

قسمت دوم
مدیریت پروژه
با GitHub

TopLearn.com

فصل نهم

GitHub Primer

بررسی اجمالی GitHub

کوتاه کردن تعریفی از GitHub واقعاً دشوار است ، زیرا همزمان بسیاری از کارها را انجام می دهد. بنابراین ، من از کلمات خود استفاده خواهم کرد: "GitHub یک بستر توسعه است که از نحوه کار شما الهام گرفته است. از OpenSource تا تجارت ، می توانید کد را میزبانی و بررسی کنید ، پروژه ها را مدیریت کنید و در کنار 36 میلیون توسعه دهنده نرم افزار بسازید. "

بنابراین GitHub نه تنها یک سیستم عامل میزبانی کد بلکه یک بستر توسعه است. معنی آن چیست؟ این بدان معناست که شما فقط از GitHub برای ذخیره کد خود استفاده نمی کنید. شما از آن برای برنامه ریزی و پیگیری تکامل آن استفاده می کنید. ما تمام ویژگی های آن را در بخش بعدی خواهیم دید ، اما نکته اصلی که باید به خاطر بسپار این است که GitHub در آنجا است تا به شما در ساخت و انتشار پروژه خود کمک کند. اگر برای استفاده از GitHub فقط یک دلیل نیاز دارید ، آن گردش کار توسعه ای است که ارائه می دهد.

مدتها گذشته است روزهایی که مدیر پروژه تمام وظایف معوق را بر روی تخته سفید نوشت و اعضای تیم برای یکدیگر ایمیل ارسال کردند تا پیگیری کنند چه کسی چه کاری انجام می داد. برای بررسی پیشرفت کار ، نیازی به زنجیره های طولانی ایمیل های برگشتی و بعدی نیست. همه اینها توسط GitHub اداره می شود.

GitHub و منبع آزاد

GitHub همیشه متحد نزدیک پروژه های منبع آزاد بوده است. در حقیقت ، GitHub بزرگترین جامعه منبع آزاد جهان است. از آنجا که توسعه دهندگان برای ساخت و به اشتراک گذاری پروژه های خود نیاز به مکانی مناسب دارند ، GitHub

یک انتخاب بدیهی است. به این ترتیب ، همه تصمیمات و مباحث مربوط به پروژه ها را می توان با هر کسی مشورت و پیوست. و این زیبایی منبع آزاد است.

با استفاده از GitHub ، بهترین کاری که می توانید برای یک پروژه منبع باز انجام دهید اکنون ساده تر از همیشه است: مشارکت وقتی در پروژه ای را که دوست دارید مشخص کنید ، می توانید آن را مانند رسانه های اجتماعی دنبال کنید و پیشرفت آن را ببینید.

اگر می خواهید روی یک ویژگی جدید کار کنید یا یک اشکال را برطرف کنید ، فقط باید یک کلون از پروژه تهیه کرده و روی آن کار کنید. این فرآیند "Forking" نام دارد و ستون فقرات پروژه های منبع باز است.

وقتی همه تغییرات را در کپی پروژه خود انجام دادید ، می توانید یک درخواست Pull (PR) را به نگهدار پروژه ارسال کنید.

این بدان معناست که شما درخواست می کنید تغییری که ایجاد کرده اید در پروژه بپیچد و ادغام شود. سایر مشارکت کنندگان سپس تغییرات شما را بررسی می کنند و ممکن است تغییرات دیگری را درخواست کنند. به جای برقراری ارتباط از طریق ایمیل یا پیام فوری ، همه این موارد در GitHub انجام می شود. پس از توافق همه طرفین در مورد تغییرات ، درخواست Pull پذیرفته می شود و تغییرات شما اکنون جزئی از پروژه هستند!

البته ، پروژه های منبع آزاد بیش از کد هستند؛ آنها به اسناد ، مترجمان ، مدیران جامعه ، نگهبانان و موارد دیگر نیاز دارند. شما می توانید با نوشتن اسناد و ارائه ترجمه یا حتی بررسی تغییری که سایر همکاران ایجاد کرده اند در پروژه ها مشارکت کنید. پروژه ها همچنین به آزمایش کنندگان و افرادی نیاز دارند که بتوانند بینش در مورد محصولات نهایی را ارائه دهند. آنها پروژه هایی هستند که میلیون ها سهم دارند ، بنابراین مدیران جامعه مورد نیاز هستند. آنها مسئولیت بهزیستی جامعه را بر عهده دارند و از آنها انتظار می رود که آیین نامه داخلی رفتار جامعه را اجرا کنند.

برخی از مشارکت کنندگان وظیفه دارند از افراد مبتدی استقبال و تدریس کنند که این کار دشوار اما بسیار ضروری برای هر پروژه است.

GitHub توسط میلیون ها پروژه از منبع آزاد انتخاب شد زیرا گردش کار از ایده تا انتشار بسیار آسان و در دسترس است. مفهوم جعل پروژه برای کمک به آن ، نیروی محرکه اصلی هر پروژه منبع باز است. و اگر یک پروژه را دوست دارید اما مسیر را دوست ندارید می توانید آن را چنگ بزنید و طعم خود را از پروژه شروع کنید. سپس شما حافظ پروژه جدید خواهید بود و دیگران می توانند در صورت تمایل به مشارکت ، درخواستهای Pull را به شما ارائه دهند. بنابراین ، هر کسی خوشحال است! همانطور که قبلاً مشخص شده بود ، پروژه های OpenSource برای مبتدیان مستندات و آموزش لازم دارند. برای پروژه های کوچک ، یک فایل متنی (با نام معمولی README خوانده می شود) کافی است.

پرونده README باید پروژه را ارائه داده و مشکلات را حل کند.

همچنین باید به کاربران نحوه نصب و استفاده از آن و همچنین نحوه کمک به آن را بیان کند. می توانید شکل 1-9 را برای نمونه ای از پرونده README بررسی کنید (که می توانید در <https://github.com/git/git> نیز بررسی کنید).

Git - fast, scalable, distributed revision control system

Git is a fast, scalable, distributed revision control system with an unusually rich command set that provides both high-level operations and full access to internals.

Git is an Open Source project covered by the GNU General Public License version 2 (some parts of it are under different licenses, compatible with the GPLv2). It was originally written by Linus Torvalds with help of a group of hackers around the net.

Please read the file `INSTALL` for installation instructions.

Many Git online resources are accessible from <https://git-scm.com/> including full documentation and Git related tools.

See `Documentation/gittutorial.txt` to get started, then see `Documentation/giteveryday.txt` for a useful minimum set of commands, and `Documentation/git.txt` for documentation of each command. If git has been correctly installed, then the tutorial can also be read with `man gittutorial` or `git help tutorial`, and the documentation of each command with `man git-<commandname>` or `git help <commandname>`.

CVS users may also want to read `Documentation/gitcvs-migration.txt` (`man gitcvs-migration` or `git help cvs-migration` if git is installed).

The user discussion and development of Git take place on the Git mailing list -- everyone is welcome to post bug reports, feature requests, comments and patches to git@vger.kernel.org (read `Documentation/SubmittingPatches` for instructions on patch submission). To subscribe to the list, send an email with just "subscribe git" in the body to majordomo@vger.kernel.org. The mailing list archives are available at <https://public-inbox.org/git/>, <http://marc.info/?l=git> and other archival sites.

Issues which are security relevant should be disclosed privately to the Git Security mailing list git-security@googlegroups.com.

The maintainer frequently sends the "What's cooking" reports that list the current status of various development topics to the mailing list. The discussion following them give a good reference for project status, development direction and remaining tasks.

The name "git" was given by Linus Torvalds when he wrote the very first version. He described the tool as "the stupid content tracker" and the name as (depending on your mood):

همانطور که در شکل 1-9 مشاهده می کنید، فایل های README می توانند قالب بندی و پیوند متن اصلی داشته باشند. آنها همچنین می توانند شامل تصاویر و نمونه کد باشند. پروژه های بزرگ بیش از پرونده های README نیاز دارند زیرا باید به درستی ارائه و مستند شوند. پروژه های GitHub دارای بخشی به نام "ویکی" است که به طور خاص برای آن نیازها تنظیم شده است.

درست مانند همه ویکی ها (از ویکی پدیا مدل شد)، ویکی های GitHub برای کمک به تازه واردان در درک چگونگی عملکرد این پروژه وجود دارند. بسیاری از ویکی ها همچنین دارای بخشی به نام سوالات متداول هستند که در آن متداول ترین سوالات کاربر پاسخ داده می شود. به طور کلی، ویکی ها توسط پروژه هایی استفاده می شوند که مستندات و

آموزش ها برای قرار گرفتن در پرونده README خیلی طولانی هستند. در شکل 2-9 می توانید نمونه ای از صفحه ویکی را مشاهده کنید که می توانید در <https://github.com/Dash-Industry-Forum/dash.js/wiki> نیز پیدا کنید. به نوار کناری توجه کنید که در آن همه پیوندها ارائه شده است.

FAQ

Jesús Oliva edited this page on 2 Oct 2018 · 21 revisions

Content Prep

Encoding/transcoding is a pretty complex topic. The FFmpeg/x264/mp4box workflow is generally fine and it is what we use in the Axinom reference encoder. My general suggestions regarding the most critical points in this regard:

- Always use DASH live profile (`mp4box -profile "dashavc264:live"`)
- Ensure that your encoder uses a fixed keyframe distance that is equal to your segment size (or a multiple of which is equal to it); FFmpeg has some defects here leading to bad output; with x264 the following works: `--keyint 50 --min-keyint 50 --no-scenecut`
- Use dash-strict mode with mp4box if you use fixed keyframe distances (`mp4box -dash-strict 4000`) - the default dash mode produced unexpected deviations last I tried it
- To avoid FFmpeg being clever with frame drop/duplication, use `-vsync passthrough`
- Watch out for aspect ratio issues! Not all input content has SAR 1:1!
- Do not use bitstream switching (`mp4box -bs-switching no`)
- If using encryption, put PSSH box information only in the manifest; (no PSSH data in crypt.xml for mp4box); also `mp4box -sample-groups-traf` made encrypted video work better in more players but I forget why

Codecs

- Safari <=9 does not support AVC3
- Internet Explorer 11 can play AVC3, but only if you signal to the browser it is AVC1
- MEDIA_ERR_DECODE indicates an issue with your stream, not dash.js.
 - [Understanding supported codecs](#)
 - In Chrome, the `chrome://media-internals` page may help you identify the problem

Browser Support

- Firefox < 49 may sometimes fail to start playback on dynamic streams, or streams with a #t= URL fragment

Pages 40

Questions

Please post questions to dash.js Google Group

Documentation

- [Dash.js API Docs](#)
- [FAQ](#)
- [How to Release Dash.js](#)
- [Dash.js 3.0 Migration Doc](#)

Samples

View the latest sample players and example implementations.

Minimum Test vectors

- [Smoke test files](#)

Meeting Minutes

- [Archives of our bi-weekly calls](#)

Background Info

- [Embedding an adaptive streaming video within your HTML5 application](#)
- [Building an Open Source DASH-AVC/264 Player](#)
- [How to: Creating a DASH-264 Player](#)

از آنجا که اسناد و مدارک در پروژه های منبع آزاد بسیار مهم هستند ، کار نوشتن و نوشتن آن به روزرسانی بسیاری از مشاغل مشترک است. به یاد داشته باشید که ویکی ها همچنین مخازن Git هستند ، بنابراین تغییرات ایجاد شده در آن نیز دقیقاً مانند هر مخزن ردیابی می شوند. این کار برای جدا کردن گردش کار توسعه از گردش کار اسناد انجام می شود.

و به عنوان گیلان در پرونده های README بالا و ویکی ها با زبانی نشانه گذاری به نام Markdown نوشته شده اند. این یک زبان بسیار ساده است که می تواند قالب بندی و پیوند ساده را ارائه دهد. نمونه ای از آن را در شکل 3-9 مشاهده می کنید. اما همچنین می توانید همه چیز را در HTML بنویسید تا تبدیل به Markdown. و همچنین می توانید یک برگه Markdowncheat را در پیوست این کتاب بیابید!

Headers

```
# This is an <h1> tag
## This is an <h2> tag
##### This is an <h6> tag
```

Emphasis

```
*This text will be italic*
_This will also be italic_

**This text will be bold**
__This will also be bold__

_You can combine them_
```

Lists

Unordered

```
* Item 1
* Item 2
  * Item 2a
  * Item 2b
```

Ordered

```
1. Item 1
1. Item 2
1. Item 3
  1. Item 3a
```

یک نکته کوچک که پروژه های منبع باز نیز باید به آن رونق پیدا کنند: بازاریابی. بله ، پرونده ها و ویکی های README منابع خوبی برای توسعه دهندگان هستند ، اما ممکن است کاربران نهایی آنها را خیلی مفید نگذارند. به همین دلیل است که بسیاری از پروژه ها دارای یک وب سایت هستند که اختصاص به جذب کاربران به محصول خود دارد. وب سایت ها همچنین روش خوبی برای ایجاد نام برای خود و قرار دادن خود در آنجا هستند. اگر یک پروژه وب حضور نداشته باشد یا توسط موتورهای جستجو ارجاع نشده باشد ، احتمال کمتری برای کشف توسط کاربران نهایی خواهد داشت. تمام آنچه گفته می شود ، نگهداری و میزبانی وب سایت کار آسانی نیست. حتی می تواند گران قیمت باشد. و بسیاری از پروژه های منبع باز از آن دسته از منابع لازم برای یک کارزار خوب بازاریابی برخوردار نیستند. آنها بیشتر به بازدید موتورهای و کلمات از دهان تکیه می کنند. به همین دلیل صفحات GitHub وجود دارد. صفحات GitHub فقط یک وب سایت است که به طور مستقیم در مخزن شما میزبان است. شما می توانید از آن برای ارائه محصول خود ، ارائه آموزش ها یا هر چیز دیگری که می خواهید استفاده کنید. این از دردسر ایجاد وب سایت و میزبانی آن خلاص می شود.

اما آیا این باکد تداخل نمی‌کند؟ به هیچ وجه ، مانند ویکی ها ، صفحات GitHub در سایر قسمت های مخزن زندگی نمی‌کنند. بنابراین آنها می‌توانند سهم مختلفی داشته باشند. می‌توانید شکل 4-9 را برای نمونه ای از یک صفحه GitHub که در <https://scd-aix-marseille-universite.github.io/latexamu> میزبانی شده است ، بررسی کنید.

همانطور که می‌بینید ، این فقط یک وب سایت ساده است اما به طور مستقیم در GitHub میزبان است. و این فقط به وب سایت های ارائه ساده محدود نمی‌شود؛ شما همچنین می‌توانید وبلاگ ها و وب سایت های مشابه ایجاد کنید.



Modèle de mise en page.

Ce modèle de mise en page pour les thèses de doctorat soutenues à Aix Marseille Université propose un ensemble de fichiers LaTeX commentés, prêt à être compilés dont une classe LaTeX [.cls].

Télécharger le PDF

La page de titre a obtenu l'approbation du Collège Doctoral Aix-Marseille Université.

همانطور که مشاهده می‌کنید ، GitHub پیشنهادات زیادی برای جامعه منبع باز دارد. و همه اینها رایگان است! اما اکنون ، ببینیم GitHub چه شخصی برای ارائه به شما شخصاً دارد.

استفاده ی شخصی

بله ، منبع آزاد عالی است ، اما مربای شما چیست ؟ یا وقتی پروژه ای دارید که می خواهید خودتان آنرا حفظ کنید ؟ GitHub نیز شما را پوشانده است!

لازم نیست همه مخازن GitHub خود را عمومی کنید. همچنین گزینه ای برای خصوصی سازی آنها وجود دارد. به این ترتیب ، فقط شما و چند همکار (که شما انتخاب می کنید) می توانید به آن دسترسی داشته باشید. می توانید تعداد نامحدودی از مخازن عمومی و خصوصی GitHub ایجاد کنید. تنها محدودیت خلاقیت و زمان شماست. با این وجود محدودیتی در تعداد مشارکت کنندگان شما در مخازن خصوصی وجود دارد: ۳- اگر می خواهید با مشارکت کنندگان بیشتری کار کنید ، می توانید در GitHub Pro ثبت نام کنید که این یک برنامه پرداختی است. اما تقریباً برای همه ، یک طرح رایگان بیش از حد کافی است.

داشتن یک حساب شخصی GitHub برای به نمایش گذاشتن کار خود نیز راهی مناسب برای بازاریابی خود است. از این طریق ، افراد می توانند OpenSource یا پروژه های شخصی را که در آن کمک می کنید بررسی کنند و حتی کد شما را نیز بررسی کنند. بسیاری از توسعه دهندگان همچنین از صفحات GitHub برای ارائه مجدد رزومه خود یا به نمایش گذاشتن نمونه کارها خود استفاده می کنند. برای نمونه ای از آن می توانید شکل 5-9 را بررسی کنید.

