PHASE 1

WEEK 2

DAY 2

ELBRUS/BOOTCAMP

План

1. Классы (classes)
2. Наследование классов (classes extends)
3. Статические методы и свойства (static method)
4. Геттеры и сеттеры (getters, setters)
5. Приватные поля и методы (private prop, methods)

Классы

синтаксический сахар над прототипным наследованием

Объявление класса

```
// объявление класса
class Student {
   constructor(name, age) {
     this.name = name;
     this.age = age;
   }
}

// создание экземпляра (instance) от класса
const anton = new Student('Антон', 24);
```

Конструктор

Функция для инициализации состояния объектов, созданных на основе класса.

```
class Student {
    constructor(name, age) {
      this.name = name;
      this.age = age;
    }
}
```

Методы класса

Задаются в теле класса.

```
class Student {
   constructor(name, age) { /* ... */ }

   sayHi() {
       console.log(`Привет, я ${this.name}`);
   }
 }

const anton = new Student('Антон', 24);
```

Наследование

Наследование

- Задаётся через ключевое слово `extends`
- В конструкторе дочернего класса нужно вызвать конструктор суперкласса

Вопрос:

Имеют ли экземпляры (`instance`) дочернего класса доступ к свойствам и методам, прописанным в суперклассе (родительский)?

Родительский класс (суперкласс)

```
// родительский класс (суперкласс)
class Point {
 constructor(x, y) { this.x = x; this.y = y; }

 getXY() {
   return (`x: ${this.x},  y: ${this.y}`);
 }
}

const a = new Point(6, 10); // { x: 6, y: 10 }
a.getXY(); // x: 6, y: 10
```

Класс-наследник (суб-, подкласс)

```
class Point3d extends Point {
 constructor(x, y, z) {
   super(x, y);  // Вызов конструктора суперкласса
   this.z = z;   // Продолжение инициализации объекта
 }

 getZ() { return (`z: ${this.z} `) }
}

const b = new Point3d(0, 0, 20); // { x: 0, y: 0, z: 20 }
b.getXY(); // метод из родительского класса
b.getZ(); // “родной” метод подкласса
```

Статические методы и свойства

Статические методы класса

```
class Point {
 static info() {
   return 'Этот класс возвращает координаты точки';
 }

 constructor(x, y) { /* ... */ }
 getXY() { /* ... */ }
}

Point.info(); // статический метод вызывается напрямую у класса
```

Статические свойства класса

```
class Elbrus {
 static phase = 1;
 static group = "Рыси 2021";
}
```

При использовании статических методов `constructor` не применяется.

Но при этом статические свойства и методы наследуются.

Геттеры, сеттеры

Геттеры и сеттеры, приватные поля

```
class Point {
 #x; // объявление приватного поля

 constructor(coordX) {
   this.#x = coordX;
 }

 // геттер
 get x() { return this.#x; }

 // сеттер
 set x(value) {
   if (value >= 0) {
     this.#x = value;
   }
 }
}
```

Геттеры и сеттеры: применение

```
const point = new Point(10);

// ошибка - к приватному полю #x нет доступа вне класса.
// запись и чтение невозможны
point.#x;

point.x;  // 10 ← возврат из функции-геттера

point.x = 35; // сработает функция-сеттер

point.x; // 35
```

Приватные поля и методы

Приватные методы: применение

```
class Test {
 // приватный метод
 #privateMethod() {
   return (this.number * this.number)
 }
 constructor(number) {
   this.number = number;
 }
 square() {
   return this.#privateMethod()
 }
}
```