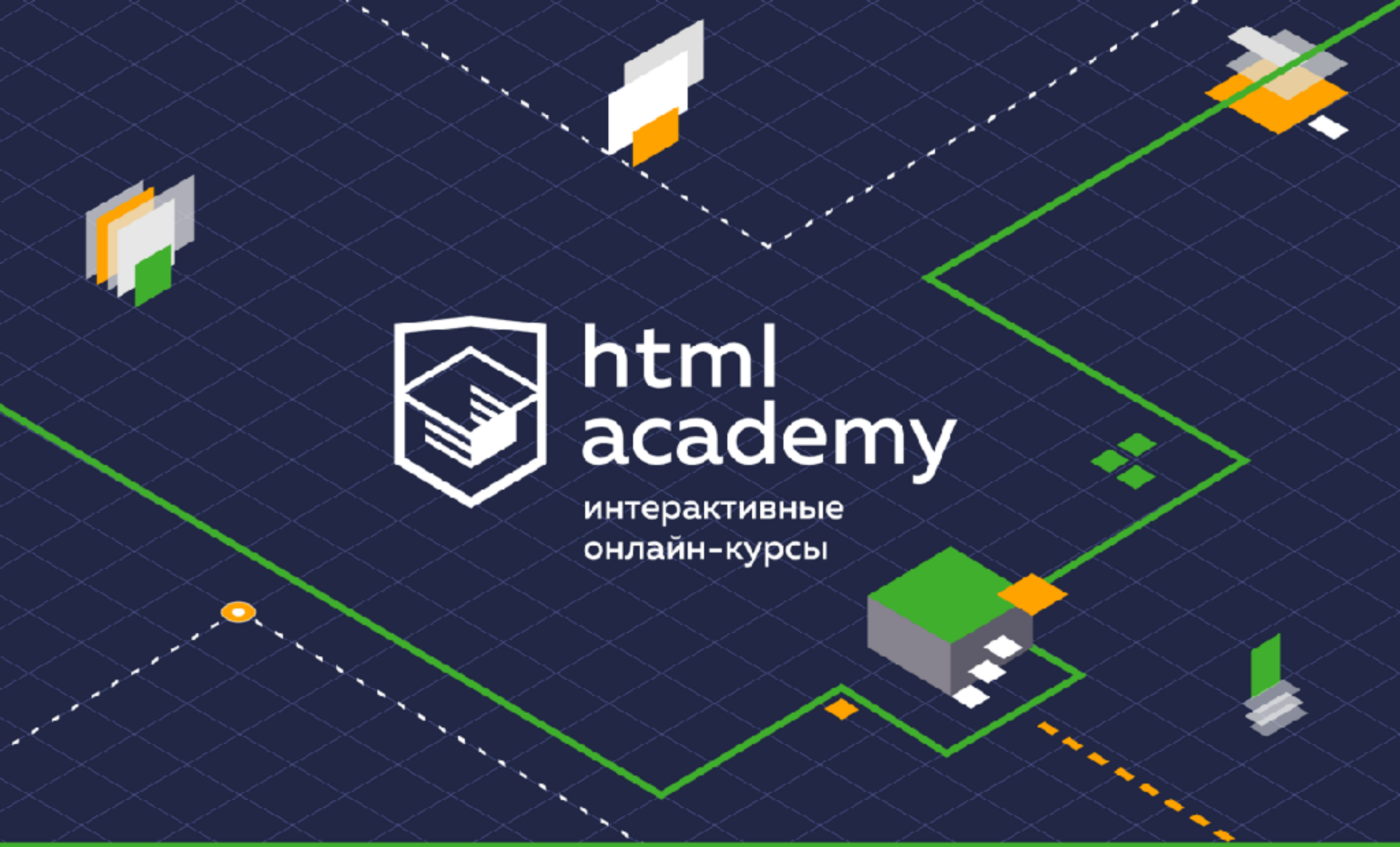



```
it("Тесты", () => assert(раздел === 0x04))
```

Разбор полетов

- данные не должны повторять шаблон, данные должны описывать сущности, которыми мы оперируем. Шаблон может измениться, а данные останутся

- все, что может быть притерировано, должно быть притерировано

```
const header = {
   logo: {
      src: 'logo.png',
      width: 100,
      height: 30
   }
};
```

Разбор полетов

- данные не должны повторять шаблон, данные должны описывать сущности, которыми мы оперируем. Шаблон может измениться, а данные останутся

- все, что может быть притерировано, должно быть притерировано

```
const header = {
  logo: {
    src: 'logo.png',
    width: 100,
    height: 30
  }
};
```

Разбор полетов

- данные не должны повторять шаблон, данные должны описывать сущности, которыми мы оперируем. Шаблон может измениться, а данные останутся

- все, что может быть притерировано, должно быть проитерировано

```
const questions = {
   one: {},
   two: {}
};
```

Разбор полетов

- данные не должны повторять шаблон, данные должны описывать сущности, которыми мы оперируем. Шаблон может измениться, а данные останутся