CAREER PROFILE

Python/JavaScript developer and Data Analyst.
Expert in Django and NodeJS/AngularJS. Fluent in NoSQL databases.
I am looking for a job where I can work with a lot of Data. I like to experiment with Machine Learning and bots.

EXPERIENCES

Full Stack Developer

Human Network International

2017 - Present

Develop new features for DataWinners, an SMS data collection tool powered by NoSQL databases and using Elasticsearch for search.
Reduced data export time from 30 mins to 15secs.
Using: Django, Numpy, Pandas, CouchDB, Elasticsearch.

Full Stack Developer

Wyllog

2016

Developing new front and backend web features. Working with designers to deliver new fully functional web apps powered by NoSQL databases.
Added three new big features to the existing Angular web app and accelerated the response rate.
Using: AngularJS 2, NodeJS, MongoDB, TypeScript, REST APIs.

Full Stack Developer

Adbreakfast

2016 - 2017

Create REST APIs accessing a relational database. Use these APIs to build a web application.
Created flexible and secured APIs with GraphQL and Flask. Used those APIs to power a ReactJS web app.
Using: Flask, PostgreSQL, ReactJS, GraphQL, Unit tests.



**Mariot
Tsitoara**

Full Stack Developer

✉ mariot.tsitoara@gmail.com

☎ 261343916159

🌐 mariot.github.io

in mariottsitoara

🌐 mariot

🐦 mariot_tsitoara

EDUCATION

MBA
University of Malawi
2017 - Present

MSc in Computer Science
Ecole Nationale
d'Informatique
2014 - 2016

و از آنجا که 36 میلیون برنامه نویس در GitHub وجود دارد، ممکن است بخواهید با برخی از آنها ارتباط برقرار کنید. یکی از راه های اتصال، دنبال کردن یک پروژه خاص است. وقتی پروژه به طول انجامید، به روزرسانی می کنید و می توانید تغییرات را بررسی کنید. توجه داشته باشید که به طور خودکار مخزنی را که در آن کمک می کنید دنبال خواهید کرد. راه دیگر برای نشان دادن قدردانی شما از یک پروژه، همچنین "ستاره سازی" آن است. شبیه به دوست داشتن مطالب در رسانه های اجتماعی است.

از این رو هرچه ستاره های بیشتری در اختیار یک مخزن قرار بگیرند، کاربران بیشتری از آن راضی هستند. GitHub همچنین یک خبرخوان را ارائه می دهد که اخبار و اعلان هایی از پروژه های خاص است. این پروژه ها انتخاب می شوند زیرا شما به آنها کمک می کنید یا آنها را "ستاره دار" می کنید. آنها همچنین با تجزیه و تحلیل زبان و ابزارهای مورد استفاده شما متناسب هستند. می توانید شکل 6-9 را برای نمونه ای از آن بررسی کنید. این یک روش خوب برای داشتن یک چشم انداز روشن از اتفاقات پیرامون شما است.

Based on your interests



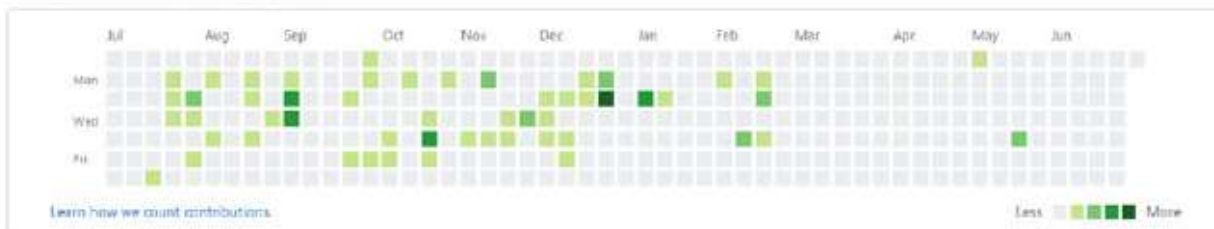
Trending repositories



قبل از اینکه به قسمت بعدی برویم، یک نکته جالب وجود دارد که می توانید با GitHub بررسی کنید: فعالیت مشارکت شما. اگر هر گزینه ای را فعال کنید، GitHub حتی به مخازن شخصی یا خصوصی نیز به عنوان سهم ثبت می شود.

این فعالیت ها در یک تصویر خوب مانند آنچه در شکل 7-9 نشان داده شده است، ارائه می شوند. آنها کمکهای شما را در طول سال نشان می دهند و دستاوردهای شما را به بازدید کنندگان نمایه شما نشان می دهند.

153 contributions in the last year



GitHub برای مشاغل

GitHub فقط برای پروژه های شخصی یا جوامع منبع باز نیست؛ مشاغل جای خود را نیز در آنجا دارند. بسیاری از مشاغل اکنون برای برخی از محصولات خود در Open Source سرمایه گذاری می کنند و کدام مکان بهتر برای یافتن توسعه دهندگان با کیفیت از GitHub است؟

یک طرح Enterprise در GitHub وجود دارد که تمام مزایای یک برنامه پرداخت شده را شامل می شود ، اما دارای بسیاری از ویژگی های اضافی است. این ویژگیها از انتخاب میزبانی ، امنیت ، پشتیبانی آنلاین پشتیبانی می شود. همه این ویژگیها ممکن است مشاغل بسیار جذابی باشند ، اما برای ما ، یک طرح ساده رایگان در حال حاضر کافی است.

خلاصه

در این فصل کاربران GitHub و برخی از ویژگی های کوچک ارائه شده است. اکنون باید ایده هایی در مورد استفاده از آن داشته باشید. در فصل بعد ، ویژگی های اصلی GitHub به همراه نکاتی در مورد نحوه استفاده از آن برای همکاری با هم تیمی ارائه می شود. ما در مورد مدیریت پروژه ، بررسی Code ها و موارد دیگر صحبت خواهیم کرد.

و برای به پایان رساندن زیبایی ، ما به سرعت با GitHub با اولین مخزن خود شروع خواهیم کرد! در فصل بعد دوباره به فعالیت خود باز خواهید گشت ، بنابراین حتما تمرین های قبلی را مرور کنید تا تیز بمانید. شروع کنیم!

فصل دهم

شروع سریع با GitHub

تاکنون فقط درمورد اینکه GitHub چیست و چه کسی به آن نیاز دارد صحبت کردیم. حال می خواهیم ببینیم که دقیقاً چه کاری انجام داده است و ویژگی های اصلی آن چیست. مهمترین ویژگی های GitHub ابزار Project Management آن است. همراه با گردش کار مناسب توسعه ، روشی مطمئن برای حرکت یک پروژه است.

برای این بخش از کتاب ، چیزی بهتر از تمرینات خوب قدیمی نیست! من می توانم تمام مزایای GitHub را برای شما تعریف کنم ، اما اگر خودتان اکتشاف را انجام دهید ، بهتر خواهید فهمید. بیایید با ایجاد یک حساب GitHub و شروع یک پروژه شروع کنیم.

مدیریت پروژه

توانایی مدیریت یک پروژه در حالی که در طی یک مسیری تثبیت شده یکی از قابل تحسین ترین ویژگی های GitHub است که قصد دارید در این بخش با من همراه باشید. بسیار مهم است که این کار را انجام دهید زیرا درک بهتری از ویژگی های آن خواهید داشت.

از آنجا که ما می خواهیم پروژه خود را با Git و GitHub مدیریت کنیم ، اولین قدم ما ایجاد یک حساب کاربری است. بسیار واضح است ، و نیازی به اطلاعات بیشتر از نام و ایمیل خود ندارید دقیقاً مانند شکل 10-1.

Built for developers

GitHub is a development platform inspired by the way you work. From open source to business, you can host and review code, manage projects, and build software alongside 36 million developers.

Username
mtsitoara ✓

Email
mariot.tsitoara@protonmail.com ✓

Password
••••••••••

Make sure it's at least 15 characters OR at least 8 characters including a number and a lowercase letter. [Learn more.](#)

Sign up for GitHub

By clicking "Sign up for GitHub", you agree to our [Terms of Service](#) and [Privacy Statement](#). We'll occasionally send you account related emails.

پس از ثبت نام ، پیوند تأیید را در سرویس گیرنده ایمیل خود دریافت خواهید کرد ، و پس از پیوند ارائه شده ، این کتیبه به پایان می رسد. سپس به صفحه اصلی GitHub خواهید رسید که باید مانند شکل 10-2 باشد.

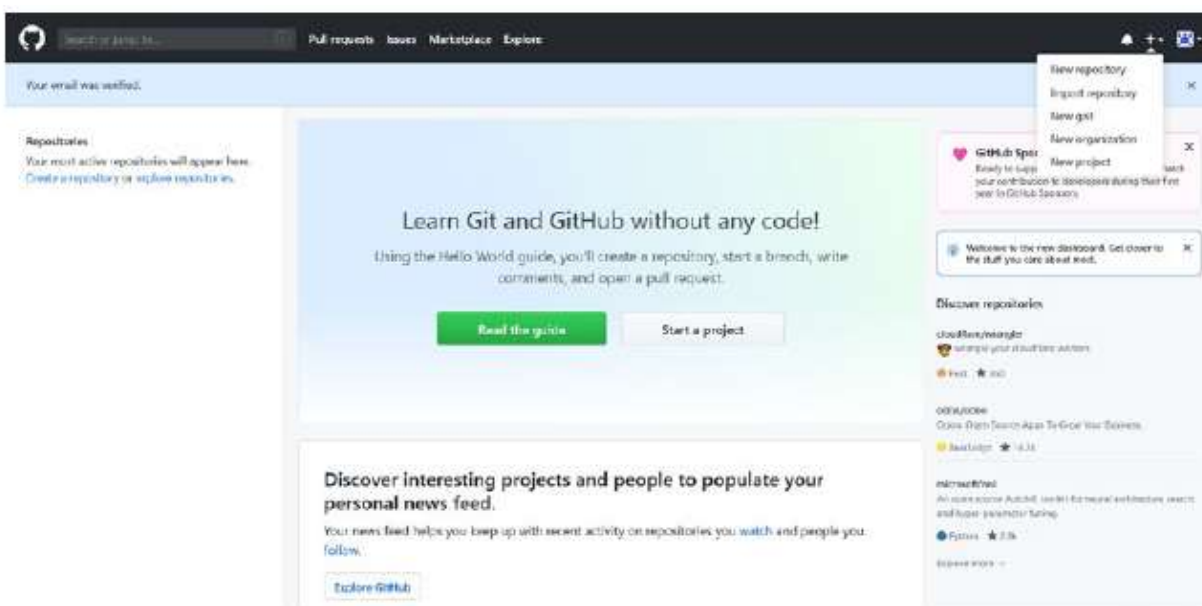


Figure 10-2. GitHub homepage

صفحه اصلی GitHub شما بسیار خالی است اما ما در تلاشیم تا آن را با پروژه های جالب پر کنیم. در سمت راست صفحه ، برخی از مخازن یا اخبار دائمی را مشاهده می کنید؛ اما ما هنوز به آنجا نخواهیم رفت

همانطور که در شکل 10-2 مشاهده می کنید ، سه لینک وجود دارد که می توانید برای ایجاد یک مخزن جدید دنبال کنید: یکی در سمت چپ ، دیگری در وسط و آخرین در نوار پیمایش. روی یکی از آنها کلیک کنید تا ما بتوانیم مخزن خود را ایجاد کنیم. فرم ایجاد مخزن نیز بسیار ساده است ، همانطور که در شکل 10-3 مشاهده می کنید. شما فقط باید فرم را با یک نام و توضیحی کوتاه از پروژه پر کنید. این توضیحات اختیاری است ، اما باید سعی کنید آن را به ساده ترین شکل ممکن انجام دهید تا کاربرانی که از مخزن شما بازدید می کنند بدانند چه مواردی را دارند.

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere?
[Import a repository.](#)

Owner



mtsitoara ▾

/

Repository name *

todo-list



Great repository names are short and memorable. Need inspiration? How about **musical-guacamole**?

Description (optional)

A todo list of my daily tasks



Public

Anyone can see this repository. You choose who can commit.



Private

You choose who can see and commit to this repository.

Skip this step if you're importing an existing repository.

☐ Initialize this repository with a README

This will let you immediately clone the repository to your computer.

Add .gitignore: None ▾

Add a license: None ▾



Create repository

اگر دوست دارید می توانید مخزن را خصوصی کنید. هیچ کس جز شما به آن دسترسی خواهید داشت. یک مخزن عمومی به این معنی نیست که هر کسی می تواند آنرا ویرایش کند. این فقط بدان معنی است که هرکسی می تواند آن را بخواند و وارد شوید ، کاربران می توانند تغییراتی در آن ارائه دهند. شما همچنان نگهدار پروژه و صاحب مخزن خواهید بود.

سپس ، شما می توانید مخزن را با یک فایل README اولیه سازی کنید. اکنون این را نادیده بگیرید زیرا ما قصد داریم یک مخزن از ابتدا ایجاد کنیم. و بعداً پرونده های README ، .gitignore و پروانه را اضافه خواهیم کرد.

پس از تمام شدن کار ، روی دکمه ارسال کلیک کنید تا اولین مخزن GitHub خود را ایجاد کنید! ساده است! سپس به صفحه پروژه خود هدایت می شوید که پیوندی بی نظیر به مخزن شما است. پیوند به صورت زیر است:
`https://github.com/your_username/your_repository`؛ به عنوان مثال ، مخزن جدیدی که من ایجاد کردم از طریق لینک زیر قابل دسترسی است: `https://github.com/mtsitoara/todo-list`. بنابراین ، شما نمی توانید دو مخزن با همین نام ایجاد کنید. صفحه پروژه شما باید مشابه صفحه نمایش داده شده در شکل 4-10 باشد.

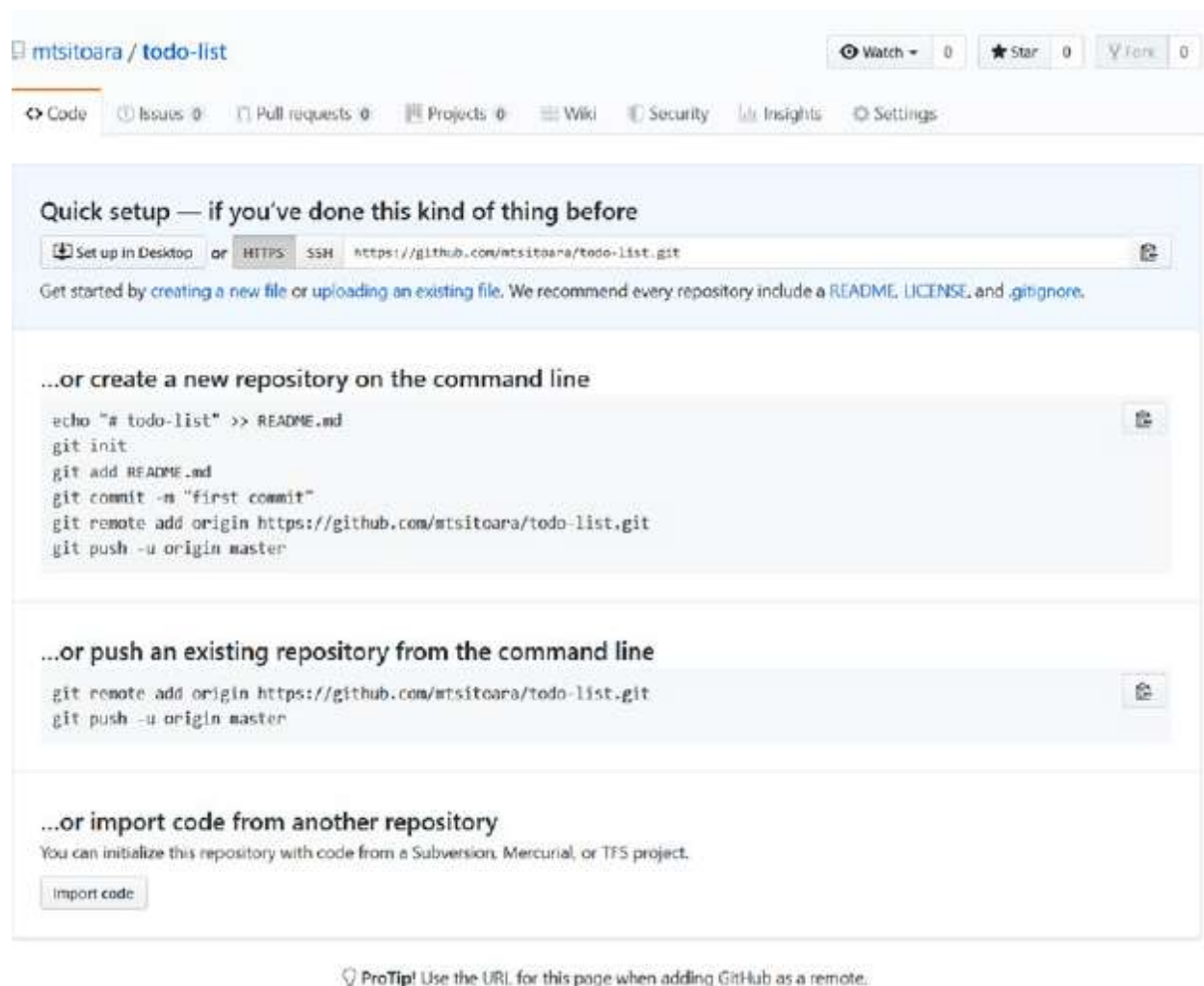


Figure 10-4. Your brand-new repository

همانطور که در شکل 10-4 مشاهده می کنید، دستورالعمل هایی در مورد چگونگی شروع کار وجود دارد که آیا می خواهید یک مخزن جدید ایجاد کنید یا یک موجود موجود را فشار دهید. از آنجا که ما مخازن خود را از ابتدا در حال ساخت هستیم، با گزینه اول پیش خواهیم رفت. گزینه دوم برای ما نیز مؤثر است زیرا ما در حال حاضر یک مخزن محلی داریم، اما می خواهیم از این به بعد نادیده بگیریم.

