ААРОН БЕНДЖАМИН(AARON BENJAMIN)

Сделайте веб-приложение камеры — учебное руководство / часть: 1

УЧЕБНОЕ РУКОВОДСТВО

Перевод: С. Кузнецов, 2022 г.

Make a Camera Web App — Tutorial / Part: 1

Aaron Benjamin

Apr 20, 2018 · 8 min

https://medium.com/prototypr/make-a-camera-web-app-tutorial-part-1-ec284af8dddf

Сделайте веб-приложение камеры — учебное руководство / часть: 1

Аарон Бенджамин

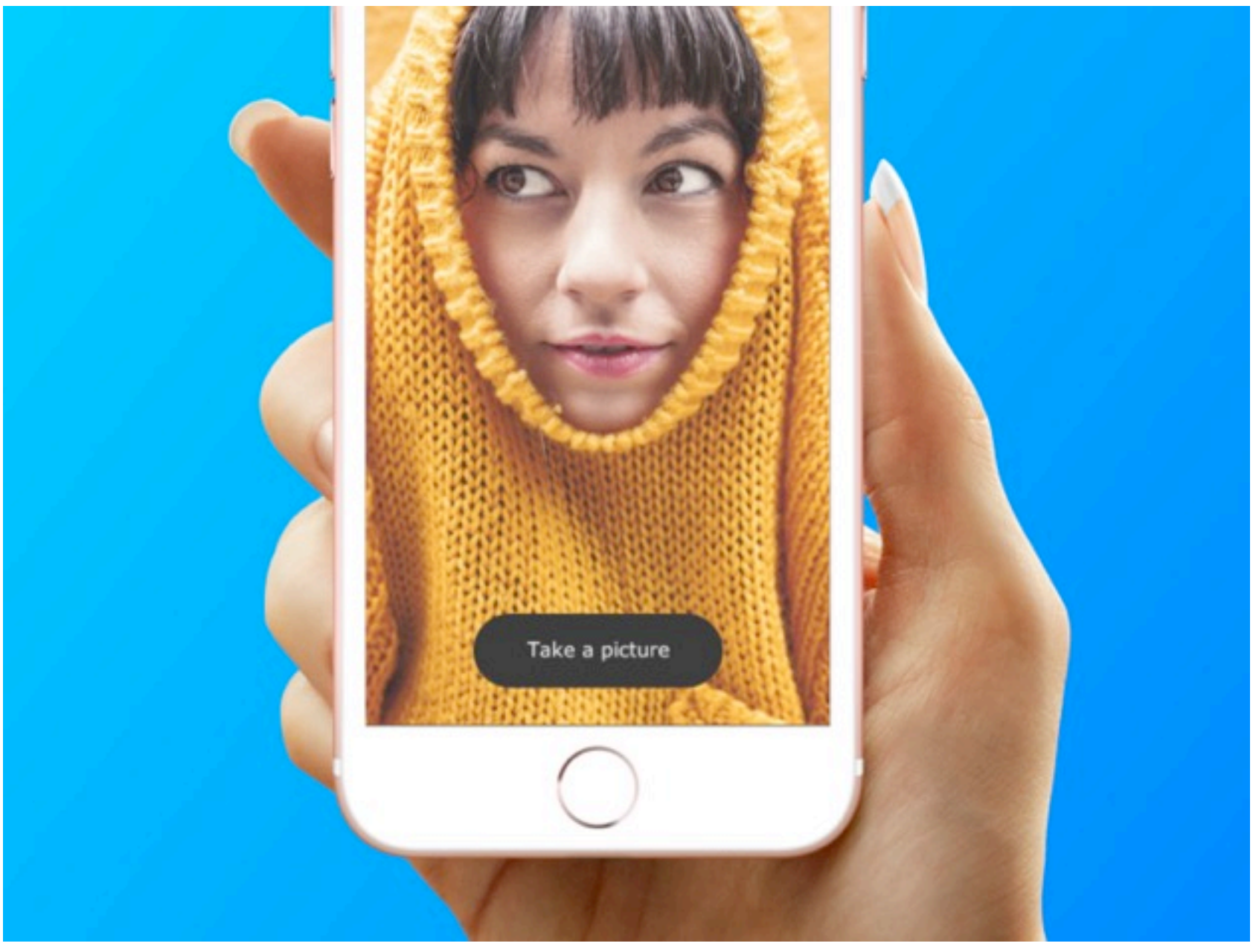
20 Апр, 2018 · 8 мин

https://medium.com/prototypr/make-a-camera-web-app-tutorial-part-1-ec284af8dddf

Перевод: С. Кузнецов, 25.09.2022

Сделайте веб-приложение камеры — учебное руководство / часть: 1

Изучите, как сделать приложение камеры с помощью **HTML**-кода, **CSS**-кода и **JavaScript**-кода



