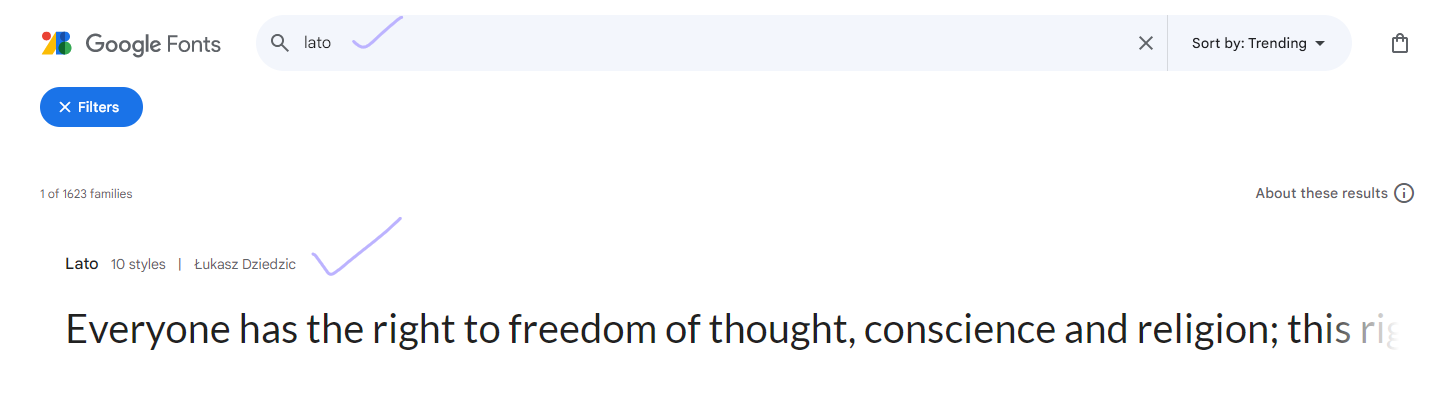
**Часть 1 - Создание навигационного меню с несколькими**

**уровнями и вложенными внутрь элементами**

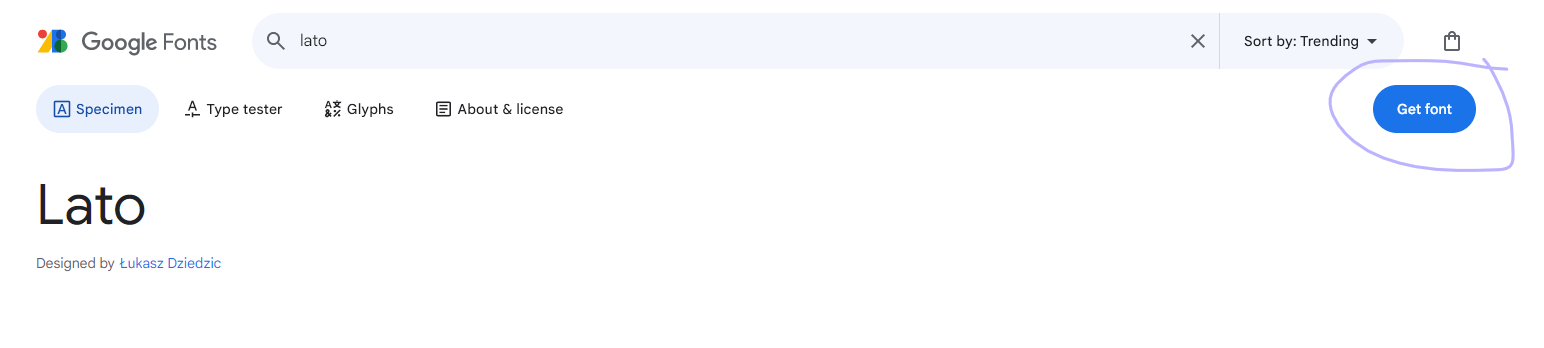
С помощью батника создайте структуру проекта. Переместите картинки в нужную папку. Текст вам предоставлен в отдельном файле.



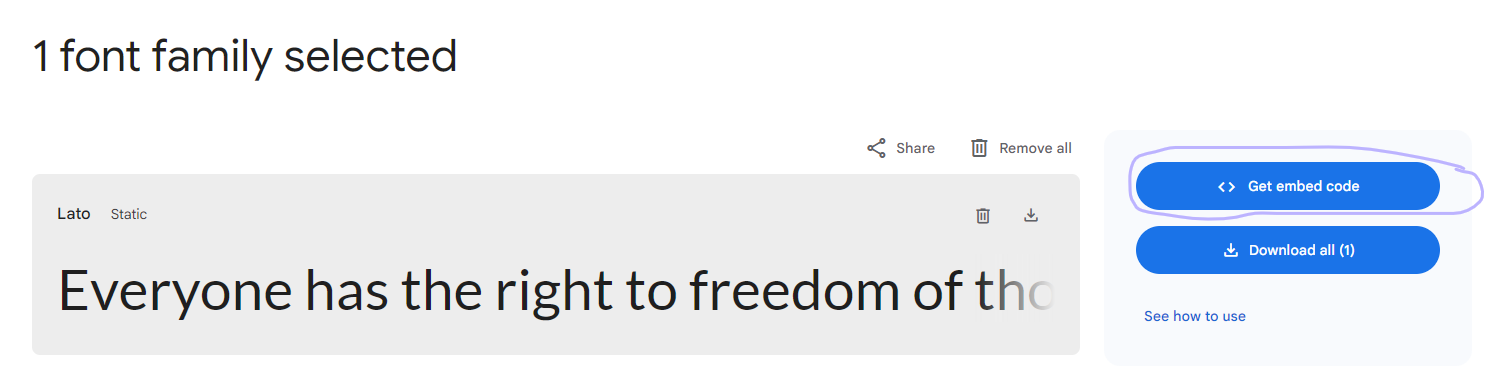
Зайдите на сайт google fonts и выберите шрифт Lato.



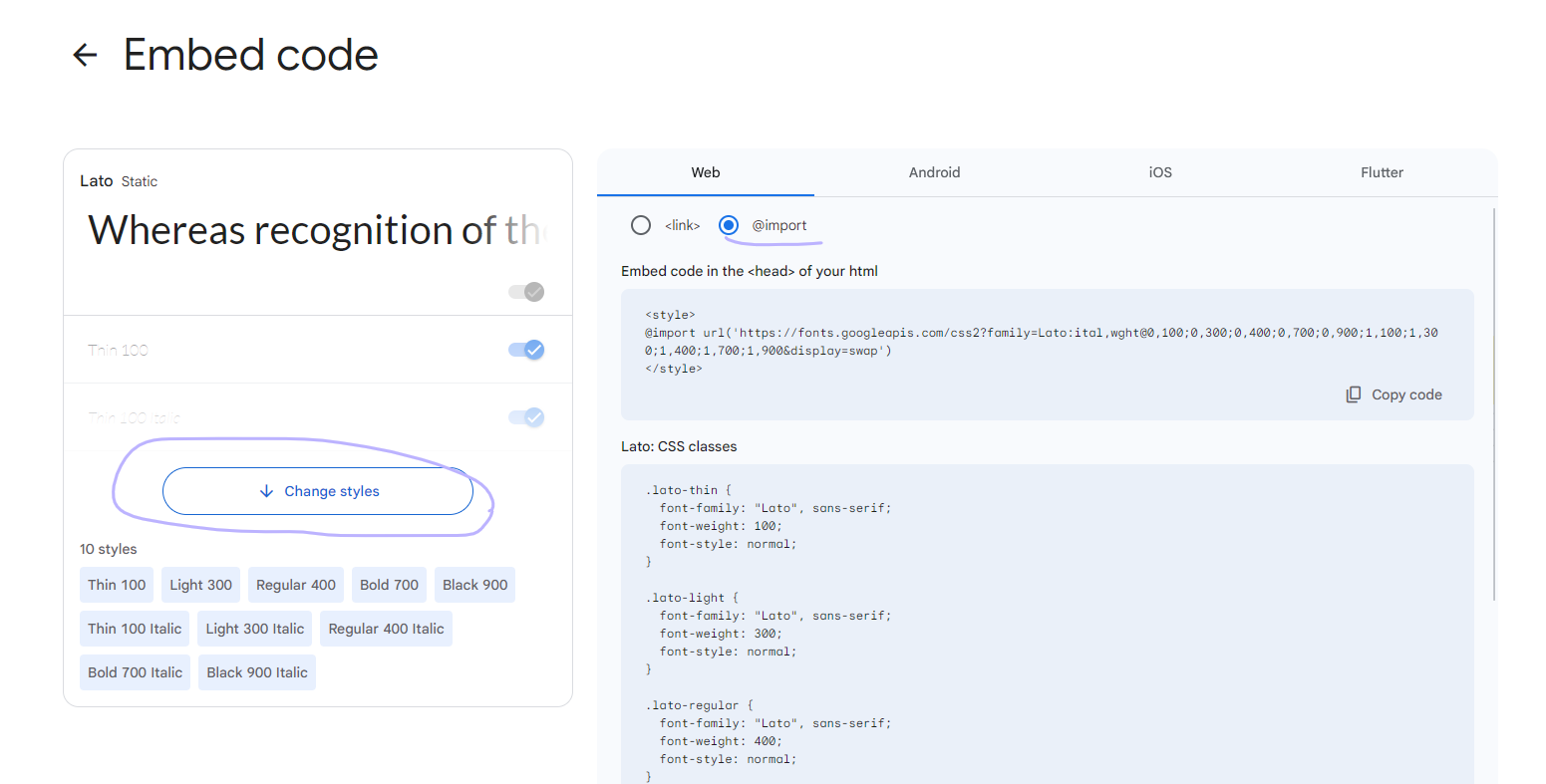
Нажмите кнопку для использования шрифта.