بنابراین، ما اولین مخزن خود را ایجاد کردیم و آماده هستیم تا طرح خود را تحت فشار قرار دهیم. اما بیا به جعبه جادو پردازیم و ببینیم دقیقاً چه اتفاقی افتاده است.

چگونه مخازن از راه دور کار می کنند

فصل 8 راجع به راه دور Git و اینکه چگونه تصمیم گرفتیم از GitHub به عنوان یک مخزن از راه دور استفاده کنیم ، به خاطر دارید؟ این بخش یک برنامه منطقی از آن فصل است زیرا ما می خواهیم بدانیم که چگونه مخازن از راه دور با GitHub کار می کنند.

وقتی مخزن خود را با استفاده از وب سایت GitHub ایجاد کردیم ، به سرورهای GitHub دستورالعمل می دادیم و از آنها می خواستیم یک مخزن خالی را آغاز کنند. و اگر فصل 3 را به یاد می آورید ، آغاز کردن مخزن بسیار ساده است: به هر فهرست بروید و اجرای `git init` را اجرا کنید. این دقیقاً اتفاقی است که در اینجا افتاده است ، مگر نه در رایانه شما بلکه به سرور میزبانی شده توسط GitHub.

بنابراین ، به نظر می رسد که ما دستورات زیر را در یک سرور دورافتاده که نصب شده `git` را اجرا کردیم.

```
$ mkdir todo-list
```

```
$ cd todo-list
```

```
$ git init
```

این همان دستوراتی است که ما برای ایجاد مخزن محلی خود استفاده خواهیم کرد. بنابراین اکنون ، یک مخزن از راه دور در سرورهای GitHub وجود دارد که ما برای به اشتراک گذاشتن پروژه خود از آنها استفاده خواهیم کرد.

مخازن از راه دور استفاده می شوند بنابراین نیازی نیست که از رایانه شخصی خود برای به اشتراک گذاشتن پروژه خود استفاده کنید. در مورد GitHub ، مخازن از راه دور توسط همه قابل دسترسی است اما فقط مالک می تواند آنها را ویرایش کند. در بخش بعدی در مورد کار گروهی صحبت خواهیم کرد.

راه اصلی خرید این است که یک مخزن از راه دور جایی است که می توانید پروژه خود را منتشر کنید تا آن را در دسترس همه قرار دهد. و هر کس می تواند مخزن شما را کلون کند ، بنابراین می تواند پیشرفت های شما را دنبال کند تا آخرین تغییرات را بدست آورد.

انتشار مخزن محلی شما به یک راه دور "pushing" نامیده می شود و بدست آوردن آخرین تعهدات از یک مخزن از راه دور به محلی ، "pulling" گفته می شود. فشار و کشیدن شاید بیشترین دستوراتی باشد که در Git استفاده خواهید کرد.

اما چگونه می توانم به GitHub بگویم که کدام مخزن از راه دور می خواهم با محلی خود ارتباط داشته باشم؟ اینجاست که به لینک منحصر به فرد به مخزن شما نیاز است. برای فشار آوردن به تغییرات محلی یا ایجاد تعهدی که قبلاً نداشتید ، از پیوند استفاده خواهید کرد.

در پایان ، GitHub یک مخزن از راه دور خالی ایجاد کرد که فقط توسط شما قابل تغییر است اما توسط همه قابل مشاهده است. کاری که اکنون باید انجام دهیم ایجاد یک مخزن محلی و پیوند دادن آن به راه دور است.

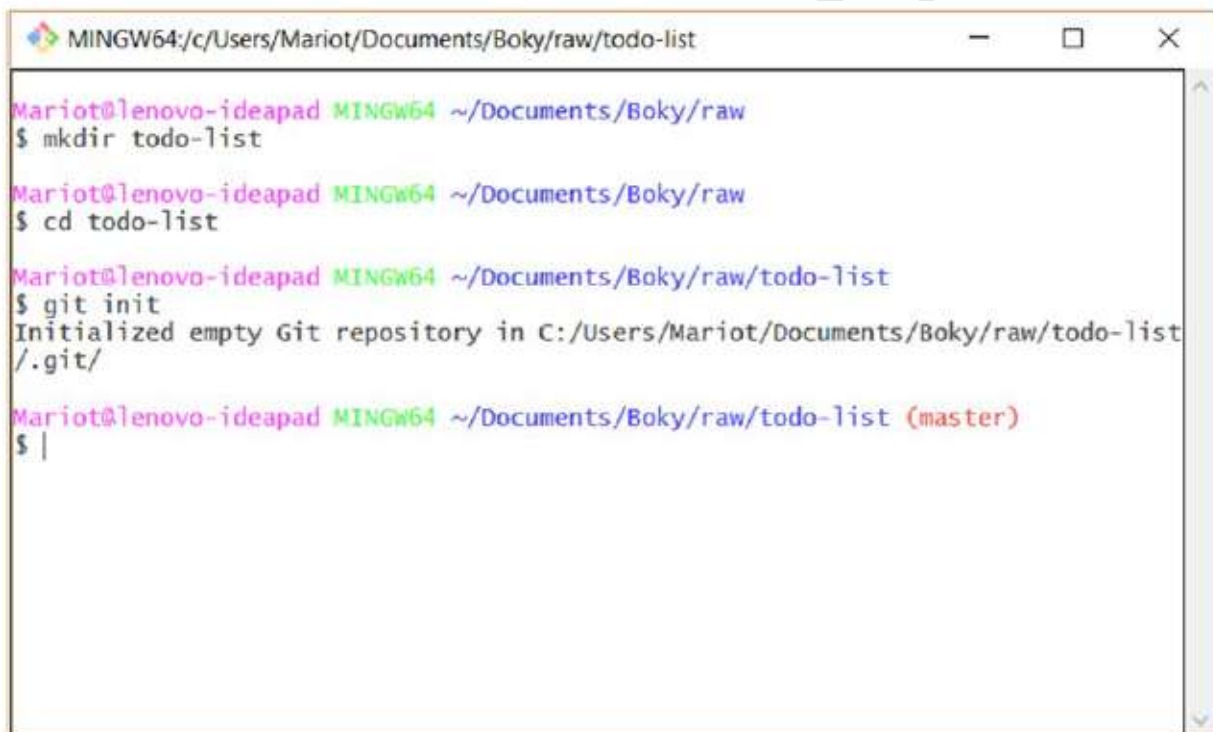
پیوند مخازن

اکنون که **GitHub** مخزن راه دور را برای ما ایجاد کرده است ، زمان آن رسیده که مخزن محلی خودمان را ایجاد کرده و آن را به ریموت وصل کنیم.

همانطور که در فصل های گذشته انجام داده ایم ، می خواهیم با دستور **git init** یک مخزن ایجاد کنیم. نام مخزن می تواند بین محلی و از راه دور متفاوت باشد ، اما این ایده خوبی است که از یک نام منحصر به فرد استفاده کنید تا دچار سردرگمی نشوید. برای این پروژه خاص ، دستورات خواهد بود.

```
$ mkdir todo-list  
$ cd todo-list  
$ git init
```

هیچ چیز جدیدی در اینجا نیست. و شما باید همان نتیجه را در شکل 10-5 دریافت کنید.



```
MINGW64/c:/Users/Mariot/Documents/Boky/raw/todo-list  
Mariot@lenovo-ideapad MINGW64 ~/Documents/Boky/raw  
$ mkdir todo-list  
Mariot@lenovo-ideapad MINGW64 ~/Documents/Boky/raw  
$ cd todo-list  
Mariot@lenovo-ideapad MINGW64 ~/Documents/Boky/raw/todo-list  
$ git init  
Initialized empty Git repository in C:/Users/Mariot/Documents/Boky/raw/todo-list/.git/  
Mariot@lenovo-ideapad MINGW64 ~/Documents/Boky/raw/todo-list (master)  
$ |
```

Figure 10-5. Initialization of a Git repository

اکنون که مخزن محلی خود را داریم ، وقت آن است که آن را به ریموت پیوند دهیم! برای لیست کردن ، اضافه کردن یا حذف Remote ها از دستور `git large` استفاده خواهیم کرد. به عنوان مثال ، بیایید Remote فعلی ما را با استفاده از این دستور پیوند دهیم:

\$ git remote

شما نباید نتیجه ای بدست آورید زیرا این یک مخزن کاملاً جدید است و ما هیچ راه دور را به آن وصل نکرده ایم. بیایید اکنون یکی را اضافه کنیم.

برای اینکه بتوانید یک مخزن محلی را به آن وصل کنید ، به لینک منحصر به فرد به مخزن خود نیاز دارید. بنابراین ، بخش خود را از قسمت قبلی بگیرید. معدن `https://github.com/mtsitoara/todo-list` است.

گیت . پایان را فراموش نکنید! همچنین شما نیاز به ایجاد نام برای مخزن از راه دور خود دارید. به این ترتیب ، می توانید چندین راه دور در یک پروژه واحد داشته باشید. ممکن است در مواردی که آزمایش و کنترل از راه دور تولید برای یکدیگر متفاوت باشد لازم باشد. نام پیش فرض "مبدا" در هر کنوانسیون است. اگرچه می توانید هر نامی را انتخاب کنید ، اما توصیه می شود از مبدا به عنوان نام ریموت در جایی که هم تیمی ها کارهای خود را با یکدیگر به اشتراک می گذارند ، استفاده کنید.

دستور اضافه کردن پیوند به ریموت ساده است. این است

`git remote add [name] [link]`

بنابراین ، برای افزودن پیوندی به مخزن تازه ایجاد شده ، باید این دستور را اجرا کنید:

`$ git remote add origin https://github.com/mtsitoara/todo-list.git`

خودشه! برای به دست آوردن اطلاعات بیشتر می توانید بررسی کنید که آیا ریموت با اجرای `git large` یا `git large-v` اضافه شده است. شما باید نتیجه ای مشابه صفحه نمایش داده شده در شکل 6-10 دریافت کنید.



```
MINGW64:/c/Users/Mariot/Documents/Boky/raw/todo-list
Mariot@lenovo-ideapad MINGW64 ~/Documents/Boky/raw/todo-list (master)
$ git remote add origin https://github.com/mtsitoara/todo-list.git

Mariot@lenovo-ideapad MINGW64 ~/Documents/Boky/raw/todo-list (master)
$ git remote
origin

Mariot@lenovo-ideapad MINGW64 ~/Documents/Boky/raw/todo-list (master)
$ git remote -v
origin https://github.com/mtsitoara/todo-list.git (fetch)
origin https://github.com/mtsitoara/todo-list.git (push)

Mariot@lenovo-ideapad MINGW64 ~/Documents/Boky/raw/todo-list (master)
$ |
```

Figure 10-6. Adding a new remote

و همین اضافه کردن ریموت جدید یک کار ساده و ساده است. اکنون که پاک شدیم ، اجازه دهید پروژه را به سمت **GitHub** سوق دهیم!

هل دادن به مخازن از راه دور

سرانجام مخازن محلی و راه دور خود را پیوند دادیم. وقت آن است که پروژه خود را به سمت **GitHub** سوق دهیم تا بتوانیم کار خود را به اشتراک بگذاریم.

فشار آوردن به مخزن از راه دور بسیار ساده است. اما اول ، بیایید برخی از تعهدات را برای فشار آوردن ایجاد کنیم در فهرست کار خود ، پرونده ای بنام **README.md** ایجاد کنید و توضیحات پروژه خود را در **Markdown** قرار دهید. به عنوان مثال ، در اینجا پرونده **README.md** من است:

```
# TODO list
A simple app to manage your daily tasks
## Features
* List of daily tasks
```

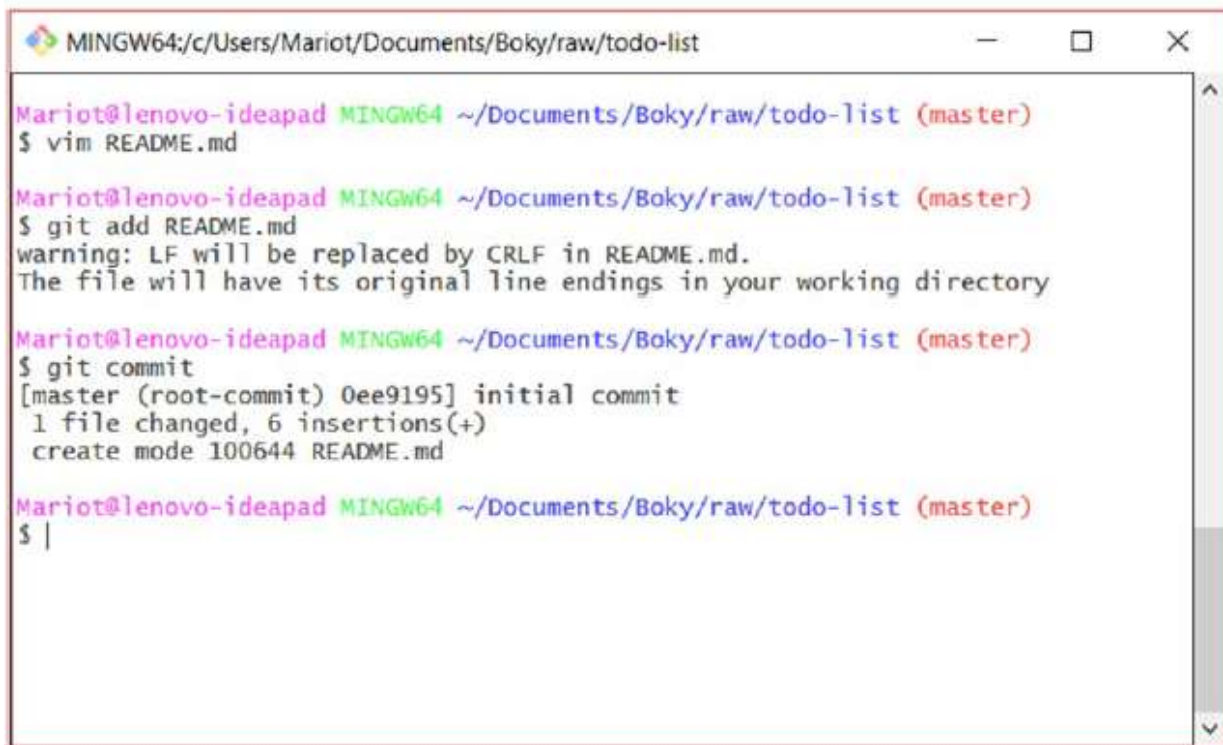
حال ، بیایید با استفاده از افزودن **git** ، پرونده تازه ایجاد شده را به قسمت صفحه اضافه کنیم.

```
$ git add README.md
```

اکنون زمان اجرای پروژه ما با تعهد **git** است. به عنوان یک پیام متعهد ، بسیاری از توسعه دهندگان "اولین تعهد Initial commit" را برای اولین بار انتخاب می کنند. این یک قانون نیست و اگر بخواهید می توانید آن را تغییر دهید.