Изображение от Райана Макгуайера(Ryan McGuire)/ https://gratisography.com

Это учебное руководство пошагово рассмотрит процесс создания **приложения камеры(camera app)**, работающего в браузере вашего смартфона. (Как волшебство)

В части 1 мы рассмотрим:

1. Установка и настройка проекта
2. Создание работающего **приложения камеры(camera app)**, **делающего снимок(take a picture)**
3. Тестирование приложения камеры на смартфоне

Некоторые примечания:

1. Сейчас, в ОС **iOS** поддерживает доступ к камере только браузер **Safari**
2. К камере можно только получить доступ со страниц, используя протоколы **TSL/HTTPS** (защищенные протоколы, подробности ниже)

Давайте начнем …

1.1 Установка и настройка проекта

На своем компьютере создайте новый каталог(папку) с именем приложения камеры **camera-app**.

Откройте ваш каталог(папку) в вашем любимом текстовом редакторе. Я собираюсь использовать редактор сублимированого текста Sublime Text 3.

Создайте 3 новых документа и сохраните их в файлах index.html, style.css и app.js.

1.2 HTML-часть приложения: HTML-код

Установите и настройте базовый HTML-документ, используя следующий шаблон:

```
<!doctype html>
<html lang=”en”>
<head>
<meta charset="utf-8">
<meta http-equiv="x-ua-compatible" content="ie=edge">
<meta name="viewport" content="width=device-width,
initial-scale=1">

<!-- Имя вашего удивительного приложения камеры -->
<!-- Name of your awesome camera app -->
<title>Camera App</title>

<!-- Ссылка на вашу главную таблицу стилей -->
<!-- Link to your main style sheet-->
<link rel="stylesheet" href="style.css">
</head>
<body>

<!-- Ссылка на ваш JavaScript-файл -->
<!-- Reference to your JavaScript file -->
<script src="app.js"></script>
</body>
</html>
```

Затем, мы создадим базовые части камеры.

Внутри тега тело <body>, создайте элемент с тегом главный <main> с ID-идентификатором камеры camera.

```
<!-- Камера -->
<!-- Camera -->
<main id="camera">

</main>
```

Внутри тега главный `<main id="camera">`, создайте элемент с тегом холст `<canvas>` с `ID`-идентификатором датчика/сенсора камеры `camera--sensor`.

Подобно реальному датчику/сенсору камеры, элемент, с тегом холст `<canvas>`, `захватит кадр/фрейм из видеопотока(grab a frame from the video stream)`, когда мы прикажем ему, и нарисует его в следующему элементу, который мы создадим.

```
<!-- Камера -->
<!-- Camera -->
<main id="camera">

    <!-- Датчик/сенсор камеры -->
    <!-- Camera sensor -->
    <canvas id="camera--sensor"></canvas>
</main>
```

Затем, создайте элемент с тегом видео `<video>` с `ID`-идентификатором вида из камеры `camera--view` и добавляют атрибут автоматическое воспроизведение `autoplay` и атрибут воспроизведения синусоиды `playsinline`.

Элемент с тегом видео `<video>` `получит доступ к камере устройства(will access the device's camera)` и `выведет на экран видеопоток(display the video stream)`.

```
<!-- Камера -->
<!-- Camera -->
<main id="camera">

    <!-- Вид из камеры -->
    <!-- Camera view -->
    <video id="camera--view" autoplay playsinline></video>

    <!-- Датчик/сенсор камеры -->
    <!-- Camera sensor -->
    <canvas id="camera--sensor"></canvas>
</main>
```

Создайте создайте элемент с тегом изображения `<img>` с `ID`-идентификатором вывод камеры `camera--output` и адрес `источника(source)` установите в `//:0` (используем `//:0`, поскольку `источник(source)` будет

препятствовать тому, чтобы пустой значок изображения показан без браузера, показывающего предупреждение системы безопасности).

Элемент изображения `img` выведет на экран картину после того, как она будет `снята(taken)`.

```
<!-- Камера -->
<!-- Camera -->
<main id="camera">

    <!-- Датчик/сенсор камеры -->
    <!-- Camera sensor -->
    <canvas id="camera--sensor"></canvas>

    <!-- Вид из камеры -->
    <!-- Camera view -->
    <video id="camera--view" autoplay playsinline></video>

    <!-- Вывод из камеры -->
    <!-- Camera output -->
    <img src="//:0" alt="" id="camera--output">
</main>
```

Последняя нужная нам часть, является кнопкой `button`, чтобы `сделать фото(snap a picture with)`.