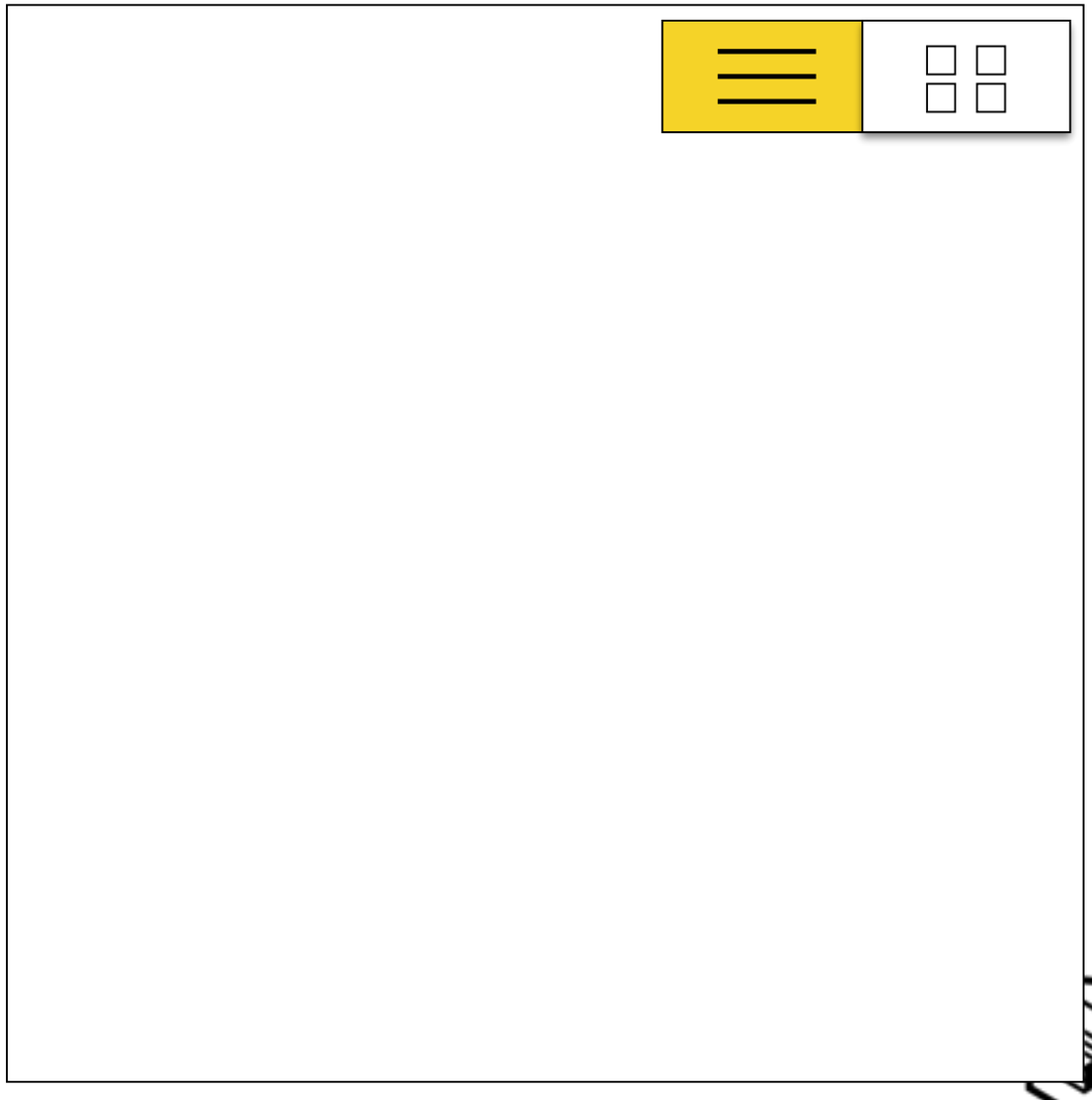
- все, что может быть притерировано, должно быть пройтерировано

```
const questions = {
  one: {},
  two: {}
};
```

Зачем отделять данные от шаблона

данные абстрактны и подходят под разные способы отображения: информация не изменится, если ее показывать по-разному

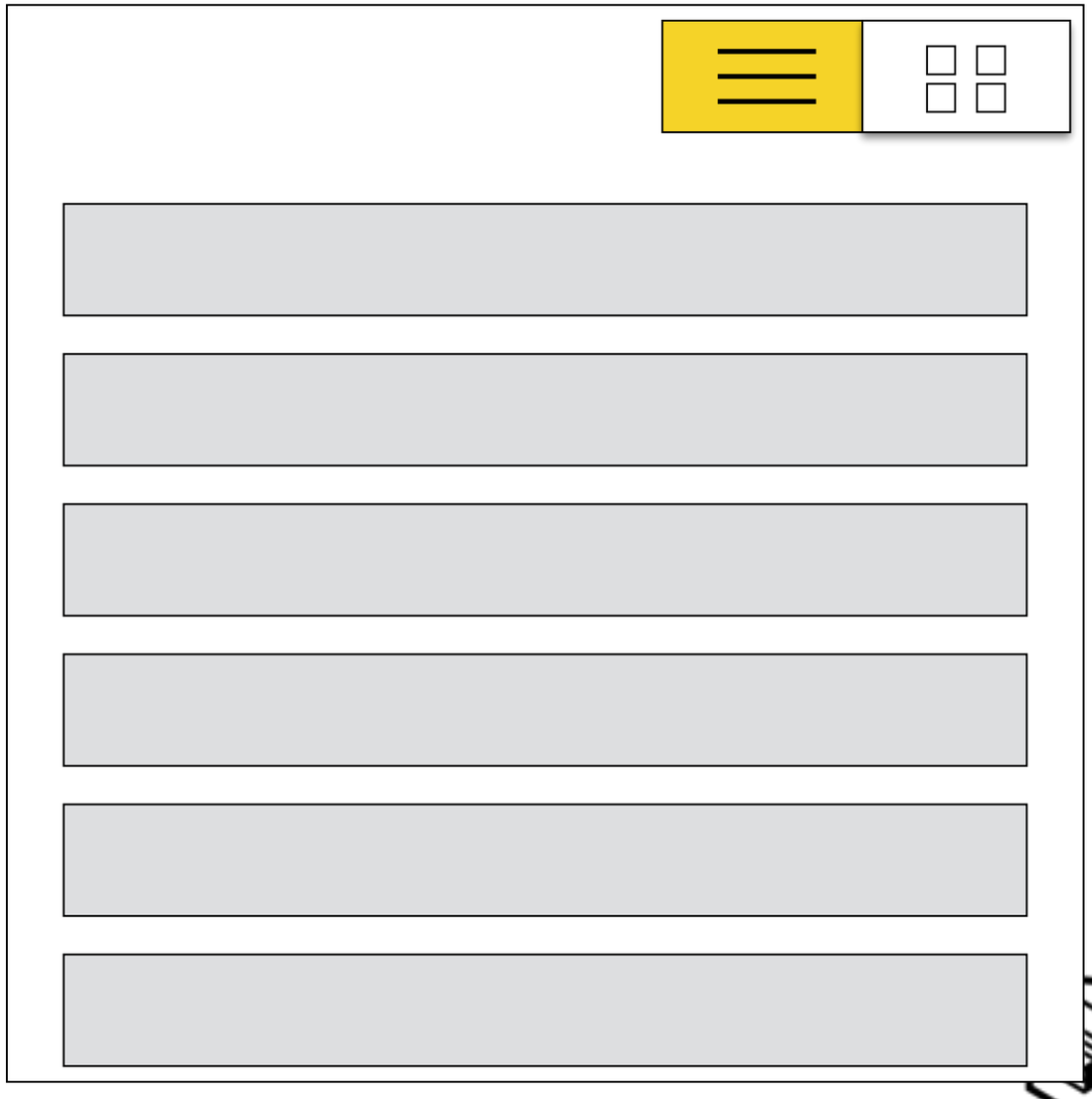
```
const productsList = [
  { name: 'Утюг' },
  { name: 'Чайник' },
  { name: 'Пылесос' },
  { name: 'Стиральная ма…
  { name: 'Кухонный комб…
  { name: 'И А-а-а-автом…
];
```