\$ git commit

از آنجا که قبلاً این کارها را بارها انجام داده ایم ، اکنون باید از صحنه نویسی و commit راحت باشید. پس از commit ، شما باید نتیجه ای مشابه شکل 7-10 داشته باشید.



```
MINGW64:/c:/Users/Mariot/Documents/Boky/raw/todo-list
Mariot@lenovo-ideapad MINGW64 ~/Documents/Boky/raw/todo-list (master)
$ vim README.md

Mariot@lenovo-ideapad MINGW64 ~/Documents/Boky/raw/todo-list (master)
$ git add README.md
warning: LF will be replaced by CRLF in README.md.
The file will have its original line endings in your working directory

Mariot@lenovo-ideapad MINGW64 ~/Documents/Boky/raw/todo-list (master)
$ git commit
[master (root-commit) 0ee9195] initial commit
1 file changed, 6 insertions(+)
create mode 100644 README.md

Mariot@lenovo-ideapad MINGW64 ~/Documents/Boky/raw/todo-list (master)
$ |
```

Figure 10-7. Creating, staging of a new file

بنابراین ، ما اولین تعهد خود را انجام داده ایم! اکنون می توانیم آن تغییرات را به مخزن راه دور منتقل کنیم. دستور فشار دادن تغییرات به ریموت ساده است؛ شما فقط نیاز به نام مخزن از راه دور و شعبه مورد فشار قرار دارید. از آنجا که ما هنوز هیچ شعبی را ایجاد نکرده ایم (در بخش بعدی در مورد شاخه ها خواهیم آموخت) ، تنها شعبه ما "استاد" خوانده می شود. دستور git push است

git push <remote_name> <branch_name>

So, in our case, the command will be

\$ git push origin master

با کمی شانس ، همه چیز خوب پیش می رود؛ اما همیشه اینطور نیست. اگر از مدیر رمز عبور استفاده می کنید یا از آنهایی که به GitHub ارائه داده اید از پیکربندی های مختلف (نام و ایمیل) استفاده می کنید ، یک مشکل احراز هویت دریافت خواهید کرد. به عنوان مثال ، من از دسترسی به مخزن خود محروم می شوم ، زیرا من از یک مدیر رمز عبور استفاده کردم و سعی کردم مرا با اعتبار قدیمی خود وارد سیستم کنم. می توانید نمونه ای از خطای تأیید اعتبار را در شکل 10-8 بررسی کنید.

A screenshot of a terminal window titled 'MINGW64:/c/Users/Mariot/Documents/Boky/raw/todo-list'. The prompt is 'Mariot@lenovo-ideapad MINGW64 ~/Documents/Boky/raw/todo-list (master)'. The user enters '\$ git push origin master'. The output shows a remote permission denial: 'remote: Permission to mtsitoara/todo-list.git denied to mariot.' followed by a fatal error: 'fatal: unable to access 'https://github.com/mtsitoara/todo-list.git/': The requested URL returned error: 403'. The prompt returns to '\$ |'.

Figure 10-8. Authentication error

برای برطرف کردن این نوع مشکلات ، مجدداً Git را با اطلاعات صحیح پیکربندی کرده ایم. می توانید در شکل 10-9 مشاهده کنید که من ایمیل خود را در پیکربندی های جهانی یعنی هر مخزن موجود در رایانه خود تغییر داده ام.

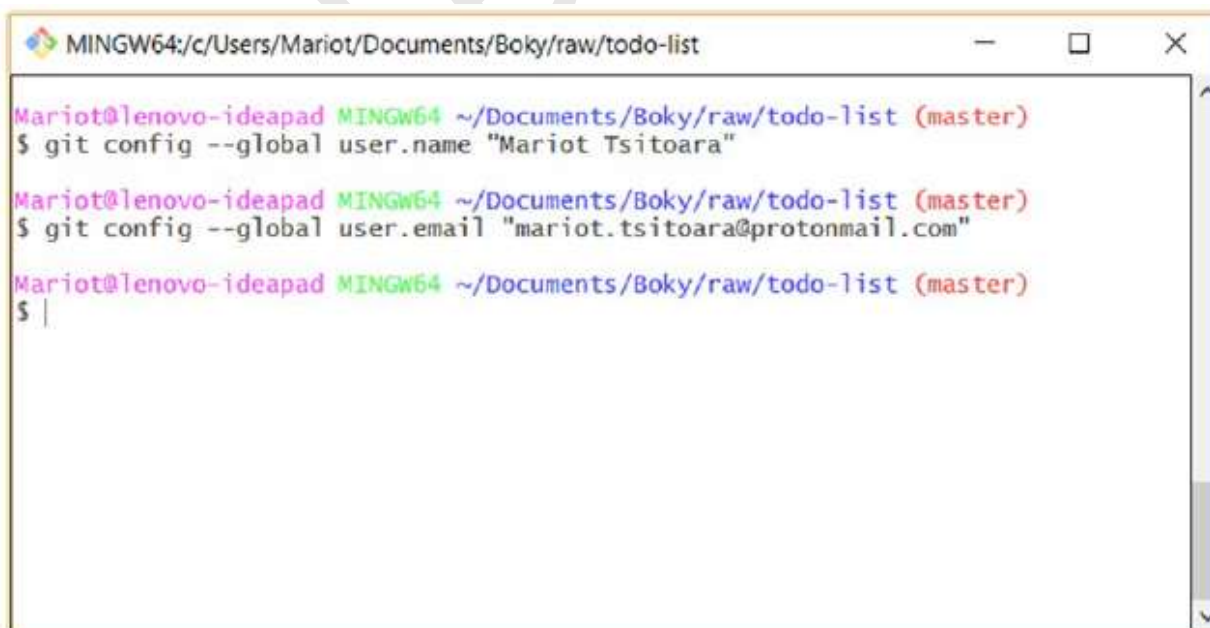
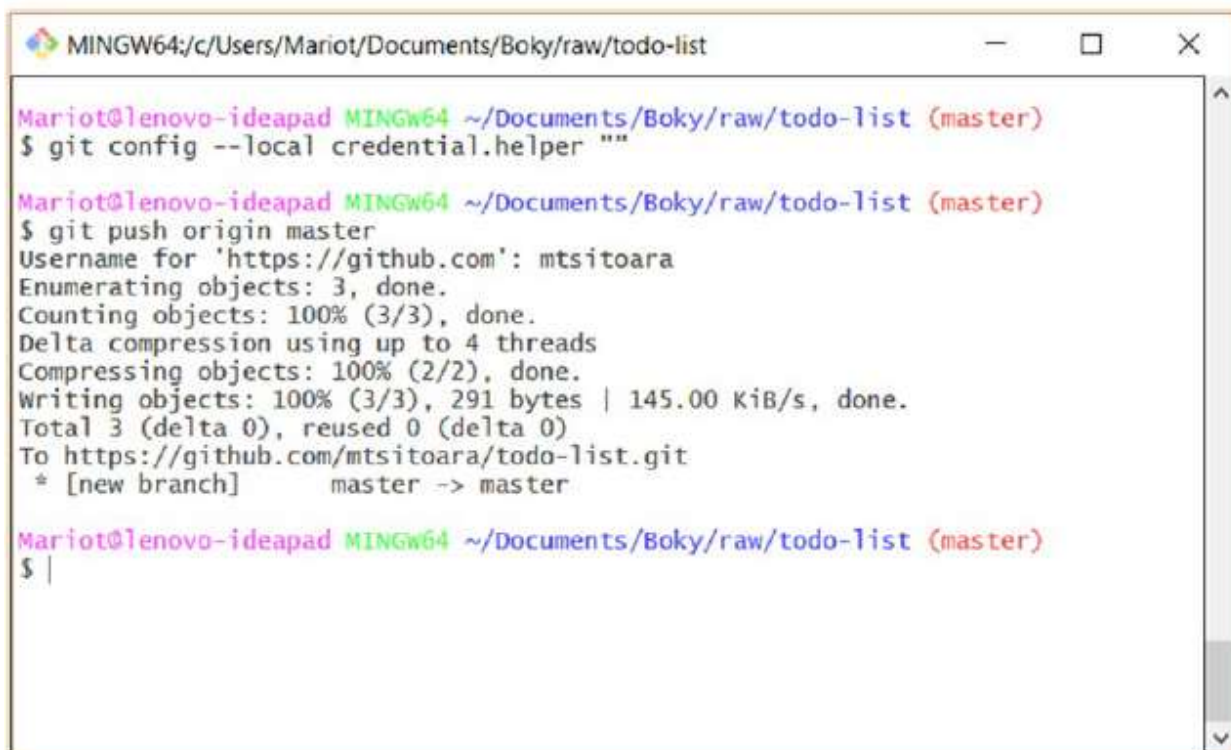
A screenshot of a terminal window titled 'MINGW64:/c/Users/Mariot/Documents/Boky/raw/todo-list'. The prompt is 'Mariot@lenovo-ideapad MINGW64 ~/Documents/Boky/raw/todo-list (master)'. The user enters '\$ git config --global user.name "Mariot Tsitoara"'. The prompt returns to '\$ |'. The user then enters '\$ git config --global user.email "mariot.tsitoara@protonmail.com"'. The prompt returns to '\$ |'.

Figure 10-9. Reconfiguration of Git

حال ، ما باید مطمئن شویم که هر پیوندی را برای مدیر رمز عبور در این مخزن حذف می کنیم. در مورد من ، از یاران معتبر (مدیران گذرواژه) در مخازن دیگر این رایانه استفاده می کنم. بنابراین من پیکربندی جهانی را تنظیم نمی کنم بلکه محلی را تنظیم می کنم.

\$ git config --local credential.helper ""

این باید مشکل ما را برطرف کند و ما می توانیم فشار خود را از سر بگیریم. پس از اجرای دستور git push ، از شما نام کاربری و رمز عبور خود را از شما سؤال می کند. سپس نتیجه ای مشابه شکل 10-10 دریافت خواهید کرد.



```
MINGW64:/c:/Users/Mariot/Documents/Boky/raw/todo-list
Mariot@lenovo-ideapad MINGW64 ~/Documents/Boky/raw/todo-list (master)
$ git config --local credential.helper ""

Mariot@lenovo-ideapad MINGW64 ~/Documents/Boky/raw/todo-list (master)
$ git push origin master
Username for 'https://github.com': mtsitoara
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 291 bytes | 145.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/mtsitoara/todo-list.git
 * [new branch]      master -> master

Mariot@lenovo-ideapad MINGW64 ~/Documents/Boky/raw/todo-list (master)
$ |
```

Figure 10-10. Successful git push

اکنون ، پروژه ما در GitHub توسط همه قابل مشاهده است! بیایید آن را در صفحه پروژه خود بررسی کنیم. اگر صفحه پروژه را تازه کنیم ، باید صفحه ای مانند صفحه نمایش داده شده در شکل 10-11 دریافت کنیم.

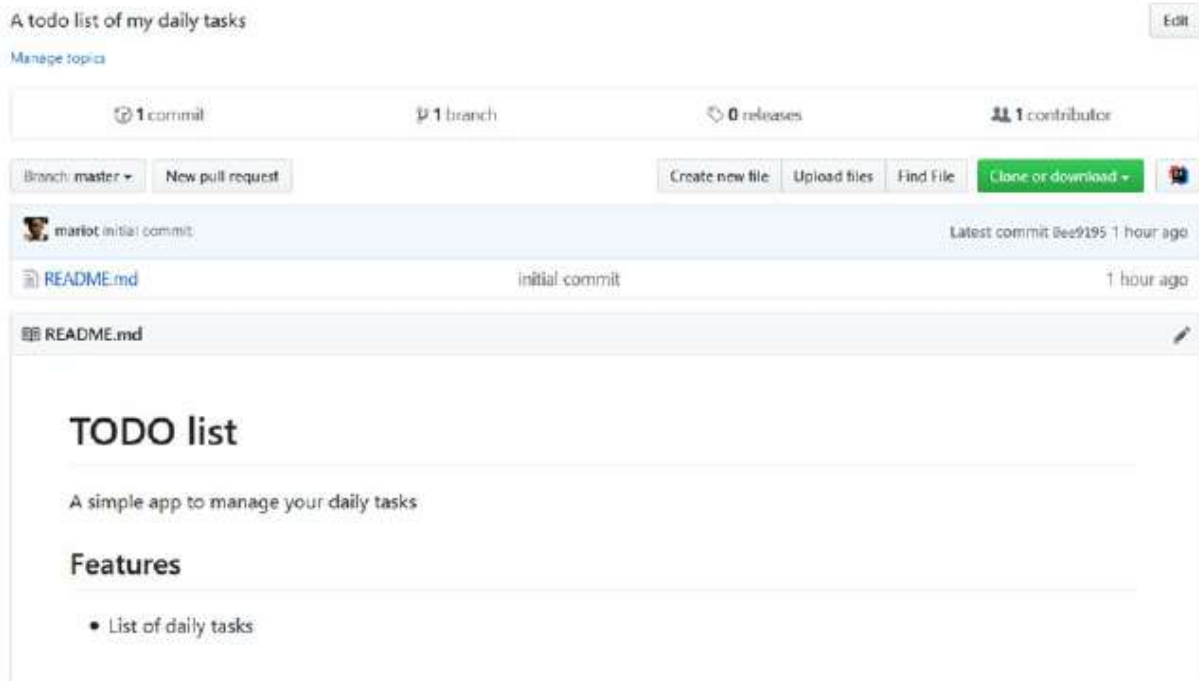


Figure 10-11. The updated project page

همانطور که در شکل 10-11 مشاهده می کنید ، صفحه مخزن اکنون بسیاری از Intel ها را نشان می دهد:

- تعداد تعهدات
- آخرین نام متعهد و مرتکب آن
- لیستی از تمام پرونده های پروژه
- پیش نمایش README.md

آنچه ما فقط انجام دادیم اساس اشتراک کد است: فشار بر تغییرات. شما هنگام کار با مخازن از راه دور بارها و بارها از این دستور استفاده خواهید کرد. این یک ویژگی بسیار ساده است ، اما ضروری است که کاملاً بفهمید که چه کاری انجام می دهد. فشار دادن فقط به معنای کپی کردن تمام تعهدات فعلی شما (در یک شعبه خاص) در یک شعبه از راه دور در یک مخزن از راه دور است. تمام پرونده های تاریخ نیز کپی می شوند.

قبل از اینکه به فصل بعد بروید ، این سؤال ها را از خود بپرسید: مخازن از راه دور کجا ذخیره می شوند؟ چه کسی فقط خواندن دسترسی به آنها دارد؟ چه کسی می تواند آنها را ویرایش کند؟ همچنین مطمئن شوید که اساس مخازن از راه دور و محلی رابط و دلیل ضروری بودن آن را درک کرده اید.

در این فصل ، ما اولین تعامل خود را با Gitrepositories از راه دور داشتیم. همانطور که قبلاً تأسیس کرده ایم ، آنها فقط مخازن معمولی هستند که به جای دستگاه محلی شما در یک سرور از راه دور ذخیره می شوند. ما دیدیم که چگونه می توان مخازن محلی و راه دور را ایجاد کرد و پیوند داد ، ویژگی ای که بارها از آن استفاده خواهیم کرد. و فرمان اصلی که آموخته ایم git push بود که وضعیت مخزن محلی شما را به یک دور دست کپی می کرد.

در فصل بعد ، قصد داریم عمیق به مدیریت پروژه برویم و ببینیم سایر ویژگی های GitHub چه چیزی را ارائه می دهد. ما همچنین یاد خواهیم گرفت که تغییرات را از مخزن از راه دور و همچنین مشکلات فشار و کشش را برطرف کنیم. بیا بریم!

فصل یازدهم

شروع پروژه

مشکلات مدیریتی

فصل گذشته ، ما با استفاده سریع از GitHub برای میزبانی و به اشتراک گذاری کد خود ، نگاهی سریع به عمل آوردیم. اما این حتی نمی تواند توصیف کند که GitHub چه کاری برای شما می تواند انجام دهد. ویژگی های بسیاری وجود دارد که می تواند به پروژه شما در بالغ شدن کمک کند. در این فصل قصد داریم تا درباره نحوه مدیریت پروژه ها با GitHub یاد بگیریم. بنابراین ، ما در ابتدا به شکل اصلی مدیریت پروژه GitHub می پردازیم: مسائل.

مروری بر موضوعات

برای مدیریت موفقیت آمیز یک پروژه ، هر پروژه ، باید از قبل برنامه ریزی کنید. فقط نسبت به ورودی های جدید واکنش نشان دهید و به طور کلی هر کاری را که احساس می کنید انجام دهید یک دستور العمل مناسب برای فاجعه است.

یک پروژه GitHub فرقی نمی کند؛ شما باید قبل از فکر کردن در مورد انجام آنها ، اقدامات خود را پیگیری کنید. به همین دلیل GitHub از ویژگی های بسیار جذاب به نام Issues برخوردار است. ما در این بخش قصد داریم در مورد آنها بحث کنیم و یاد بگیریم که چگونه آنها را به درستی مدیریت کنیم.

در تمام فصل های این کتاب ، شما هم توسعه دهنده هستید و هم مدیر پروژه. اما در یک پروژه بزرگ ، ممکن است شما در مراحل برنامه ریزی گنجانده نشوید. اما در حال حاضر ، شما به طور موقت به مدیر پروژه و توسعه دهنده پیشرو (علاوه بر اینکه تنها توسعه دهنده آن هستید) تبلیغ می شوید ، تبریک می گویم! یکی از وظایف مدیر پروژه این است که در پیشبرد تمام کارهایی که باید انجام شود برنامه ریزی کند. برنامه ها لازم نیست بسیار دقیق باشند (در دنیای واقعی آنها هرگز نباشند) ، اما لازم است لیستی از کلیه کارهایی که باید انجام شود وجود داشته باشد. این وظایف می تواند یا ویژگی های جدید ، رفع اشکال یا فقط بحث تیم باشد. در GitHub به آن وظایف Issues گفته می شود.