Создайте элемент кнопки `<button>` с `ID`-идентификатором триггера/спускового крючка/триггера камеры `camera--trigger` и внутри него добавьте `Take a picture(Сделать фото)`(или что-то другое, как вы хотите маркировать свою кнопку).

```
<!-- Камера -->
<!-- Camera -->
<main id="camera">

    <!-- Датчик/сенсор камеры -->
    <!-- Camera sensor -->
    <canvas id="camera--sensor"></canvas>

    <!-- Вид из камеры -->
    <!-- Camera view -->
    <video id="camera--view" autoplay playsinline></video>

    <!-- Вывод из камеры -->
    <!-- Camera output -->
    <img src="//:0" alt="" id="camera--output">
```

```
        <!-- Спусковой крючок/триггер камеры -->
        <!-- Camera trigger -->
        <button id="camera--trigger">Take a picture</button>
    </main>
```

Ваш HTML-документ должен быть похож на этот код:

```
    <html lang="en">
    <head>
        <meta charset="utf-8">
        <meta http-equiv="x-ua-compatible" content="ie=edge">
        <meta name="viewport" content="width=device-width,
initial-scale=1">

        <!-- Имя вашего удивительного приложения камеры -->
        <!-- Name of your awesome camera app -->
        <title>Camera App</title>

        <!-- Ссылка на вашу главную таблицу стилей -->
        <!-- Link to your main style sheet-->
        <link rel="stylesheet" href="style.css">
    </head>
    <body>

        <!-- Камера -->
        <!-- Camera -->
        <main id="camera">

            <!-- Датчик/сенсор камеры -->
            <!-- Camera sensor -->
            <canvas id="camera--sensor"></canvas>

            <!-- Вид из камеры -->
            <!-- Camera view -->
            <video id="camera--view" autoplay playsinline></video>

            <!-- Вывод из камеры -->
            <!-- Camera output -->
            <img src="//:0" alt="" id="camera--output">

            <!-- Спусковой крючок/триггер камеры -->
            <!-- Camera trigger -->
            <button id="camera--trigger">Take a picture</button>
        </main>

        <!-- Ссылка на ваш JavaScript-файл -->
        <!-- Reference to your JavaScript file -->
        <script src="app.js"></script>
</body>
</html>
```

Вы сделали это! **HTML**-часть приложения - полно комплектна… далее **CSS**-часть приложения, таблица стилей!

1.3 CSS-часть приложения: CSS-таблица стилей

Чтобы удостовериться все размечено корректно и выглядит притягательно, мы к странице добавим некоторые стили в созданном нами ранее файле **style.css**.

В вашем файле **style.css** добавьте селектор для **html**, **body** и значения **margin(кромка)** и **padding(отступ)** установите в **0**. Этим гарантируем, что в **области просмотра(видовом порту; viewport)** нет никакого незапланированного пустого интервала. Значения **width(ширины)** и **height высоты()** установите в **100%**.

```
html, body{
    margin: 0;
    padding: 0;
    height: 100%;
    width: 100%:
}
```

Затем, мы должны будем соответствовать в размерах элементам, которые мы создали на экране.

Добавьте селектор для камеры **#camera**, для вида из камеры **#camera--view**, для датчика/сенсора камеры **#camera--sensor**, для вывода из камеры **#camera--output** и значение свойства **позиции(position)** установите в **fixed(фиксированная)**. Значения **width(ширины)** и **height(высоты)** установите в **100%**. Значения свойства **object-fit(объект-подгонка)** установите в **cover(покрыть)** так, чтобы видео заняло весь экран.

```
#camera, #camera--view, #camera--sensor, #camera--output{
    position: fixed;
    height: 100%;
    width: 100%:
    object-fit: cover;
}
```

Так как мы снимаем на телефон, используя камеру **"selfie"("селфи")**, то мы захотим изображение отразить зеркально, чтобы придать ему более естественное чувство.

```
#camera--view, #camera--sensor, #camera--output{
    transform: scaleX(-1);
    filter: FlipH;
}
```

В селекторе для датчика/сенсора камеры **#camera--sensor**, значение свойства **позиции(position)** установите в **fixed(фиксированная)** и добавьте некоторый другой стиль(см. ниже) для центрирования его внизу экрана.

```
#camera--trigger{
    width: 200px;
    background-color: black;
    color: white;
    font-size: 16px;
    border-radius: 30px;
    border: none;
    padding: 15px 20px;
    text-align: center;
    box-shadow: 0 5px 10px 0 rgba(0,0,0,0.2);
    position: fixed;
    bottom: 30px;
    left: calc(50% - 100px);
}
```