Зачем отделять данные от шаблона

данные абстрактны и подходят под разные способы отображения: информация не изменится, если ее показывать по-разному

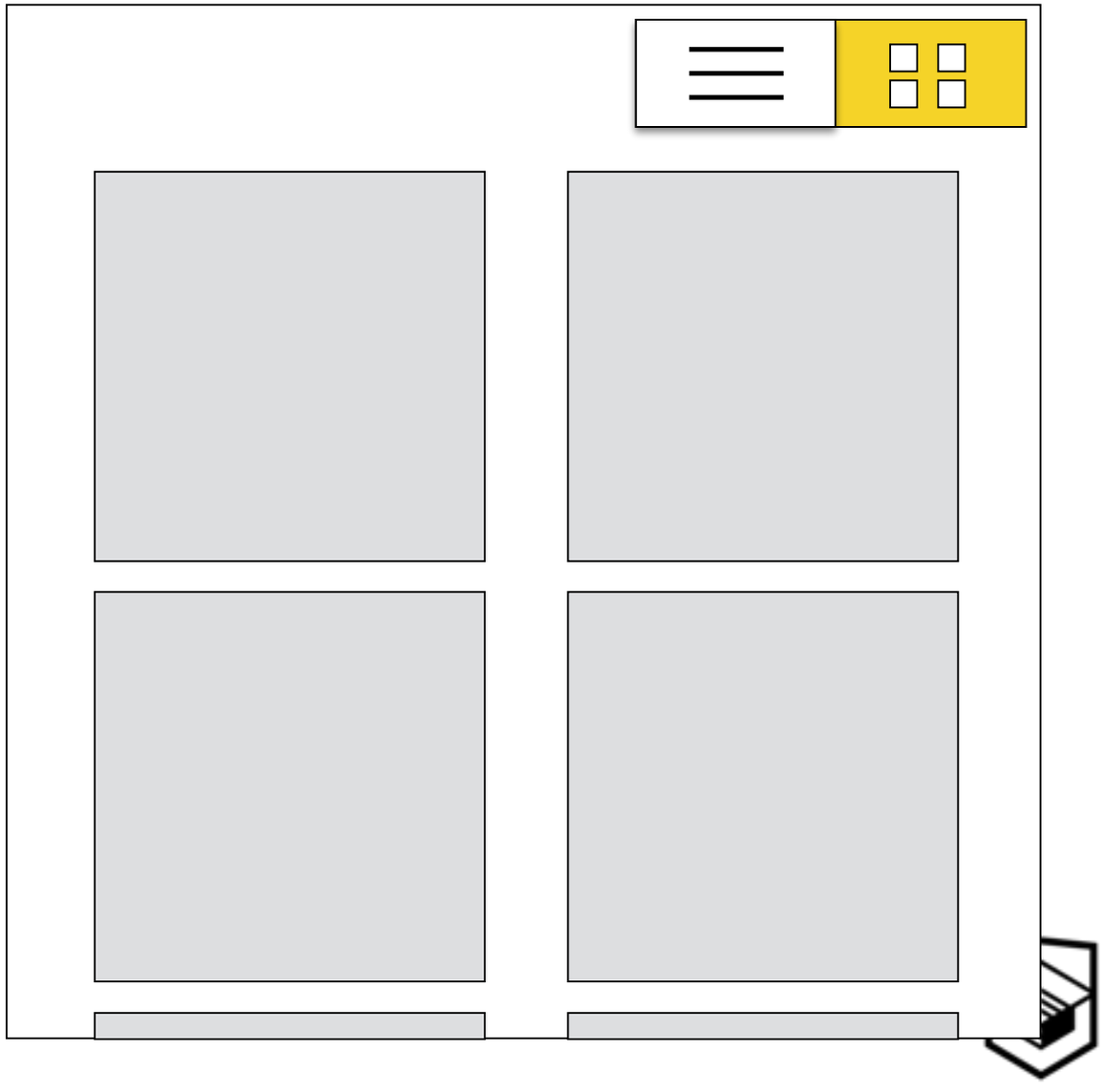
```
const productsList = [
  { name: 'Утюг' },
  { name: 'Чайник' },
  { name: 'Пылесос' },
  { name: 'Стиральная ма…
  { name: 'Кухонный комб…
  { name: 'И А-а-а-автом…
];
```



Зачем отделять данные от шаблона

данные абстрактны и подходят под разные способы отображения: информация не изменится, если ее показывать по-разному

```
const productsList = [
  { name: 'Утюг' },
  { name: 'Чайник' },
  { name: 'Пылесос' },
  { name: 'Стиральная ма…
  { name: 'Кухонный комб…
  { name: 'И А-а-а-автом…
];
```



✅ Задача на сегодня

проверить правильность работы алгоритмов

для работы с данными, написанных в прошлом разделе

Тестирование

Тестирование

написание программ, которые пытаются разломать другие программы 🤖🔫🤖

Тест –

это процедура, которая позволяет либо подтвердить, либо опровергнуть работоспособность кода

Википедия

https://ru.wikipedia.org/wiki/Разработка_через_тестирование

Тестирование

Тестирование

- **ручное**
  тестирование производится человеком

- **автоматическое**
  тестирование производится программами

- **функциональное тестирование**
  проверка, правильно ли работает программа

- **тестирование производительности**
  проверка, достаточно ли быстро работает программа

- **тестирование безопасности**
  проверка, безопасна ли программа

Тестирование

- **ручное**
  тестирование производится человеком

- **автоматическое**
  тестирование производится программами

- **функциональное тестирование**
  проверка, правильно ли работает программа

- **тестирование производительности**
  проверка, достаточно ли быстро работает программа

- **тестирование безопасности**
  проверка, безопасна ли программа

Тестирование

- **юнит-тестирование**
  тестирование отдельных модулей программы.
  Проверяется, правильно ли работает каждая из частей программы в отдельности

- **интеграционное тестирование**
  проверка, правильно ли взаимодействуют компоненты системы

- **системное тестирование**
  проверка, правильно ли работает вся система

Тестирование

- **юнит-тестирование**
  тестирование отдельных модулей программы.
  Проверяется, правильно ли работает каждая из частей программы в отдельности

- **интеграционное тестирование**
  проверка, правильно ли взаимодействуют компоненты системы

- **системное тестирование**
  проверка, правильно ли работает вся система

`*.test.js`

тесты создаются как отдельные JS-файлы, использующие основные файлы как зависимости и запускающие проверки кода

`*.test.js`

- **тесты в консоли**
  тесты запускаются в консоли с помощью Node.js командой npm test, как таск Галпа и т. д.

