**Модульное тестирование JavaScript**

Будем использовать инструмент QUnit.

[QUnit](http://docs.jquery.com/QUnit) - это мощная рабочая среда для модульного тестирования JavaScript, которая существенно помогает отлаживать код. Она разработана командой [jQuery](http://jquery.com/) и является официальным инструментом для тестирования jQuery.

Работать можно как в браузере, так и в Node.js

Мануал по QUnit - <https://unetway.com/tutorials/qunit>

Далее будут рассмотрены примеры, когда тестирование проводится в браузере.

Чтобы начать работу с QUnit в браузере, создайте простой HTML-файл с именем test.html и включите следующую разметку:

***<!DOCTYPE html>***

<meta charset="utf-8">

<title>Test Suite</title>

<link rel="stylesheet"

//qunit.css библиотеки Qunit стилирует страницу набора тестов, чтобы отобразить результаты теста.

href="https://code.jquery.com/qunit/qunit-2.19.4.css">

<body>

// Эти элементы div необходимы и обеспечивают приспособление для испытаний.

<div id="qunit"></div>

<div id="qunit-fixture"></div>

//qunit.js библиотеки Qunit представляет собой тестовый интерфейс и тестовую структуру.

<script src="https://code.jquery.com/qunit/qunit-2.19.4.js"></script>

</body>

Это вся разметка, которая вам нужна, чтобы начать писать тесты. Обратите внимание, что при этом загружается библиотека из [jQuery CDN](https://code.jquery.com/qunit/).

\*\*\*

Теперь добавим следующий скрипт, который проверяет пример add()функции для сложения двух чисел вместе:

<script>

**function** add(a, b) {

**return** a **+** b;

}

// Открываем группу тестов под меткой add

QUnit.module('add', function() {

//Добавляем тест для запуска

QUnit.test('two numbers', function(assert) {

// Нестрогое сравнение

assert.equal(add(1, 2), 3);

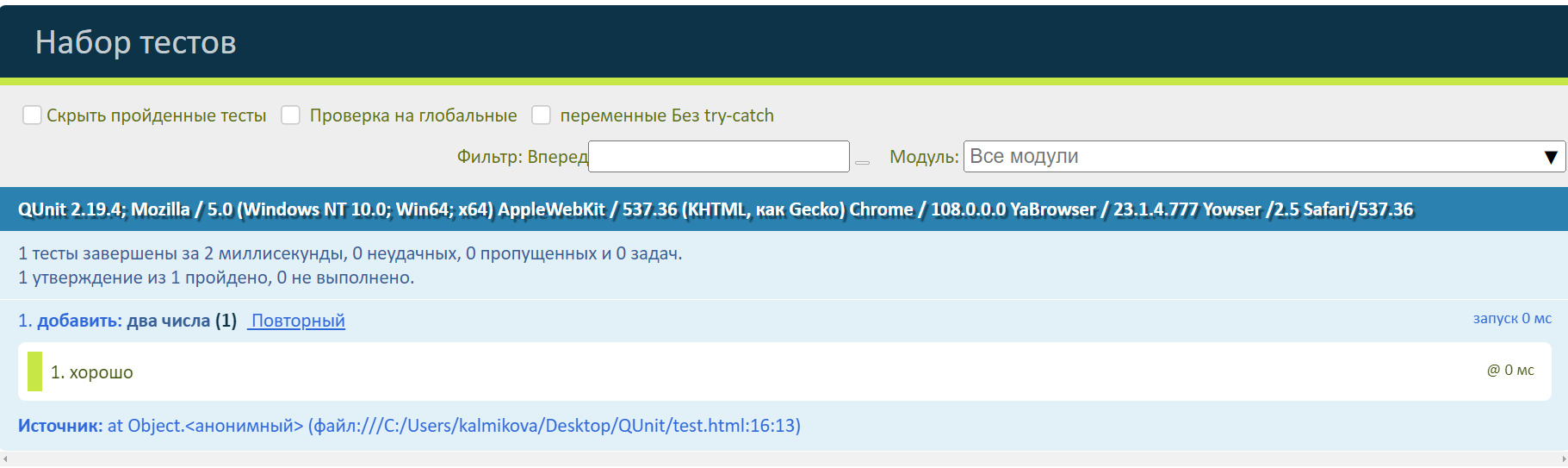
});

});

</script>

Этот код определяет тестовый модуль для add()функции и проверяет результат сложения двух чисел.

Если вы откроете это в браузере, вы найдете подробный отчет о выполненных тестах и их утверждениях, а также различные варианты фильтрации и повторного запуска отдельных тестов, чтобы помочь во время разработки. Вот так:



**\*\*\***

**Некоторые из важных категорий QUnit:**

## **Assert (Утверждение)**

| Код | Описание |
| --- | --- |
| equal() | Нестрогое сравнение, примерно эквивалентное утверждению JUnit's assertEquals. |
| expect() | Укажите, сколько утверждений ожидается в рамках теста. |
| notEqual() | Нечеткое сравнение, проверяющее неравенство. |
| notOk() | Логическая проверка, обратная к ok () и assert.ok () CommonJS, и эквивалентная assertNalse () JUnit. Пропускает, если первый аргумент является ложным. |
| ok() | Булева проверка, эквивалентная assert.ok () и JUnit's assertTrue (). Проходит, если первый аргумент равен true. |
| propEqual() | Строгий тип и сравнение значений свойств объекта. |
| push() | Сообщите результат пользовательского утверждения. |
| strictEqual() | Строгое сравнение типов и значений. |

**Самостоятельно:**

1. Напишите тесты для проверки функции, определяющей четность/нечетность числа.
2. Напишите тесты для проверки функции, определяющей хватит ли в баке автомобиля бензина, чтобы доехать до ближайшей заправочной станции.
3. Напишите функцию для решения следующей задачи:

Пользователь вводит многозначное число через promt. Напишите функцию colonOdd(num), которая принимает число num в качестве аргумента и вставляет двоеточие (:) между двумя нечетными числами. Например, если вводится число 55639217, то на выход должно быть 5:563:921:7.

Протестируйте полученную функцию.