Наконец, в селекторе **.taken** мы добавим некоторую стилизацию для изображения после того, как **снимем фото(taken)**...

```
.taken{
    height: 100px!important;
    width: 100px!important;
    transition: all 0.5s ease-in;
    border: solid 3px white;
    box-shadow: 0 5px 10px 0 rgba(0,0,0,0.2);
    top: 20px;
    right: 20px;
    z-index: 2;
}
```

Код **CSS**-таблицы стилей в файле **style.css** должен быть похожим на этот код...

```
html, body{
  margin: 0;
  padding: 0;
  height: 100%;
  width: 100%;
```

```
}
#camera, #camera--view, #camera--sensor, #camera--output{
    position: fixed;
    height: 100%;
    width: 100%;
    object-fit: cover;
}
#camera--view, #camera--sensor, #camera--output{
    transform: scaleX(-1);
    filter: FlipH;
}
#camera--trigger{
    width: 200px;
    background-color: black;
    color: white;
    font-size: 16px;
    border-radius: 30px;
    border: none;
    padding: 15px 20px;
    text-align: center;
    box-shadow: 0 5px 10px 0 rgba(0,0,0,0.2);
    position: fixed;
    bottom: 30px;
    left: calc(50% - 100px);
}
.taken{
        height: 100px!important;
        width: 100px!important;
        transition: all 0.5s ease-in;
        border: solid 3px white;
        box-shadow: 0 5px 10px 0 rgba(0,0,0,0.2);
        top: 20px;
        right: 20px;
        z-index: 2;
    }
```

CSS-часть приложения, таблица стилей - полно комплектна! Давайте запишем некоторую JavaScript-часть приложения: JavaScript-код…

1.4 JavaScript-часть приложения: JavaScript-код

JavaScript-код в JavaScript-файле свяжет все части приложения вместе и заставит волшебство произойти. Откройте свой файл app.js, и давайте начнем действовать.

Сначала мы должны создать переменную для наших **ограничений (constraints; или параметров настройки; settings)** для **видеопотока(video stream)**. Мы хотим принять значение по умолчанию состояния камеры: **передняя камера/камера обращенная на пользователя(user camera; camera facing the user)**(вы можете значение параметра **facingMode** установить в **"environment"("окружение камеры")**, если вы приняли бы значение по умолчанию состояния камеры: **задняя камера/камера обращенная от пользователя(rear camera)**. Мы также хотим сказать браузеру, что не нуждаемся в аудио от устройства.

Наверху кода файла **app.js** запишите…

```
    var constraints = { video: { facingMode: "user" }, audio: false
};
```

Затем, мы должны определить константы для всех созданных нами частей.

На следующей строке запишите …

```
const cameraView = document.querySelector("#camera--view"),
      cameraOutput = document.querySelector("#camera--output"),
      cameraSensor = document.querySelector("#camera--sensor"),
      cameraTrigger = document.querySelector("#camera--trigger")
```

Теперь, когда мы собрали все нужные нам части, мы должны создать функцию, которую назовем именем старта камеры **cameraStart** и она получит доступ к камере и видео-поток камеры передаст созданному нами элементу вида из камеры **camera--view**.

Чтобы получить доступ к камере, этот код использует метод чтения медиа потока передней камеры **getUserMedia**, используя заданные нами ограничения **constraints**. Мы также добавим оператор перехвата ошибок **.catch**, чтобы удостовериться, что сообщено об ошибке, если камера не работает.

```
function cameraStart() {
    navigator.mediaDevices
        .getUserMedia(constraints)
        .then(function(stream) {
        track = stream.getTracks()[0];
```

```
                cameraView.srcObject = stream;
            })
            .catch(function(error) {
                console.error("Oops. Something is broken./Ой.Что-то
    повреждено.", error);
            });
        }
```

Как только у нас есть видеопоток, для последующей работы с ним, мы можем запрограммировать кнопку, чтобы захватить кадр(фрейм) из потока, который мы будем использовать в качестве нашего **вывода изображения(image output)**.

```
    cameraTrigger.onclick = function() {
        cameraSensor.width = cameraView.videoWidth;
        cameraSensor.height = cameraView.videoHeight;
        cameraSensor.getContext("2d").drawImage(cameraView, 0, 0);
        cameraOutput.src = cameraSensor.toDataURL("image/webp");
        cameraOutput.classList.add("taken");
    };
```

Наконец, мы должны будем инициировать функцию старта камеры **cameraStart**, когда будет закончена загрузка окна. Это будет похоже на этот код ...

```
    window.addEventListener("load", cameraStart, false);
```

