CSS: RWD, Transition and Animation

Lec-7

Responsive Web Design (RWD)

- Responsive Web Design refers to the ability of a website to resize and reorganize it's contents based on:

1. **The size of either content on website.**
2. **The size of screen the website is being viewed on.**

- You've probably noticed the unit of **pixels (px)** used in website. Pixels are used to size container to exact dimensions.
- **Pixels** however are **fixed/hard coded** values.
- With CSS, you've to avoid hard coded measurements and use relative measurements instead. Relative measurements allow for the proportions of a website to remain intact regardless of screen size or layout.

Em

- If the base font of a browser is 16px (which is normally the default size of text in a browser), then
- **1em = 16px** and **2em = 32px**

Rem

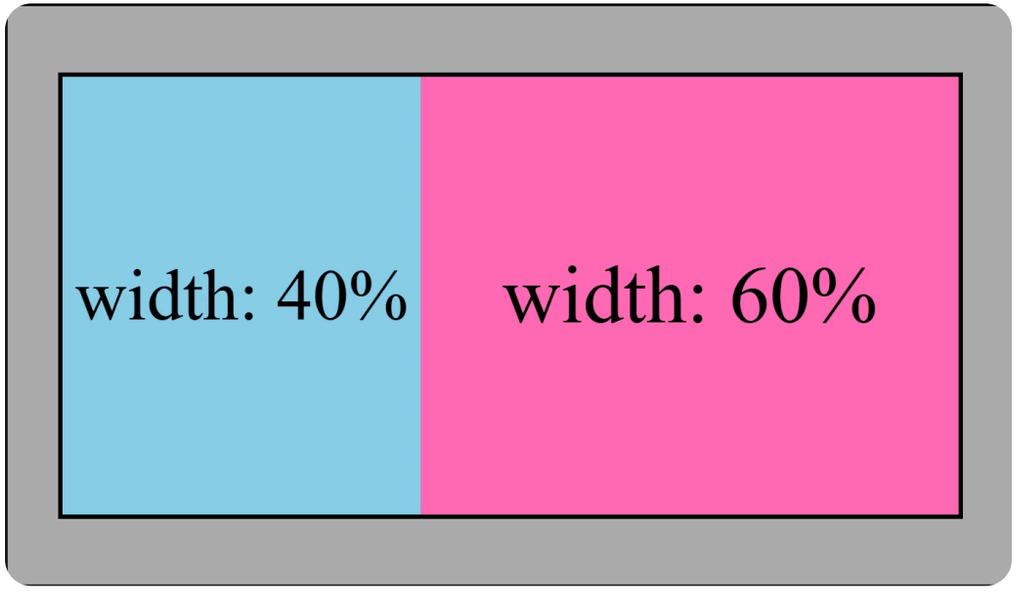
- **Rem** stands for **root em**. It is similar to **em,** but **instead of checking the parent element** it **checks the root element** i.e. the **HTML tag.**
- Most browsers set the font size of **<html>** tag to **16px,** so by default the **rem measurements** will be compared to that value.

- If you're interested in sizing elements consistently across an entire website, **the em measurement** is the **best unit for the job.**
- If you're interested in sizing elements in comparison to other elements nearby then, **the rem measurement** is the **best unit for the job.**