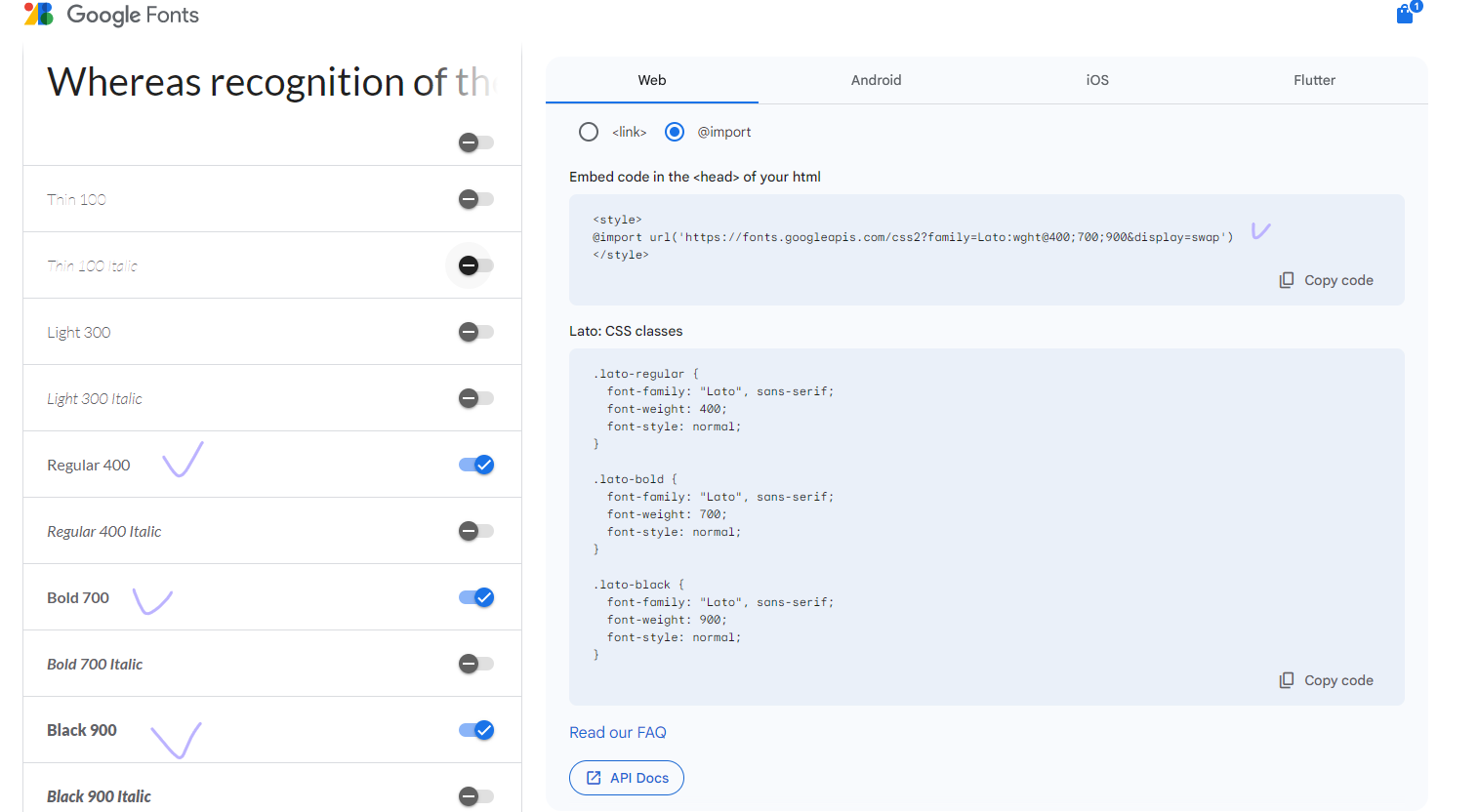


Снова нажмите кнопку для вставки шрифта через код.

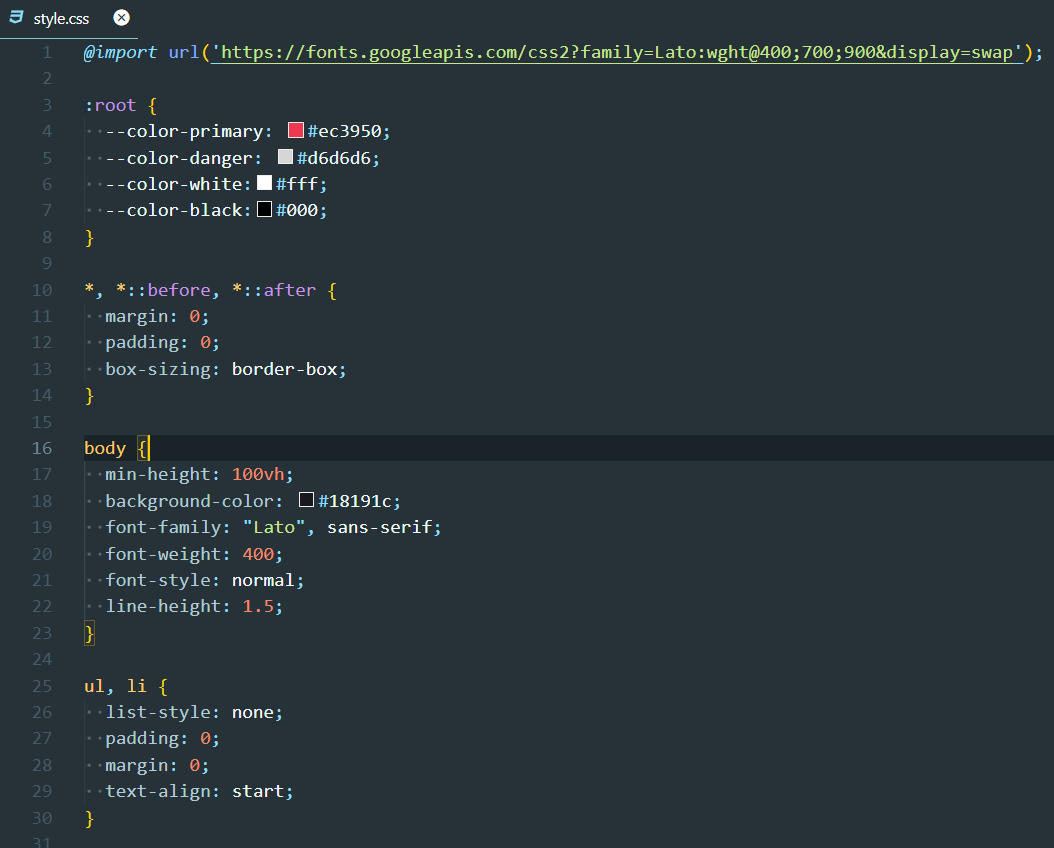


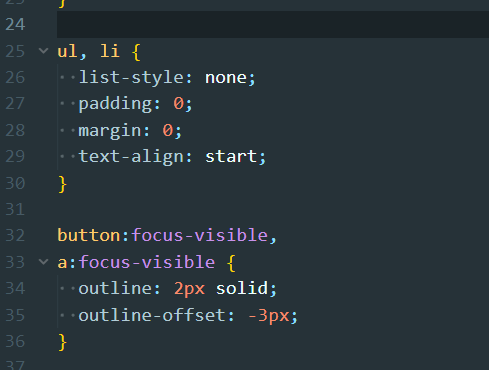
Все варианты шрифта нам не нужны, поэтому выполните настройку.



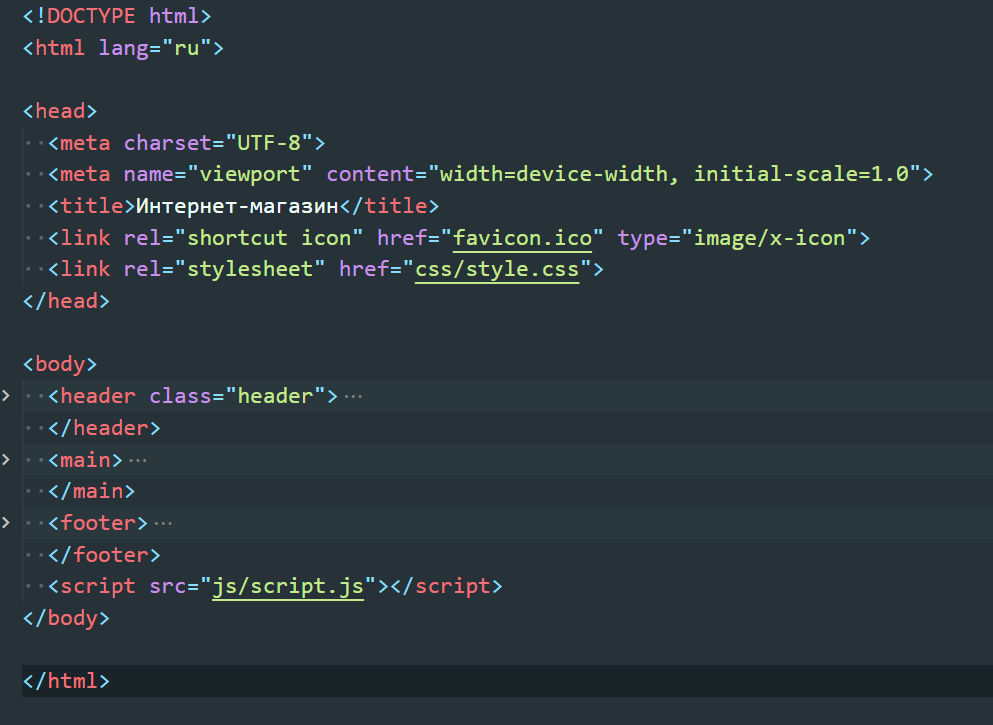


Вставьте полученный код в css. Далее назначьте шрифт по умолчанию для всего документа и сделайте сброс.