- **тесты в браузере**
  запускается браузер (браузеры), в котором выполняются файлы тестов

`*.test.js`

- **тесты в консоли**
  тесты запускаются в консоли с помощью Node.js командой npm test, как таск Галпа и т. д.

- **тесты в браузере**
  запускается браузер (браузеры), в котором выполняются файлы тестов

Тесты в JS

Тесты

1. что тестировать
2. как тестировать

mocha

(мокка, сорт кофе) тест-фреймворк — набор методов, который позволяет описывать тесты в коде. Отвечает на вопрос «что тестировать»: позволяет описывать наборы тестов и тестовые случаи

Фреймворк mocha

1. **Наборы тестов** *(test suites)*: `describe(string, Function)` группы проверок, связанные одной общей темой, например проверка отдельного набора функций модуля, выполняющего одну задачу

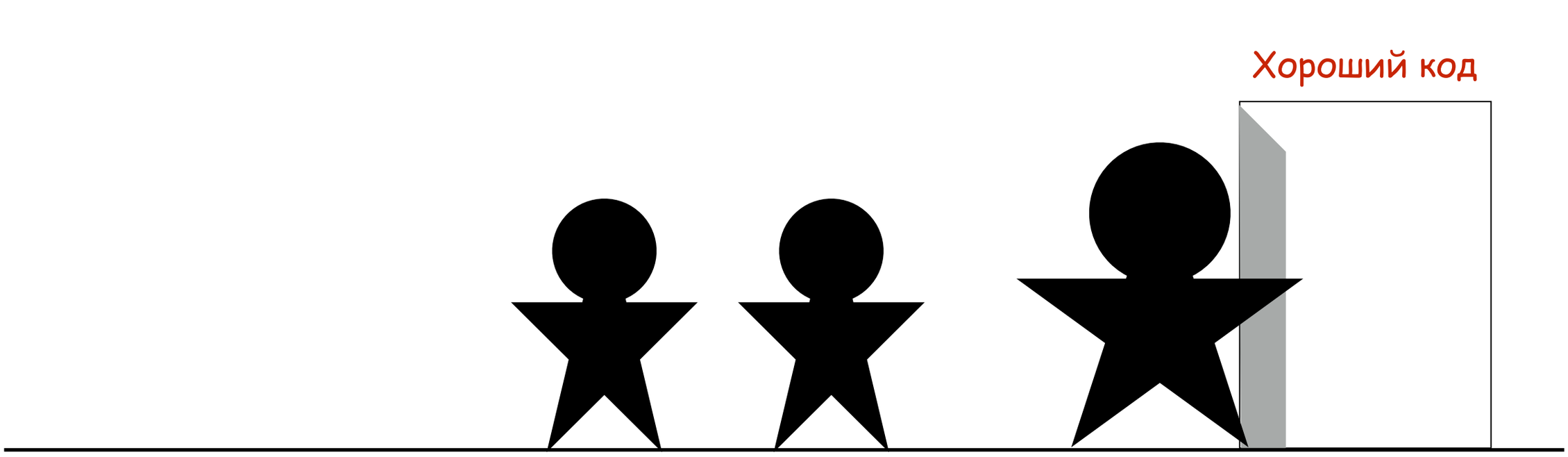
2. **Тестовые случаи** *(test cases)*: `it(string, Function)` конкретные проверки правильно ли работают определенные выражения в коде

assert

стиль проверки выражений — подразумевает под собой наличие эталона с которым будет сравнивать получившийся вариант

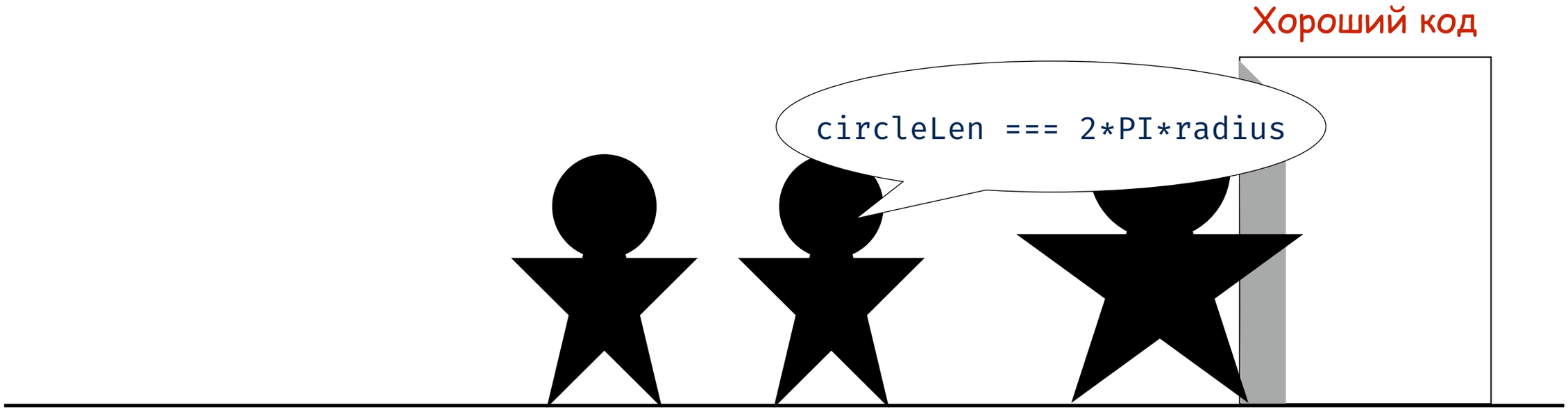
`assert`

принимает на вход значение, которое переводит в `boolean`. Если значение равно `true`, продолжает выполнение, если `false` — выдает ошибку *(останавливает JS)*