Percentages, Height and Width

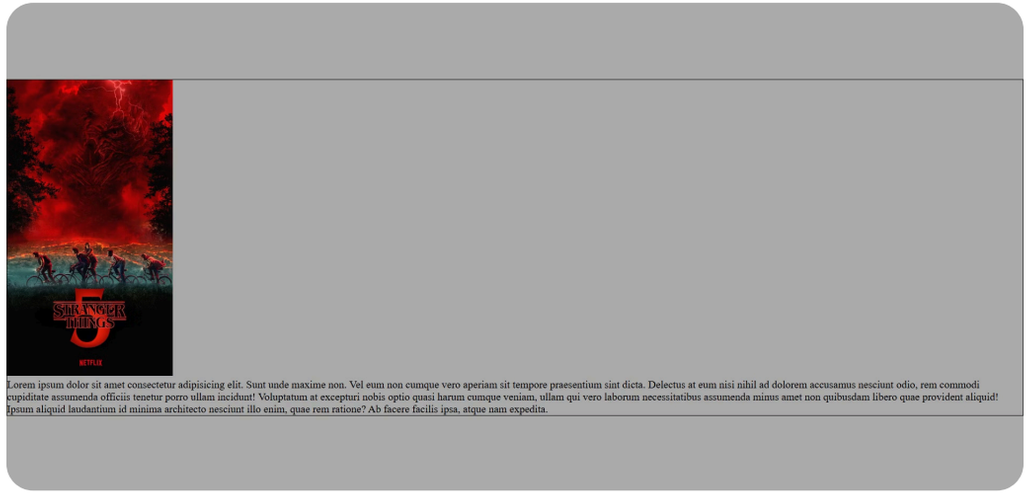
- To size **non-text HTML elements relative to their parent elements** on the page you can use percentages.
- Percentages are **often used to size box model values**, like **width, height, padding, border** and **margins**. They can also be used to set **positioning offset properties- top, bottom, left, right**.
- When **percentages are used**, **elements are sized relative to the dimensions of their parent elements** (also known as **container**).

```
.container{
  border: 5px solid black;
}
.Box-1{
  width: 40%;
  background-color: skyblue;
}
.Box-2{
  width: 60%;
  background-color: hotpink;
}
```

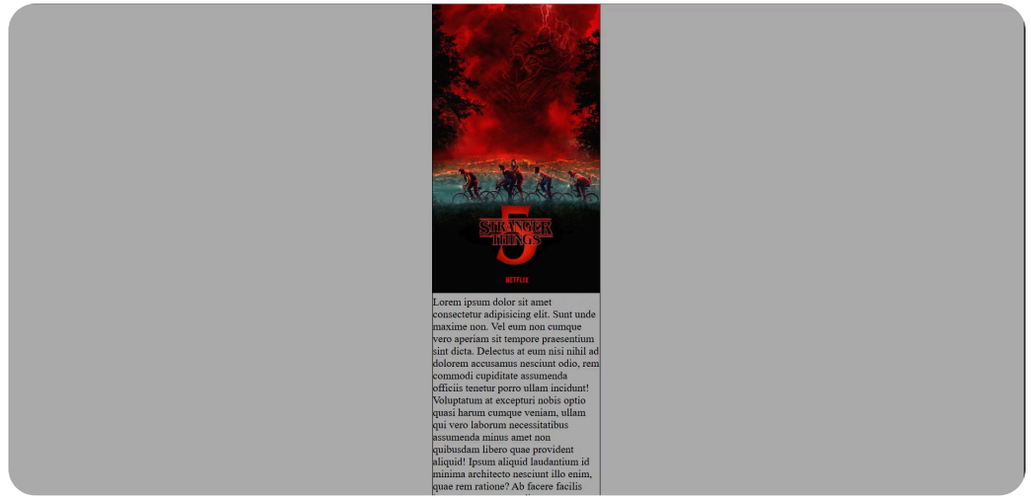


Min and Max Width and Height

1. **min-width:** Ensures a minimum width for an element.
2. **max-width:** Ensures a maximum width for an element.




```
p{
    min-width: 150px;
    max-width: 250px;
}
```



Note: The unit of pixels is to ensure the hard limits on the dimensions of elements.

Viewport Meta Tag

- **Viewport** is the total viewable area for a user.
- This area varies depending upon the device.
- The viewport is smaller on a mobile device and larger on a desktop.
- Based on the size of viewport, the **meta tag <meta>** is used to instruct the browser on how to render the webpage's scale and dimensions.

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

1. **name = "viewport":** This attribute tells the **browser** to **display the web page at same width as it's screen**.
2. **content:** This attribute defines the values for **<meta>** tag including **width** and **initial-scale.**

(i) width = device-width: It is a **key value pair** which **controls the size of the viewport** in which it **sets the width of the viewport equal to the width of device**.

