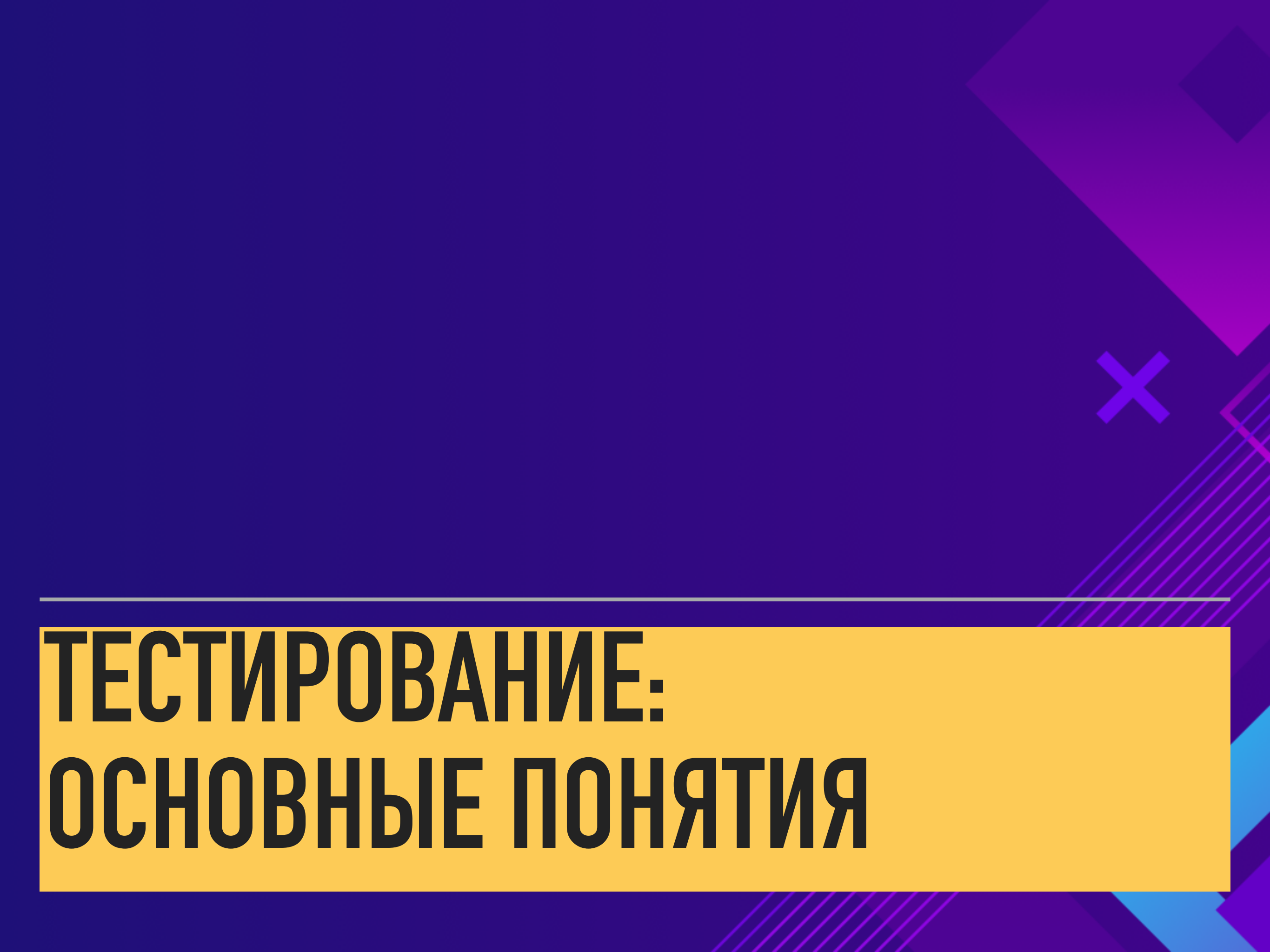


Автоматизация тестирования: введение



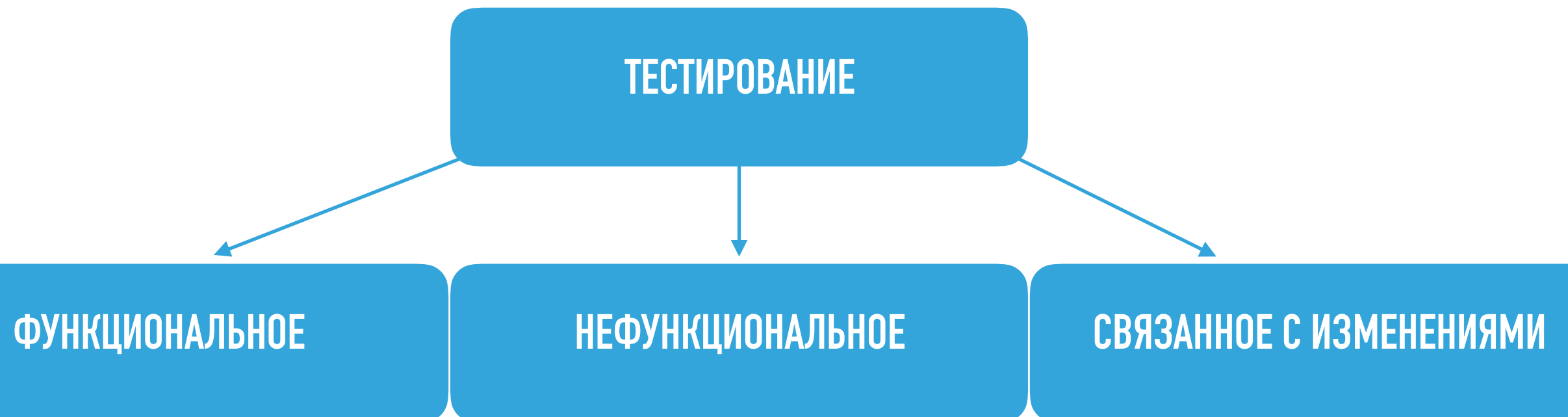


ТЕСТИРОВАНИЕ: ОСНОВНЫЕ ПОНЯТИЯ

КАЧЕСТВО ПРОДУКТА



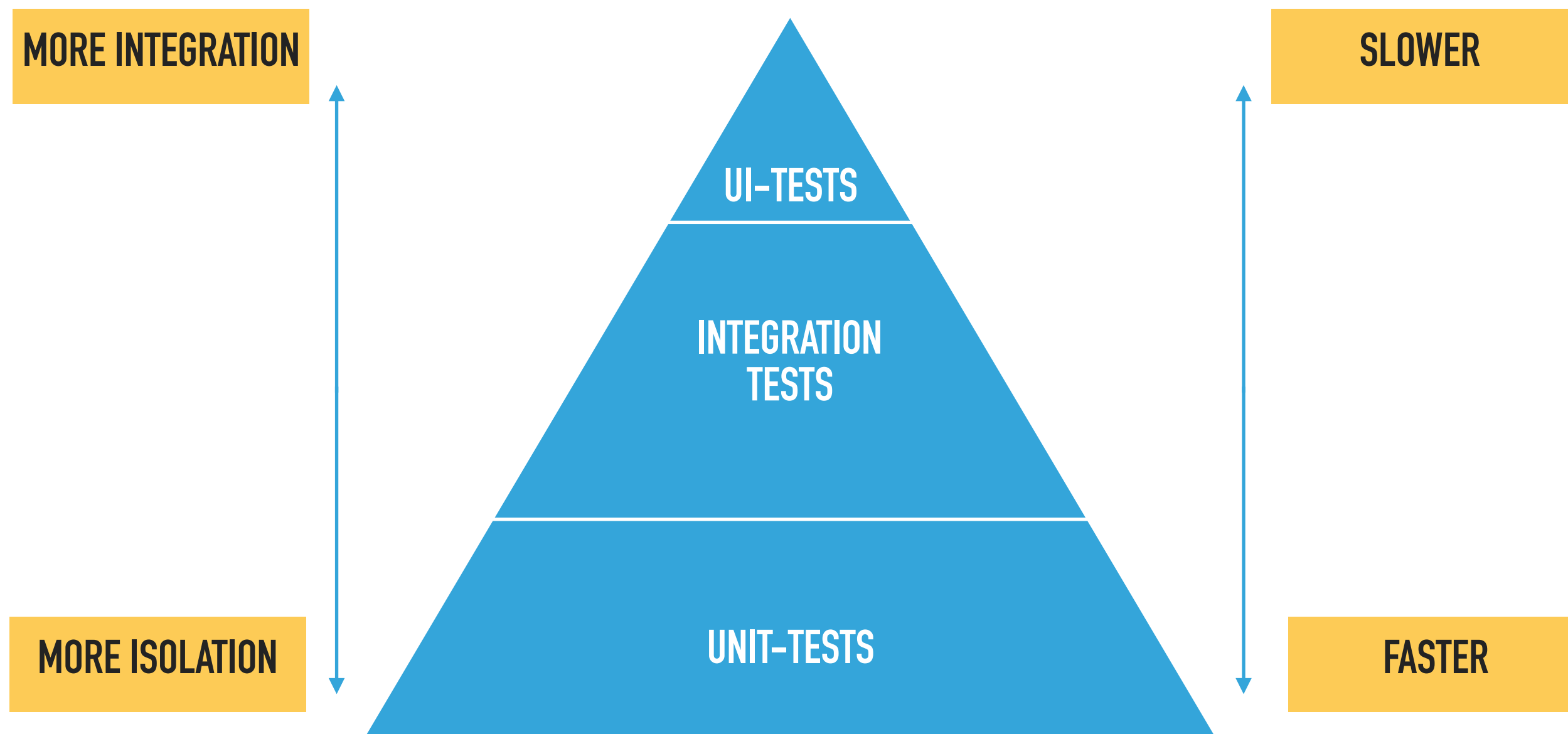
ВИДЫ ТЕСТИРОВАНИЯ





ЦЕЛИ АВТОМАТИЗАЦИИ. ПИРАМИДА ТЕСТОВ

ПИРАМИДА ТЕСТОВ



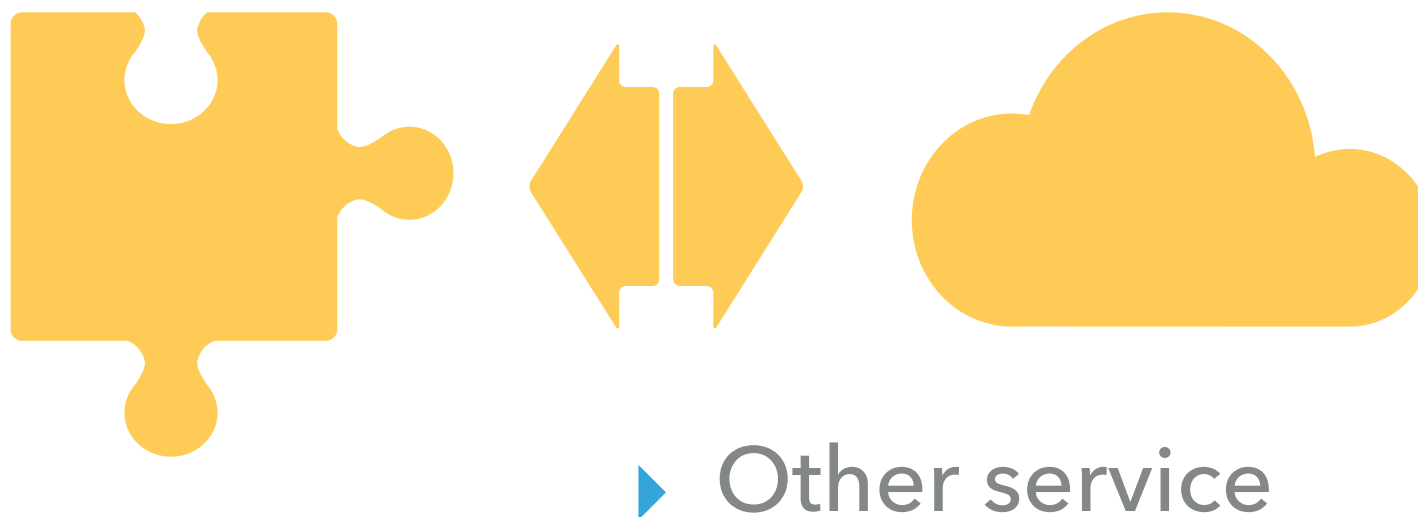
ТЕСТ ИНТЕГРАЦИИ С БАЗОЙ ДАННЫХ

- ▶ Запуск базы данных
- ▶ Подключение приложения к БД
- ▶ Запуск функции в коде, которая записывает данные в БД
