12 потрясающих возможностей GitHub

[**Статья из группы Random**](https://javarush.ru/groups/general)

Присоединиться

Хоть убейте, не могу придумать никакого вступления, так что...

|  |
| --- |
| **Маленький словарик**  Поскольку термины Git и другие программистские словечки чаще всего употребляются без перевода, то я решил их не переводить. Приведу их тут, для порядка, краткий перевод терминов из этой статьи с «расшифровкой».  **Fork** – «развилка». По сути вы копируете себе проект, чтобы на его основе что-то доработать  **Pull request** — запрос на изменение. Отправка внесенных вами изменений в репозиторий на проверку (то есть в основной проект этот код будет внесен только после подтверждения владельцем репозитория или коллегами по работе)  **Pull** – «втянуть» (в IDE на вашем компьютере, например) проект с GitHub  **Push** – «вытолкнуть» проект c локальной машины на GitHub |

**#1 Редактирование кода на GitHub.com**

Начну с того, что, как мне кажется, и так всем известно (хотя лично я и не подозревал об этом еще неделю назад). При просмотре любого текстового файла на сайте GitHub, в любом репозитории, справа сверху можно видеть маленький карандашик. Если щёлкнуть по нему, можно будет отредактировать этот файл. По завершении, нажмите Propose file change ("Предложить изменение файла") и GitHub создаст разветвление репозитория (fork) и запрос на внесение изменений (Pull Request). Поразительно, не правда ли? Он сам создает fork! Нет нужды делать форк и загружать к себе код, вносить локально изменения и отправлять обратно на GitHub c Pull Request'ом. Очень удобно, если нужно внести минимальные правки.

*не совсем настоящий Pull Request*

**#2 Вставка изображений**

Описания проблем не ограничиваются только текстовыми комментариями. Знаете ли вы, что можно вставлять изображения прямо из буфера обмена? При вставке вы увидите, его загрузку (несомненно, в "облако") и превращение в разметку для отображения изображения. Изящно!

**#3 Форматирование кода**

Если вам нужно написать блок кода, начните с трёх обратных одинарных кавычек — и GitHub попытается угадать, на каком языке программирования вы пишете. Но если вы размещаете фрагмент кода на таких языках программирования, как Vue, Typescript или JSX, то можете явным образом указать язык, чтобы подсветка синтаксиса была правильной. Обратите внимание на ```jsx в первой строке:

...обеспечивающий правильное отображение фрагмента кода:

(это распространяется и на Gist, кстати. Если указать для гиста расширение .jsf, будет подсвечиваться синтаксис JSF). Вот список [всех поддерживаемых синтаксисов](https://github.com/github/linguist/blob/fc1404985abb95d5bc33a0eba518724f1c3c252e/vendor/README.md).

**#4 Закрытие проблем при помощи "магических слов" в Pull Request’ах**

Допустим, вы создаете Pull Request, исправляющий проблему #234. Вы можете вставить текст "исправляет проблему #234" в описание вашего запроса (или в любом месте любого комментария к запросу на изменение). После этого слияние Pull Request’а «автомагически» закроет проблему. Круто, не так ли? Вот больше [информации об этом в документации](https://help.github.com/articles/closing-issues-using-keywords/).

**#5 Ссылка на комментарии**

Требовалось ли вам когда-нибудь создать ссылку на конкретный комментарий, а вы не знали, как это сделать? Эти дни давно прошли, поскольку я раскрою вам секрет: для создания ссылки на комментарий нужно просто щелкнуть на дате/времени рядом с названием.