Полностью код в файле **app.js** должен быть похож на этот код ...

```
    // Установить ограничения для видеопотока
    // Set constraints for the video stream
    var constraints = { video: { facingMode: "user" }, audio: false
};
    // Определите константы
    // Define constants
    const cameraView = document.querySelector("#camera--view"),
        cameraOutput = document.querySelector("#camera--output"),
        cameraSensor = document.querySelector("#camera--sensor"),
        cameraTrigger = document.querySelector("#camera--trigger")

    // Получите доступ к камере устройства и потоку,
    // к элементу вида из камеры cameraView
    // Access the device camera and stream to cameraView
    function cameraStart() {
        navigator.mediaDevices
            .getUserMedia(constraints)
            .then(function(stream) {
            track = stream.getTracks()[0];
            cameraView.srcObject = stream;
        })
```

```
        .catch(function(error) {
            console.error("Oops. Something is broken. /Ой.Что-то
повреждено.", error);
        });
    }

    // Делайте фото/снимок, когда нажат
    // спусковой крючок/триггер камеры cameraTrigger
    // Take a picture when cameraTrigger is tapped
    cameraTrigger.onclick = function() {
        cameraSensor.width = cameraView.videoWidth;
        cameraSensor.height = cameraView.videoHeight;
        cameraSensor.getContext("2d").drawImage(cameraView, 0, 0);
        cameraOutput.src = cameraSensor.toDataURL("image/webp");
        cameraOutput.classList.add("taken");
    };

    // Запустите видеопоток, когда окно загружено
    // Start the video stream when the window loads
    window.addEventListener("load", cameraStart, false);
```

1.5 Хостинг: размещение на хосте/сервере в сети

Давайте протестируем наше приложение!

Сначала мы должны разместить его на **безопасном HTTPS-сервере(secure HTTPS server)**. Вы можете сделать это локально на вашей машине, если вам нравится …, но это - другое учебное руководство.

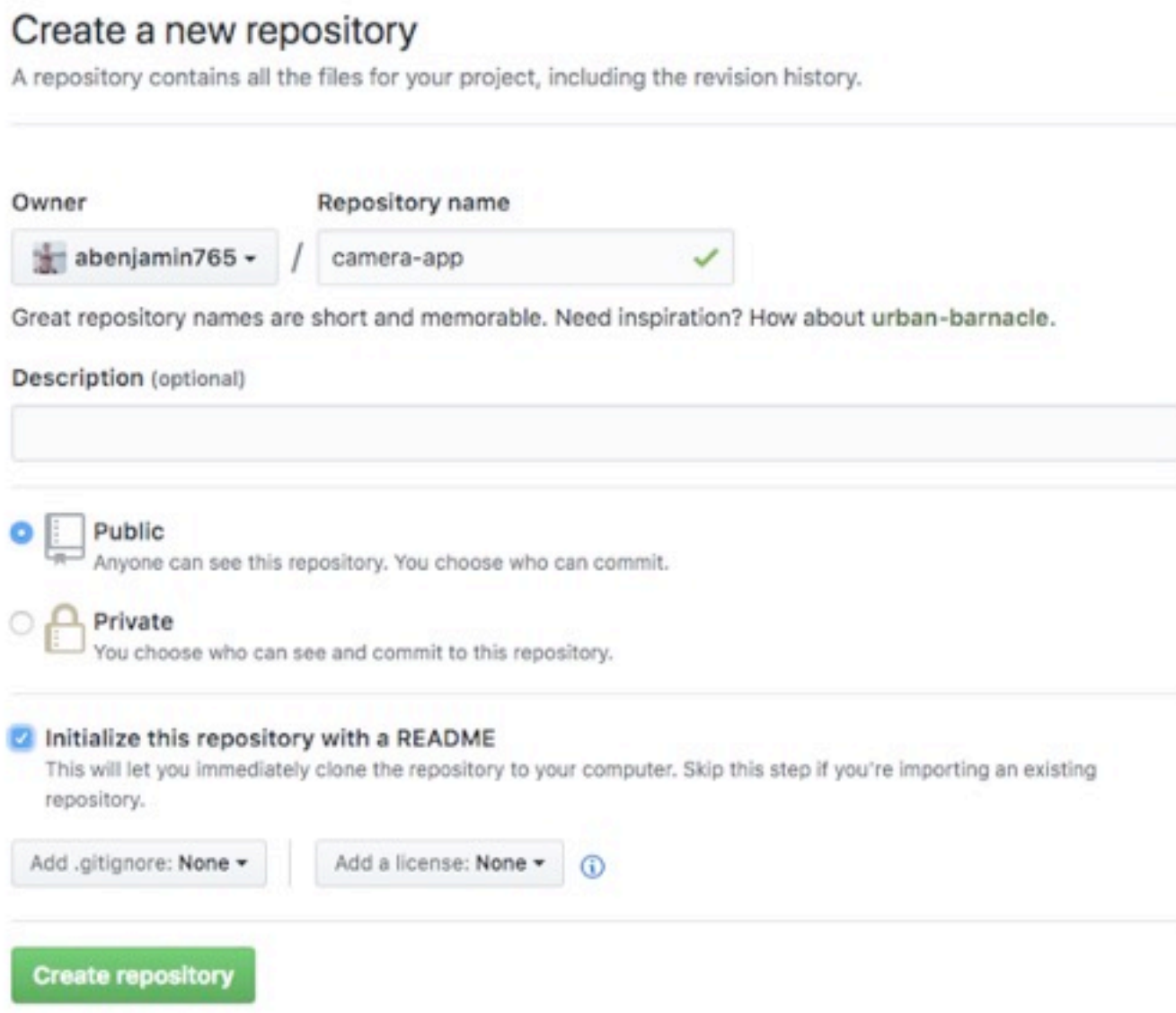
Мы собираемся использовать страницы сервиса **GitHub**.