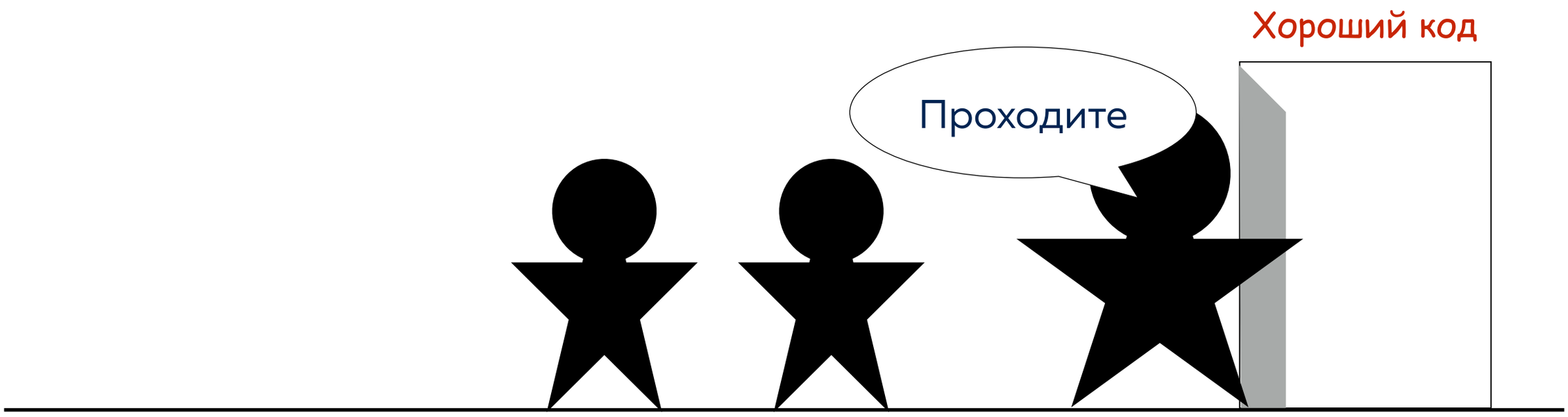
`assert`

принимает на вход значение, которое переводит в `boolean`. Если значение равно `true`, продолжает выполнение, если `false` — выдает ошибку *(останавливает JS)*



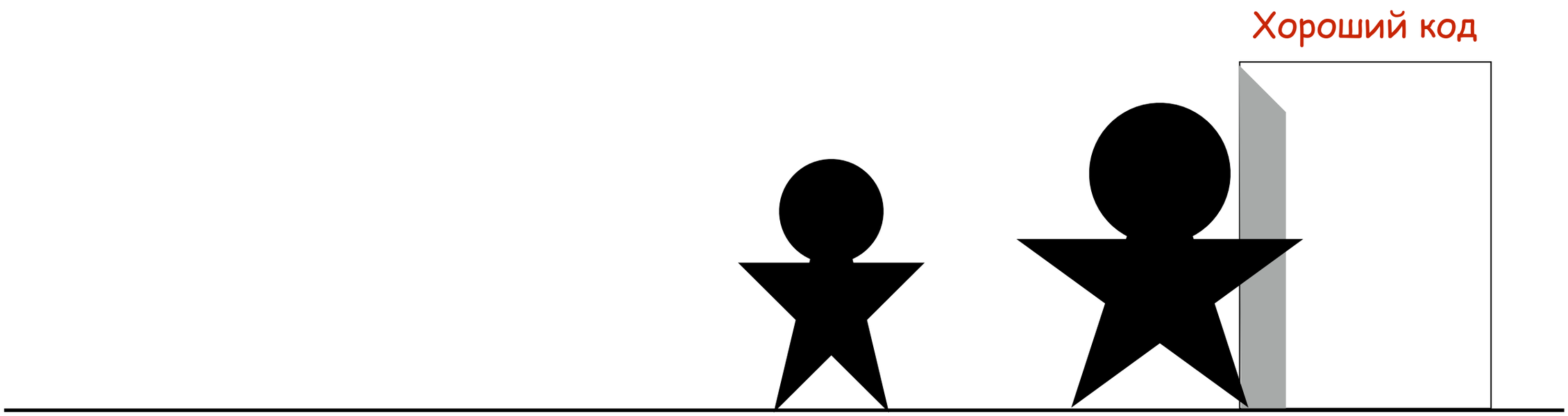
`assert`

принимает на вход значение, которое переводит в `boolean`. Если значение равно `true`, продолжает выполнение, если `false` — выдает ошибку *(останавливает JS)*



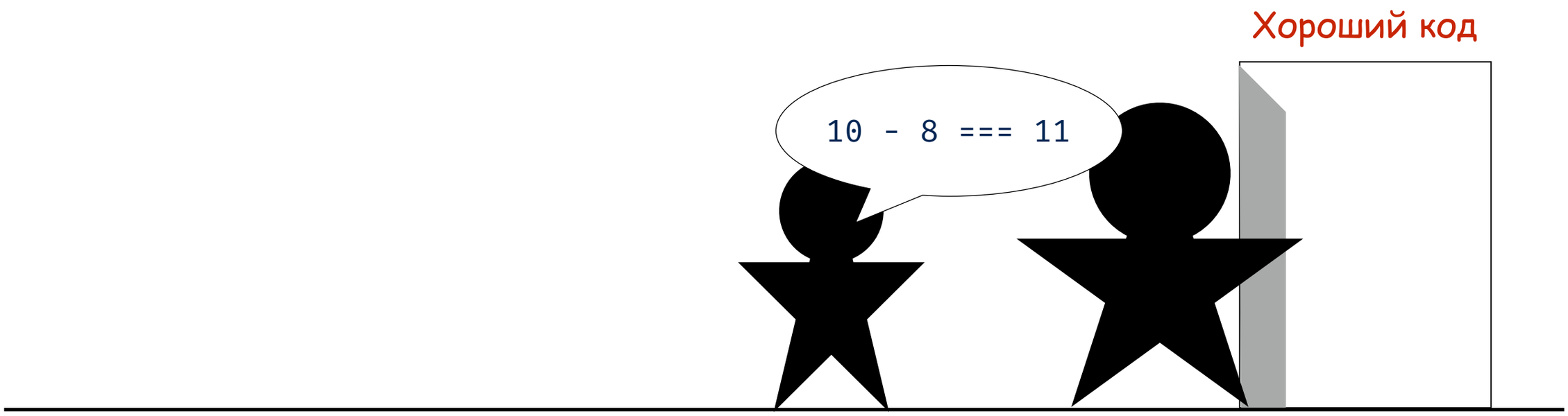
`assert`

принимает на вход значение, которое переводит в `boolean`. Если значение равно `true`, продолжает выполнение, если `false` — выдает ошибку *(останавливает JS)*



