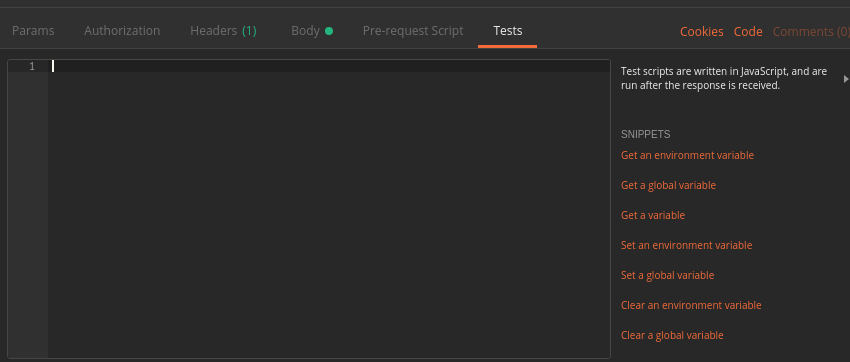
**Тестирование ответа**[**​**](https://for-each.dev/lessons/b/-postman-testing-collections#6-%D1%82%D0%B5%D1%81%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5-%D0%BE%D1%82%D0%B2%D0%B5%D1%82%D0%B0)

**Добавление тестов в запрос**[**​**](https://for-each.dev/lessons/b/-postman-testing-collections#61--%D0%B4%D0%BE%D0%B1%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5-%D1%82%D0%B5%D1%81%D1%82%D0%BE%D0%B2-%D0%B2-%D0%B7%D0%B0%D0%BF%D1%80%D0%BE%D1%81)

Чтобы создать тест, вернемся к панели редактирования запроса, где мы построили наш метод POST. Мы нажимаем вкладку « Тесты », которая находится под URL-адресом. Когда мы это сделаем, появится панель Tests:



В панели Tests пишем JavaScript, который будет выполняться при получении ответа от сервера.

**Postman предлагает встроенные переменные, обеспечивающие доступ к запросу и ответу** . Кроме того, с помощью синтаксиса require() можно импортировать ряд библиотек JavaScript .

В этом руководстве слишком много функций сценариев. Однако [официальная документация Postman](https://learning.getpostman.com/docs/postman/scripts/test_scripts) — отличный ресурс по этой теме.

Продолжим, добавив в наш запрос три теста:

pm.test("success status", () => pm.response.to.be.success );  
pm.test("name is correct", () =>   
 pm.expect(pm.response.json().name).to.equal("Transformers"));  
pm.test("id was assigned", () =>   
 pm.expect(pm.response.json().id).to.be.not.null );

Как мы видим, **эти тесты используют глобальный модуль pm, предоставленный Postman** . В частности, в тестах используются pm.test(), pm.expect() и pm.response .

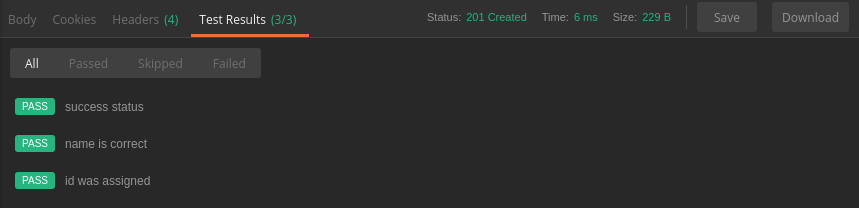
Функция pm.test() принимает метку и функцию утверждения , например, expect() . Мы используем pm.expect() для утверждения условий содержимого ответа JSON.

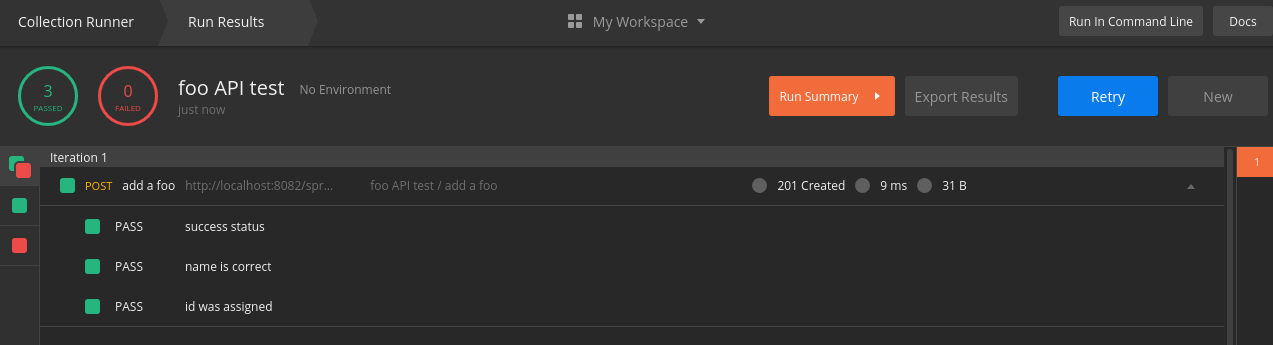
Объект pm.response обеспечивает доступ к различным свойствам и операциям над ответом, возвращенным с сервера. Доступные свойства включают, среди прочего, статус ответа и содержимое JSON.

Как всегда, сохраняем наши изменения с помощью Ctrl-S или кнопки Сохранить .

**Запуск тестов**[**​**](https://for-each.dev/lessons/b/-postman-testing-collections#62--%D0%B7%D0%B0%D0%BF%D1%83%D1%81%D0%BA-%D1%82%D0%B5%D1%81%D1%82%D0%BE%D0%B2)

Теперь, когда у нас есть тесты, давайте снова запустим запрос. При нажатии на кнопку « Отправить » результаты отображаются на вкладке « Результаты теста » панели ответов:

Точно так же бегун коллекции теперь отображает результаты наших тестов. В частности, сводка в левом верхнем углу показывает обновленные итоговые значения успешно пройденных и не пройденных экзаменов. Под сводкой находится список, в котором показан каждый тест с его статусом:



**Просмотр консоли почтальона**[**​**](https://for-each.dev/lessons/b/-postman-testing-collections#63--%D0%BF%D1%80%D0%BE%D1%81%D0%BC%D0%BE%D1%82%D1%80-%D0%BA%D0%BE%D0%BD%D1%81%D0%BE%D0%BB%D0%B8-%D0%BF%D0%BE%D1%87%D1%82%D0%B0%D0%BB%D1%8C%D0%BE%D0%BD%D0%B0)

Postman Console — полезный инструмент для создания и отладки скриптов. Мы можем найти консоль в меню « Вид » с названием пункта « Показать консоль почтальона» . При запуске консоль открывается в новом окне.

**Пока консоль открыта, она записывает все HTTP-запросы и ответы** . Кроме того, когда скрипты используют console.log(), Postman Console отображает следующие сообщения:

