- **Модуль:** Typescript. Классы
- **Урок:** Наследование классов в TS (extends)

TypeScript Handbook from Иван Петриченко
udemy

# НАСЛЕДОВАНИЕ КЛАССОВ В TS

Наследование – это когда мы можем создать цепочку классов, где есть связь родитель – потомок. Потомок будет содержать **все свойства и методы родителя**, и на усмотрение разработчика содержать что-то дополнительное или немного по другому реализует родительский функционал. Для наследования мы используем ключевое слово **extends**:

```
class Box {
    width: number;
    height: number = 500;
    // ...
```

```
class PresentBox extends Box {
    wrap: string;
    height: number = 600;
    // ...
```

Потомки могут конструироваться по другому, а значит и будут иметь свой конструктор. Не забывайте использовать **суперконструктор** перед использованием this:

```
class PresentBox extends Box {
    wrap: string;
    height: number = 600;

    constructor(wrap: string, width: number) {
        super(width);
        this.wrap = wrap;
    }
```

В наследуемом классе свойства и методы **должны совпадать** по типам с аналогами в родителе. В методах возвращаемое значение должно быть таким же, а новые аргументы обязательно с **модификатором опциональности:**

```
// Parent
async content(value: string) {
    const date = await new Date().toTimeString();
```

```
// Derived
async content(value: string, text?: string) {
    const date = await new Date().toTimeString();
```

Чтобы сказать, что метод был “перезаписан” и в потомке он уже имеет другой функционал, существует модификатор override:

```
// Derived
override async content(value: string, text?: string) {
    const date = await new Date().toTimeString();
```

Его суть в двух моментах:

- 👉 Четко сказать разработчику, что это перезаписанный метод родителя
- 👉 Если из родителя исчезает этот метод – в потомке будет **ошибка**. Так мы убедимся, что потомок не будет использовать собственный метод и всегда сможем это подправить