1. Этот сервис - свободный
2. Этот сервис - довольно простой в использовании
3. Этот сервис предлагает протокол **HTTPS**

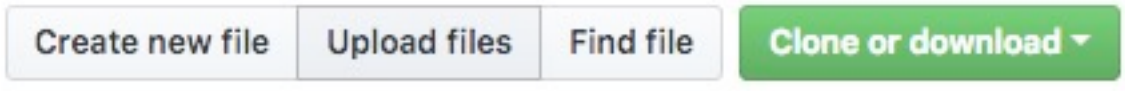
Сначала вы должны на сервисе **GitHub** создать учетную запись(если у вас еще нет), на сайте по следующей ссылке create a GitHub account.

Когда вы будете зарегистрированы и войдете в систему, то найдите зеленую кнопку **“New repository”(“Новый репозитарий”)** и щелкните по ней.

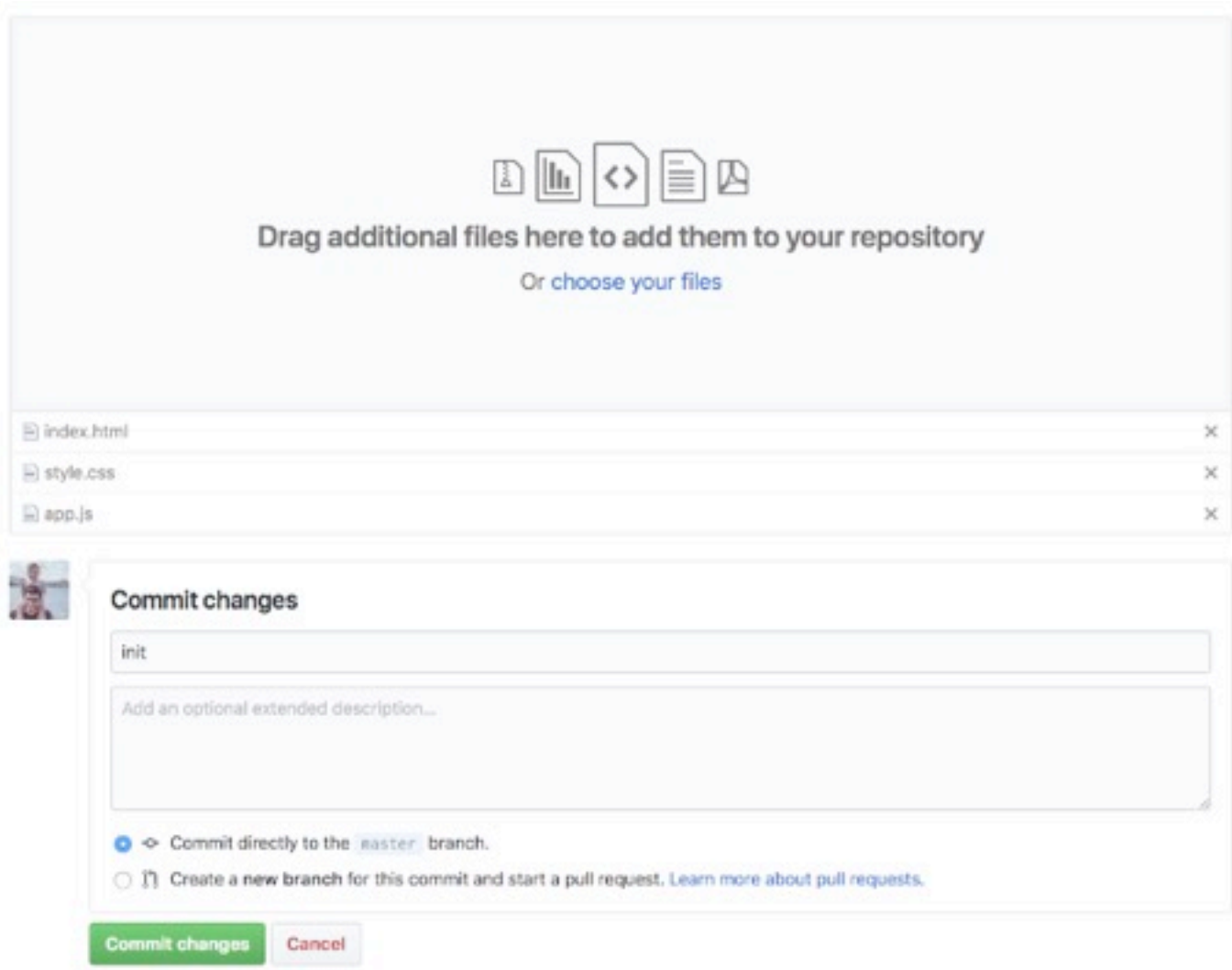
Дайте вашему репозитарию имя. Я назову мой репозитарий приложением камеры `camera-app`. Удостоверьтесь, что устанавливаете флажок в поле `"Initialize this repository with a README"("Инициализировать этот репозиторий с файлом описанием README")`.