*Смотрите, у gaearon'а теперь есть фотка!*

**#6 Ссылка на код**

Итак, вы хотите создать ссылку на конкретную строку кода. В таком случае, попробуйте следующее: щелкните на номере строки рядом с нужным кодом в открытом файле. Ух ты, видите? URL поменялся, в нём теперь виден номер строки! Если удерживать нажатой клавишу SHIFT и нажать на номер другой строки, то – вуаля! – URL поменяется еще раз, и будет подсвечен диапазон строк. Этот URL теперь будет указывать на данный файл и данный диапазон строк. Но погодите, он указывает на текущую ветку. А что, если файл поменяется? Наверное, вам нужна, в этом случае, постоянная ссылка на файл в его текущем состоянии. Я очень ленивый, так что сделал один снимок экрана для всего вышеописанного:

Кстати, насчет URL'ов...

**#7 Использование URL GitHub в качестве командной строки**

Перемещение по GitHub с помощью UI организовано очень удобно. Но иногда, чтобы попасть в определенное место, быстрее окажется просто набрать его в URL. Например, если мне нужно перейти в ветку, над которой я работаю и посмотреть отличия её от ветки master, я могу просто ввести **/compare/имя\_ветки** после имени репозитория. После этого я попаду на страницу различий для данной ветки:

Но это отличия от ветки master, если же я работал до этого с веткой интеграции, то могу ввести в URL **/compare/integration-branch...my-branch**

Для любителей горячих клавиш: CTRL+L или CMD+L переводит курсор в строку URL (по крайней мере в браузерах Chrome и Firefox). Это, в сочетании с автодополнением браузера, значительно упрощает перемещение между ветками. **Совет от профи:** воспользуйтесь стрелками для перемещения по предложениям автодополнения Chrome и нажимайте SHIFT+DELETE для удаления элементов из истории (например, после слияния ветки). (Не знаю, будет ли легче читать эти горячие сочетания клавиш, если буду ставить в них пробел, вот так SHIFT + DELETE. Но формально "+" не является их частью, так что мне этот вариант не нравится. Именно из-за таких вещей я и не сплю по ночам, Ронда.)

**#8 Создание списков для проблем**

Хотите ли вы, чтобы в вашем описании проблемы присутствовал чекбокс?

И хотите ли вы, чтобы при просмотре проблемы из списка отображалась элегантная полоска вроде "2 из 5"?

Никаких проблем! Создавать интерактивные кнопки-флажки можно с помощью следующего синтаксиса:

* **[ ] Screen width (integer)**
* **[x] Service worker support**
* **[x] Fetch support**
* **[ ] CSS flexbox support**
* **[ ] Custom elements**

**Синтаксис:** пробел, дефис, пробел, открывающаяся квадратная скобка, пробел (или x), закрывающаяся квадратная скобка, пробел и какие-нибудь слова. После этого вы сможете действительно ставить/убирать галочки с этих кнопок! Мне это, почему-то, кажется каким-то техническим волшебством. Вы можете ставить галочки! И при этом исходный текст меняется! Страшно подумать, что они придумают дальше. А, и если эта проблема есть на панели проекта, то там тоже будет отображаться ход выполнения:

Если вам непонятно, что я имею в виду под "панелью проекта" – читайте ниже. Всего на пару сантиметров ниже на этой странице.

**#9 Панели проектов в GitHub**

Для больших проектов я всегда использовал Jira. А для своих личных проектов я всегда использовал Trello. Оба этих инструмента мне очень нравятся. Когда несколько недель назад я узнал, что GitHub предлагает свой собственный вариант, прямо на закладке **Projects** репозитория, я подумал, что иметь смысл продублировать набор задач, с которыми я уже работаю в Trello.

Ничего веселого тут нет А теперь то же самое в проекте GitHub:

*Постепенно ваши глаза привыкнут к неконтрастному изображению*

Ради ускорения я добавил все вышеуказанное в виде заметок (notes), то есть они не являются "настоящими" проблемами (issues) GitHub. Но мощь управления задачами в GitHub состоит в интеграции с остальным репозиторием – так что, вероятно, лучше добавить существующие проблемы из репозитория на панель. Нажмите **Add Cards** в верхнем правом углу и найдите то, что хотели бы добавить. Здесь пригодится специальный [синтаксис поиска](https://help.github.com/articles/searching-issues-and-pull-requests/): например, наберите **is:pr is:open** и вы сможете перетащить любой открытый Pull Request на панель, или **label:bug**, если нужно исправить какие-то ошибки.

