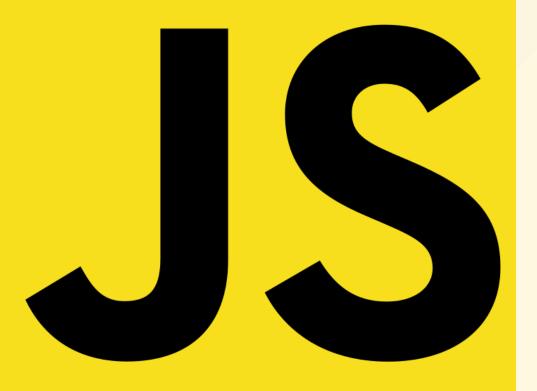
Классы



Домашнее задание типичные ошибки

Функция сортировки

Класс - это абстрактное представление объектов.

Конкретное воплощение класса - объект

Для хранения сосьояния объекта используются свойства. Они могут просто быть объявлены или с начальными значениями.

```
class Course {
  topic = 'JS';
  during = 4.5;
  print() {
    console.log(`This is ${this.topic} (${this.during} month.)`);
  }
}
```

Конструктор

Для первичной инициализации свойств объекта класса используют конструктор

```
class Course {
  constructor(courseTopic, courseDuring) {
    this.topic = courseTopic;
    this.during = courseDuring;
  }
}
const myCourse = new Course('TS', 6);
```

Приватные поля и методы

Для того чтобы запретить доступ к некторым свойствам и методам вне класса, был введен механизм приватных полей.

```
class Course {
  constructor(courseTopic, courseAmount) {
    this.#topic = this.#checkTopic(courseTopic);
    this.amountPeople = courseAmount;
  #checkTopic(value) {
   if (['JS', 'TS', 'C#'].include(value)) {
      return value;
const myCourse = new Course('TS', 15);
```

Статические поля и методы

```
class Course {
   static maxPeople = 15;
   constructor(courseAmount) {
     this.amountPeople = courseAmount;
   }
   static print() {
     console.log(`Max amount - ${this.maxPeople}`);
   }
}
const myCourse = new Course(12);
```

!!! В статических методах this используется только к статическим полям

При необходимости в статическом методе использования значений нестатических свойств - передается ссылка на этот объект

```
class Course {
  constructor(courseTopic, courseDuring) {
    this.topic = courseTopic;
    this.during = courseDuring;
  satic print(obj){
      console.log(`Info: ${obj.topic}, ${obj.during}`)
const myCourse = new Course('TS', 6);
Course.print(myCourse);
```

Доступ к свойствам и контроль присваиваемых им значений

```
class Course {
  #during = 2;
  constructor(courseDuring) {
    this.during = courseDuring;
  set during(value) {
   if (value > 0 && value <= 6) this.#during = value;
  get during() {
    return this.#during;
const myCourse = new Course(-8);
console.log(myCourse.during);
```

Наследование (наследники получают весь функционал родительских классов)

```
class Course {
  constructor(courseTopic, courseDuring) {
    this.topic = courseTopic;
    this.during = courseDuring;
  print() {
    console.log(`${this.topic} ${this.during}`);
class Group extends Course {
  constructor(topic, during, groupAmout) {
    super(topic, during);
    this.amount = groupAmout;
```

Beetroot. Academy. JavaScript. Тема 2. Основы JS