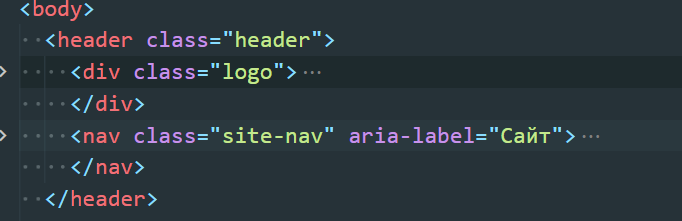




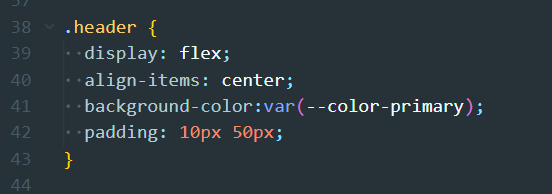
Сделаем разметку.



Далее в шапке сделайте блоки для логотипа и меню.



Для шапки задайте стили.



Внутри блока logo разместите логотип **с помощью тега svg**. Вам достался сломанный логотип, почините его, исправив значения в коде. Нельзя использовать никакой графический редактор.

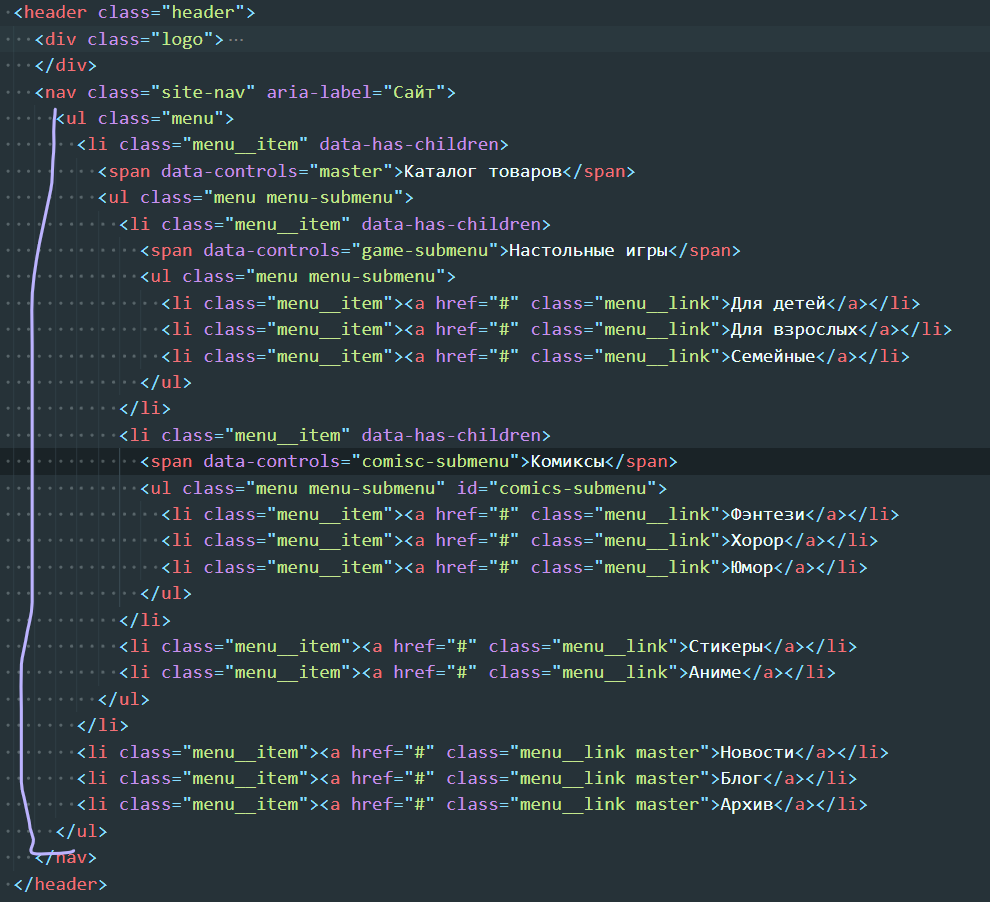
Размер лого: 440\*50.

Цвета: #fff – белый, #424242 – черный, #ec3950 – красный.

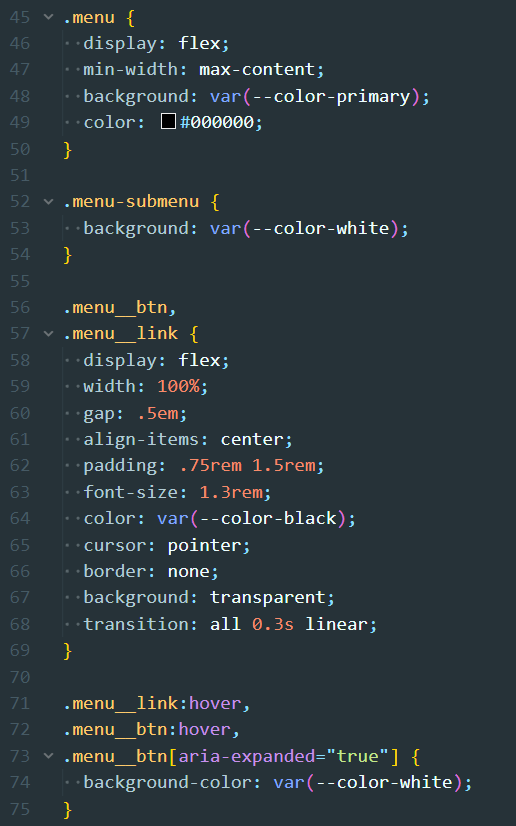
Слева логотип, который должен у вас получиться.

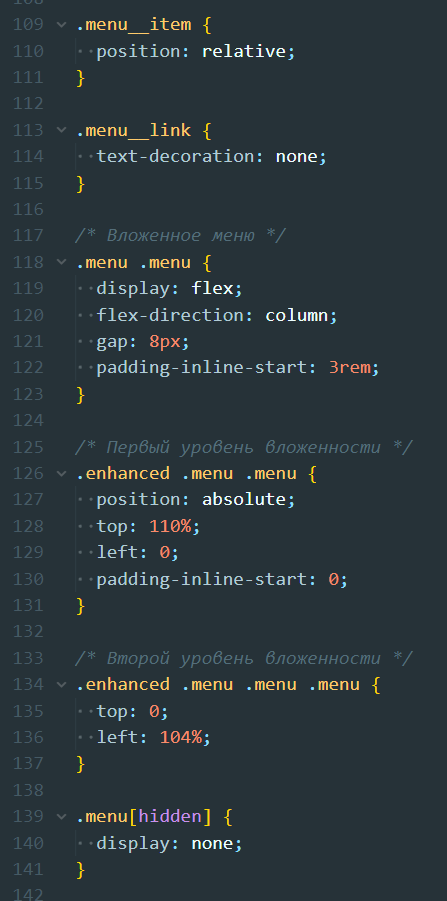
 

Далее сделайте разметку для меню. Будьте внимательны, очень точно соблюдайте вложенность тегов, чтобы корректно сформировать 3 уровня вложенности в меню.

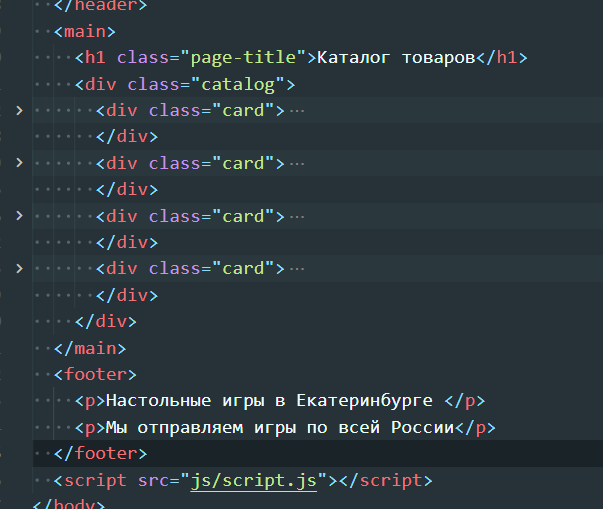


Далее выполните оформление меню. Если какие-то классы встречаются в разметке, но не встречаются в css, значит они нужны для взаимодействия (js).

