Тема 18. Сборка и публикация проекта.

хекслет колледж

Цель занятия:

Познакомить студентов с современным циклом подготовки фронтенд-проекта к релизу: от оптимизации кода до размещения на бесплатном хостинге.

Учебные вопросы:

1. Зачем нужна сборка проекта?
2. Инструменты сборки
3. Подготовка проекта к публикации
4. Хостинг для фронтенд-разработчика
5. Публикация на GitHub Pages

1. Зачем нужна сборка проекта?

Два режима жизни проекта

В современной разработке разделяют две среды обитания кода:

- **Development** (Разработка): Код должен быть удобным для человека. Много файлов, длинные названия переменных, комментарии, отступы.
- **Production** (Продакшн): Код должен быть удобным для браузера. Он должен быть максимально легким, быстрым и сжатым.

Главные задачи сборки

А. Минификация (Minification)

Это процесс удаления всего «лишнего» из кода без изменения его логики. Сборщик удаляет:

- Пробелы и переносы строк.
- Комментарии.
- Сокращает имена переменных (например, let userRegistrationStatus превращается в let a).

Результат: Файл весит на 60–80% меньше, что ускоряет загрузку сайта.

Б. Бандлинг (Bundling)

В процессе разработки мы разделяем код на десятки модулей (файлов), чтобы не запутаться.

Но каждый отдельный файл — это отдельный HTTP-запрос к серверу. Много запросов замедляют сайт.

Результат: Сборщик «склеивает» все ваши JS-файлы в один (бандл), а все CSS — в другой.

B. Транспиляция (Transpilation)

Мы хотим писать на самом современном JS (ES6+), но не все браузеры (особенно старые) его понимают.

Сборщик автоматически переводит современный код на «старый лад», чтобы сайт работал везде.

1. Понятие

Г. Оптимизация ресурсов

Сборщик может автоматически:

- Сжимать изображения.
- Конвертировать картинки в современные форматы (WebP).
- Удалять неиспользуемый код (Tree Shaking).

Наглядное сравнение

Характеристика	Исходный код (Dev)	Сборка (Prod)
Читаемость	Высокая (для людей)	Нулевая (для машин)
Размер файлов	Большой	Минимальный
Количество файлов	Много (модули)	1–2 (бандлы)
Скорость загрузки	Медленно	Максимально быстро

Итог: Что мы получаем в конце?
После запуска команды сборки (обычно npm run build) вместо папки с исходниками мы получаем папку dist (от англ. distribution — распространение) или build.
Это и есть готовый продукт. Именно содержимое этой папки, а не весь ваш репозиторий, отправляется на сервер (хостинг).

Вывод:
Сборка — это технический процесс превращения «черновика», удобного для программиста, в «чистовик», оптимизированный для быстрой работы в интернете.
Без сборки профессиональный веб-проект сегодня немыслим, так как скорость загрузки напрямую влияет на посещаемость и позиции сайта в поиске.

2. Инструменты сборки (обзор)

Эволюция инструментов

Раньше разработчики всё делали вручную: сами сжимали картинки через сервисы и копировали код в один файл. Затем появились инструменты автоматизации.

- Webpack: «Дедушка» современных сборщиков. Очень мощный, но сложный в настройке (огромные файлы конфигурации). До сих пор стандарт в больших корпорациях.
- Parcel: «Сборщик без конфигурации». Популярен для небольших проектов, так как почти не требует настройки.
- Vite: (франц. «Быстрый»). Современный стандарт. Невероятно быстрый, прост в освоении и уже настроен под все нужды современного JS.

Как работает современный сборщик (на примере Vite)

Когда мы запускаем процесс сборки, инструмент делает следующее:

- Анализирует связи: Он смотрит на ваш index.html, видит подключенные JS-файлы и идет по цепочке импортов.
- Обрабатывает типы файлов: Если он видит современный JS, он его транспилирует. Если видит CSS, он его минимизирует.
- Генерирует хеши: Он добавляет к именам файлов уникальные коды (например, style.a7f2b1.css). Это нужно, чтобы браузер пользователя понимал: «О, файл обновился, надо скачать новую версию, а не брать старую из кэша».