А ещё можно преобразовать существующие заметки в проблемы.

И, наконец, из формы существующей проблемы, добавить её в проект в правой панели.

*Она попадёт в список triage данной панели проекта, так что вы сможете выбрать, в какой столбец её поместить*

Когда описание "таска" находится в том же репозитории, что и реализующий этот таск код, это очень (ооооочень) удобно. Это значит, что через много лет вы сможете выполнить команду **git blame** для какой-либо строки кода и выяснить всю подоплёку задачи, которая к этой строке привела, без того, чтобы отслеживать это всё в Jira/Trello/еще где-нибудь.

**Недостатки**

Последние три недели я экспериментировал с выполнением всех задач в GitHub вместо Jira (на маленьком проекте, примерно в стиле Канбан) и мне это понравилось. Но я не могу себе представить это для scrum-проекта, где необходимо оценивать и подсчитывать должным образом скорость разработки и тому подобное. Хорошая новость: у проектов GitHub настолько мало "особенностей", что переход на другую систему не займет, в случае чего, много времени. Так что попробуйте, посмотрим, насколько вам понравится. Не знаю, насколько это важно, но я *слышал* про [ZenHub](https://www.zenhub.com/) и открыл его впервые 10 минут назад. Фактически это расширение GitHub, в котором можно оценивать проблемы и создавать "epics" и зависимости. Там есть графики скорости разработки и выгорания; похоже, что это просто потрясающая вещь. Дальнейшее чтение: [документация GitHub по Projects.](https://help.github.com/articles/tracking-the-progress-of-your-work-with-project-boards/)

**#10 Gwiki**

Для неструктурированного набора страниц – подобного Википедии – GitHub Wiki (которую я буду далее называть просто Gwiki) просто великолепна. Для структурированного же набора страниц – например, как ваша документация – уже не настолько. Отсутствует возможность указать, что "вот эта страница – дочерняя по отношению к вот той", нет таких удобных вещей, как кнопки "Следующий раздел" и "Предыдущий раздел". Гензель и Гретель тут бы точно заблудились, потому что "хлебных крошек" (специальных отладочных операторов – прим. перев.) тут тоже нет. (Примечание автора: Вы *читали* эту историю? Она просто бесчеловечна. Двое юных отморозков убивают несчастную голодную старушку, сжигая её заживо в её *собственной печи*. И конечно же, оставляя непонятно кому полный беспорядок. Мне кажется, именно поэтому молодежь в наши дни адски чувствительна – в наши дни сказки, читаемые детям перед сном, недостаточно жестоки!) Продолжаем – чтобы попробовать Gwiki на деле, я ввел несколько страниц из NodeJS в качестве страниц вики, после чего создал пользовательскую боковую панель, чтобы смоделировать реальную структуру сайта. Эта боковая панель находится там постоянно, хотя текущая страница и не подсвечивается. Ссылки придется поддерживать вручную, но в целом все работает отлично. Если хотите, [можете взглянуть](https://github.com/davidgilbertson/about-github/wiki):

Это вряд ли может соперничать с чем-то вроде GitBook (который используется в документации [Redux](http://redux.js.org/)) или сделанным на заказ веб-сайтом. Но это уже добрых 80% от него и все прямо в вашем репозитории. Я просто фанат этого. Предлагаю: если вы уже переросли этап использования одного файла README.md и вам нужно несколько различных страниц для руководств пользователя или более подробной документации, имеет смысл остановиться на Gwiki. Если же отсутствие структуры/навигации вам мешает, переходите на что-либо ещё.