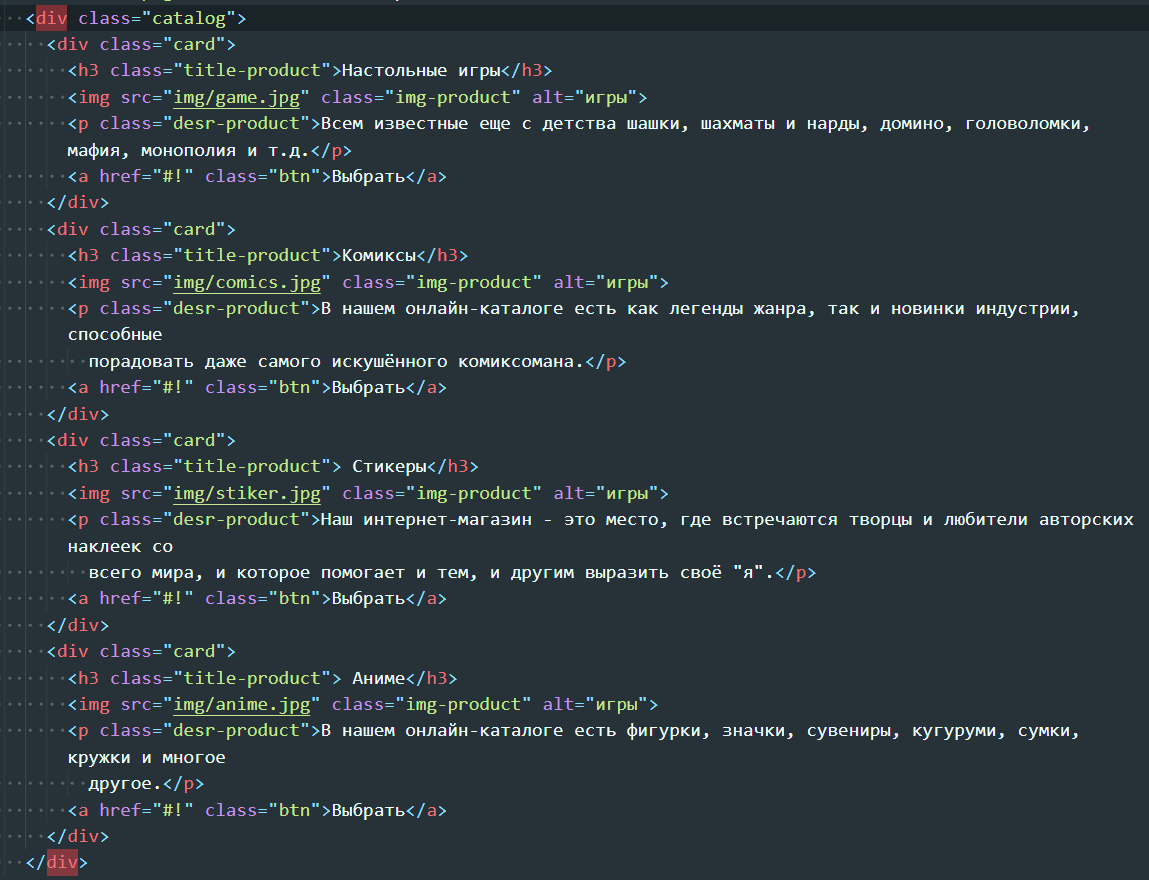
 



Далее делаем начальную разметку для main и footer.

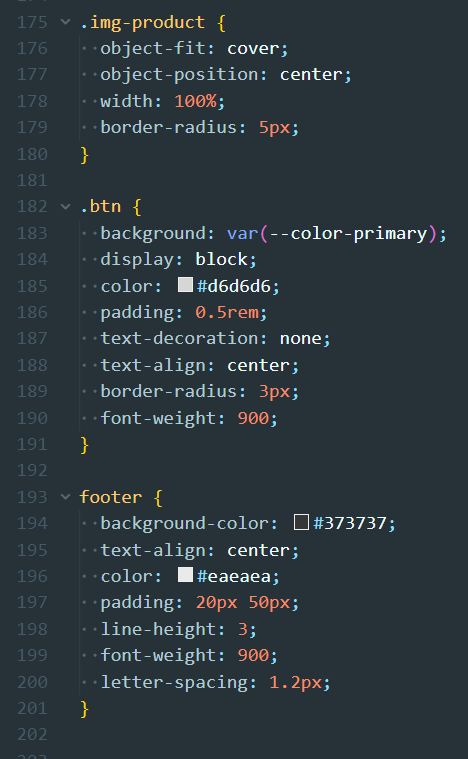


Добавим в каталог содержимое каждой карточки.



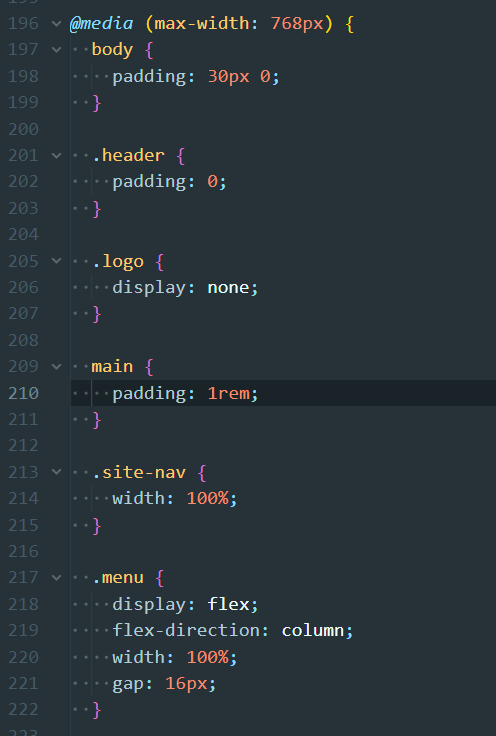
Выполним оформление.





Далее проверьте как выглядит страница на других экранах. Вы видите, что меню крашится, а вот область основного контента адекватно адаптируется, так как мы применили в гридах, тексте и отступах формулы и функции.

Починим меню на маленьких экранах. Для этого напишем классический медиа-запрос.





Меню до сих пор странное, напишем скрипт, чтобы вложенные пункты меню появлялись по щелчку.

*// Показываем пользователю простую навигацию в случае, если не удаётся загрузить JavaScript.*

*// Если JavaScript-файл загружен, заменяем span на кнопки и вешаем дополнительные ARIA-атрибуты.*

    const nav = document.querySelector('.site-nav')

    nav.classList.add('enhanced')

    const submenus = document.querySelectorAll('.menu\_\_item[data-has-children]')

    const dropdowns = document.querySelectorAll('.menu\_\_item[data-has-children] > .menu')

    const icon = `

      <svg

        width="24px"

        height="24px"

        viewBox="0 0 24 24"

        aria-hidden="true"

        class="menu\_\_btn-icon"

      >

      <path fill="currentColor" d="M5.64645 8.64645c.19526-.19527.51184-.19527.7071 0L12 14.2929l5.6464-5.64645c.1953-.19527.5119-.19527.7072 0 .1952.19526.1952.51184 0 .7071L12 15.7071 5.64645 9.35355c-.19527-.19526-.19527-.51184 0-.7071Z"></path>

      </svg>

    `

*// Находим подменю, заменяем в нём span на кнопку*

    submenus.forEach((*item*) => {

      const dropdown = *item*.querySelector(':scope > .menu')

      dropdown.setAttribute('hidden', '')

      const span = *item*.querySelector(':scope > span')

      const text = span.innerText

      const ariaControlsId = span.dataset.controls

      const button = document.createElement('button')

*// Добавляем класс и необходимые ARIA-атрибуты*

      button.classList.add('menu\_\_btn')

      button.setAttribute('aria-expanded', 'false')

      button.setAttribute('aria-controls', ariaControlsId)

      button.innerText = text

*// Добавляем иконку к кнопке, чтобы визуально было понятно открыто меню или нет*

      button.innerHTML += icon

      span.replaceWith(button)

      button.addEventListener('click', function (*e*) {

        toggleDropdown(button, dropdown)

      })

*// Обрабатываем нажатие на Esc*

      dropdown.addEventListener('keydown', (*e*) => {

*e*.stopImmediatePropagation()

*if* (*e*.keyCode === 27 && focusIsInside(dropdown)) {

          toggleDropdown(button, dropdown)

          button.focus()

        }

      }, false)

    })

    function toggleDropdown(*button*, *dropdown*) {

*if* (*button*.getAttribute('aria-expanded') === 'true') {

*button*.setAttribute('aria-expanded', 'false')

*dropdown*.setAttribute('hidden', '')

      } *else* {

*button*.setAttribute('aria-expanded', 'true')

*dropdown*.removeAttribute('hidden')

      }

    }

    function focusIsInside(*element*) {

*return* *element*.contains(document.activeElement)

    }

    function collapseDropdownsWhenTabbingOutsideNav(*e*) {

*if* (*e*.keyCode === 9 && !focusIsInside(nav)) {

        dropdowns.forEach(function (*dropdown*) {

*dropdown*.setAttribute('hidden', '')

          const btn = *dropdown*.parentNode.querySelector('button')

          btn.setAttribute('aria-expanded', 'false')

        })

      }

    }

    function collapseDropdownsWhenClickingOutsideNav(*e*) {

      const target = *e*.target

      dropdowns.forEach(function (*dropdown*) {

*if* (!*dropdown*.parentNode.contains(target)) {

*dropdown*.setAttribute('hidden', '')

          const btn = *dropdown*.parentNode.querySelector('button')

          btn.setAttribute('aria-expanded', 'false')

        }

      })

    }

*// Закрываем навигацию, если протапались за её пределы*

    document.addEventListener('keyup', collapseDropdownsWhenTabbingOutsideNav)

*// Закрываем навигацию, если кликнули вне навигации*

    window.addEventListener('click', collapseDropdownsWhenClickingOutsideNav)

Теперь вы получили выпадающее меню и адаптивные карточки.