(ii) initial-scale = 1.0: It is also a **key value pair,** it **sets the initial zoom level.**

- The **viewport meta tag** allows our web pages to be **responsive and adapt a site's content** to the user's screen size.
- In order to develop a **responsive website**, our **HTML file** should mandatorily contain **<meta> tag.**

Media Queries

- CSS **uses media queries to adapt a website's content to different screen sizes.**
- With media queries, **CSS can detect the size of current screen and apply different CSS styles depending on the width of the screen.**
- **Syntax:**

```
@media only screen and (max-width: 480px)
{
        body{
                font-size: 12px;
            }
}
```

The above ruleset defines a rule for screens smaller than 480px **(approximate width of smartphone's in landscape orientation).**

Here's the breakdown of this syntax:

- **@media -** This keyword **begins a media query rule** and **instructs the CSS compiler on how to parse the rest of the rule.**
- **only -** This keyword is added **to indicate that this rule only applies to one media type which is screen.**
- **screen -** This is the **media type** always **used for displaying content, no matter the type of device.**
- **and -** It is an **operator**, it is **used to chain more than one media features**, by **placing it between two media features**, **the browser will require both media features to be true** before it renders the CSS within media query.
- **(max-width: 480px) -** This part is called as **Media Feature,** it **instructs CSS compiler** to apply the **CSS styles to devices with a width of 480px or smaller.**

Range in Media Query

- By using multiple width's and heights, a range can be set for media query.

```
@media only screen and (min-width: 320px)
{
      // Your CSS Ruleset goes here
}
@media only screen and (min-width: 480px)
{
      // Your CSS Ruleset goes here
}
```

Distinctive ruleset

```
@media only screen and (min-width: 320px) and (max-width: 480px)
{
      // Your Ruleset goes here
}
```

Combined ruleset

In above examples, the **CSS ruleset will get applied when the screen size is between 320px to 4800px**. This allowed us to **chain multiple media features.**

Comma Separated List

- If **only one of the multiple media feature in a media query must be met** we use comma separated list.

```
@media only screen and (min-width: 320px), (orientation: landscape)
{
      // Your CSS Ruleset goes here
}
```

Common Breakpoints

- < 600px
- 601px to 768px
- 992px to 1200px

CSS Transitions

- CSS transitions **allows us to control the timing of visual state changes**. We can **control the following 4 aspects** of an element's transition:

1. **The Transitioned Property**
2. **Transition Duration**
3. **Timing function**
4. **Transition delay**

The Transition Property

- It is a property which denotes the type of transition.
- Common Transitioned Property:

1. **Layout:** width, height, margin, padding, top, bottom, left, right.
2. **Appearance:** opacity, visibility, z-index, background-color, color.
3. **Transforms:** transform (for rotation, scaling, skewing, translating).
4. **Borders:** border-color, border-width, outline

- **Eg:**

```
transition-property: width;
```

Transition Duration

- Duration is specified in **s** or **ms** such as **3s, 0.7s, 500ms.**
- **Eg:**

```
transition-duration: 200ms;
```

How To Trigger Transition ?

- The transition is triggered **when there is a change in the element's properties.**
- This often happens within **pseudo-classes (:hover, :active, :focus, or :checked).**
- **Eg:**

```
html

  <div class="box-1"></div>
```

```
css

.box-1{
        width: 160px;
        height: 160px;
        background-color: hotpink;
        transition: width 500ms;
    }

.box-1:hover{
        width: 360px;
        cursor: pointer;
    }
```

Timing Function

- It describes the **pace of transition.**
- It takes values:

1. **ease:** starts slow, accelerates quickly, stops slowly.
2. **ease-in:** starts slow, accelerates quickly, stops abruptly.
3. **ease-out:** starts slow, slows down, ends slowly.
4. **ease-in-out:** starts slow, get's fast in the middle, ends slowly.
5. **linear:** constant speed throughout.

- **Eg:**

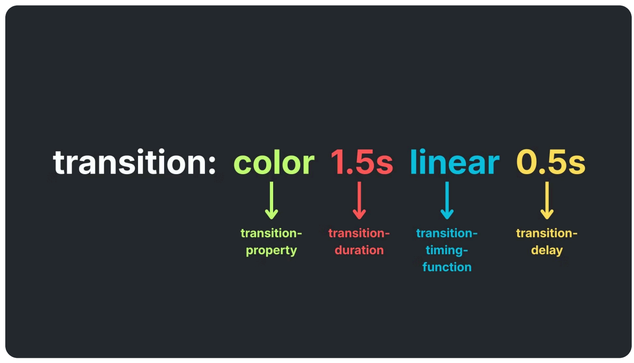
```
transition-timing-function: ease
```