Когда сборщик НЕ нужен (в каких случаях можно обойтись без него)

Не всегда стоит «стрелять из пушки по воробьям». Можно обойтись без сборки, если:

- Учебные задачи: Когда вы только учите основы синтаксиса JS в одном файле.
- Простые лендинги: Одностраничный сайт с 20 строками JS и парой CSS-правил. Сборка здесь займет больше времени, чем сама разработка.
- Старые проекты: Поддержка legacy-кода, где всё изначально написано на простых скриптах.
- Быстрое прототипирование: Когда нужно за 5 минут набросать идею в CodePen или простой папке.

Когда использование сборщика ОБЯЗАТЕЛЬНО

- Использование NPM-пакетов: Если вы хотите установить через npm библиотеку (например, Chart.js для графиков или Leaflet для карт), вам нужен сборщик, чтобы подключить её код.
- Модульная структура: Когда в проекте больше 3–5 JS-файлов. Без сборщика управлять зависимостями между ними крайне сложно.
- Использование препроцессоров: Если вы хотите писать на Sass/SCSS или использовать современные возможности CSS (которые еще не везде поддерживаются).
- Командная разработка: Сборщик гарантирует, что у всех участников команды проект соберется и будет работать одинаково.
- Курсовая работа и портфолио: Для работодателя использование сборщика (хотя бы Vite) — это признак того, что вы знакомы с современным рабочим процессом (Workflow).

Вывод:

- Инструменты сборки — это не усложнение, а профессиональный стандарт.
- Если ваш проект выходит за рамки «одной кнопки, меняющей цвет фона», вам нужен сборщик.
- Для современных задач Vite является оптимальным выбором: он дает максимальную скорость при минимальных усилиях на настройку.

3. Подготовка проекта к публикации

Этот этап критически важен, потому что именно здесь новички чаще всего «ломают» свой проект.
Код может идеально работать на компьютере разработчика, но «рассыпаться» сразу после загрузки на сервер.

Когда мы открываем файл прямо из папки, браузер прощает многие ошибки.
Но сервер — это строгая среда.
Перед тем как нажать кнопку «Опубликовать», нужно пройтись по списку критических проверок.

Чек-лист подготовки (6 шагов)

Шаг 1: Относительные пути (Самая частая ошибка)
Проверьте, как подключены картинки, стили и скрипты.

- Плохо: C:/Users/Admin/Project/img/logo.png (этот путь существует только у вас на диске).
- Плохо: /img/logo.png (абсолютный путь от корня может не сработать на GitHub Pages).
- Хорошо: ./img/logo.png или img/logo.png (относительный путь).

Шаг 2: Регистр имен файлов

Windows игнорирует регистр (ему все равно, Photo.jpg или photo.jpg). Серверы (Linux) — нет.

- Если в коде написано <img src="cat.jpg">, а файл называется Cat.jpg, на сервере картинка не отобразится.
- Правило: Всегда называйте файлы маленькими латинскими буквами без пробелов.

Шаг 3: Очистка кода (Hygiene)

Код для пользователя должен быть чистым.

- Удалите все console.log(), которые вы использовали для отладки.
- Удалите закомментированные куски кода (для этого есть Git).
- Проверьте наличие favicon.ico (иконка сайта во вкладке), чтобы в консоли не висела ошибка 404.

Шаг 4: Мета-теги и заголовок

Убедитесь, что в <head> заполнены:

- <title> — название сайта, которое увидят люди в поиске и на вкладке.
- <meta name="description" content="..."> — описание для поисковиков.

Шаг 5: Проверка консоли (DevTools)

Откройте проект, нажмите F12 и перейдите во вкладку Console.

- Там не должно быть никаких красных сообщений об ошибках.
- Каждая ошибка в консоли — это риск, что какая-то функция (например, сохранение в localStorage) не сработает у пользователя.

Шаг 6: Оптимизация ресурсов

- Если у вас фоновая картинка весит 5 МБ — сайт будет тормозить. Сожмите изображения через сервисы типа TinyPNG или переведите их в формат WebP.

Вывод:

- Подготовка к публикации — это проверка проекта на «автономность».
- Сайт должен содержать только относительные пути и корректные имена файлов, чтобы успешно работать на любом удаленном сервере, независимо от того, какая операционная система там установлена.

4. Хостинг для фронтенд-разработчика

Что такое хостинг и деплой?