**Протестируйте работу всех элементов и их внешний вид на разных экранах.**

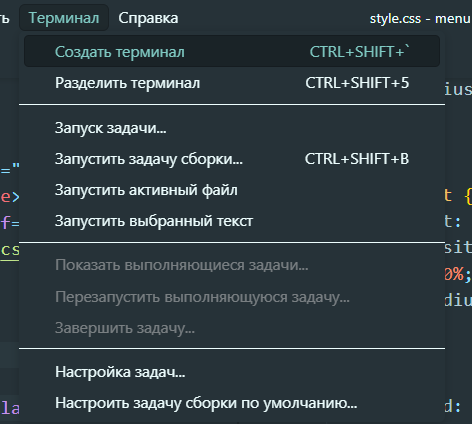
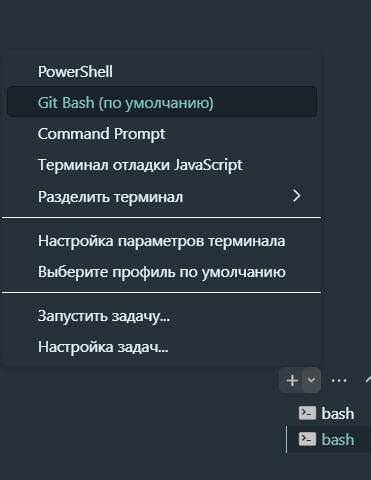
**Самостоятельная работа:** основываясь на своем профессиональном и пользовательском опыте улучшите внешний вид данной страницы и настройте взаимодействие с элементами (ховеры, анимации, тени, адаптив…). Вы можете использовать только те цвета, которые уже есть в коде. Также отследите как выглядит меню на разных экранах. При необходимости допишите медиазапросы.

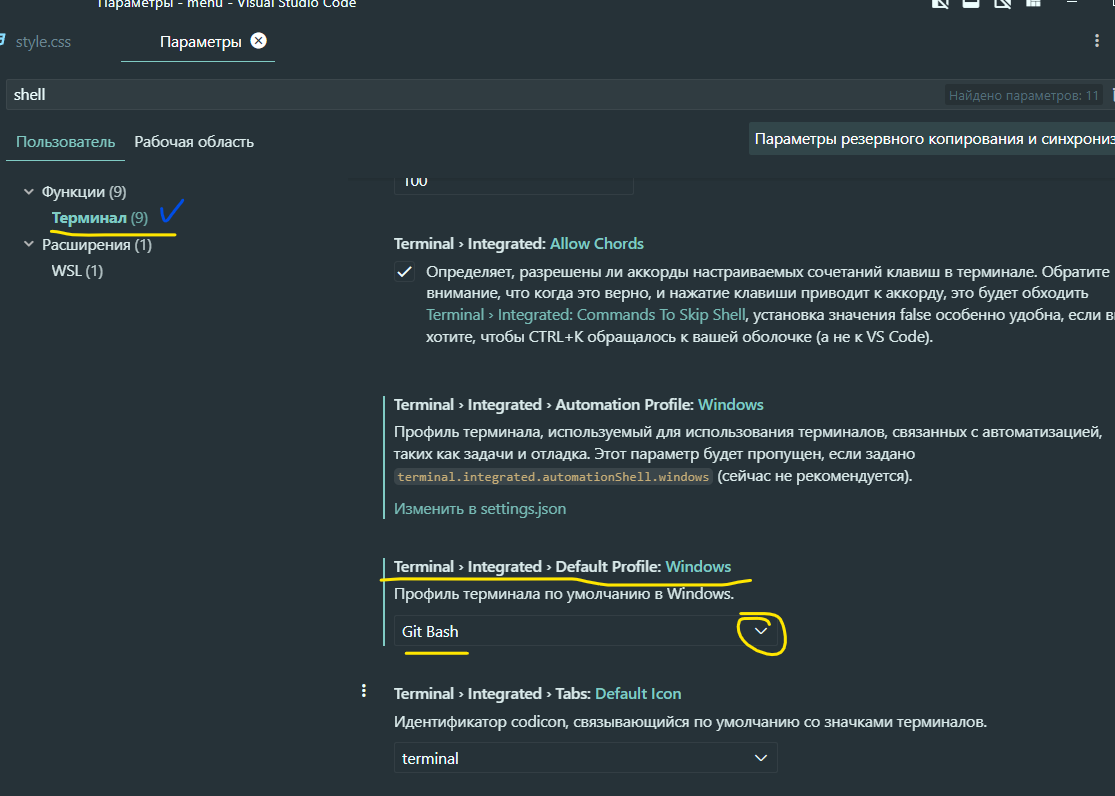
После того, как вы выполните первую часть работы переходите ко второй.

**Часть 2 – Заливка сайта на github и публикация через Pages**

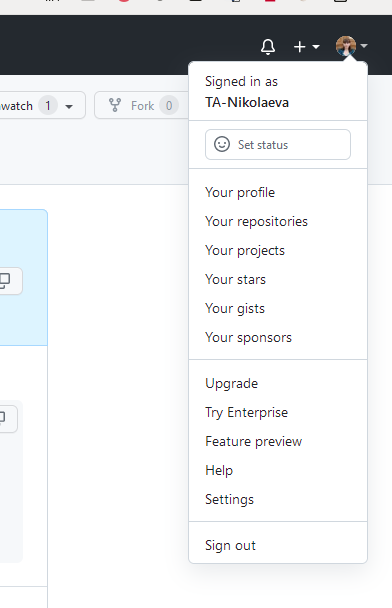
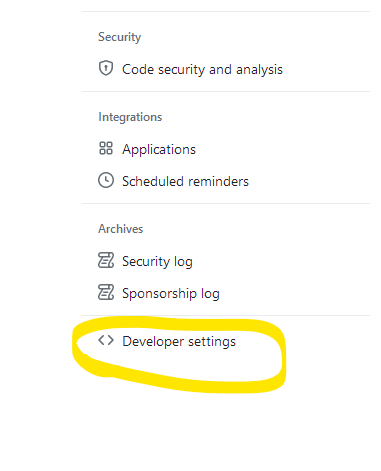
Скачайте и установите программу git <https://git-scm.com/downloads>. Все параметры при установке оставьте по умолчанию, просто нажимайте далее.

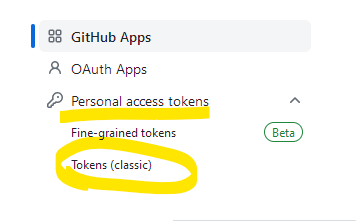
В VS Code запустите терминал и выберете терминалом по умолчанию git bash.

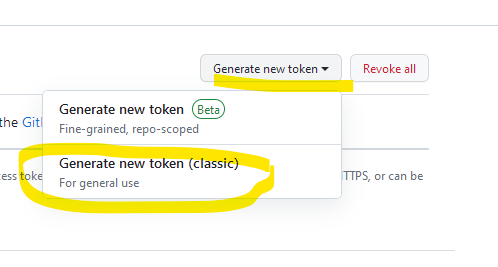
 

Чтобы в VS Code сделать терминалом по умолчанию git bash зайдите в параметры 

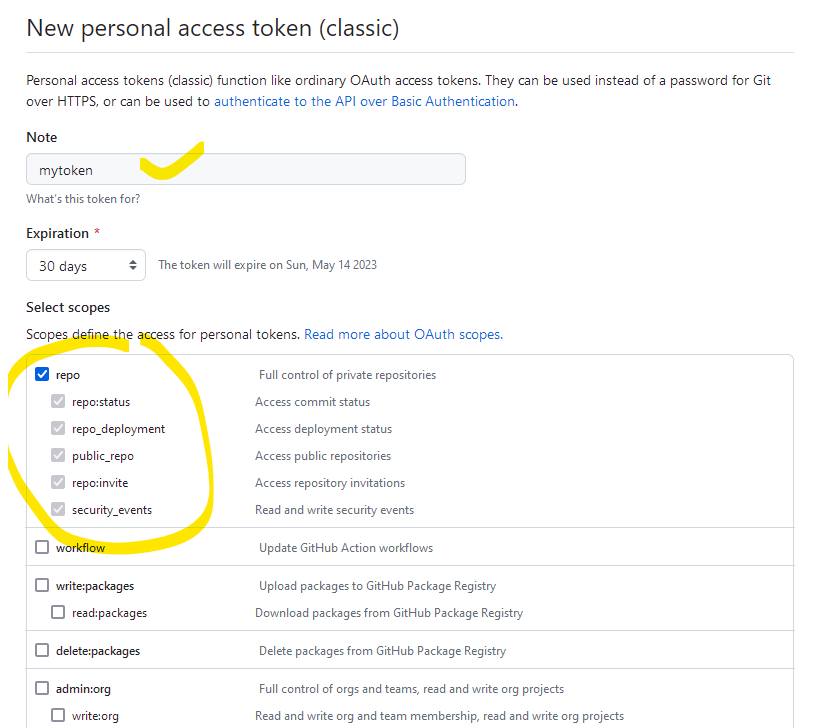
Перейдите на гитхаб, залогиньтесь, зайдите в настройки (Settings) и создайте токен