از یک شماره برای ردیابی توسعه ویژگی های جدید ، رفع اشکال یا ایده های جدیدی که یک عضو تیم پیشنهاد کرده استفاده می شود. آنها آجر و ملات مدیریت پروژه GitHub هستند. از نظر تئوری ، هیچ کاری نباید انجام شود با موضوعی که به آن وصل شود. هدف هر عملی که انجام دهید باید حل مسئله باشد.

روزهایی که می خواستند برنامه های بعدی توسط جلسات خسته کننده تیم انجام شود ، گذشته است. اکنون شما دقیقاً می دانید مراحل بعدی شما چه خواهد بود و مهمتر از همه اینکه دیگران چه می کنند. پیشنهاد ایده های جدید به همکارانتان آسان تر از همیشه است؛ فقط یک مسئله را باز کنید تا بدون استفاده از برنامه یا مشتری دیگر ایمیل ، در مورد آن با تیم خود صحبت کنید. بزرگترین امتیاز استفاده از موضوعات این است که تاریخ برای همیشه حفظ می شود - هر ویژگی ، هر اشکال و هر بحث.

ایجاد شماره

بهترین راه برای یادگیری درمورد موضوعات ، تعامل مستقیم با آنهاست. بنابراین برگردیم به صفحه پروژه GitHub و برخورد با آنها. هنگامی که صفحه پروژه GitHub خود را باز می کنید ، مستقیماً وارد قسمت "کد" پروژه می شوید. این بخشی است که پرونده های پروژه شما نشان داده می شود. در حال حاضر ، صفحه پروژه شما باید مانند من باشد ، که در شکل 11-1 نشان داده شده است.

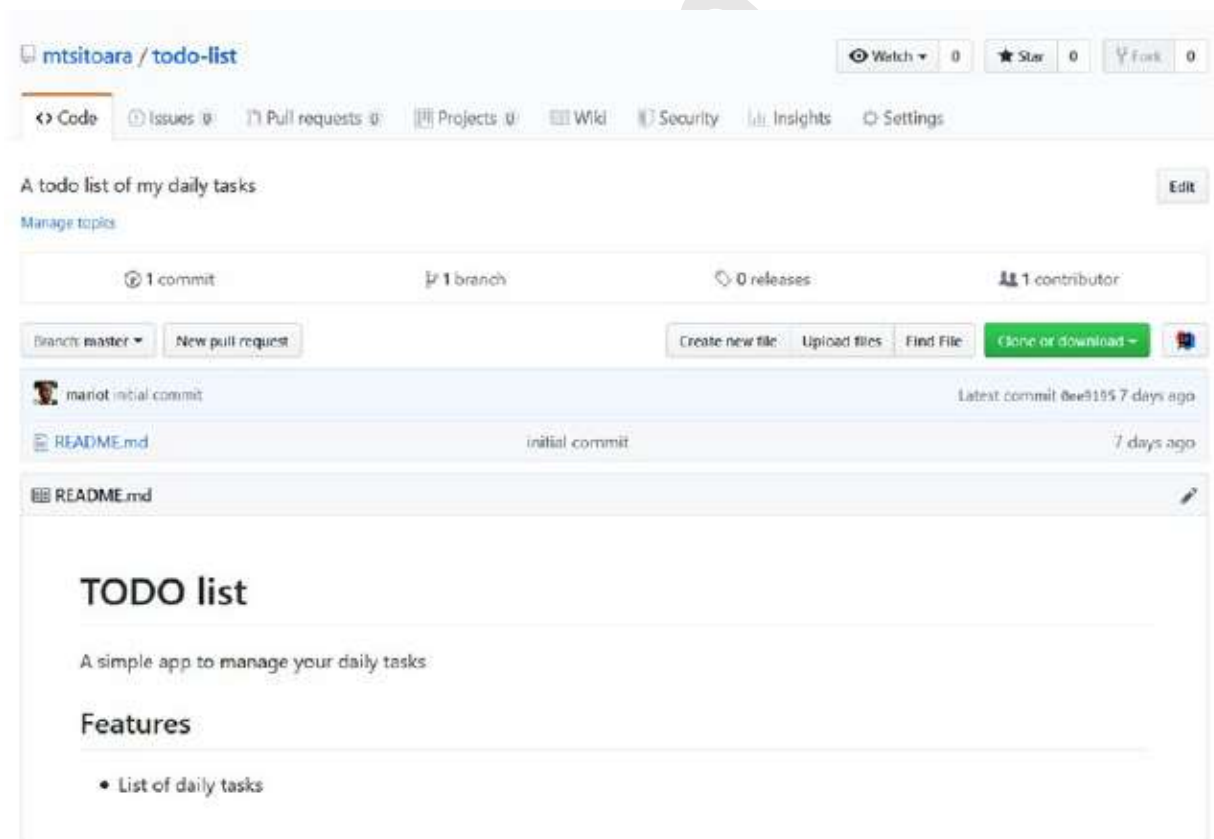


Figure 11-1. Project page open on the “code” section

درست زیر نام پروژه ، برگه های بسیاری وجود دارد که تمام بخش های پروژه شما را نشان می دهد. شما اکثراً روی "کد" ، "مسائل" ، "درخواست های بیرون کشیدن" و "پروژه ها" کار خواهید کرد. اما فعلاً ، بیایید به موضوع توجه کنیم. برای شروع پیش بروید و روی آن کلیک کنید شما باید به یک قسمت خالی مانند آنچه در شکل 11-2 نشان داده شده است برسید زیرا پروژه شما هنوز هیچ مشکلی ندارد.

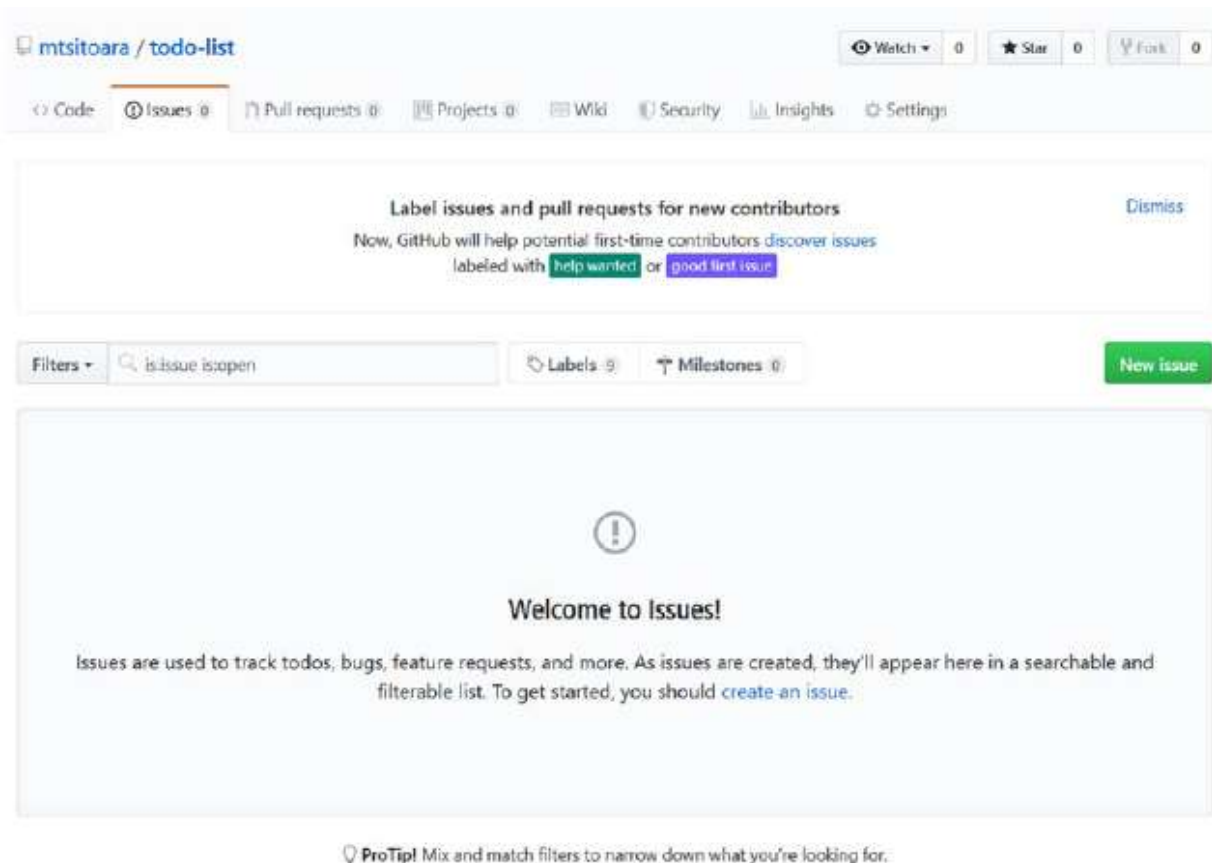


Figure 11-2. The Issues section

در مورد ایجاد موضوع جدید فراخوان های زیادی برای اقدام وجود دارد. یکی از آنها را کلیک کنید ، و فرم مشابه با من را مشاهده می کنید همانطور که در شکل 11-3 نشان داده شده است.

The image shows the GitHub interface for creating a new issue in the repository 'mtsitoara/todo-list'. At the top, there are navigation links for Code, Issues (selected), Pull requests, Projects, Wiki, Security, Insights, and Settings. Below the repository name, there are buttons for Watch, Star, and Fork. The main form has a 'Title' input field, a rich text editor with 'Write' and 'Preview' tabs, and a 'Leave a comment' placeholder. To the right of the form, there are dropdown menus for Assignees (No one—assign yourself), Labels (None yet), Projects (None yet), and Milestone (No milestone). At the bottom right, there is a green 'Submit new issue' button. A note at the bottom left states 'Styling with Markdown is supported'.

Figure 11-3. New issue form

فرم بسیار ساده است. و فقط عنوان الزامی است در صورت نیاز به فضای بیشتر برای توضیح، در زیر عنوان نیز یک بخش نظر وجود دارد. بیا بید جلو برویم و اولین شماره ما را با موارد اساسی پر کنیم؛ هنوز مقادیر را در سمت راست تغییر ندهید. برای اولین شماره، ما شروع به بحث در مورد فن آوری مورد استفاده برای محصول خود می کنیم. مشکلات فقط به ویژگی ها و ردیابی اشکال ها لازم نیستند. آنها همچنین برای شروع بحث و به اشتراک گذاشتن ایده ها استفاده می شوند. پیش بروید و اولین شماره خود را مانند موارد زیر که در شکل 11-4 نشان داده شده است، پر کنید. من عنوان خودم را انتخاب کردم "فناوریهای را انتخاب کنید که برای برنامه استفاده شود" زیرا این اولین گام برای هر پروژه است.

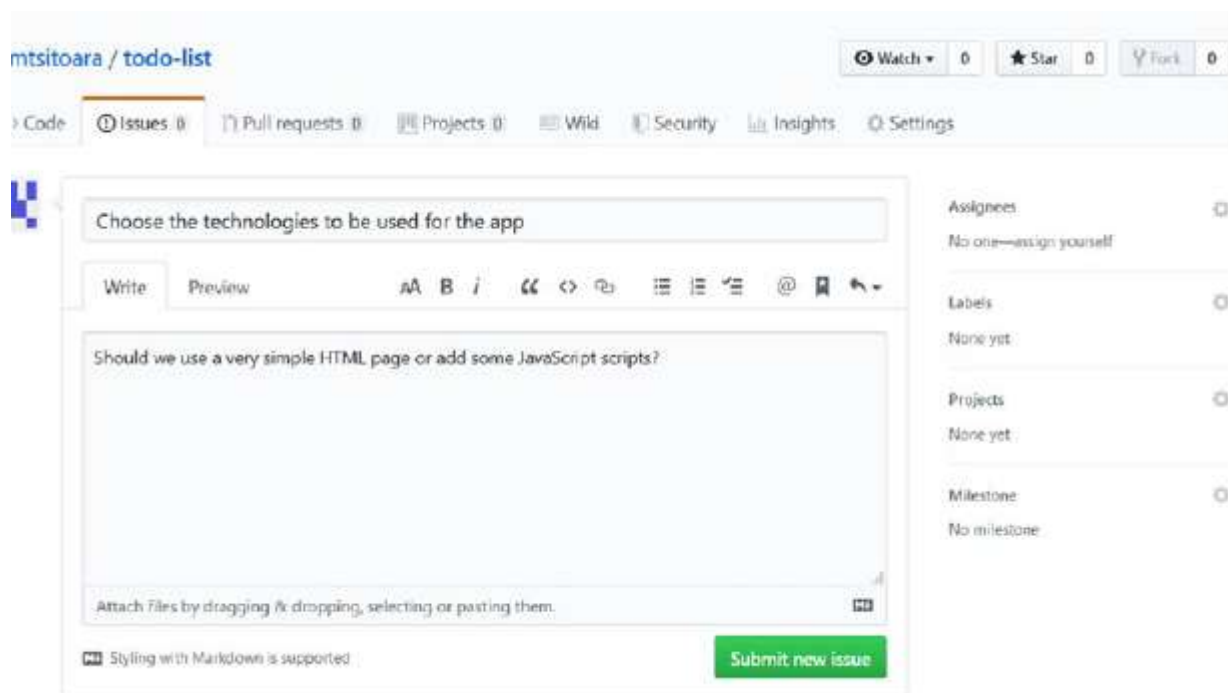


Figure 11-4. Our first issue

اکنون که اطلاعات اصلی در مورد مسئله را پر کردیم ، آنرا ارسال کنید. سپس به نمای تفصیلی شماره جدید خود هدایت می شوید. باید مشابه مسئله من باشد که در شکل 5-11 نشان داده شده است.



Figure 11-5. Details of an issue

اولین چیزی که باید توجه کنید این است که شماره شما داده شده است. هر شماره یک شماره منحصر به فرد دارد و آن شماره ها بازیافت نمی شوند ، به این معنی که حتی اگر یک شماره را حذف کنید ، هرگز از شماره آن استفاده مجدد نمی شود. همانطور که در این بخش مشاهده خواهید کرد این شماره دارای اهمیت است.

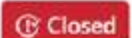
صفحه جزئیات همچنین شامل یک بخش نظرات است که اعضای تیم می توانند در مورد ایده بحث کنند. حتی تعداد محدودی از ایموجی ها را شامل می شود که می توانید از آنها به عنوان جایگزین برای اظهار نظر استفاده کنید. برای مثال ، اگر با کسی موافق باشید ، دادن شست به آنها بهتر از اظهار نظر یا نوشتن "من" نیز هست! این ارتباطات را مسدود کرده و مکالمه را متوقف می کند.

در سمت راست پایین صفحه ، می توانید دکمه اشتراک را ببینید. اگر تصمیم به عضویت در یک شماره گرفتید ، اعلان هایی درباره تغییرات انجام شده در آن دریافت خواهید کرد. همچنین نظرات و اخبار جدیدی درباره نقاط عطف به دست آمده دریافت خواهید کرد.

از آنجا که شما تنها عضو تیم هستید ، بحث زیادی نخواهید کرد. فقط یک نظر یا یک عکس العمل اضافه کنید و مسئله را ببندید. بسته شدن این مسئله آن را حذف نمی کند؛ فقط آن را به عنوان تکمیل علامت گذاری می کند. حذف مسائل توصیه نمی شود زیرا به سابقه تاریخی پروژه نیاز است و موضوعات بهترین راه برای پیگیری تغییرات است. و به یاد داشته باشید: اگر مخزن شما عمومی است ، هر کسی می تواند نظرات شما را بخواند. بنابراین لطفاً مهربان باشید و هرگونه ناخوشایندی را که ممکن است بوجود بیاید ، بپیچید.

پس از اظهار نظر و بسته شدن موضوع ، به صفحه جزئیات شماره باز خواهید گشت و همانطور که در شکل 11-6 نشان داده شده است ، به من شبیه خواهد بود.

Choose the technologies to be used for the app #1

 Closed mtsitoara opened this issue 18 minutes ago · 1 comment



mtsitoara commented 18 minutes ago

Owner



Should we use a very simple HTML page or add some JavaScript scripts?



mtsitoara commented now

Author

Owner



No need to add JS just yet. Let's just stick to HTML5 and some styles.

@mtsitoara will lead the development



1



mtsitoara closed this now

Figure 11-6. A closed issue

شما می توانید در مورد یک موضوع بسته اظهار نظر کنید ، اما از آنجایی که همه موضوع را کامل و ادامه داده اند ، دلسرد می شود. یک مسئله همچنین می تواند قفل شود و هیچ کس دیگر نمی تواند درباره آن اظهار نظر کند. این به عنوان آخرین روش تلاش برای حفظ صلح در نظر گرفته شده است. همه ما نظرات خود را داریم و بحث در مورد آنها هیچگاه آسان نیست ، خصوصاً در یک انجمن آزاد. اما سعی کنید همیشه حرفه ای باشید زیرا هر آنچه می گوئید برای همه قابل مشاهده خواهد بود.