- Хостинг — это удаленный компьютер (сервер), который работает 24/7, чтобы ваш сайт был доступен в любое время.
- Деплой (Deployment) — процесс «развертывания» или переноса вашего кода с локального компьютера на этот сервер.

Типы хостинга для фронтенда

Для фронтенд-разработчиков (особенно для статических сайтов на HTML/CSS/JS) существуют специализированные платформы.
Они бесплатны, работают быстро и часто подключаются напрямую к GitHub.

Платформа	Особенности	Для чего лучше
GitHub Pages	Встроено в GitHub. Бесплатно.	Портфолио, учебные проекты, курсовые.
Netlify	Удобный Drag-and-drop. Авто-обновление при пуше.	Быстрые запуски, коммерческие лендинги.
Vercel	Лидер индустрии. Идеально для современных фреймворков.	Проекты на Vite, React, Next.js.

Механизм работы: Статика vs Динамика

Важно понимать: на этих хостингах мы размещаем только статические файлы (HTML, CSS, JS, картинки).

- Как это работает: Пользователь запрашивает сайт -> Сервер просто отдает ему готовый файл.
- На таком хостинге нельзя запустить базу данных или серверный код (Node.js, Python, PHP). Для этого нужны другие сервисы.

1. Понятие

Понятие домена

- Когда вы публикуете сайт бесплатно, вы получаете поддомен сервиса:
  - username.github.io/project-name
  - project-name.netlify.app
- Это отличный вариант для учебных работ. В будущем к этим же сервисам можно будет привязать «красивое» имя (например, my-portfolio.com).

Как происходит процесс (Workflow)

Современный деплой выглядит так:

- Вы пишете код и делаете git commit.
- Вы делаете git push в репозиторий на GitHub.
- Хостинг «видит» изменения, сам запускает сборку (npm run build) и обновляет сайт.

Это называется Continuous Deployment (Непрерывное развертывание).

Вывод:

- Для фронтенд-разработчика сегодня нет необходимости арендовать сложные серверы.
- Такие инструменты, как GitHub Pages и Netlify, делают процесс публикации мгновенным и бесплатным.
- Главное — правильно подготовить файлы (см. чек-лист выше).

5. Публикация на GitHub Pages

GitHub Pages — это бесплатный сервис хостинга статических сайтов прямо из ваших репозиториев на GitHub.
Если в вашем проекте только HTML, CSS и JS (без папок dist и сборщиков), этот метод — самый надежный.

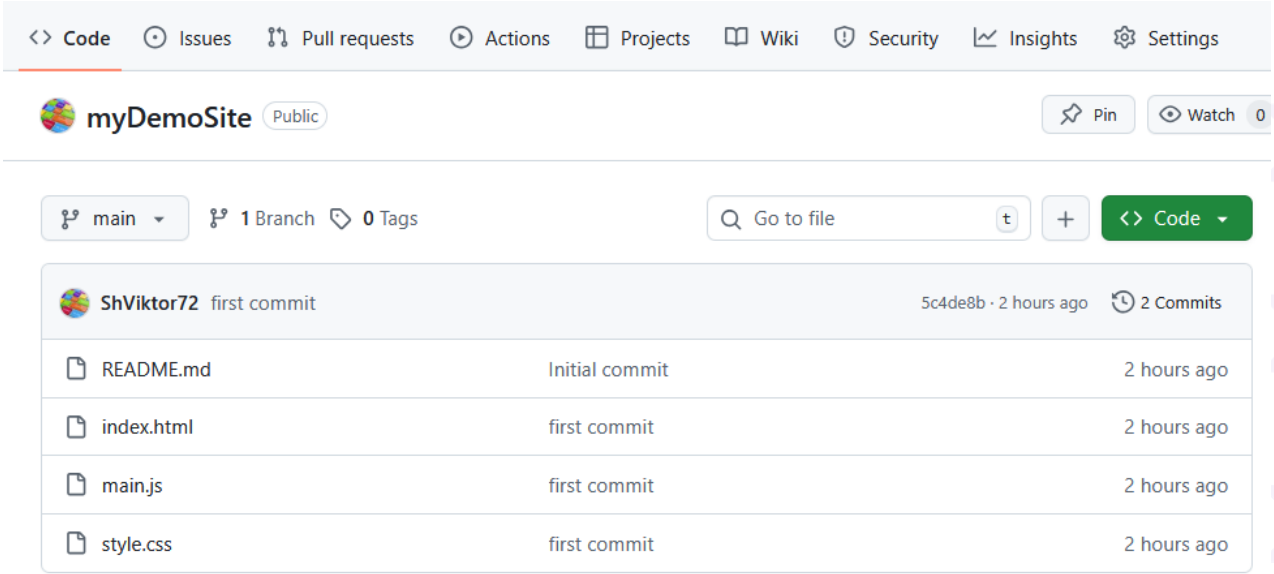
Подготовка репозитория

Чтобы сайт заработал, GitHub должен понимать, какой файл открывать первым.