- ▶ Проверка, что ожидаемые данные записаны в базу путём их чтения из БД



ТЕСТ ИНТЕГРАЦИИ С СЕРВИСОМ ЧЕРЕЗ REST API

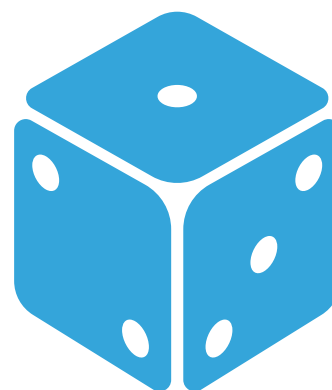
- ▶ Запуск приложения
- ▶ Запуск инстанса отдельной службы
- ▶ Запуск функции в коде, которая берёт данные из API внешней службы
- ▶ Проверка, что приложение правильно разбирает ответ



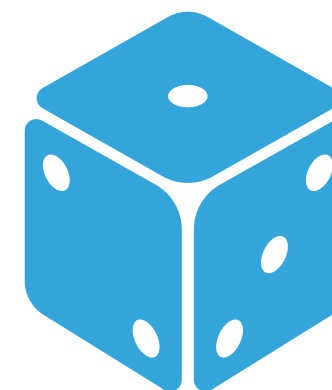
СКВОЗНЫЕ ТЕСТЫ



► User Interface



► Service 1



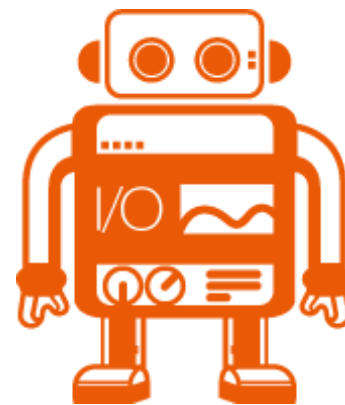
► Service 2

СКВОЗНОЙ ТЕСТ REST API

```
1  const fetch = require("node-fetch");
2  const assert = require("assert");
3
4  let response;
5  let responseJson;
6
7  describe("Restapiexample test", () => {
8    before(async () => {
9      response = await fetch(
10        "http://dummy.restapiexample.com/api/v1/employees",
11        { method: "GET" }
12      );
13      responseJson = await response.json();
14    });
15
16    it("The number of employees is equal 24", async () => {
17      let employeesNumber = responseJson.data.length;
18      assert.equal(employeesNumber, 24);
19    });
20
21    it("The first employee's name is Tiger Nixon", async () => {
22      let firstEmployee = responseJson.data[0];
23      assert.equal(firstEmployee.employee_name, "Tiger Nixon");
24    });
25  });
```

ЧТО ИЗУЧИМ

- ▶ Mocha – тестовый фреймворк
- ▶ Selenium WebDriver – библиотека для автоматизации действий веб-браузера
- ▶ WebdriverIO (3 лекция)
- ▶ Yandex Allure (4 лекция)