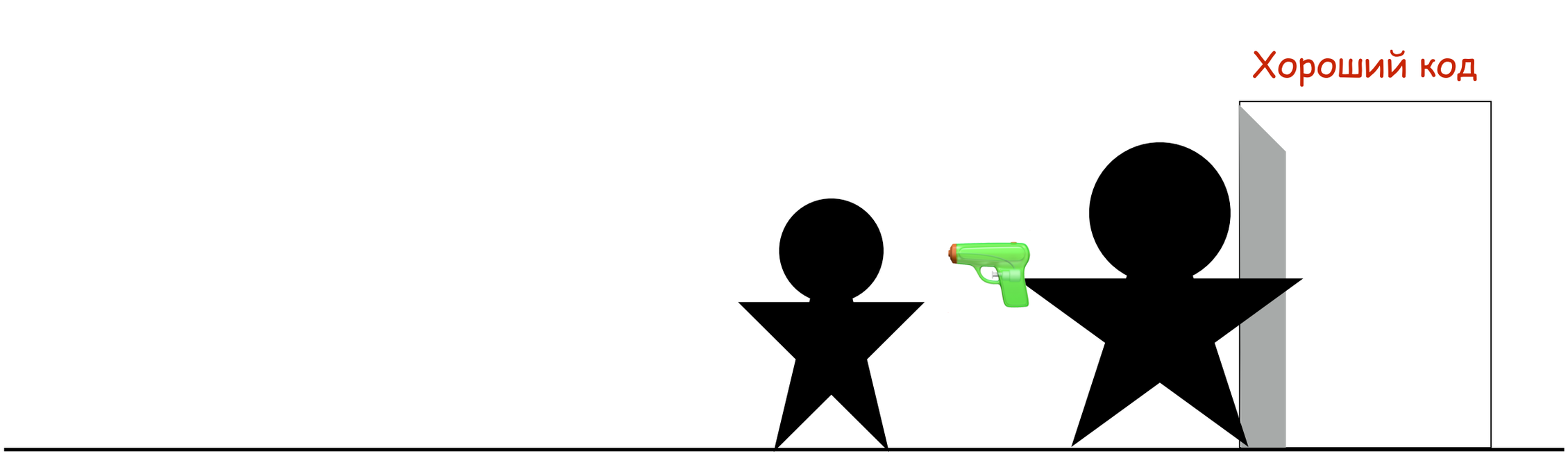
`assert`

принимает на вход значение, которое переводит в `boolean`. Если значение равно `true`, продолжает выполнение, если `false` — выдает ошибку *(останавливает JS)*



`assert`

принимает на вход значение, которое переводит в `boolean`. Если значение равно `true`, продолжает выполнение, если `false` — выдает ошибку *(останавливает JS)*



assert

- `assert`
  проверяет на правдивость выражение

- `assert.equal/notEqual/strictEqual/notStrictEqual`
  проверяет, равны ли два значения

- `assert.deepEqual/notDeepEqual`
  проверяет равны ли объекты по значению

- `assert.throws/assert.doesNotThrow`
  проверяет, падает ли функция с ошибкой

Test Last Development *(TLD)*

Test Last Development — *(тесты потом)* разработка по принципу написания тестов для уже готового кода. После того как код написан, пишутся тесты, которые проверяют, работает ли программа так, как задумано

Test Last Development

подходит в ситуации, когда нужно сформировать набор тестов для уже имеющегося кода

Test Last Development

Test-driven Development (TDD)

методология подхода к написанию ПО на основе тестов

Test-driven development (TDD) —

сначала пишутся тесты для того, чтобы закрепить поведение программы, а потом пишется код, на котором тесты будут выполняться без ошибок

Behavior-driven development (BDD)

набор практик и подходов к написанию тестов. Диктует стиль написания и формулирования тестов

codeutopia.net/blog/2015/03/01/unit-testing-tdd-and-bdd

[ ] Тест 1
слайды корректно переключаются

[ ] Тест 2
не происходит переполнения вправо

[ ] Тест 3
не происходит переполнения влево

[x] Тест 1
слайды корректно переключаются

[ ] Тест 2
не происходит переполнения вправо

[ ] Тест 3
не происходит переполнения влево

[x] Тест 1
слайды корректно переключаются

[x] Тест 2
не происходит переполнения вправо

[ ] Тест 3
не происходит переполнения влево

[x] Тест 1
слайды корректно переключаются

[x] Тест 2
не происходит переполнения вправо

[x] Тест 3
не происходит переполнения влево

Test-driven Development

разработка ведется короткими итерациями,
по определенному алгоритму

Test Driven Development —

1. **написать тест**
   тест используется для описания функциональности, формализует задачу

2. **запустить тесты и проверить, проходят ли они**
   не должны

3. **написать код**
   который будет решать поставленную задачу так, чтобы тесты начали проходить. Код не должен быть идеальным, он просто должен выполнять поставленную задачу

4. **запустить тесты**
   проверить, проходят ли тесты теперь

5. **оптимизировать код**
   после решения задачи, можно улучшить код: оптимизировать алгоритмы, оптимизировать расположение модулей

Test-driven Development

ключевой момент в разработке через тестирование даже не само написание тестов до кода, а именно *короткие итерации написания полностью протестированного кода*

habrahabr.ru/post/314994/

Behaviour Last Development

(поведение потом) разработка после того как придет менеджер и заставит что-то делать

XD

Тестирование данных

gulp-mocha

- установить плагин *gulp-mocha*
- установить *babel-core* и плагины
- настроить *babel* в *package.json*
- дописать таск *test* в *gulpfile.js*
- поправить окружение *eslint.yml*