- Главное правило: В корне вашего проекта (не в папке src или html) обязательно должен лежать файл с названием index.html. Именно его сервер ищет по умолчанию.
- Убедитесь, что репозиторий Public (Публичный). В бесплатном тарифе GitHub Pages работает только для открытых проектов.

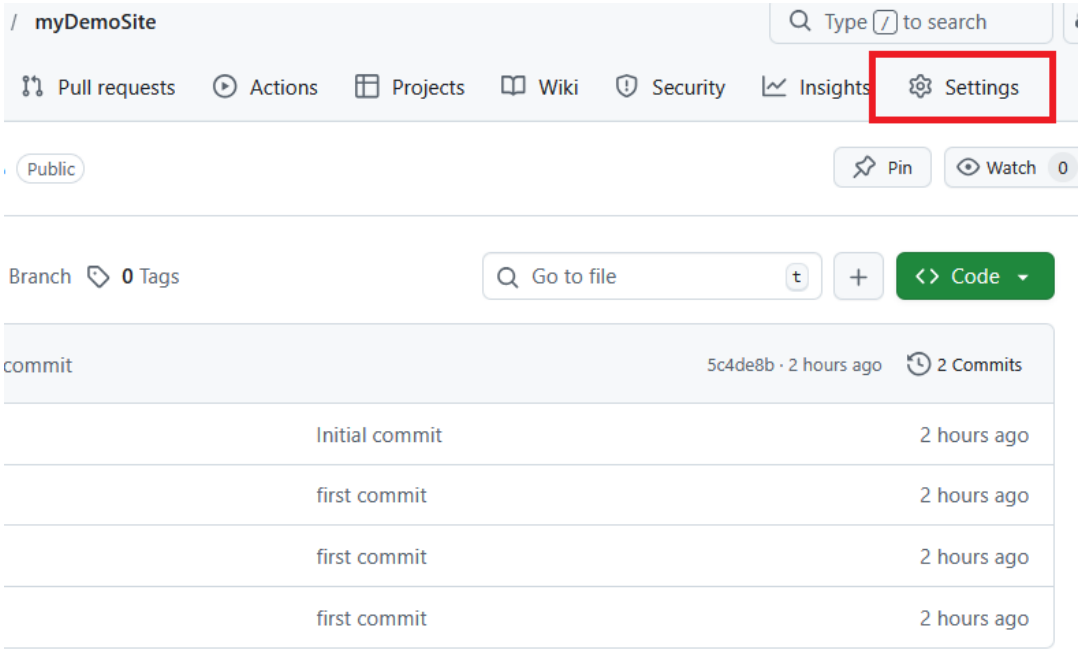
Пошаговый алгоритм публикации

Шаг 1: Загрузка кода
Загрузите ваш проект в репозиторий GitHub привычным способом (через Git или кнопку "Upload files").