Delay

- Transition Delay is the **property which specifies the time to wait before starting the transition.**
- **Eg:**

```
transition-delay: 1s;
```

Transition Shorthand Property



CSS Animations

- An animation **lets an element gradually change from one style to another**.
- You can change as many CSS properties you want, as many times as you want.
- To **use CSS animation**, you **must specify some keyframes** for the animation.
- **Keyframes hold what styles the element will have at certain times**.
- When you specify CSS styles inside the **@keyframes rule**, the **animation will gradually change** from the **current style to the new style** at certain times.
- **Eg:**

```
@keyframes mianimation {
  from{
    background-color: red;
  }
  to{
    background-color: blue;
  }
}
```

Animation Name

- The **animation-name** property specifies name for an animation.
- **Eg:**

```
animation-name: mianimation
```

Animation Duration

- The **animation-duration** property defines how long an animation should take to complete.
- If this property is **not specified**, **no animation will occur**, because the **default value is 0s** (0 seconds).
- **Eg:**

```
animation-duration: 10s;
```

Animation Delay

- This property **specifies a delay for the start of an animation**.
- **Eg:**

```
animation-delay: 2s;
```

Animation Iteration Count

- This property **specifies the number of times an animation should run.**
- **Eg:**

```
animation-iteration-count: 5;
```

- We can add value as **"infinite"** to make animation **continue to run forever**.

Animation Direction

- This property **specifies whether an animation should be played forwards, backwards or in alternate cycles**.
- The animation-direction property can have the following values:

1. **normal-** The animation is played as normal (forwards).
2. **reverse -** The animation is played in reverse direction (backwards).
3. **alternate -** The animation is played forwards first, then backwards.
4. **alternate-reverse -** The animation is played backwards first, then forwards**.**

- **Eg:**

```
animation-direction: alternate;
```

Animation Timing Function

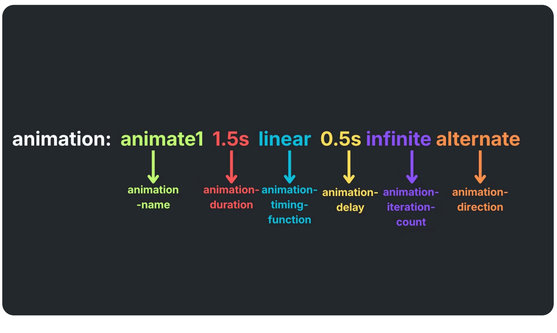
- This property specifies the **speed curve of the animation**.
- The animation-timing-function property can have the following values:

1. **ease -** Specifies an animation with a slow start, then fast, then end slowly (this is default)
2. **linear -** Specifies an animation with the same speed from start to end.
3. **ease-in -** Specifies an animation with a slow start.
4. **ease-out -** Specifies an animation with a slow end.
5. **ease-in-out -** Specifies an animation with a slow start and end.

- **Eg:**

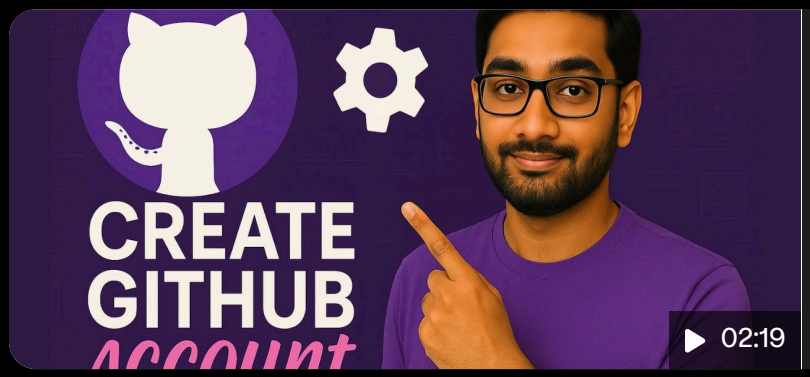
```
animation-timing-function: linear;
```

Animation Shorthand Property



How To Host Website using GitHub Pages ?

- **Step-1:** Create your GitHub Account by following the steps shown in this 👇video.

