

THIS

Ключевое слово **`this`** в JavaScript используется для **обращения к текущему объекту**. Контекст `this` зависит от того, как вызывается функция.

```
const person = {
  name: 'John',
  introduce: function() {
    console.log(`Hello, my name is ${this.name}.`);
  }
};

person.introduce(); // "Hello, my name is John."
```

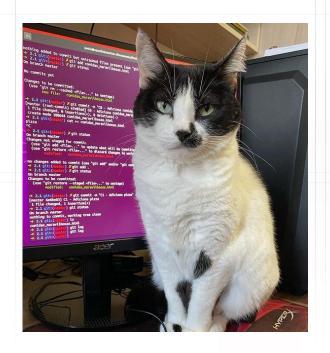
поиграем;)

Что такое класс? Из чего он состоит?

Что такое экземпляр класса?

Какие поля называются приватными?

ДОМАШНЕЕ ЗАДАНИЕ



СТАВИМ +, ЕСЛИ ВАМ ПОНЯТНО ДОМАШНЕЕ ЗАДАНИЕ



Ваша новая IT-профессия – Ваш новый уровень жизни

Программирование с нуля в немецкой школе AIT TR GmbH

