

План

1. Введение
2. Что умеют наши автотесты
3. Структура проекта и рабочее окружение
4. Локальная работа с тестами и проектом
5. Работа с тестами в GitLab

Введение

Привет! Хорошая новость: мы запустили автотесты на курсе «JavaScript. Базовый уровень»!

Что такое автотест? Это программа, которая сама проверяет ваш код по каким-либо критериям. Это поможет вам проверять себя как локально, на своём компьютере, так и при пуше кода в GitLab и сразу исправлять ошибки, если какой-то тест не прошёл.

Что умеют наши автотесты

На данный момент мы запустили две функции автотестирования:


1. Проверка кода с помощью ESLint. ESLint — это инструмент, проверяющий код на соответствие заданным правилам оформления кода и его синтаксису. Например, ставится ли пробел между элементами массива, какие кавычки используются для строк и тому подобное. В разных компаниях код-стайл может быть разным. В нашем случае это правила, установленные для курса JavaScript внутри Skillbox.
2. Unit-тесты с использованием Jest. Такие тесты позволяют проверять получаемый результат работы программы. Например, при тестировании функции, переворачивающей строку, тест может проверять, что при передаче параметра '123456' функция вернёт (return) строку '654321'.

Ниже мы расскажем, как всем этим пользоваться.

Структура проекта и рабочее окружение

В каждом модуле курса вам будет предложено выполнить практическое задание. Обычно на один модуль создаётся проект в GitLab со всем необходимым для начала работы. Но бывают практические задания, которые связаны между собой от модуля к модулю — они будут выполняться в одном и том же проекте. Каждое практическое задание — отдельная ветка, которая должна быть названа определённым образом. Все названия веток будут прописаны у вас в задании, например:

Сдача домашнего задания

 Создайте ветку и Merge Request

1. Создайте ветку с названием `second_homework`.
2. Создайте Merge Request ветки `second_homework` в ветку `master`.

[Перейти в GitLab](#) 

Соблюдать названия очень важно, иначе тесты не сработают (и, соответственно, вы не сдадите работу).

Структура файлов в подобном репозитории также особая, давайте кратко в ней разберёмся. Обращаю ваше внимание: **нельзя** изменять никакие файлы в этом репозитории, кроме, конечно, папки `src`, в которой будет код вашего сайта.

В папке имеется:

1. Папка `src` — это папка, в которую вы должны поместить файлы своего сайта. Не папку с сайтом, а именно файлы. Здесь же могут находиться тесты — это всегда файлы с расширением `.test.js`.
2. Папка `.vscode` — папка с настройками для редактора VSCode, в ней лежит файл `extensions.json`, где указаны рекомендованные для работы плагины. Они будут предлагаться вам редактором кода, если ещё не установлены.

3. `.eslintrc.js` — управляет настройками ESLint, который проверяет JS-файлы.
4. `.gitlab-ci.yml` — конфигурирует запускаемые в GitLab тесты.
5. `jest.config.js` — конфигурирует раннер тестов (Jest).
6. `package.json` — содержит общую информацию о проекте, а также команды для запуска тестов.

О всех этих файлах вы узнаете подробнее в продвинутом курсе JS.

