Вступление

Про компанию и кто я

```
"dependencies": {
  "@company/child-library": "git+https://github.com/compan
},
```

Виды API

- веселые
- скучные

Веселые API

- Ошибки со статусом 200.
- False одним из вариантов: `0`, `'0'`, `'false'`.
- Приходят левые данные, которые никто не использует.
- Формат сохраняемых данных отличается от формата полученных.
- много других подобных штук...

Скучные API

- API без признаков `веселых`.

Как добиться качества?

1. Поругаться с бекендерами.
2. Отгораживаться.

Отгораживаемся от api

- классы
- мапперы
- репозитории
- тесты апи

Классы > JS

- декларация структуры данных
- поддержка IDE

```
export class Client {
  constructor (data) {
    this.id = undefined
    this.name = ''
    this.owner = null
  }
}
```

Классы > TypeScript

```
export class Company {
  id: number
  name: string = ''

  constructor (data: Partial<Company> = {}) {
    Object.assign(this, data)
  }
}
```

Мапперы

- Суть - хелперы по преобразованию данных.
- Удобно тестировать через снепшоты.

```
export class CompanyMapper {
  static toClass (data) {
    return new Company({ id: data.id, name: data.company_n
  }
  static toObject (company) {
    return { id: company.id, company_name: company.name }
  }
}
```

Репозиторий

Суть - обертка над ресурсом.

```
export class CompanyRepository {
  async create (company) {
    const payload = CompanyMapper.toObject(company)
    await axios.post('companies', payload)
  }
}
```

Пример:

```
await CompanyRepository.create(company)
```

Тесты API

- Суть - тесты репозитория. Бонус - блекбокс тестирование фронта.
- Используются фабрики, полезные также для фронтовых компонентов.

Пример:

```
const fresh = CompanyFactory.getNew()
const created = await CompanyRepository.create(fresh)
const loaded = await CompanyRepository.getOne(created)
expect(created.name).toBe(loaded.name)
```