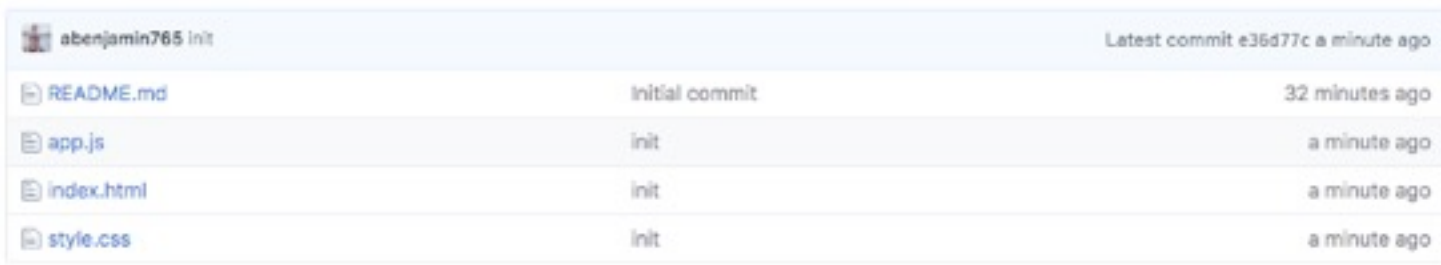
После того, как вы нажмете кнопку `"Create repository"("Создать репозитарий")`, на панели инструментов выберите кнопку `"Upload files"("Загрузить файлы в репозитарий")`.



Загрузите созданные нами файлы в ваш репозитарий `GitHub`, выбирая их и щелкая по кнопке `"Commit changes"("Фиксировать изменения")`.

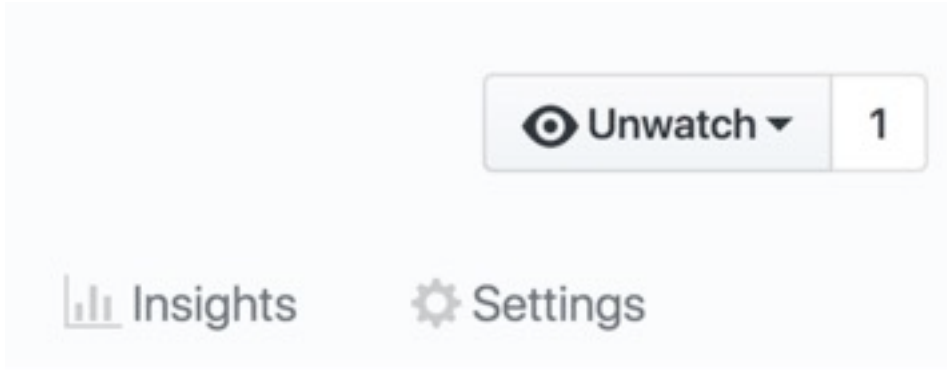


Когда файлы фиксированы, то вы должны видеть их в своем репозитории.