تعامل با یک مسئله

ما مسئله ای را با موفقیت ایجاد کرده و بسته ایم ، اما زیاد درگیر آنها نبوده ایم. اگر تأثیری در پروژه نداشته باشد ، چه فایده ای دارد؟ در این بخش ، ما به طور مستقیم با موضوعات مربوط به GitHub و در کد خود ارتباط برقرار خواهیم کرد.

برای قسمت اول این بخش ، شما کلاه پروژه Project خود را نگه دارید زیرا ما قصد داریم پروژه خود را برنامه ریزی کنیم. تا این لحظه ، برنامه لیست TODO ما فقط چندین فایل متنی در کنار یکدیگر بود. سپس تصمیم گرفتیم که از HTML5 استفاده کنیم تا آنها را به روشی بهتر ارائه کنیم. برای کدگذاری این ، ما به یک برنامه عمل نیاز داریم. و این وظیفه شما به عنوان ProjectManager است که این طرح را تهیه کنید.

از آنجایی که این یک برنامه HTML5 ساده است ، ما به یک برنامه خیلی بزرگ نیاز نخواهیم داشت ، فقط به برخی از نقاط مهم گلوله نیاز دارید ، بنابراین ، برای ایجاد این برنامه ، ما باید

- اسکلت برنامه را با HTML5 بنویسید

- برخی از سبک ها را اضافه کنید تا با CSS3 زیباتر شود

- برنامه را در README.md توصیف کنید

- کد را مستند کنید

- ایجاد یک صفحه وب برای برنامه

اینها مراحل اساسی است که برای تحقق هدف خود باید انجام دهیم: ارسال یک برنامه TODO.

از آنجا که شما از قبل می دانید چگونه می توانید مسائل ایجاد کنید ، من به شما امکان می دهم برای هر یک از این نقاط گلوله مسئله ایجاد کنید. بعد از اتمام کار ، صفحه شماره شما باید مانند من باشد ، همانطور که در شکل 11-7 نشان داده شده است.

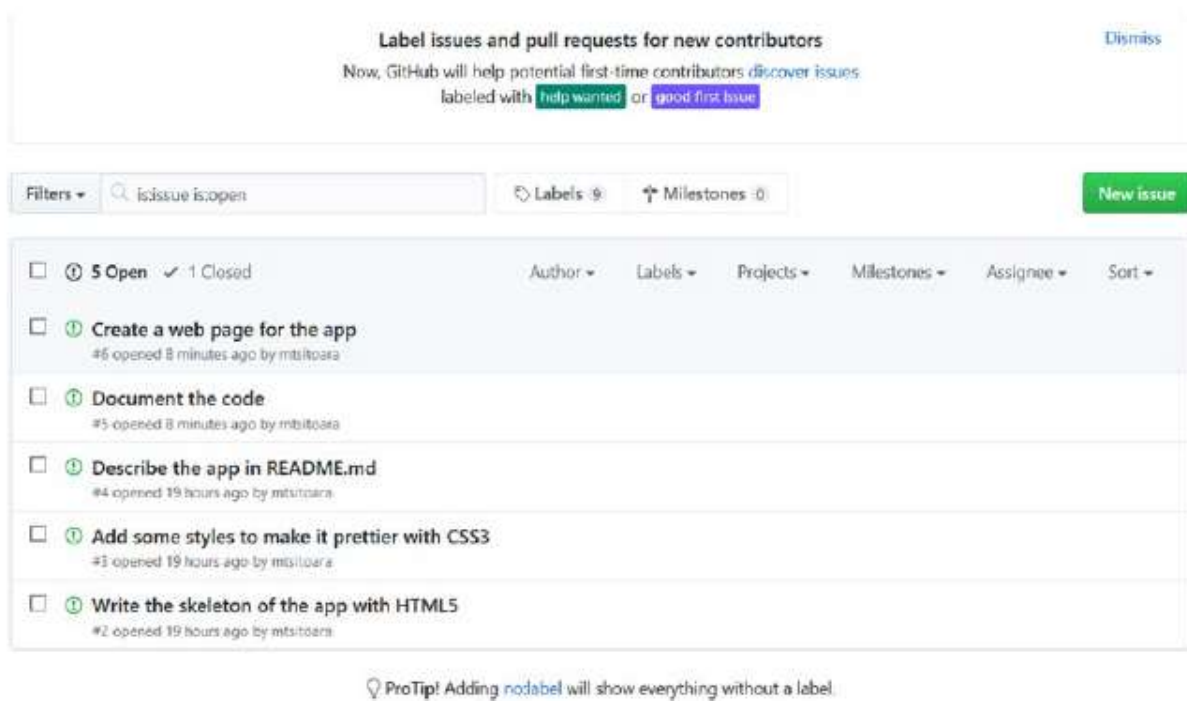


Figure 11-7. All open tasks

همانطور که مشاهده می کنید، وظایف به روشی که معرفی شده اند نشان داده می شود. به جز تعداد آنها، هیچ راهی برای تشخیص آنها وجود ندارد، و در صورت وجود مشکلات زیاد، گم شدن بسیار آسان است. بنابراین، برای داشتن یک نگاه واضح تر به تمام وظایف خود، قصد داریم از برچسب ها استفاده کنیم.

برچسب ها

برچسب ها دقیقاً همان چیزی هستند که انتظار دارید از آنها باشد: متن هایی که به شما کمک می کنند تا به سرعت در میان موضوعات خود فیلتر کنید. بیایید مستقیماً از آنها استفاده کنیم تا بتوانید با مفهوم آن آشنا شوید.

همانطور که در شکل 11-7 مشاهده می کنید، یک نوار جستجو در صفحه Issues وجود دارد و می توانید از آن برای فیلتر کردن مشکلات استفاده کنید. اما از آنجایی که ما هنوز هیچ برچسبی نداریم، نمی توانیم فیلترگذاری کنیم. فقط جستجوی اولیه برای نشان دادن تمام برچسب های موجود روی دکمه برچسب ها در کنار نوار جستجو کلیک کنید. سپس لیستی از برچسب های پیش فرض را که می توانید استفاده کنید، مشاهده خواهید کرد. شکل 11-8 را برای نمونه ای از این موضوع بررسی کنید.

Labels	Milestones	Search all labels	New label
9 labels		Sort ▾	
bug	Something isn't working	Edit	Delete
documentation	Improvements or additions to documentation	Edit	Delete
duplicate	This issue or pull request already exists	Edit	Delete
enhancement	New feature or request	Edit	Delete
good first issue	Good for newcomers	Edit	Delete
help wanted	Extra attention is needed	Edit	Delete
invalid	This doesn't seem right	Edit	Delete
question	Further information is requested	Edit	Delete
wontfix	This will not be worked on	Edit	Delete

Figure 11-8. List of the default labels

اینها برچسبهای متداول در جامعه توسعه دهندگان هستند. اما این بدان معنی نیست که اجباری یا تغییر ناپذیر باشند. شما می توانید آنها را در خواست و نیاز خود تغییر دهید.

فقط هنگامی که روی یک پروژه منبع باز کار می کنید، توصیه نمی شود که آنها را تغییر دهید زیرا اکثر توسعه دهندگان به آنها عادت دارند.

اما از آنجایی که این پروژه شخصی شماست و شما مدیر پروژه هستید، می توانید هر برچسب مورد نظر خود را اضافه کنید، ویرایش یا حذف کنید به عنوان مثال، اگر تنها در یک محیط خصوصی کار کنید، برچسب "help kërkuar" بی فایده خواهد بود. در صورت جدی بودن مشکل، می توانید از برچسب ها برای نشان دادن شدت مسئله استفاده کنید، بسیاری از پروژه ها از برچسب هایی مانند "فوری" یا "شکستن" استفاده می کنند. اگر پروژه به اندازه کافی بزرگ باشد، می توانید از لیبل ها برای تمایز منشأ مسئله استفاده کنید. یک پروژه بزرگ می تواند از برچسب های "backend، frontend" یا "پایگاه داده" برای جدا کردن موضوعات به گروه استفاده کند.

بعد از اینکه تغییرات خود را روی برچسب ها انجام دادید (اگرچه توصیه می کنم فقط موارد جدید مورد نیاز خود را اضافه کرده و موارد پیش فرض را ترک کنید)، دوباره به مشکلات خود برگردید و صفحه جزئیات را باز کنید. سپس با کلیک بر روی دکمه Labels یک یا چند برچسب را روی هر یک از آنها اعمال کنید. برای مثال می توانید شکل 9-10 را بررسی کنید.

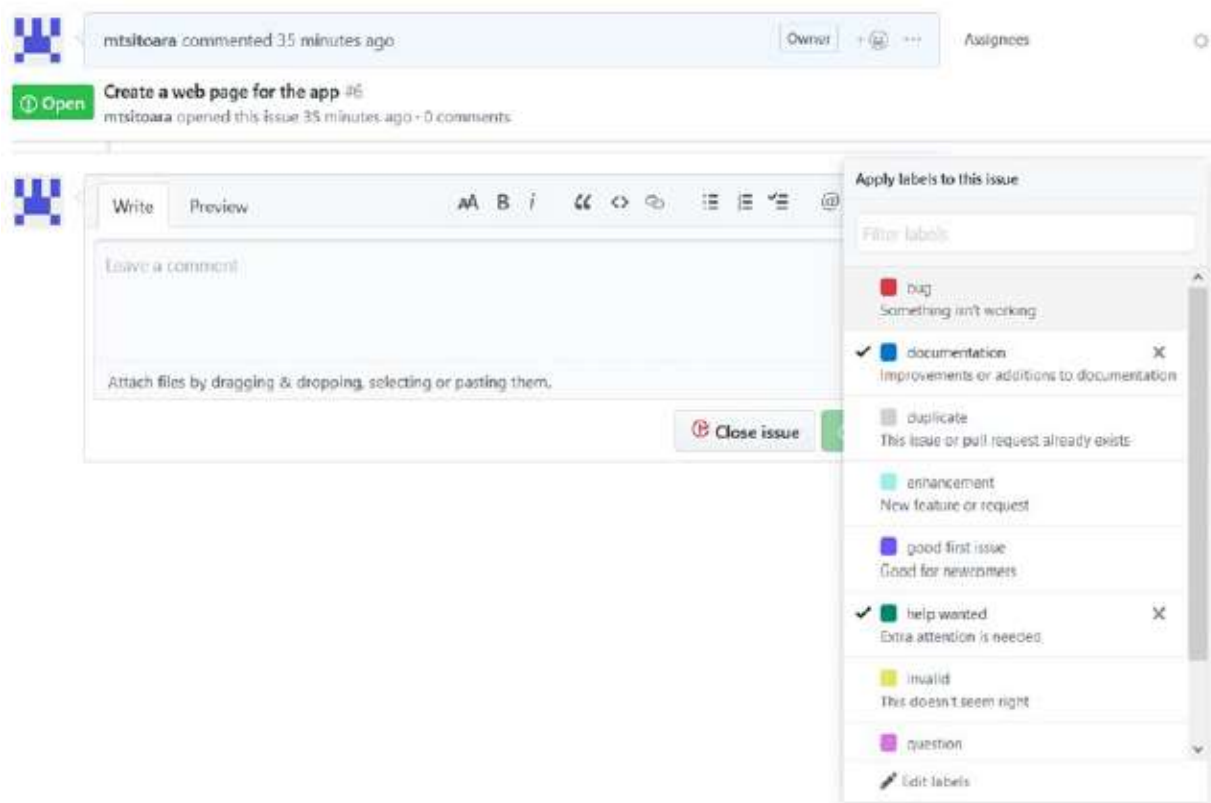


Figure 11-9. Adding a label to an issue

پس از افزودن برچسب ها ، یک اعلان در قسمت نظرات صفحه Issues ظاهر می شود. برای مثال می توانید شکل 10-11 را بررسی کنید.

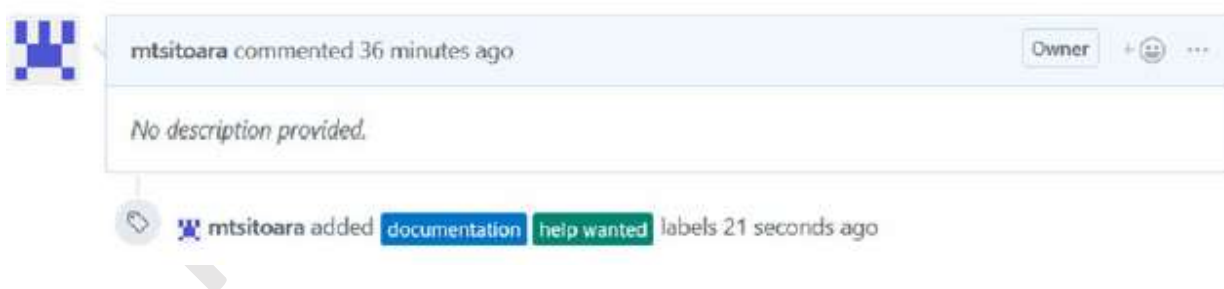
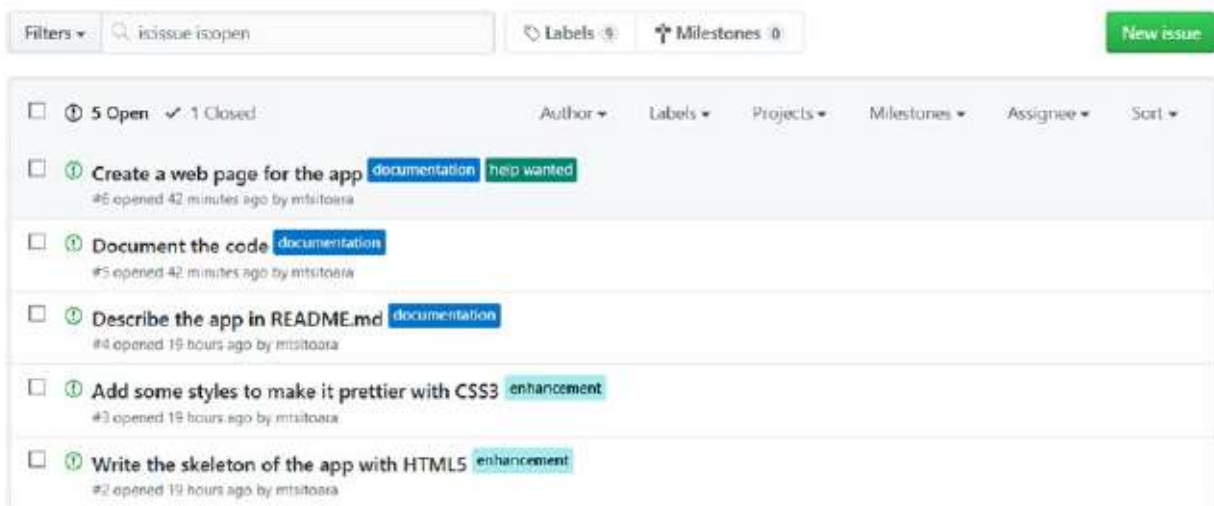


Figure 11-10. Notification about the newly added labels

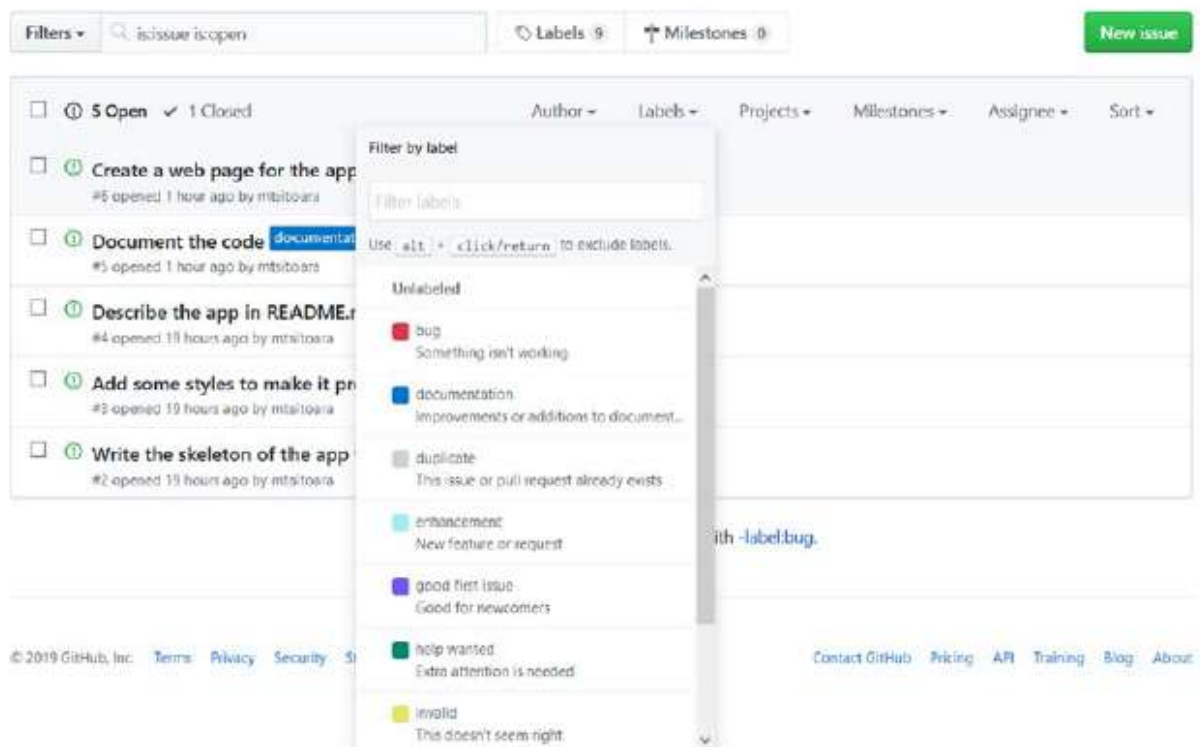
اکنون ، هر یک از مشکلات خود را طی کرده و برچسب های خود را بر روی آنها اعمال کنید. سپس ، پس از اتمام ، به صفحه Issues برگردید. باید مانند من باشد ، همانطور که در شکل 11-11 نشان داده شده است.