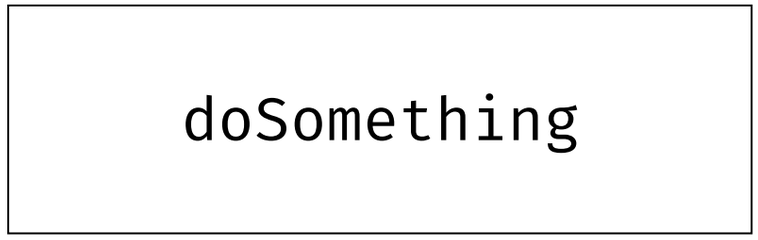
🤓 Функциональное программирование

Функция

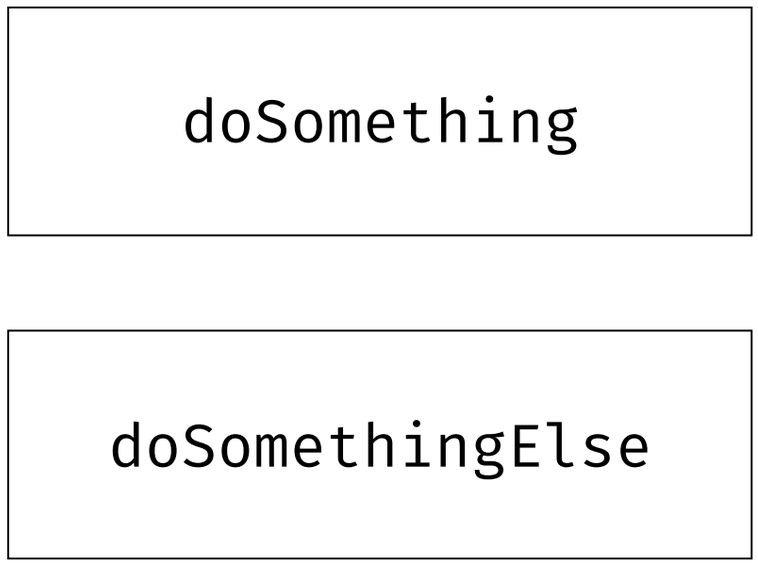
Функция в программировании — подпрограмма. Часть программного кода, к которому можно обратиться из другого места программы

Функция в программировании

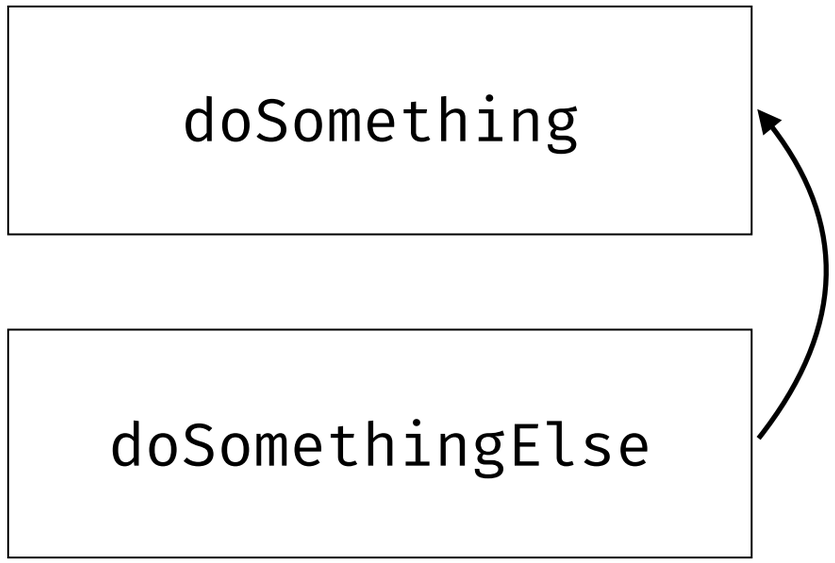
Функция в программировании



Функция в программировании



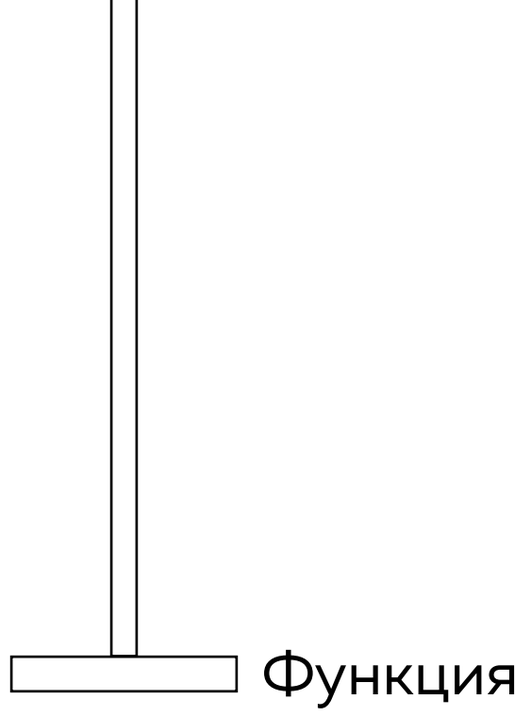
Функция в программировании



Функция в программировании

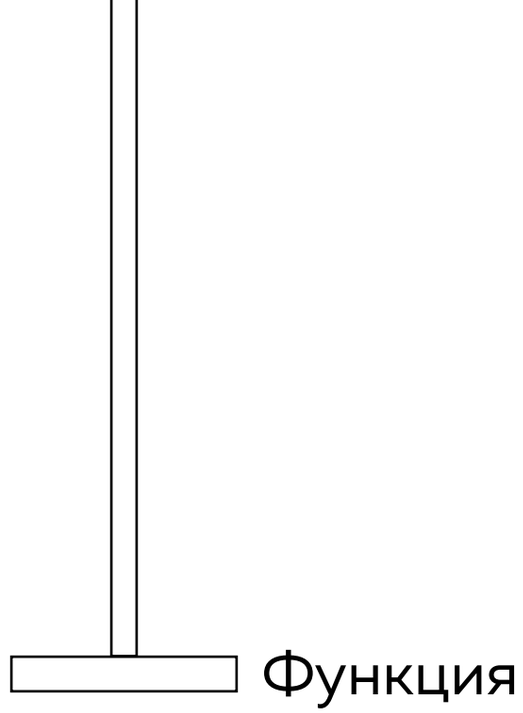
кроме того, что функция это участок повторно вызываемого кода, больше никаких ограничений нет. Функция может делать все, что угодно, может изменять любые значения и производить любые действия