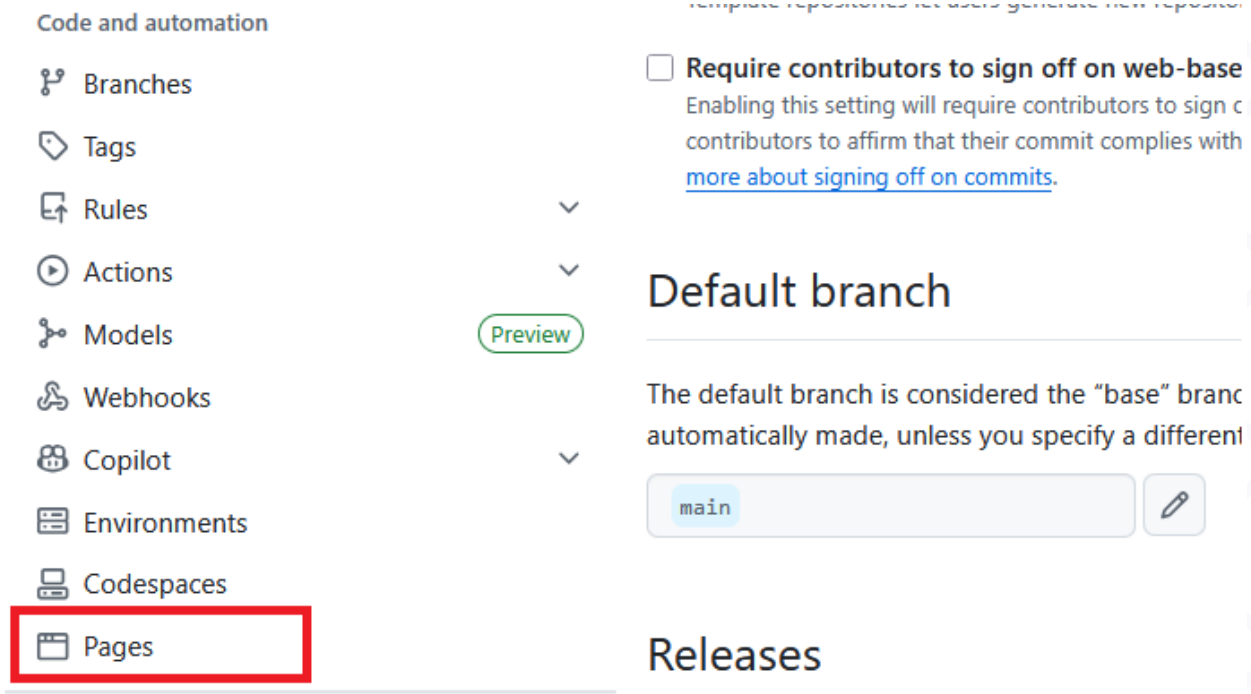
Шаг 2: Вход в настройки

В верхней панели репозитория выберите вкладку Settings (Настройки).



Шаг 3: Переход в раздел Pages

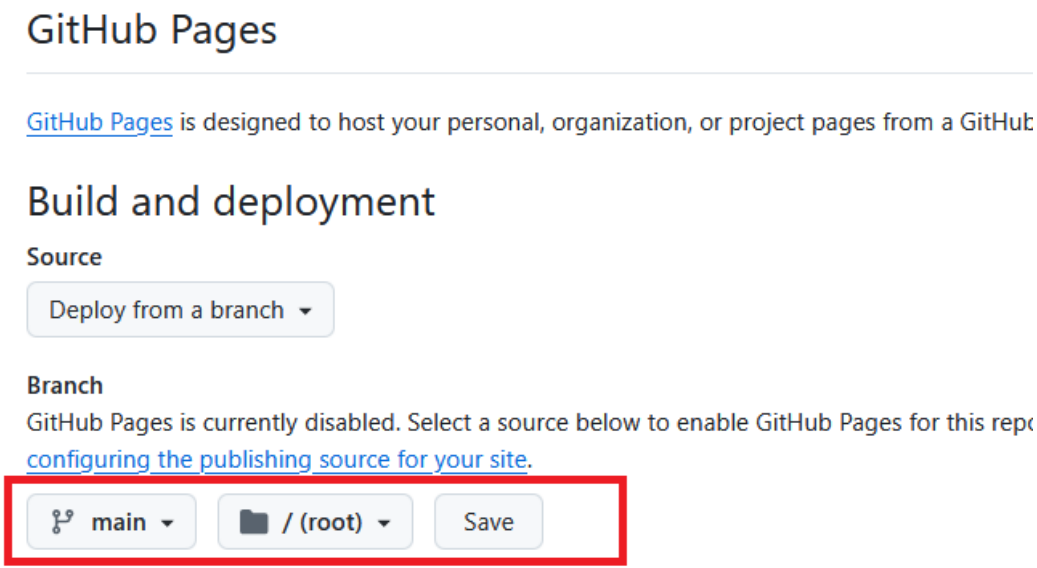
В боковом меню слева найдите блок Code and automation и выберите пункт Pages.