**#11 GitHub Pages**

Возможно, вы уже знали, что GitHub Pages можно использовать для размещения статического сайта. А если не знали, то знаете теперь. Однако этот раздел посвящен более узкому вопросу: использованию *Jekyll* для создания сайта. В простейшем варианте, GitHub Pages + Jekyll могут визуализировать файл README.md с использованием приятной глазу темы. Например, взгляните на мою страницу readme из [about-github](https://github.com/davidgilbertson/about-github):

Если нажать на закладку settings ("настройки") для этого сайта на GitHub, включить GitHub Pages и выбрать тему Jekyll...

То мы получим [страницу в стиле темы Jekyll](https://davidgilbertson.github.io/about-github/):

После этого можно создать целый статический сайт на основе, главным образом, легко редактируемых файлов разметки, по существу, превращая GitHub в CMS. Хотя фактически я этого не использовал, именно так создаются сайты на фреймворке Bootstrap с использованием React, так что ничего ужасного в этом нет. Отмечу, что на локальной машине должен быть запущен Ruby (пользователи Windows тут обменяются понимающими взглядами и пойдут другим путём, пользователи macOS скажут: "В чем проблема, вы куда? Ruby – универсальная платформа! Там же есть система управления пакетами GEMS!") (Стоит отметить также, что "Агрессивный или угрожающий контент или поведение" в GitHub Pages недопустимы, так что вы не сможете разместить там свою версию истории про Гензеля и Гретель).

**Мое мнение**

Чем детальнее я изучал связку GitHub Pages + Jekyll (для этой статьи), тем больше мне казалось, что вся эта идея странно попахивает. Идея "сделать свой собственный веб-сайт с приложением минимальных усилий" замечательна. Но для работы над ним все равно требуется текущий вариант на локальной машине. И для чего-то столь "простого" тут слишком много команд командной строки. Я бегло просмотрел семь страниц из раздела [Getting Started](https://jekyllrb.com/docs/home/) и почувствовал, что единственное, что тут простого – это я *сам*. И это я еще даже не разобрался с простым синтаксисом для главной страницы или азами простого "Механизма шаблонизации на основе языка Liquid". Уж лучше я напишу веб-сайт самостоятельно! Если честно, я немного удивлен, что Facebook использует это для документации React, в то время как они могли бы, вероятно, формировать страницы их системы помощи при помощи React и [выполнять предварительную визуализацию в виде статических HTML-файлов каждый день](https://github.com/facebookincubator/create-react-app/blob/master/packages/react-scripts/template/README.md#pre-rendering-into-static-html-files). Все, что им нужно – всего лишь найти способ получать существующие файлы разметки так, как если бы они поступали из CMS. А что, если...

**#12 Использование GitHub в качестве CMS**

Допустим, у нас есть веб-сайт с каким-то текстом, но нам не хотелось бы хранить этот текст в виде HTML-разметки. Вместо этого, мы хотели бы хранить где-то куски текста, чтобы их могли легко редактировать и обычные пользователи, не разработчики. Желательно, с каким-то вариантом управления версиями. Возможно, даже с какой-нибудь процедурой рецензирования. Вот что я предлагаю: использовать хранимые в репозитории файлы разметки для хранения текста. И использовать компонент в клиентской части, который бы извлекал эти куски текста и отображал их на странице. Я поклонник React, так что вот пример соответствующего компонента <Markdown>, который, при заданном пути к файлу разметки, извлечет его, выполнит синтаксический разбор и визуализацию в виде HTML.

**class** Markdown **extends** React.Component {

constructor(props) {

**super**(props);

// Конечно, вам нужно заменить это на свой URL

**this**.baseUrl = 'https://raw.githubusercontent.com/davidgilbertson/about-github/master/text-snippets';

**this**.state = {

markdown: '',

};

}

componentDidMount() {

fetch(`${**this**.baseUrl}/${**this**.props.url}`)

.then(response => response.text())

.then((markdown) => {

**this**.setState({markdown});

});

}

render() {

**return** (

<div dangerouslySetInnerHTML={{\_\_html: marked(**this**.state.markdown)}} />

);

}

}

(я использую тут пакет npm [marked](https://www.npmjs.com/package/marked) для синтаксического разбора разметки в HTML) URL указывает на мой репозиторий примеров, в каталоге [/text-snippets](https://github.com/davidgilbertson/about-github/tree/master/text-snippets) которого лежат файлы разметки. (Можно также использовать API GitHub для [получения контента](https://developer.github.com/v3/repos/contents/#get-contents), но я сомневаюсь, что это вам пригодится) Использовать подобный компонент можно следующим образом:

**const** Page = () => (

<div className="page">

<div className="about-us">

<Markdown url="about-us.md" />

</div>

<div className="disclaimer">

<p>A very important disclaimer:</p>

<Markdown url="disclaimers/home-page-disclaimer.md" />

</div>

</div>

);

Так что теперь GitHub играет роль, в некотором роде, вашего CMS, для тех кусков текста, которые вы хотели бы разместить. Вышеприведенный пример извлекает разметку только после того, как компонент будет загружен в браузере. Если вам нужен статический сайт, то придется визуализировать его на сервере. Хорошая новость! Ничто не мешает вам извлечь все файлы разметки на сервере (при использовании любой устраивающей вас стратегии кэширования). Если вы решите пойти этим путем, то вам имеет смысл воспользоваться API GitHub, чтобы получить список всех файлов разметки в каталоге. **Бонус – утилиты GitHub!** Я уже довольно давно использую расширение [Octotree](https://chrome.google.com/webstore/detail/octotree/bkhaagjahfmjljalopjnoealnfndnagc?hl=en-US) для браузера Chrome и рекомендую его вам. Не без оговорок, но все же рекомендую. Оно отображает слева панель с древовидным представлением просматриваемого репозитория.

А из [этого видео](https://www.youtube.com/watch?v=NhlzMcSyQek&index=2&list=PLNYkxOF6rcIB3ci6nwNyLYNU6RDOU3YyL) я узнал про [octobox](https://octobox.io/), который также пока что представляется мне весьма неплохой утилитой. Это папка для входящих для ваших проблем GitHub. Это всё, что вам нужно о нём знать. Говоря о цветах, я сделал все вышеприведенные снимки экрана в светлой теме, чтобы вас не пугать. Но если во всём остальном я предпочитаю темные цвета, то зачем терпеть мертвенно-бледный GitHub?

Здесь я использовал сочетание расширения [Stylish](https://chrome.google.com/webstore/detail/stylish-custom-themes-for/fjnbnpbmkenffdnngjfgmeleoegfcffe/related?hl=en) для браузера Chrome (умеющее применять темы к любому веб-сайту) и стиля [GitHub Dark](https://userstyles.org/styles/37035/github-dark). И на закуску, темная тема инструментов разработчика GitHub (встроенная, только нужно включить) и тема [Atom One Dark](https://chrome.google.com/webstore/detail/atom-one-dark-theme/obfjhhknlilnfgfakanjeimidgocmkim?hl=en) для Chrome.

**Bitbucket**

Строго говоря, он здесь не совсем уместен, но я просто не могу не упомянуть Bitbucket. Два года назад я начинал проект и провел полдня за выбором оптимального git-хостинга. Так вот, Bitbucket выиграл с существенным отрывом. Их поток рецензирования кода ушел далеко вперед от конкурентов (это было задолго до того, как в GitHub появилось хотя бы понятие рецензента). С тех пор GitHub тоже обзавелся рецензиями. К сожалению, последний год мне не доводилось использовать Bitbucket – возможно, они опять ушли вперед в чём-нибудь. Так что я рекомендую тем, кто отвечает за выбор git-хостинга, обратить внимание и на Bitbucket.

**Заключение**

Вот и всё! Надеюсь, что сумел рассказать вам хотя бы три ранее незнакомых вам вещи. И ещё надеюсь, что вы хорошо провели время за чтением моей статьи. Перевод статьи: [12 cool things you can do with GitHub](https://hackernoon.com/12-cool-things-you-can-do-with-github-f3e0424cf2f0)