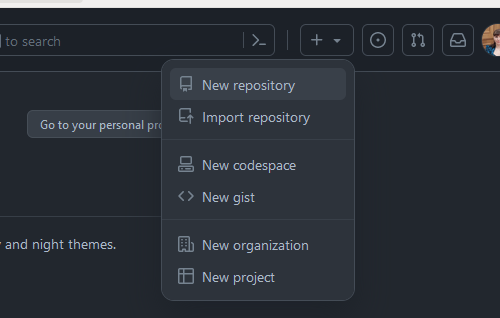


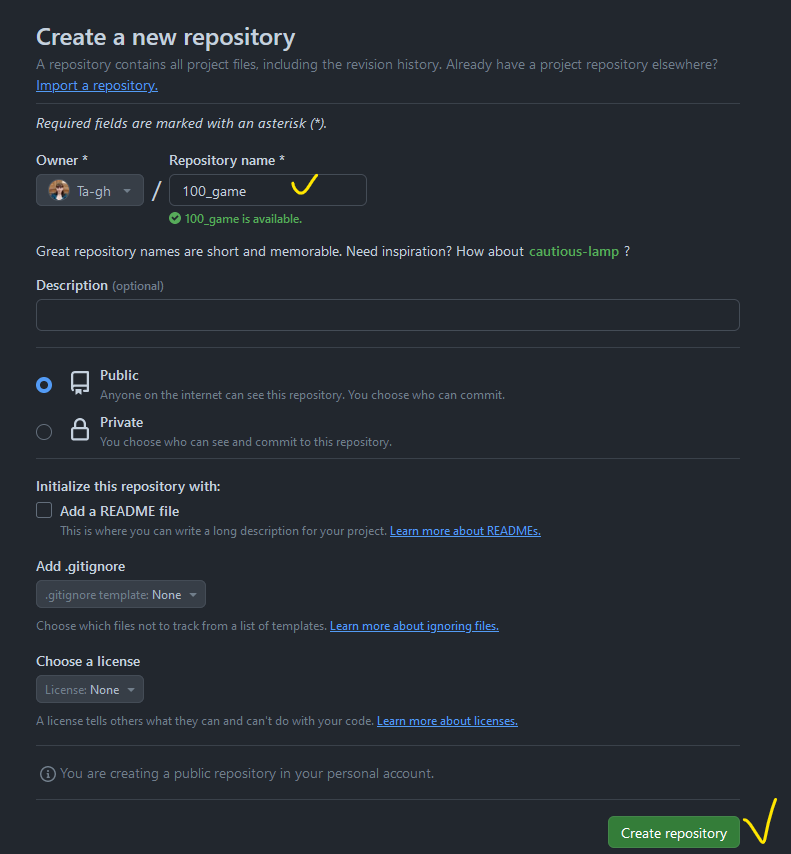
Укажите имя токена (любое) и права доступа



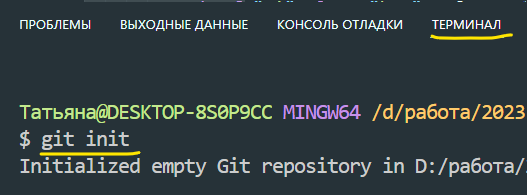
У вас сгенерируется случайная последовательность бит, это и будет токен. Он нужен для аутентификации и подтверждения прав на репозиторий. Скопируйте этот токен и сохраните куда-нибудь в укромное место (в текстовый документ), так как дальше он вам пригодится.

Cоздайте на gihub новый репозиторий.



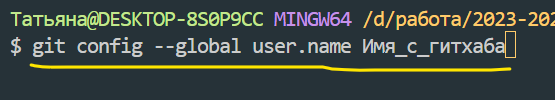


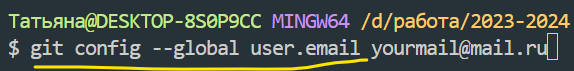
Инициализируйте работу Git (сделайте папку репозиторием).



Так как многие из вас еще дома не работали с git, то нужно задать конфигурационный файл.

Подтвердите права на работу с github. Введите имя с гитхаба и почту с помощью, которой вы зарегались на гитхабе. Надеюсь вы сообразите заменить данные на свои.





Далее все команды я беру с гитхаба.

Пометьте **все** файлы для отслеживания



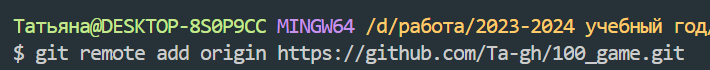
Зафиксируйте текущее состояние проекта с помощью коммита.



Переименуйте основную ветку проекта (была master станет main)



Свяжите локальный и удаленный репозитории

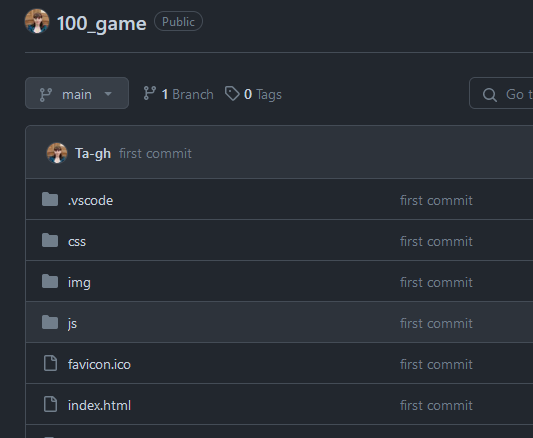


Пушните коммит в удаленный репозиторий

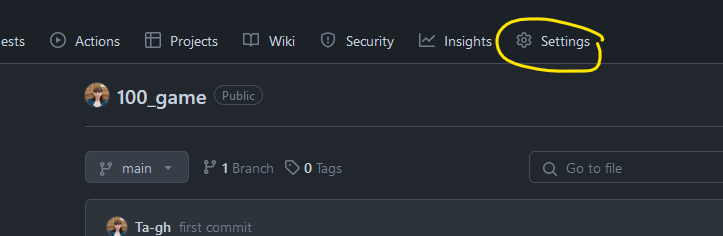


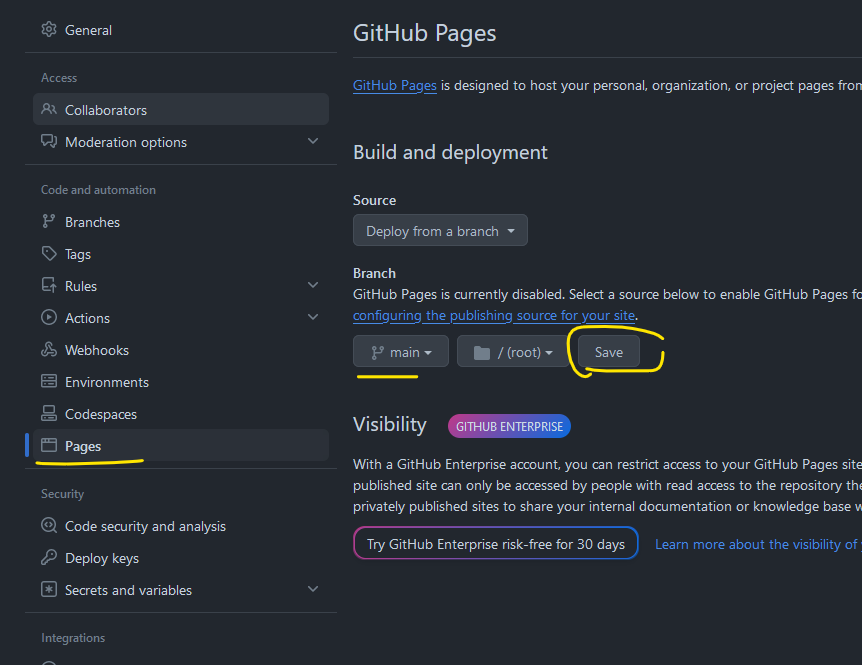
Если попросит ввести имя пользователя и пароль, то введите имя как вы зареганы на github и вместо пароля вставьте токен (сгенерировали ранее).

Обновите страницу на github и убедитесь, что все файлы залиты.



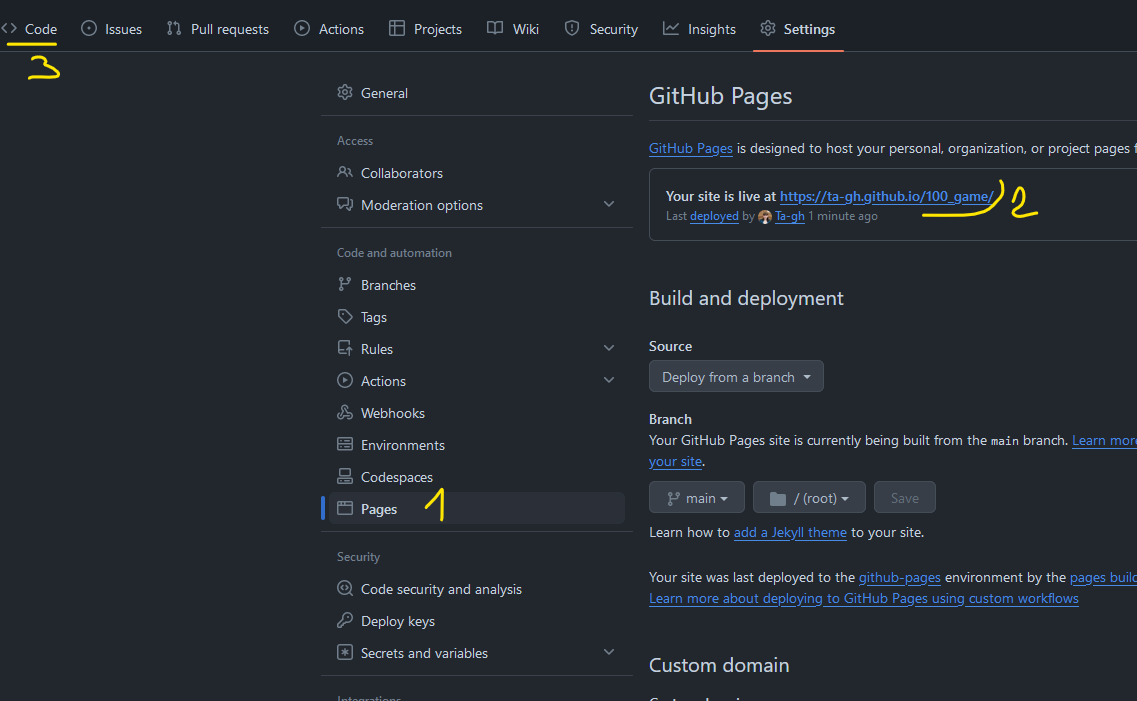
Далее переходим в настройки и находим раздел Pages. Выбираем ветку main для публикации.