Локальная работа с тестами и проектом

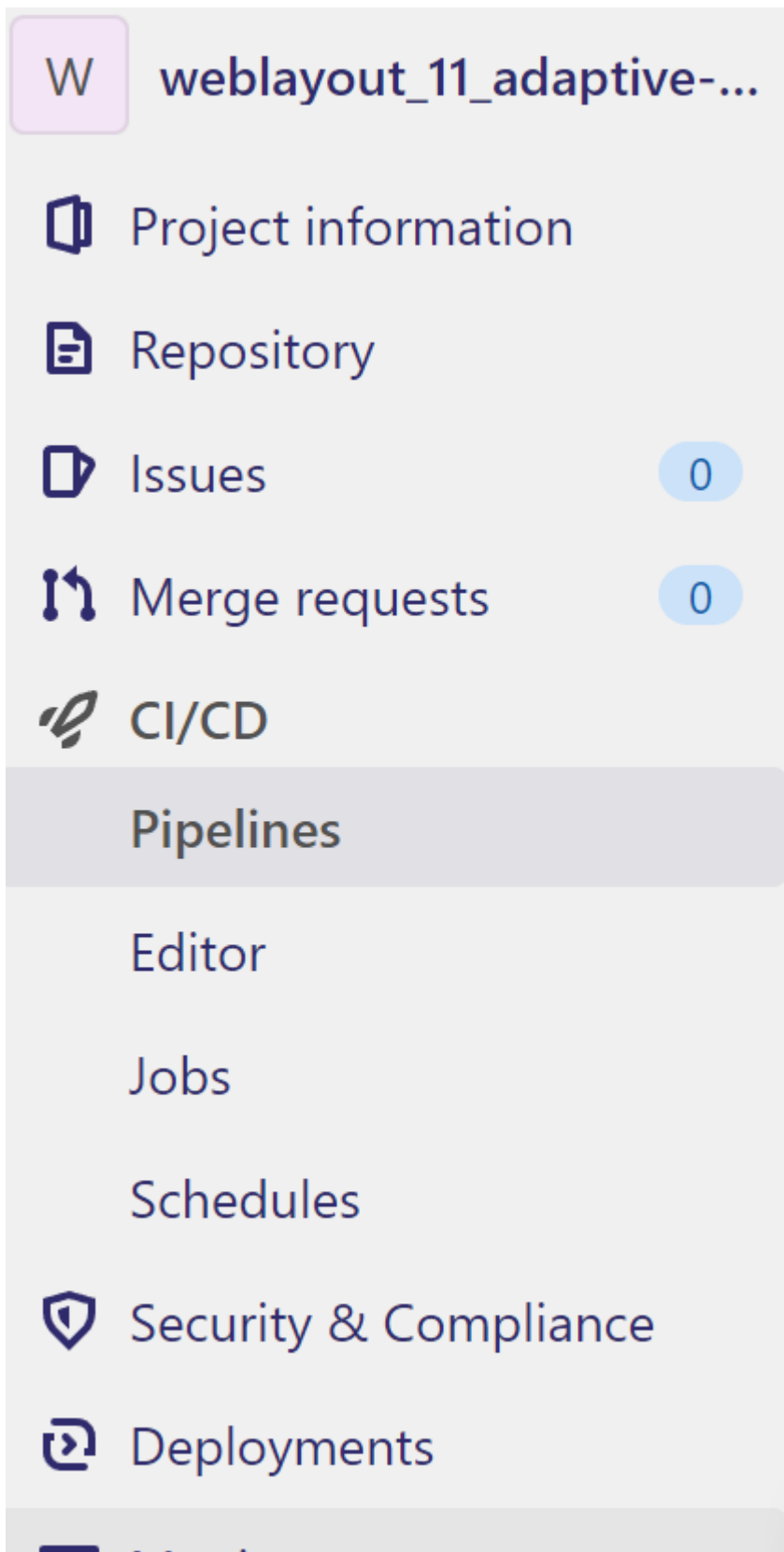
Для запуска тестов достаточно в терминале перейти в папку с проектом (в ту папку, которая содержит файл с расширением `.test.js`.) и запустить одну из двух команд:

- `npm test` для запуска unit-тестов, проверяющих результат работы программы (если для практической работы они предусмотрены, что бывает не всегда).
- `npm run lint` для проверки кода на соответствие правилам оформления в Skillbox. **Но!** Обратите внимание, что вы можете поставить расширение ESLint для вашего редактора, чтобы видеть все ошибки сразу в окне редактора. Так что если вы всё сделаете правильно, **вам не нужно будет запускать линтер вручную.**



Работа с тестами в GitLab

Теперь, когда вы можете локально тестировать себя, стоит научиться делать то же самое в GitLab. Процесс немного отличается. Вам не нужно вводить никакие команды тестов, необходимо просто запустить ваш код как обычно.

Увидеть статус вашего теста вы сможете в разделе CI/CD в репозитории.



Там, в разделе Pipelines вы увидите список проверок, где последняя — ваш текущий пуш, который в данный момент совершает работу — устанавливает все зависимости и запускает тесты прямо внутри GitLab.

Status	Pipeline
	<u>#22932</u> latest 

🕒 1 job for `master` in 1 minute and 16 seconds (queued for 4 seconds)




📌 latest

🔗 b2b77acc

🔍 No related merge requests found.

Pipeline Needs Jobs 1 Failed Jobs 1 Tests 0

Test

 test  

Нажав на номер Pipeline, а затем на test, вы можете наблюдать за работой тестов. Возможно, придётся обновить страницу, чтобы увидеть результат, который выдаёт консоль. Но в итоге вы увидите то же самое, что увидели бы у себя в локальной консоли: те же ошибки, те же предупреждения и так далее. Если они есть и преподаватель их увидит, ваша работа не будет принята, поэтому стоит всё исправить, а уже потом делать пуш.

Всё это может показаться сложным и лишним, но именно так работают в реальных компаниях с Git и тестами, и наша задача — освоить это ещё на этапе обучения. Как говорится, тяжело в учении — легко в бою!

Надеюсь, теперь вы умеете пользоваться нашими автотестами и они принесут в процесс сдачи работы лишь пользу. Удачи!