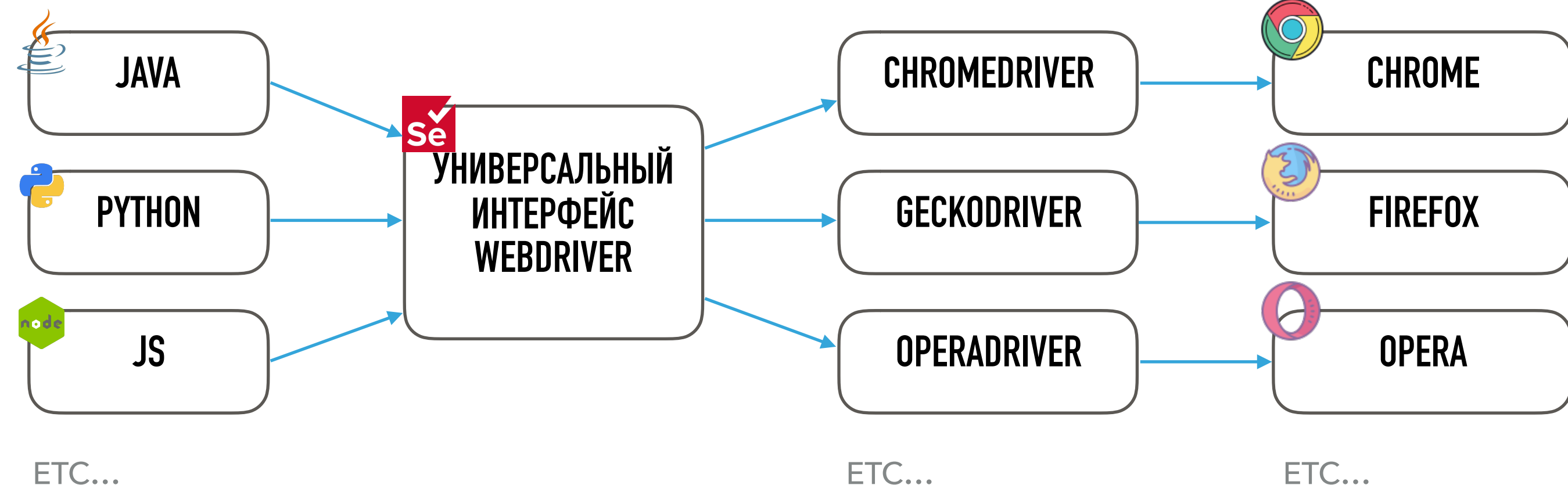
SELENIUM WEBDRIVER

SELENIUM WEBDRIVER – ЭТО:

КЛИЕНТСКАЯ
БИБЛИОТЕКА
НА ЯП

ДРАЙВЕР
БРАУЗЕРА

БРАУЗЕР



СКВОЗНЫЕ GUI-ТЕСТЫ

```
require('chromedriver');
const assert = require('assert');
const {Builder, Key, By, until} = require('selenium-webdriver');

describe("Checkout Google", () => {
  before(async function() {
    driver = await new Builder().forBrowser('chrome').build();
  });

  it('Search on Google', async function() {
    await driver.get('https://google.com');
    await driver.findElement(By.xpath("//input[@name='q']")).click();
    await driver.findElement(By.xpath("//input[@name='q']")).sendKeys('Beeline', Key.RETURN);
    await driver.wait(until.elementLocated(By.id('rcnt')), 10000);
    let title = await driver.getTitle();
    assert.equal(title, 'Beeline - Поиск в Google');
  });

  after(() => driver.quit());
});
```

ПАРАМЕТРИЗИРОВАННЫЕ СКВОЗНЫЕ GUI-ТЕСТЫ

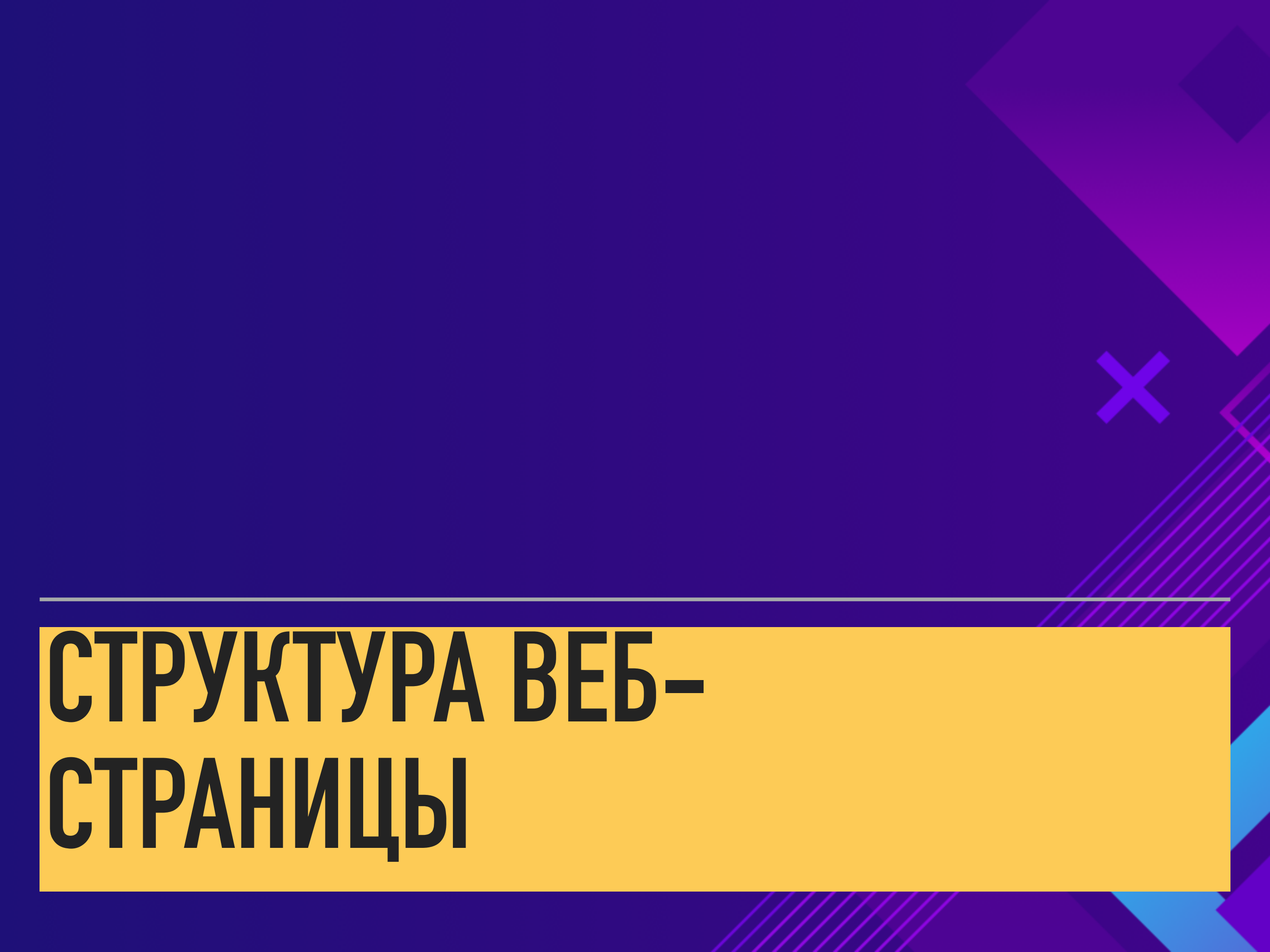
```
require("chromedriver");
const assert = require("assert");
const {Builder, Key, By, until} = require("selenium-webdriver");

const queries = ["Beeline", "Интернет-магазин Beeline", "Билайн"];

queries.forEach((query) => {
  describe("Checkout Google", () => {
    before(async function () {
      driver = await new Builder().forBrowser("chrome").build();
    });

    it("Search on Google: Title", async function () {
      await driver.get("https://google.com");
      await driver.findElement(By.xpath("//input[@name='q']")).click();
      await driver.findElement(By.xpath("//input[@name='q']")).sendKeys(query, Key.RETURN);
      await driver.wait(until.elementLocated(By.id('rcnt')), 10000);
      let title = await driver.getTitle();
      assert.equal(title, `${query} - Поиск в Google`);
    });

    after(() => driver.quit());
  });
});
```