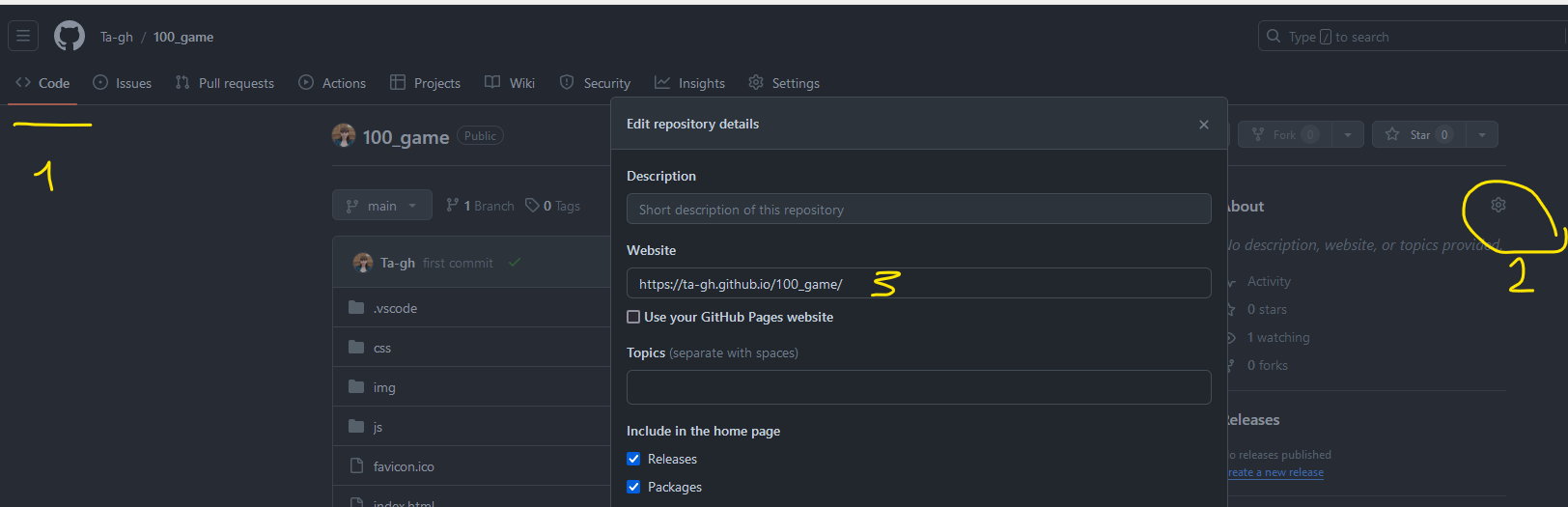




Дальше нужно немного подождать пока проект опубликуется и обновить страницу. У вас должна появиться ссылка. Не забывайте обновлять страницу.

Скопируйте эту ссылку и прейдите на вкладку Code. Нажмите на шестеренку и в поле website вставьте ссылку.





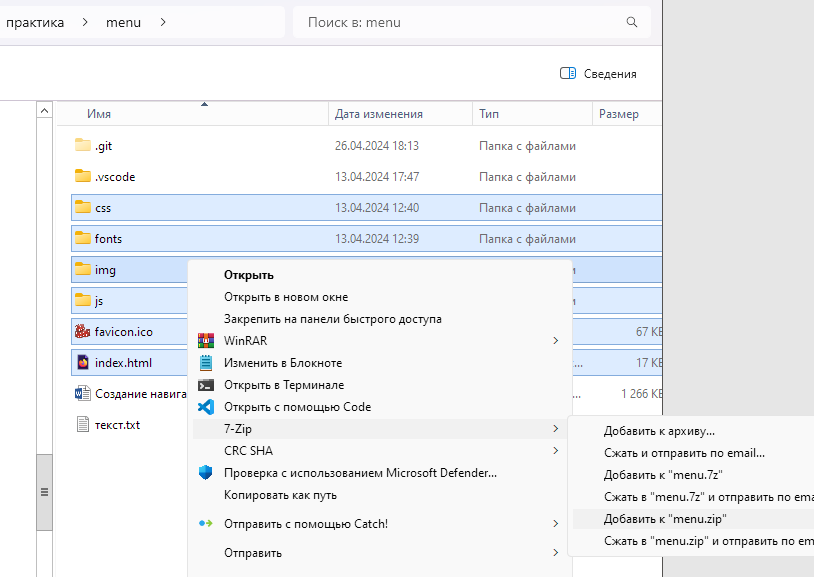
Перейдите на сайт и убедитесь, что все загрузилось, стили и скрипты работают, контент весь доступен.

После того, как тестирование сайта закончено, нужно разместить его на хостинге. Если проект реальный и коммерческий, лучше выбрать платный хостинг. В нашем случае хостинг с пробным тестовым периодом, но вот домен мы не покупаем.

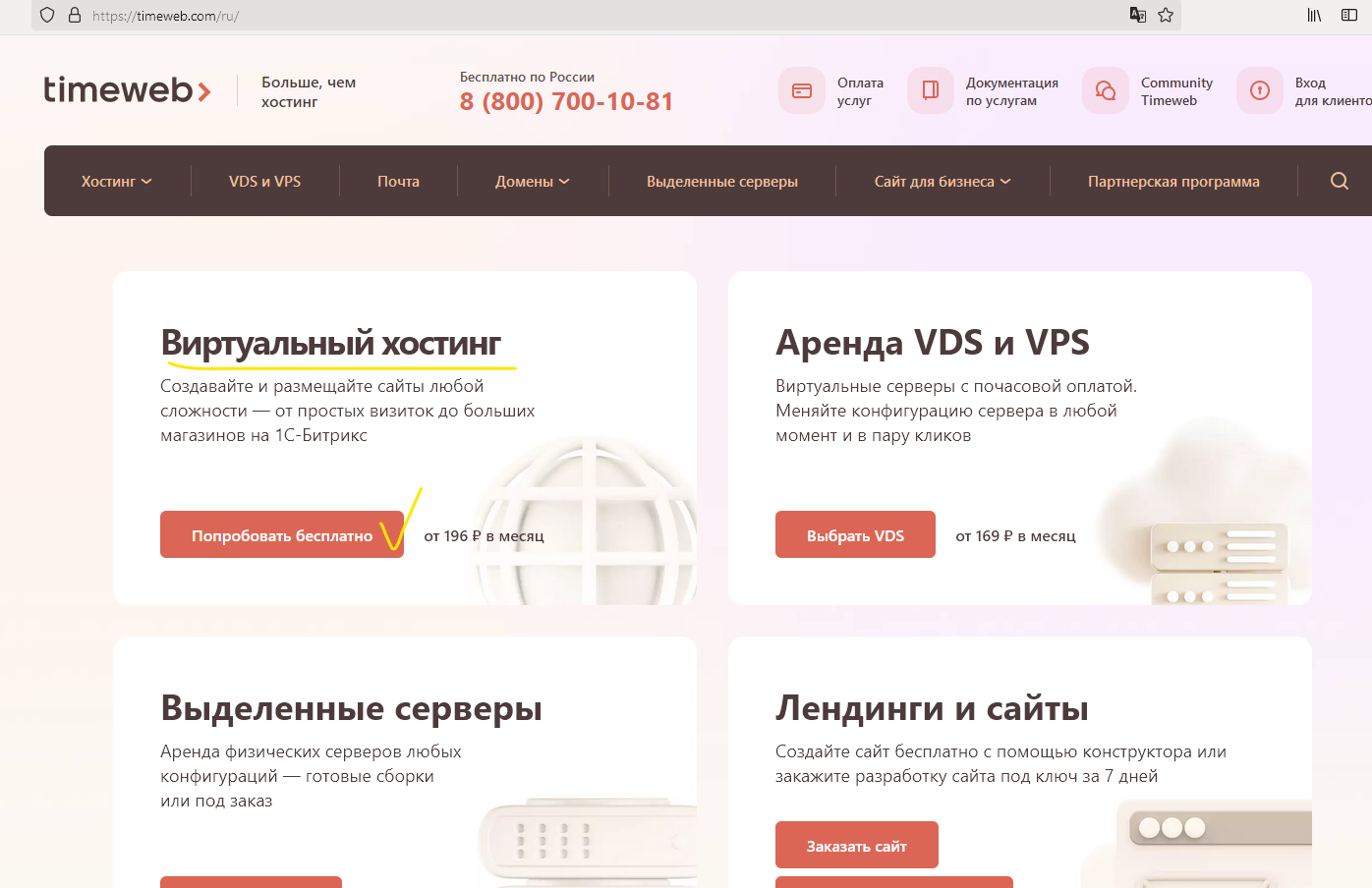
Покупка домена, заливка сайта на хостинг и регистрация его в поисковых системах, входит в обязанности веб-разработчика, так как не каждый заказчик умеет и может это сделать.

**Часть 3 – Заливаем сайт на хостинг**

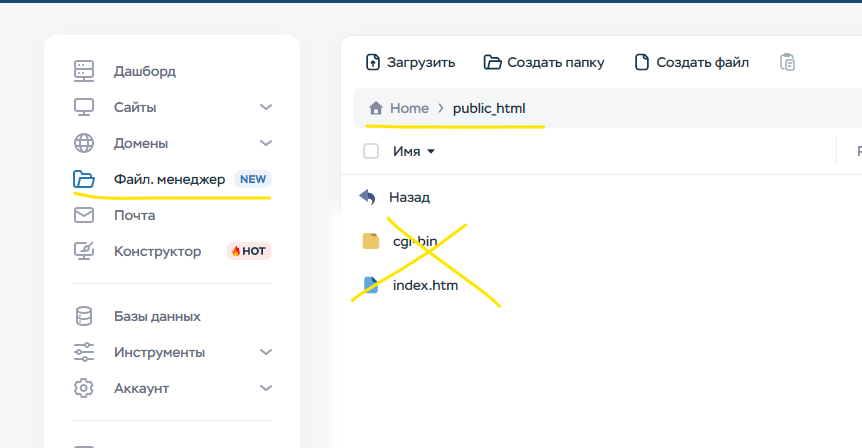
Перейдите в проводнике в папку проекта, выделите только файлы проекта и нажмите пкм, выберите сжать в zip и укажите название архива menu.