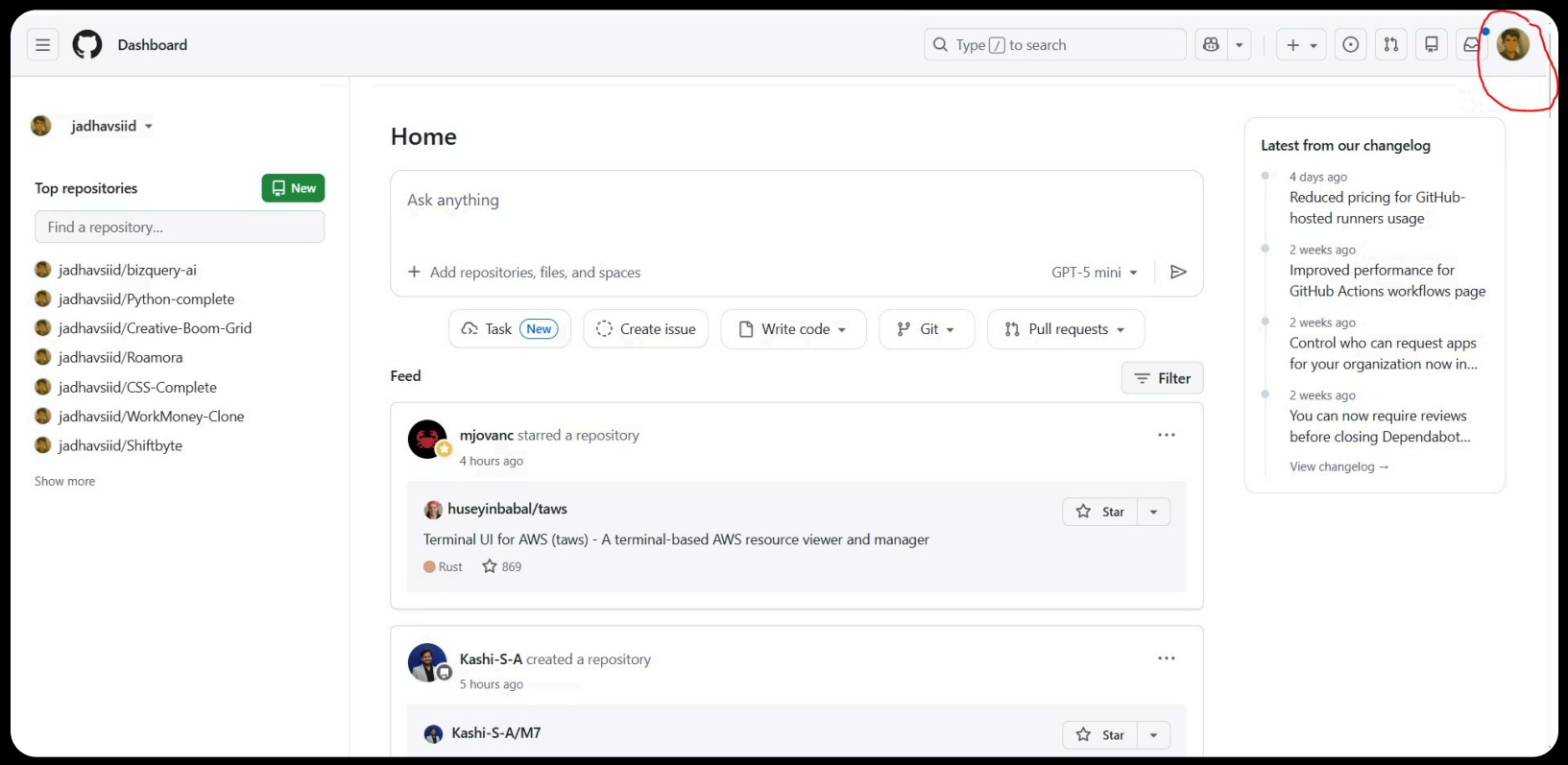


YouTube

How to Create Github Account | Github Account Creation Tutorial in Hindi...