СТРУКТУРА ВЕБ- СТРАНИЦЫ

ЧТО ТАКОЕ HTML

- ▶ HTML (HyperText Markup Language) – язык гипертекстовой разметки
- ▶ Не является языком программирования
- ▶ Используется для определения структуры веб-страниц

ОСНОВНЫЕ ЧАСТИ ЭЛЕМЕНТА

- ▶ Открывающий тэг `<p>`
- ▶ Закрывающий тэг `</p>`
- ▶ Содержимое Привет!
- ▶ Элемент = открывающий тэг + содержимое + закрывающий тэг

```
<!DOCTYPE html>
<html lang="ru">
  <head>
    <meta charset="utf-8">
    <title>Начало работы с HTML</title>
    <script>...</script>
  </head>
  <body>
    <script>...</script>
    <p>Привет!</p> = $0
  </body>
</html>
```

ВЛОЖЕННОСТЬ

- ▶ Вложенный тэг ``: `<p>Привет!</p>`

```
<!DOCTYPE html>
▼ <html lang="ru">
  ▼ <head>
    <meta charset="utf-8">
    ▶ <script>...</script>
    <title>Начало работы с HTML</title>
  </head>
  ▼ <body>
    ▼ <p>
      <strong>Привет!</strong> = $0
    </p>
  </body>
</html>
```

АТРИБУТЫ HTML-ТЕГОВ

- ▶ Пробел между атрибутом и именем элемента (или предыдущим атрибутом, если у элемента уже есть один или несколько атрибутов)
- ▶ Имя атрибута и следующий за ним знак равенства
- ▶ Значение атрибута, заключённое в кавычки

СТРУКТУРА HTML-ДОКУМЕНТА

```
<!DOCTYPE html>
▼ <html lang="ru">
  ▼ <head>
    <meta charset="utf-8">
    ▶ <script>...</script>
    <title>Начало работы с HTML</title>
  </head>
  ▼ <body>
    ▼ <p>
      <strong>Привет!</strong>
    </p>
    <a href="https://google.ru" title="Гугл" target="_blank" ">Google</a> = $0
  </body>
</html>
```

ПОЛЕЗНЫЕ ССЫЛКИ

- ▶ Martin Fowler о пирамиде тестирования: <https://martinfowler.com/articles/practical-test-pyramid.html>
- ▶ Selenium: <https://www.selenium.dev/>
- ▶ Mocha: <https://mochajs.org/>
- ▶ Node.JS: <https://nodejs.org/ru/>

ИНСТРУКЦИИ ПО НАЧАЛУ РАБОТЫ

- ▶ Установить редактор/среду разработки
- ▶ Установить Node.JS: <https://nodejs.org/ru/download/>
- ▶ Создать папку проекта
- ▶ Проверка корректности установки: `node -version`
- ▶ Создание package.json: `npm init -y`
- ▶ Установка Mocha: `npm i mocha`
- ▶ Установка Chromedriver: `npm i chromedriver`
- ▶ Установка Selenium-webdriver: `npm i selenium-webdriver`

ДОМАШНЕЕ ЗАДАНИЕ

- ▶ Установить все инструменты, необходимые для работы
- ▶ Проверить установку
- ▶ Написать такой же базовый тест (без параметризации), изменив вариант поискового запроса
- ▶ Запустить тест и убедиться, что он отработал верно
- ▶ Сделать скриншот успешного результата

**СЛЕДУЮЩАЯ ЛЕКЦИЯ — 10.03.2020,
17:00**

**ВАШИ
ВОПРОСЫ**