Функция в математике — способ описания отношения между значениями. Ставит в соответствие одному значению другое



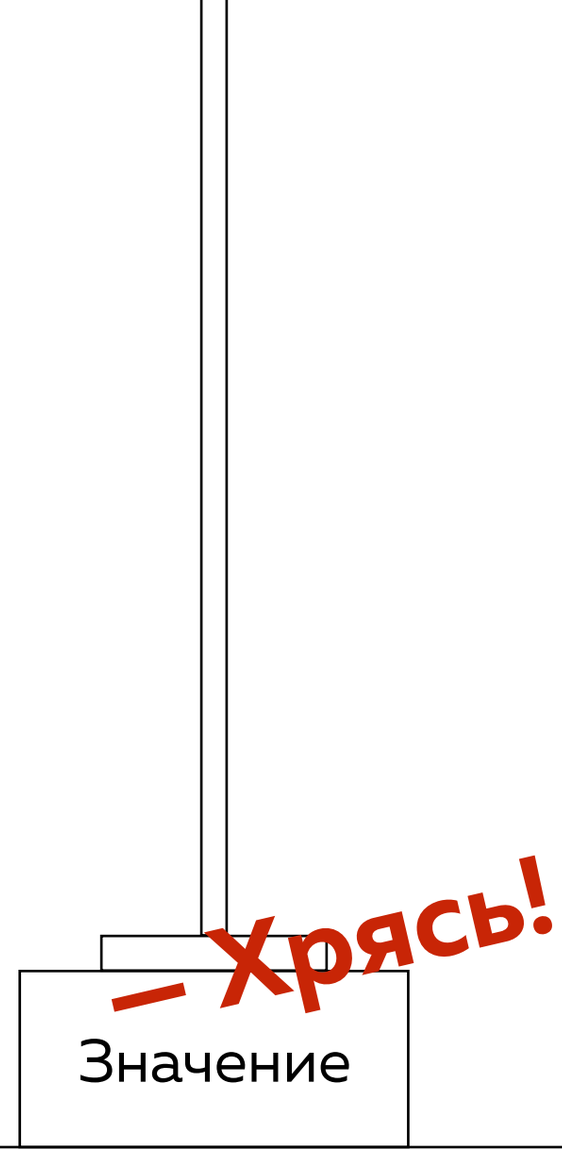
Значение

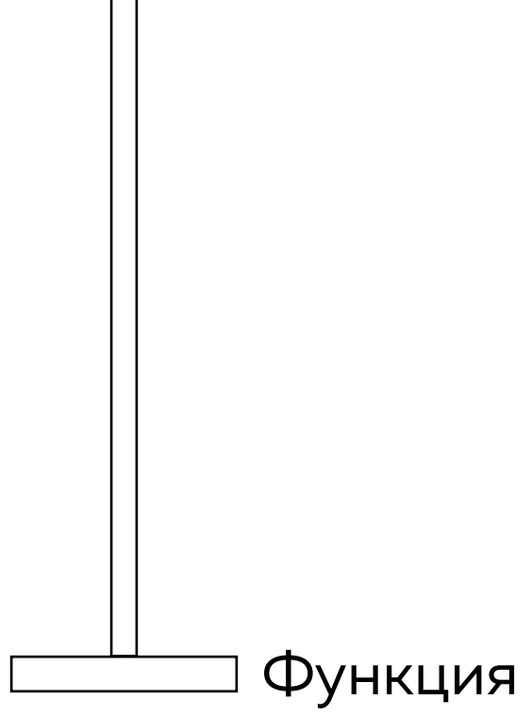
Программа



Значение

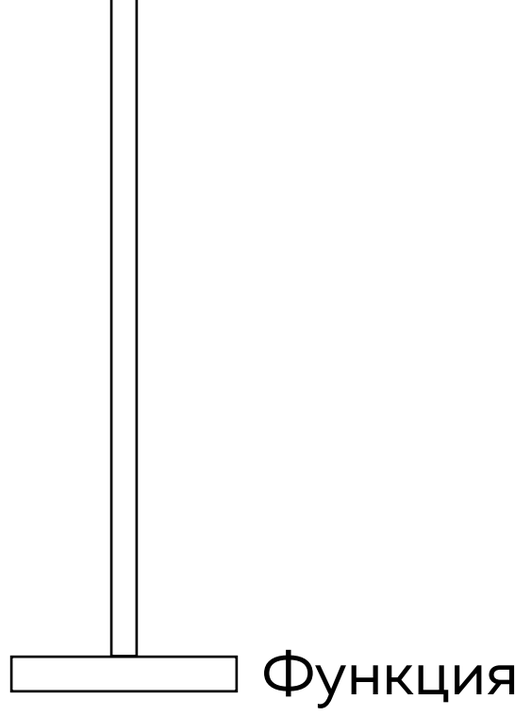
Программа


— Хрясь!
Значение



Другое значение

Программа



Другое значение

Функциональное программирование

оперирует функциями в их математическом понимании. Каждая функция используется для получения предсказуемого вывода, соответствующего входу. При этом для функций есть несколько ограничений

- чистота функций
- «неизменяемость» данных

🛁 Чистые функции —

(pure functions, функции-пюре) функции, которые для получения вывода используют только те значения, которые были поданы им на вход

🛁 Чистые функции

- предсказуемый результат выполнения
- возможность композиции с другими функциями
- удобное тестирование

👽🔫 «Неизменяемость» данных — *(immutability, немутантизм)* все операции над структурами данных не изменяют исходных объектов, а возвращают новые

Принцип copy on write

(копируй при записи) все операции над структурами данных создают новые объекты, основанные на старых объектах, отличающиеся только измененными данными

Как не надо тестировать

1. Не надо повторять код алгоритма в тесте и сравнивать результаты (тест-близнец)

2. Не надо тестировать то, что не надо тестировать. Тестами должны покрываться места в коде, где можно совершить ошибку

3. Не надо писать слишком сложные тесты

https://events.yandex.ru/lib/talks/3932/


(pprint "— Ciao!")
p totip
GRA
PADANO