ProTip! Exclude everything labeled bug with `-label:bug`.

Figure 11-11. Labeled issues

اکنون که برچسب ها را در این زمینه قرار داده ایم ، می توانیم از طریق آنها فیلتر کنیم. به عنوان مثال ، برای دیدن هر شماره با عنوان "پیشرفت" ، فقط روی فیلتر کلیک کنید (که در شکل 11-12 نشان داده شده است) ، و نتیجه ای مشابه با من دریافت خواهید کرد همانطور که در شکل 11-13 نشان داده شده است.



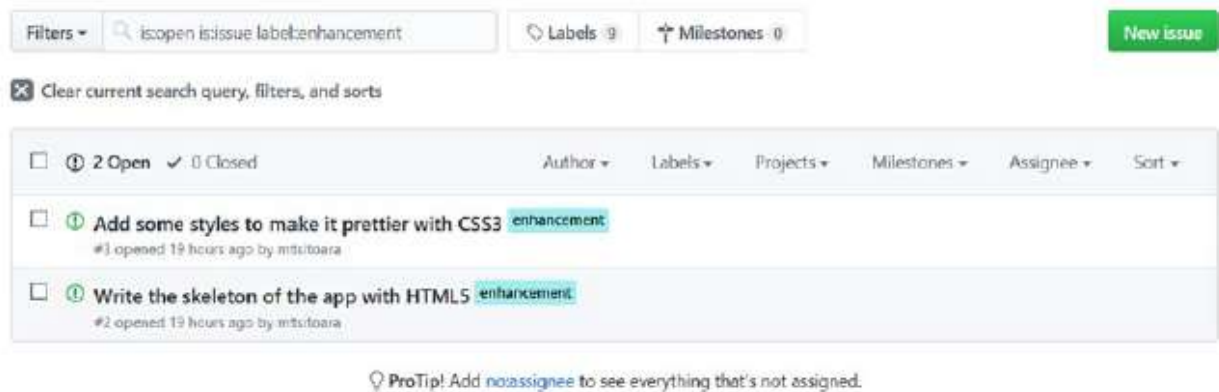


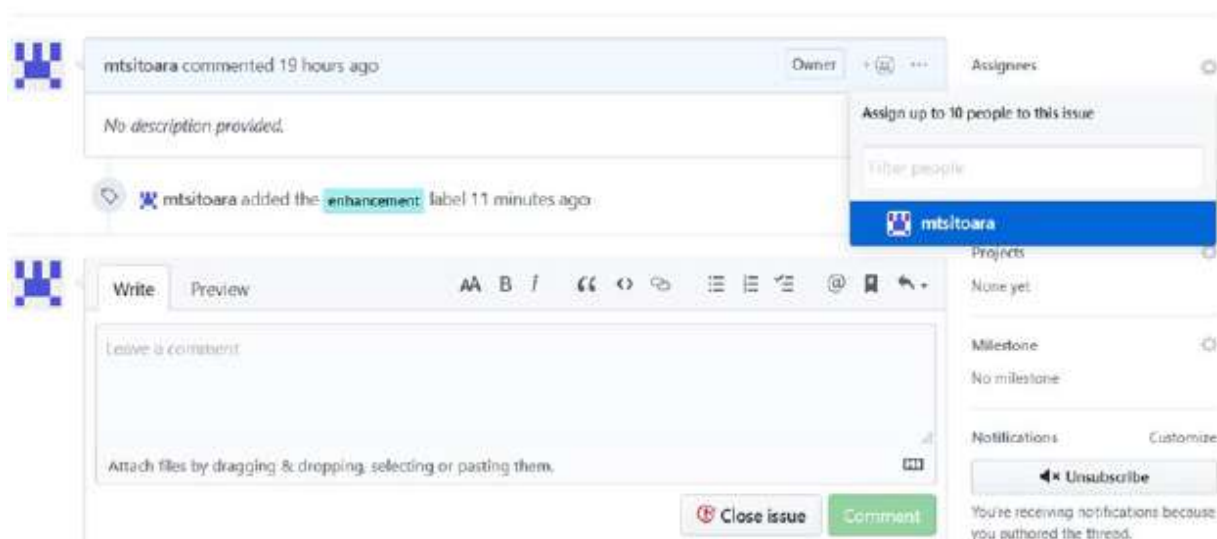
Figure 11-13. Filtered issues

آیا فیلتر سرگرم کننده نیست؟! اما می دانید چه چیز سرگرم کننده دیگری است؟ موضوع را به دیگران اختصاص دهید! بیاید این کار را انجام دهیم

Assignees

اکنون که مشکلات ما به درستی برچسب خورده است، وقت آن رسیده است که آنها را توسعه دهیم. این کار بسیار ساده ای است و تفاوت چندانی با برچسب ها ندارد.

می توانید یک شماره را به ده عضو تیم خود اختصاص دهید. اما از آنجا که شما در حال حاضر تنها هستید، فقط می توانید خود را اختصاص دهید. بیاید این کار را انجام دهیم! به موضوعاتی تحت عنوان "اسکلت برنامه را با HTML5 بنویسید" و "برخی از سبک ها را اضافه کنید تا با CSS3 زیباتر شود" بروید و آنها را به خود اختصاص دهید. اختصاص دادن شماره به یک عضو تیم دقیقاً مانند اضافه کردن برچسب ها کار می کند. برای مثال می توانید شکل 11-14 را بررسی کنید.



بعد از اینکه آن دو موضوع را به خود اختصاص دادید ، نتیجه ای مانند من دریافت خواهید کرد همانطور که در شکل 11-15 در صفحه شماره های خود نشان داده شده است. اکنون می توانید مشکلات خود را با برچسب ها و افراد اختصاصی فیلتر کنید.

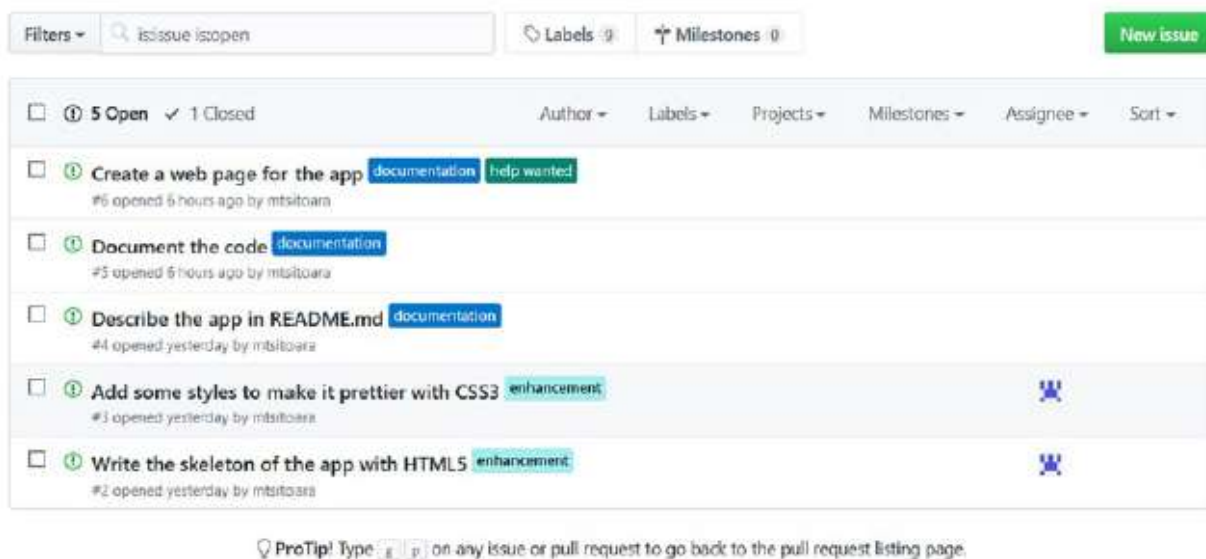


Figure 11-15. A complete issues list

پیوند دادن مسائل با تعهدات

همانطور که در ابتدای این فصل گفتیم ، هر اقدامی که با Git انجام می دهید باید حل مسئله به عنوان هدف خود باشد. بیشتر اوقات ، هنگام استفاده از Git ، با تعهداتی کار خواهید کرد. بنابراین ، هر یک از این تعهدات باید با یک موضوع گره خورده باشند. در این بخش ، ما می خواهیم یاد بگیریم که چگونه تعهدات خود را به مسائل پیوند دهیم.

ابتدا ، تصمیم می گیریم در مورد چه موضوعاتی کار خواهیم کرد. همانطور که در شکل 11-15 مشاهده کردیم ، دو مسئله به ما اختصاص داده شده است: "اسکلت برنامه را با HTML5 بنویسید" و "برخی از سبک ها را اضافه کنید تا با CSS3 زیباتر شود." ما در ابتدا قصد داریم روی نوشتن اسکلت کار کنیم زیرا این کار خیلی مهم است که با آن شروع کنیم. بنابراین صفحه جزئیات این شماره را باز کنید و به شماره آن توجه کنید. همانطور که در شکل 11-16 می بینید ، شماره 2 مخزن است.

Write the skeleton of the app with HTML5 #2

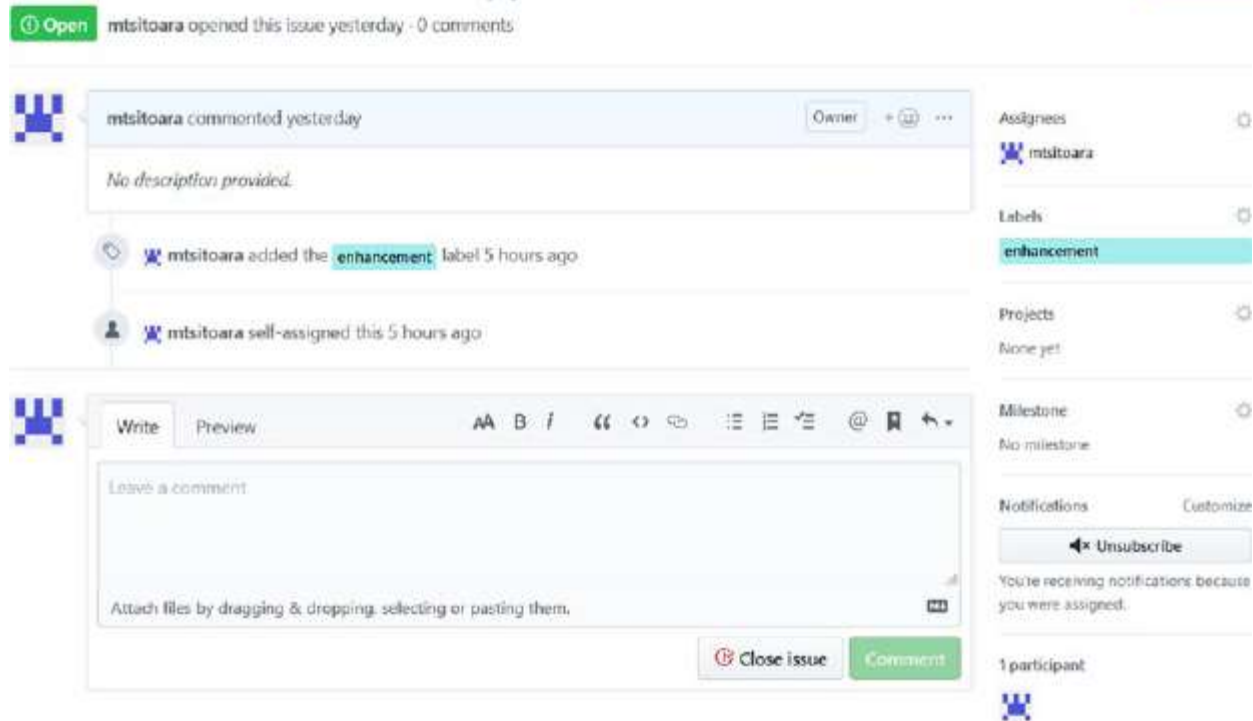


Figure 11-16. Issue number 2 details page

کار روی تعهد

اکنون که مشکلی برای حل و فصل آن داریم ، زمان آن رسیده که تعهد خود را آماده کنیم. از آنجا که تصمیم گرفتیم از HTML5 ساده برای این برنامه استفاده کنیم ، فقط به یک اسکلت واحد نیاز داریم. بنابراین ، یک پرونده با نام index.html در فهرست کار خود ایجاد کنید و در این کد جایگذاری کنید:

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>TODO list</title>
</head>
<body>
<h1>TODO list</h1>
<h3>Todo</h3>
<ul>

<li>Buy a hat for the bat</li>
<li>Clear the fogs for the frogs</li>
<li>Bring a box to the fox</li>
</ul>
<h3>Done</h3>
```

```
<ul>
<li>Put the mittens on the kittens</li>
</ul>
</body>
</html>
```

اکنون ، به شما اجازه خواهیم داد پرونده جدید ایجاد شده را تنظیم کنید ، اما هنوز commit نکنید. ما باید در مورد پیام متعهد صحبت کنیم.

ارجاع یک شماره

ما آماده هستیم تا پروژه را در وضعیت فعلی خود انجام دهیم ، اما باید پیام متعهد را ببندیم تا این تعهد با یک موضوع مرتبط شود. متداول ترین روش پیوند تعهد به یک موضوع ذکر شماره شماره در پیام متعهد است.

تا این لحظه ، ما فقط از پیامهای متعهد بسیار کوتاه استفاده می کردیم زیرا سعی می کردیم آنها را تحت یک خط نگه داریم. اما از آنجایی که ما برای توصیف تعهدات خود نیاز به اتاق داریم تا روش پیچیده تری داشته باشیم ، ما از این پس می خواهیم پیام های متعهد خود را از این طریق بسازیم: یک عنوان ، یک بدن و یک پاورقی که با یک خط خالی از هم جدا شده است. برای کمک به درک شما ، می توانید تصویری از آن را در شکل 11-17 پیدا کنید.

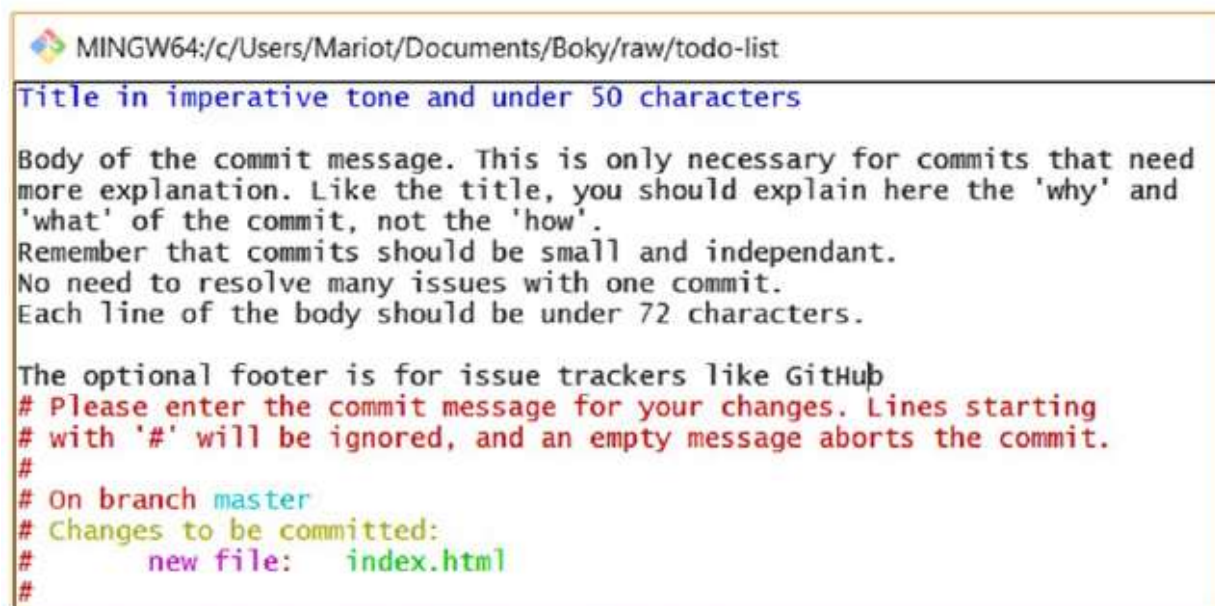


Figure 11-17. The commit message structure

بدنه و پاورقی اختیاری است. فقط در صورت لزوم از آنها استفاده کنید ، مخصوصاً بدنه. مردم تنبل هستند؛ آنها احتمالاً فقط عنوان را می خوانند و به حرکت خود ادامه می دهند. پاورقی همان چیزی است که هم اکنون ما را مورد علاقه قرار می دهد. این بخش برای ردیاب های مسئله ای مانند GitHub اختصاص داده شده است. ما از پاورقی استفاده می کنیم تا با استفاده از شماره آنها به مواردی مراجعه کنیم. به عنوان مثال ، برای اشاره به موضوعی که ما روی آن کار می کنیم ، فقط می خواهیم شماره آن را در صفحه قرار دهیم که مقدم بر "#" باشد. هنگامی که GitHub این مسئله را می بیند ، بلافاصله تعهد را با موضوع ارجاع شده پیوند می دهد.

با ترکیب همه اینها ، بیایید تعهد خود را با یک پیام متعهد مناسب انجام دهیم ، مثلاً تعهد من در شکل 11-18 نشان داده شود.