Шаг 4: Настройка источника (Source)

В разделе Build and deployment убедитесь, что выбрано:

- Source: Deploy from a branch.
- Branch: Выберите вашу основную ветку (обычно main или master).
- Folder: Выберите /(root). Это значит, что сайт находится прямо в корне репозитория.
- Нажмите Save



Как понять, что всё заработало?

После нажатия Save на этой же странице (сверху) появится статусная панель:

- Желтый круг: GitHub готовит ваш сайт (процесс Deployment).
- Зеленая галочка: Сайт готов! Появится ссылка вида: https://username.github.io/repository-name/.

Совет: Если вы перешли по ссылке и видите ошибку 404 — подождите 1–2 минуты. GitHub нужно время, чтобы обновить свои серверы.

GitHub Pages

GitHub Pages is designed to host your personal, organization, or project pages from a GitHub reposito

Your site is live a https://shviktor72.github.io/myDemoSite/

Last deployed by ShViktor72 2 hours ago

Visit site

Build and deployment

Source

Deploy from a branch

Branch

Your GitHub Pages site is currently being built from the `main` branch. Learn more about configuring the

shviktor72.github.io/myDemoSite/

Привет, я будущий фронтенд-разработчик!

Я изучаю HTML, CSS и JavaScript. Этот сайт — моя первая работа.

Мои навыки

Типичные «ловушки» при простой публикации

Ошибка 404 (File not found).
Если по ссылке открывается ошибка 404, проверьте название главного файла. Он должен называться точно index.html (все маленькими буквами). Если он называется Index.html или main.html, сервер его не увидит.

Пропавшие картинки.
Самая частая проблема. На компьютере картинки есть, на сайте — нет.
Причина: Неверные пути.
Решение: Проверьте, чтобы ссылки в HTML были относительными (img/photo.jpg), а не начинались с косой черты (/img/photo.jpg) или буквы диска (C:/...).

Кэширование.
Вы изменили код, запушили на GitHub, а на сайте всё по-старому.
Решение: Браузер «запомнил» старую версию. Нажмите Ctrl + F5 (жесткая перезагрузка с очисткой кэша), чтобы увидеть изменения.

Преимущества этого способа

- Автоматизация: Как только вы делаете git push в основную ветку, GitHub автоматически пересобирает сайт. Через пару минут изменения уже видны в интернете.
- Бесплатный HTTPS: Ваш сайт сразу получает замочек в адресной строке (безопасное соединение).
- Вечное хранение: Сайт будет работать до тех пор, пока существует ваш репозиторий.

Вывод:

- Простая публикация на GitHub Pages — это идеальный способ хостинга для новичка.
- Она приучает к порядку в именах файлов и структуре проекта.
- Если ваш index.html лежит в корне, а пути относительны — ваш сайт будет доступен всему миру за два клика в настройках.

Итоги лекции:

- Сборка — это мост: Это процесс превращения кода, удобного для разработчика, в код, оптимизированный для браузера (минификация, сжатие, удаление мусора).
- Production vs Development: В разработке мы ценим читаемость, в релизе — скорость загрузки и минимальный вес файлов.
- Чек-лист — залог успеха: Большинство ошибок при публикации связаны с неверными путями (абсолютные вместо относительных) и регистром имен файлов.
- Хостинг стал доступным: Сервисы вроде GitHub Pages, Netlify и Vercel позволяют бесплатно опубликовать проект и настроить автоматическое обновление сайта при каждом обновлении кода (CD).
- Главное правило публикации: Всегда проверяй index.html в корне проекта и используй только относительные пути к ресурсам.

Контрольные вопросы

- Зачем удалять console.log перед публикацией?
- Почему на Windows сайт может работать, а после деплоя на сервер — нет?
- Что такое минификация? (Ответ: Удаление пробелов, комментариев и сокращение имен переменных для уменьшения веса кода).
- Как должен называться главный файл сайта и где он должен лежать для успешного деплоя на GitHub Pages?
- В чем разница между путем /img/logo.png и ./img/logo.png при публикации?
- Что произойдет, если в HTML указан путь C:/Users/MyDoc/project/style.css после загрузки на хостинг?
- Как принудительно обновить сайт в браузере, если вы запушили изменения, но видите старую версию?
- В каких случаях можно обойтись без сборщика (Vite/Webpack)?

Домашнее задание:

https://ru.hexlet.io/courses/js-basics

хекслет колледж

@HEXLY.KZ