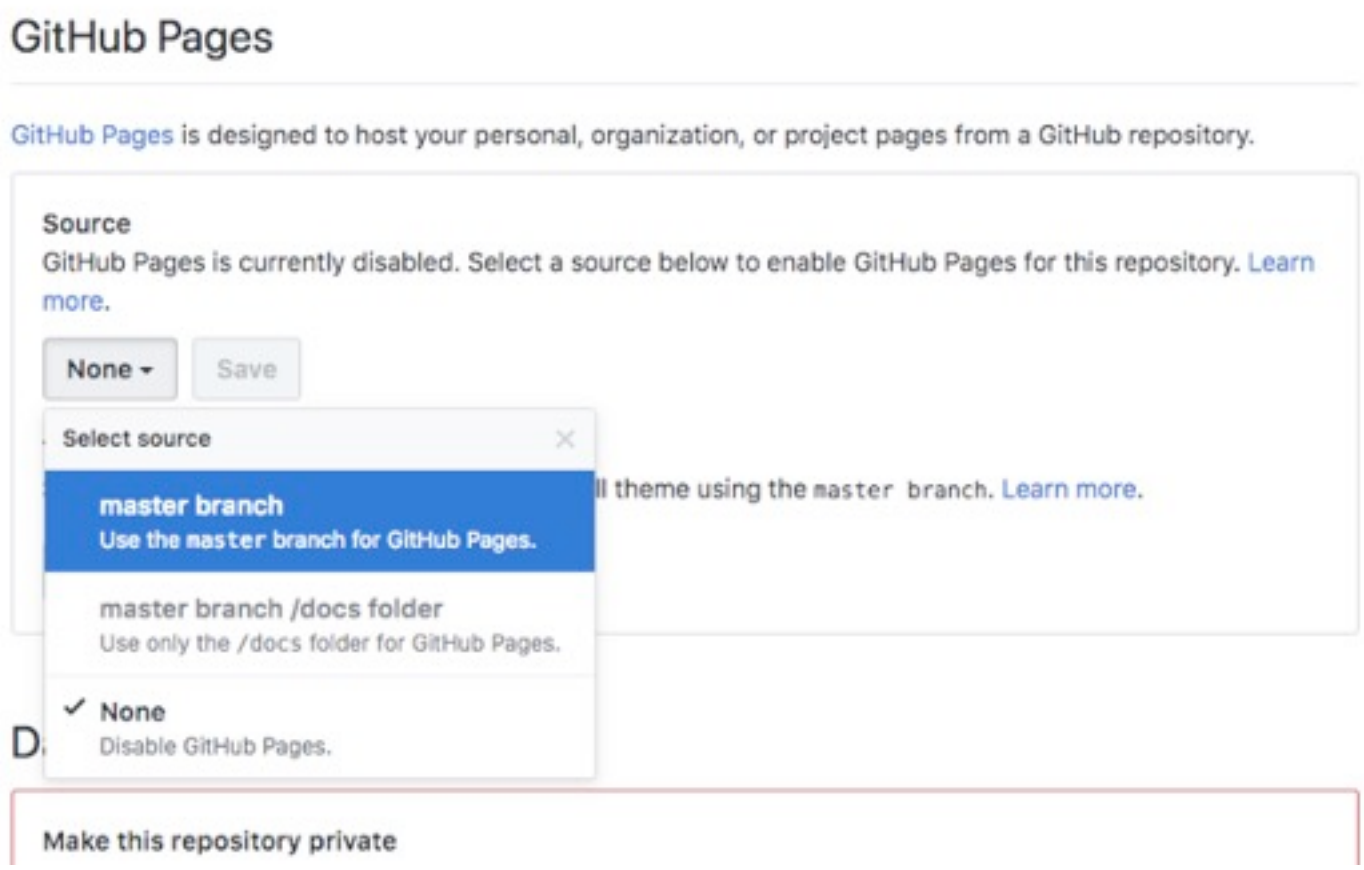


Следующий шаг должен установить и настроить страницы сервиса **GitHub**.

Наверху экрана щелкните по вкладке **“Settings”(“Параметры настройки”)**.



В **разделе страниц(pages section)** сервиса **GitHub**, в меню выберите пункт **master branch(главное ответвление)** и нажмите кнопку **"Save"("Сохранить")**.



Из вашего компьютера или мобильного устройства, посетите страницу по адресу https://yourGitHubUsername.github.io/camera-app

Вы увидите свое работающее приложение камеры. Сделайте селфи!

Сделайте селфи!

Вы можете найти эти файлы на **GitHub** в https://github.com/abenjamin765/camera-app

1.6 В части 2 мы рассмотрим ...

1. Создание и использование **значка приложения(App Icon)**
2. Переключение камер: **передней камеры/камеры обращенной на пользователя(user camera; camera facing the user)** и **задней камеры/камеры обращенной от пользователя(rear camera)**.
3. Сохранение картинки