```
MINGW64/c/Users/Mariot/Documents/Boky/raw/todo-list
Add index.html that contains the project skeleton
See: #2
# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
#
# Date:      Sun Jul 14 20:26:37 2019 +0200
#
# On branch master
# Changes to be committed:
#   new file:   index.html
#
```

Figure 11-18. Commit message linked to issue #2

در پیام متعهد من قسمت بدن را رد کردم زیرا غیر ضروری بود. من فقط نیاز به پیوند این تعهد به شماره 2 داشتم بنابراین آن شماره را در صفحه قرار دادم. اکنون ، آن را فشار دهید! اگر فراموش کردید که چگونه (اشاره: git push master) را فراموش کرده اید.

حالا برگردیم به صفحه جزئیات شماره ما. اولین چیزی که توجه خواهید کرد این است که نظر جدیدی به آن اضافه شده است: این اشاره به تعهد ما است. این باید مانند معدن باشد که در شکل 11-19 نمایش داده شده است.

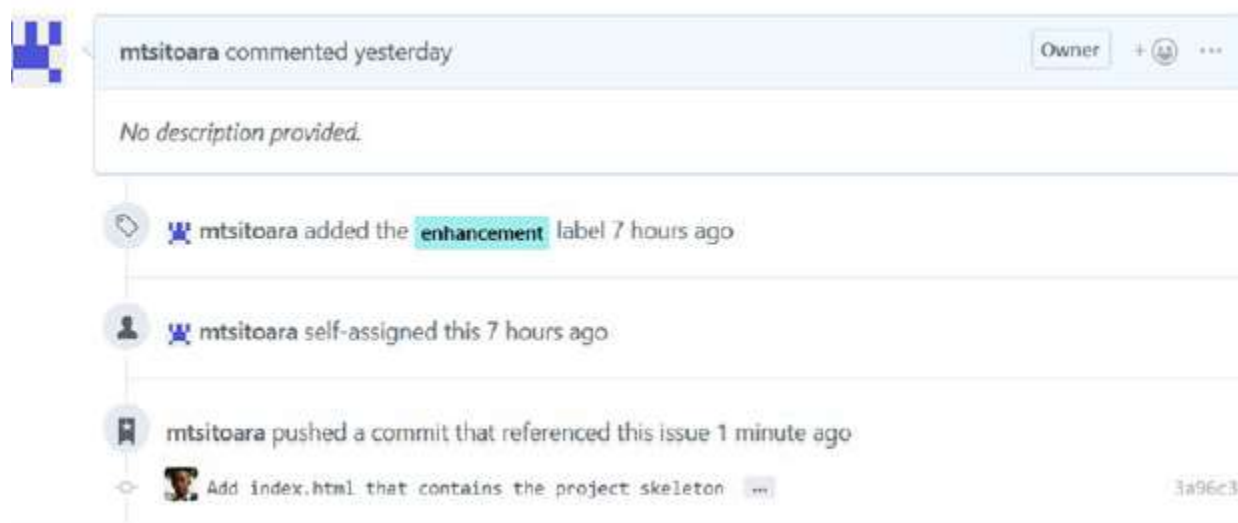


Figure 11-19. A reference to our last commit

این ویژگی بسیار مفیدی از GitHub است که مطمئناً از آن زیاد استفاده خواهید کرد: تمام تعهدات مرتبط با یک موضوع خاص را نشان دهید. به همین دلیل باید بدون پیوند دادن به مسئله، هیچ متعهدی را تحت فشار قرار دهند. برای مدیریت پروژه بهتر است.

اگر روی عنوان تعهد نشان داده شده در مرجع کلیک کنید (شکل 11-19 را ببینید)، یک صفحه آشنا خواهید دید. به شما اجازه می‌دهد خودتان کشف کنید که کدام صفحه در شکل 11-20 نمایش داده شده است.

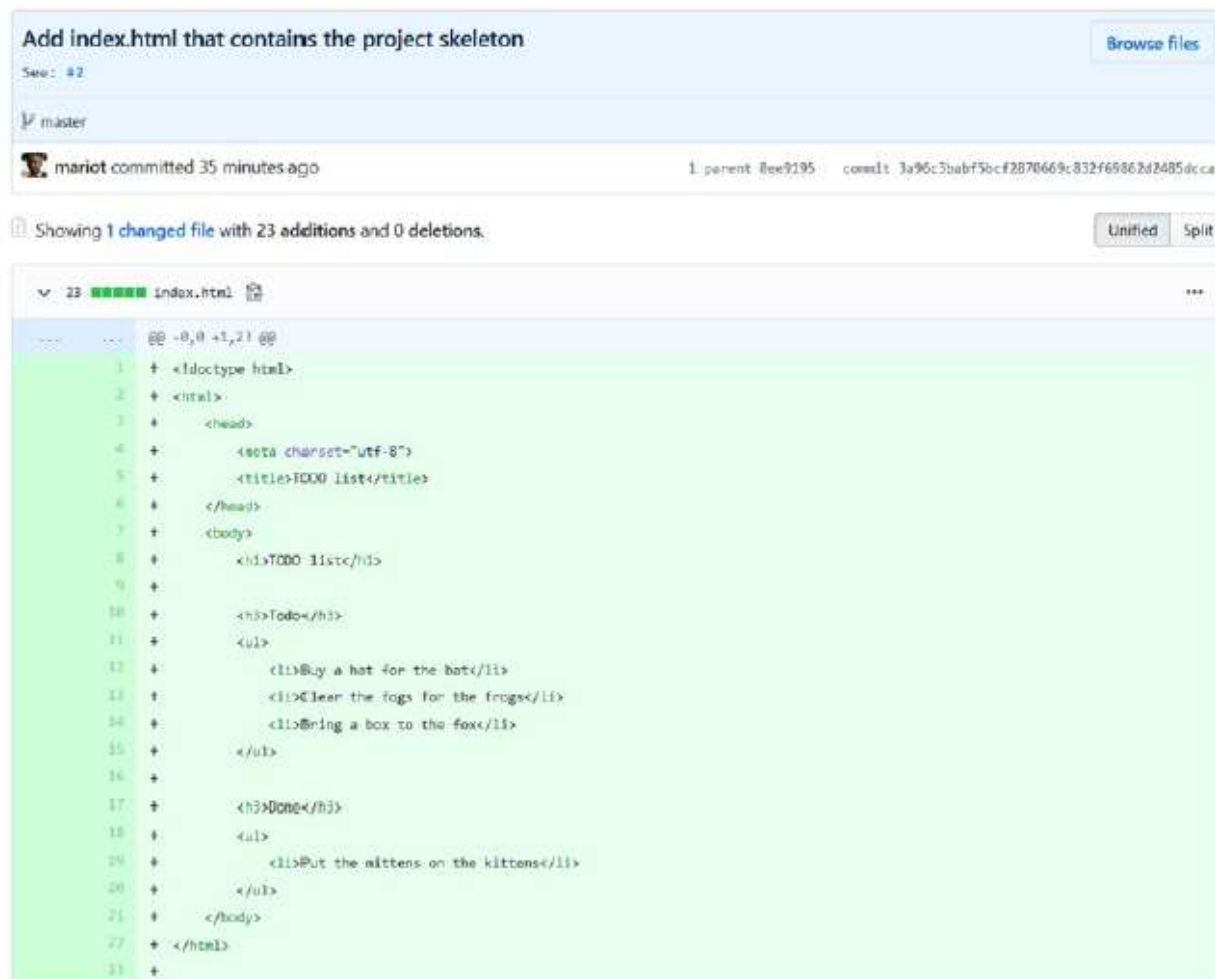


Figure 11-20. A detailed view of a commit

درست است! این نمایش "نمایش گیت" است. بدون نیاز به گم شدن در دستورات Git برای دیدن آنچه commit کرده‌اید، می‌توانید مستقیماً آن را در GitHub مشاهده کنید! اکنون که مسئله را با موفقیت حل کردیم، به صفحه جزئیات آن برگردید و آن را ببندید.

بباید مسئله بعدی را حل کنیم!

بسته شدن یک مسئله با استفاده از کلمات کلیدی

خوب بود که روی یک موضوع کار کنیم و آن را ببندیم ، درست است؟ خوب ، هنوز هم یک چیز جالب تر وجود دارد: بستن یک شماره با استفاده از کلمات کلیدی در یک پیام متعهد!

اول ، ما باید تصمیم بگیریم که کدام مسئله را حل کند. شماره بعدی ما "برخی از سبک ها اضافه کنید تا با CSS3 زیباتر شود" که دارای شماره 3 است. بیایید حلش کنیم! index.html را باز کنید و محتوا را به این ترتیب تغییر دهید:

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>TODO list</title>
<style>
h1 {
text-align:center;
}
h3 {
text-transform: uppercase;
}
li {
overflow: hidden;
padding: 20px 0;
border-bottom: 1px solid #eee;
}
</style>
</head>
<body>

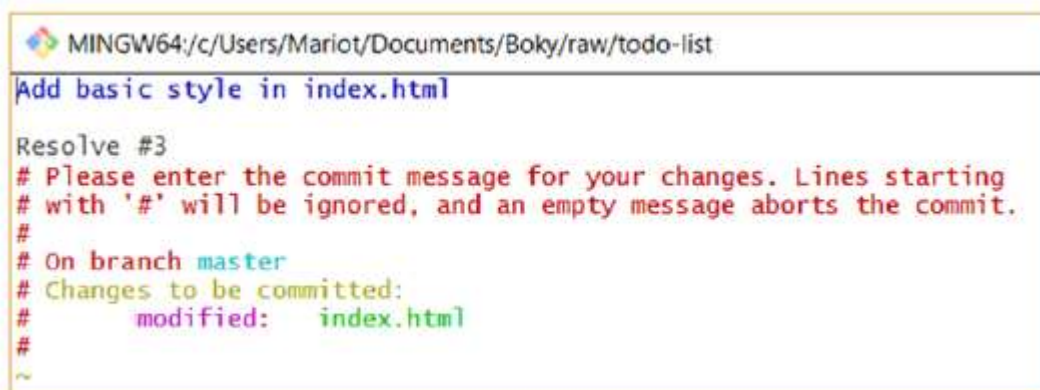
    <h1>TODO list</h1>
<h3>Todo</h3>
<ul>
<li>Buy a hat for the bat</li>
<li>Clear the fogs for the frogs</li>
<li>Bring a box to the fox</li>
</ul>
<h3>Done</h3>

    <ul>
<li>Put the mittens on the kittens</li>
</ul>
</body>
</html>
```

پرونده را مرحله بندی کنید اما هنوز commit نکنید. اینها کلمات کلیدی برای بستن یک شماره هستند

- بستن
- بسته می شود
- بسته
- ثابت
- رفع می کند
- درست شد
- برطرف کردن
- حل می کند
- حل شد

با استفاده از یکی از این کلمات که به دنبال آن شماره وجود دارد ، آن را به عنوان علامت گذاری و بسته شدن آن علامت گذاری می کنید. تعهد ما مسئله شماره 3 را برطرف می کند ، بنابراین آن را در صفحه پیام متعهد قرار خواهیم داد. پیام متعهد شما باید مانند من باشد ، همانطور که در شکل 11-21 نشان داده شده است.



```
MINGW64:/c/Users/Mariot/Documents/Boky/raw/todo-list
Add basic style in index.html

Resolve #3
# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
#
# On branch master
# Changes to be committed:
#   modified:   index.html
#
~
```

Figure 11-21. Resolving an issue by commit message

درست مانند پیام های متعهد ، مراجع موضوع باید از لحن ضروری استفاده کنند. بنابراین ترجیح داده می شود به جای " resolve " از " resolved." استفاده شود. حالا وقت آن رسیده که به تعهد خود فشار بیاوریم و خودمان را ببینیم!

به موضوعی که در آن کار کرده اید بروید (شما آن را در موارد باز پیدا نخواهید کرد ، از فیلتر استفاده می کنید تا مشکلات بسته را ببینید) و صفحه جزئیات را باز کنید. دقیقاً مانند نظر من ، باید نظر جدیدی در مورد آن ببینید ، همانطور که در شکل 11-22 نشان داده شده است.

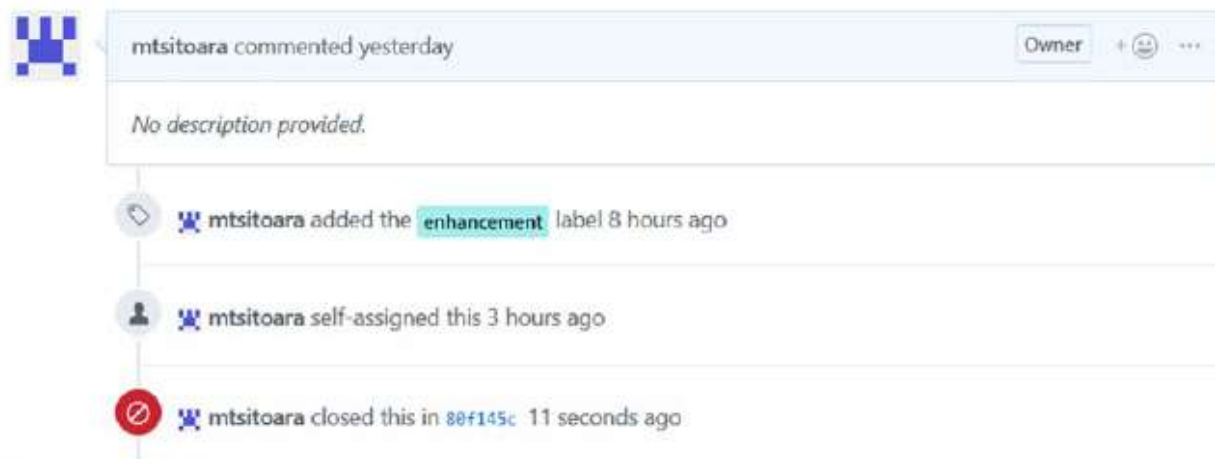


Figure 11-22. Issue closed by keywords

اگر روی نام commit کلیک کنید ، دوباره نمای "git show" از تعهد را مشاهده خواهید کرد. ویژگی کمی GitHub مفید است اما هنگام استفاده بسیار مراقب باشید. فقط هنگامی که کاملاً مطمئن هستید که آن را برطرف کرده اید ، مسئله را ببندید. بستن و بازگشایی موضوعات باعث سردرگمی مردم و ایجاد علامت گذاری های زیادی می شود. و مسئله دیگری را به اشتباه نبندید! 83٪ از کل خشونت در محل کار ناشی از اشتباهات بسته شده است. و فقط به این دلیل که من این آمار را اختراع کردم به این معنی نیست که باید آن را جدی بگیرید!

خلاصه

اوه! آن فصل کمی طولانی بود ، نه؟ ما در مورد موضوعات زیادی آموختیم اما مهمتر از همه ، نحوه پیوند دادن آنها به تعهدات. همیشه به یاد داشته باشید که قبل از اقدام به آنها ، تمام اقدامات خود را در مواردی قرار دهید. و فراموش نکنید که آنها را با برچسب ها و تکالیف گنجانید.

این فصل ما در مورد مدیریت اولیه پروژه است. شما باید بدانید که چگونه باید برنامه های بعدی حرکت خود را در GitHub برنامه ریزی کنید. اما مدیریت پروژه فقط کارهای قبلی را برنامه ریزی نمی کند. شما همچنین باید دید واضح و روشن از آنچه در گذشته اتفاق افتاده و به چه نقاط عطف رسیده است. بنابراین ، ما با پروژه ها به "مدیریت صحیح" پروژه خواهیم رسید. این بخش همچنین شامل خلاصه ای بسیار کوتاه از اکثر اشکال گردش کار GitHub است. بیا بریم!