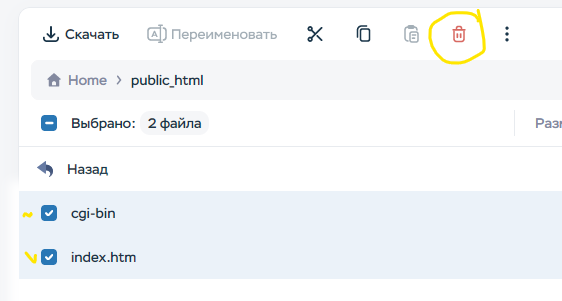


Зарегистрируйтесь на хостинге[**https://timeweb.com/ru/**](https://timeweb.com/ru/)с помощью учетки google или github. Так как у нас простой сайт и на него никто не будет заходить, то выбираем виртуальный хостинг (что это такое мы изучили на лекции).

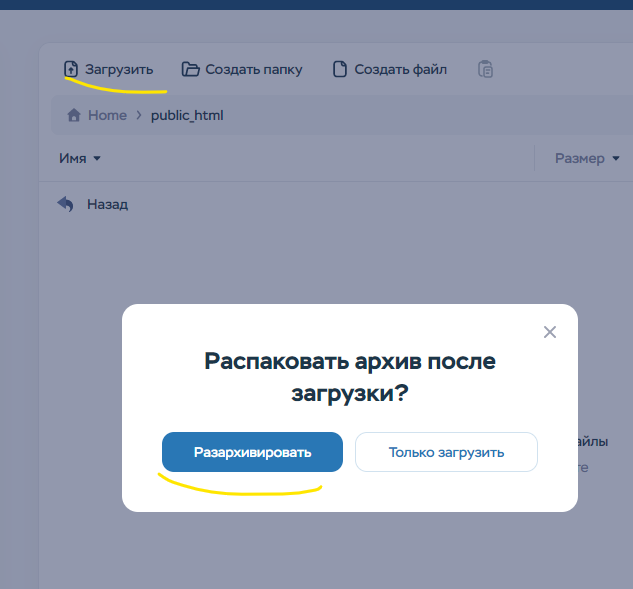


Зайдите в раздел файл менеджер, в папку public\_html и удалите папку и файл index.

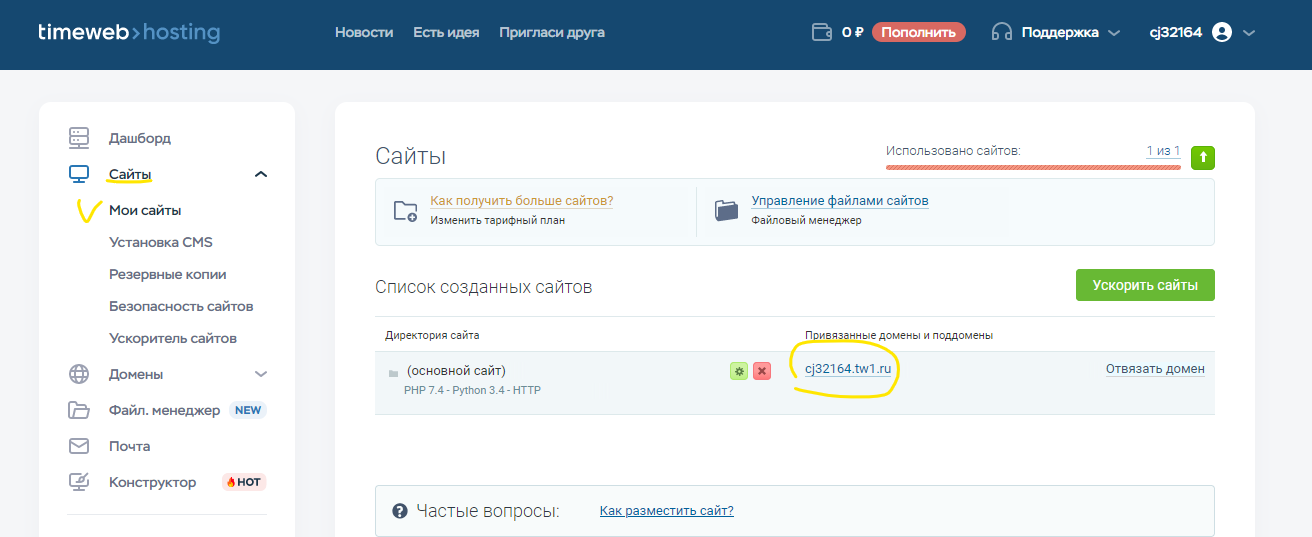




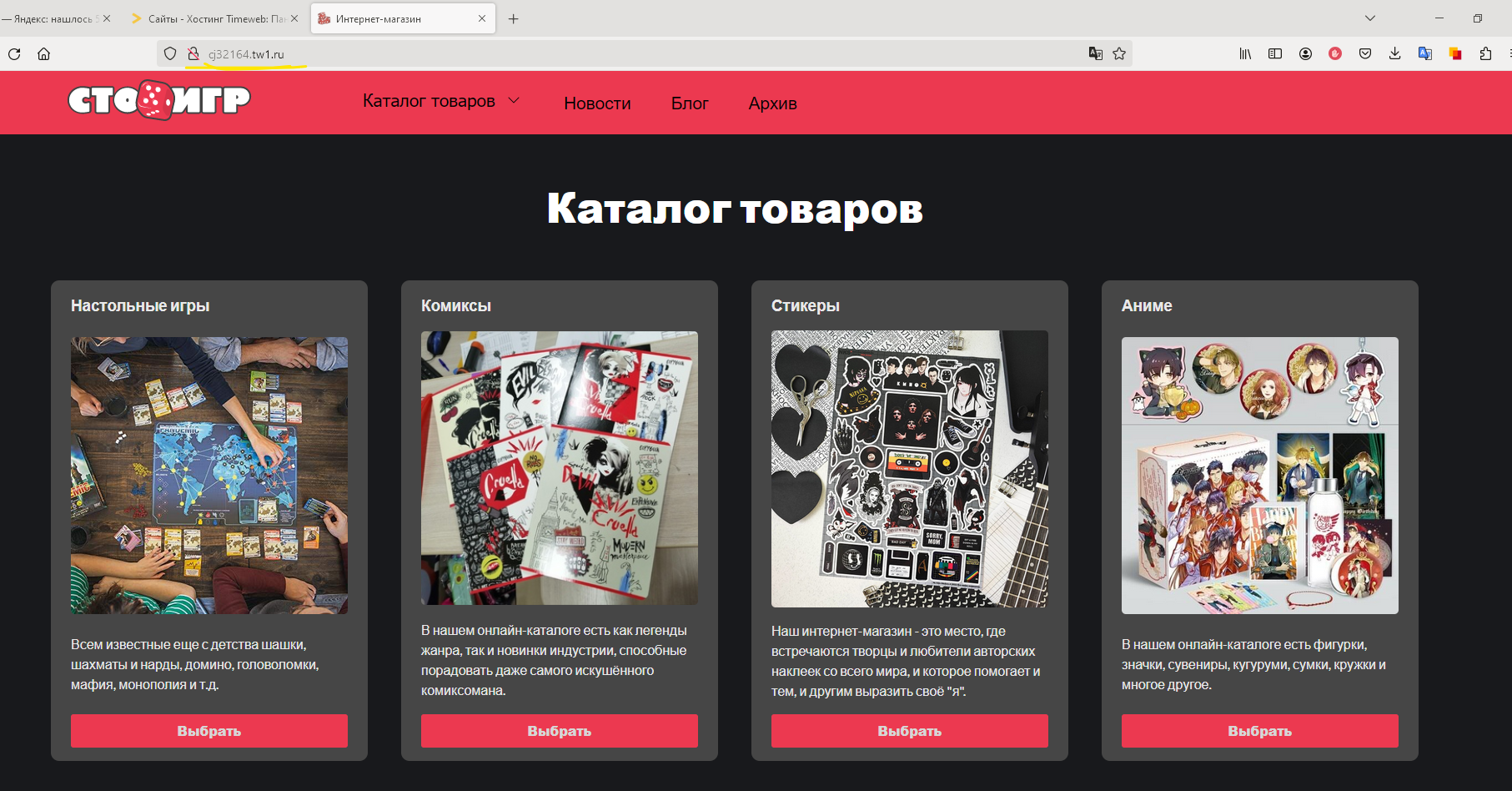
На хостинге нажмите загрузить и выберите архив, который создали ранее. И сразу же нажмите разархивировать.



Проверьте что сайт работает.



На других хостингах принцип публикации статичного сайта будет примерно таким же.



**Для проверки выполненной работы скиньте мне в личку 3 ссылки:**

1) ссылку на репозиторий на github (пример ссылки https://github.com/Ta-gh/100\_game.git)

2) ссылку на опубликованную страницу через Pages на github (пример ссылки ta-gh.github.io/100\_game/)

3) ссылку на опубликованный сайт на хостинге (пример ссылки http://cj32164.tw1.ru/ )