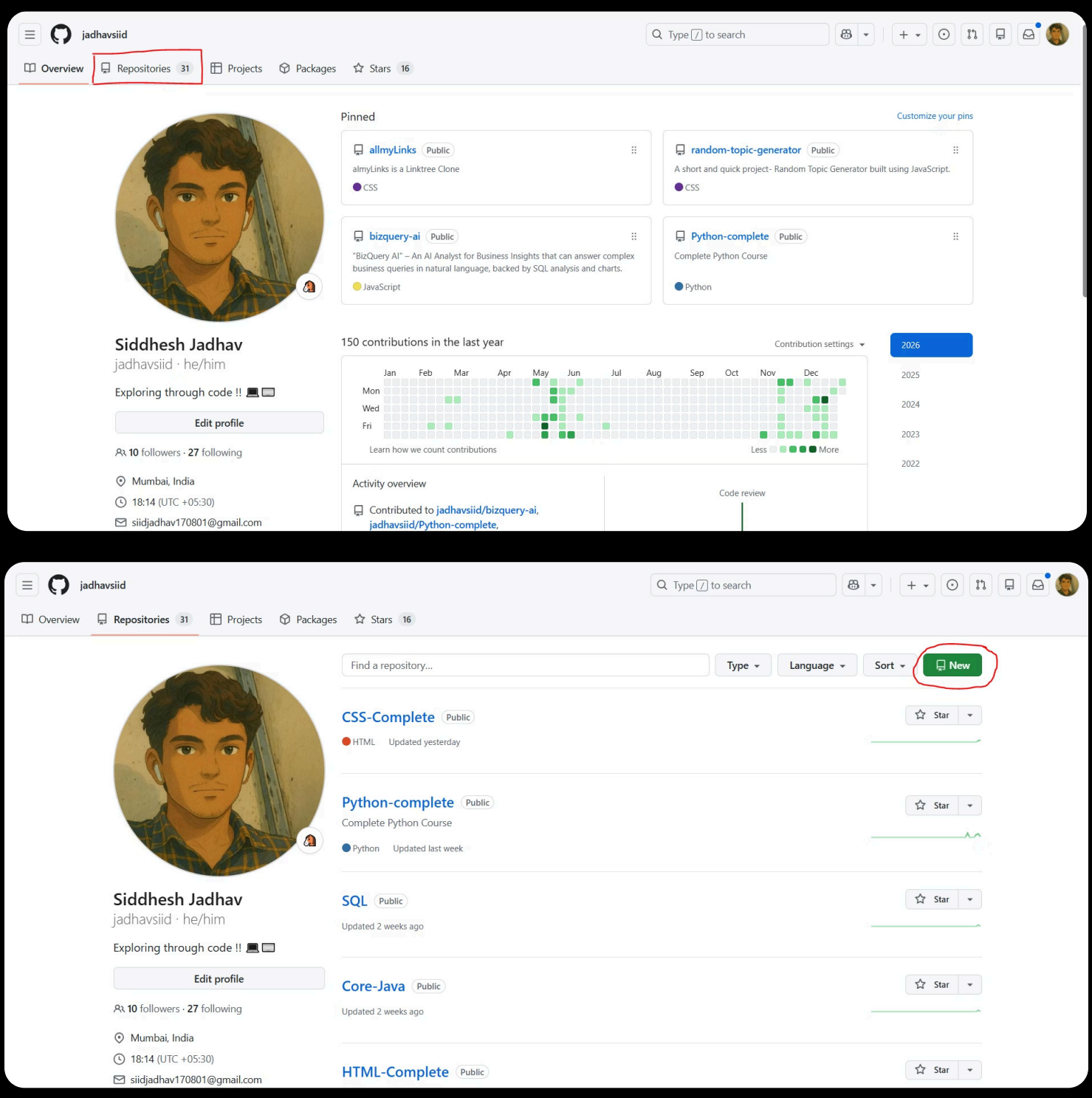
How to Create Github Account | Github Account Creation Tutorial in Hindi (2026) 🔥 Learn how to create a GitHub account with this beginner tutorial in Hindi. Follow...

- **Step-2:** After Successfully creating your GitHub account, you'll be directed towards the GitHub Dashboard page which can also be accessed by the following link: https://github.com/dashboard
- **Step-3 :** On Your GitHub's dashboard page, towards the top right you'll se your profile icon, click on it and visit your profile page.

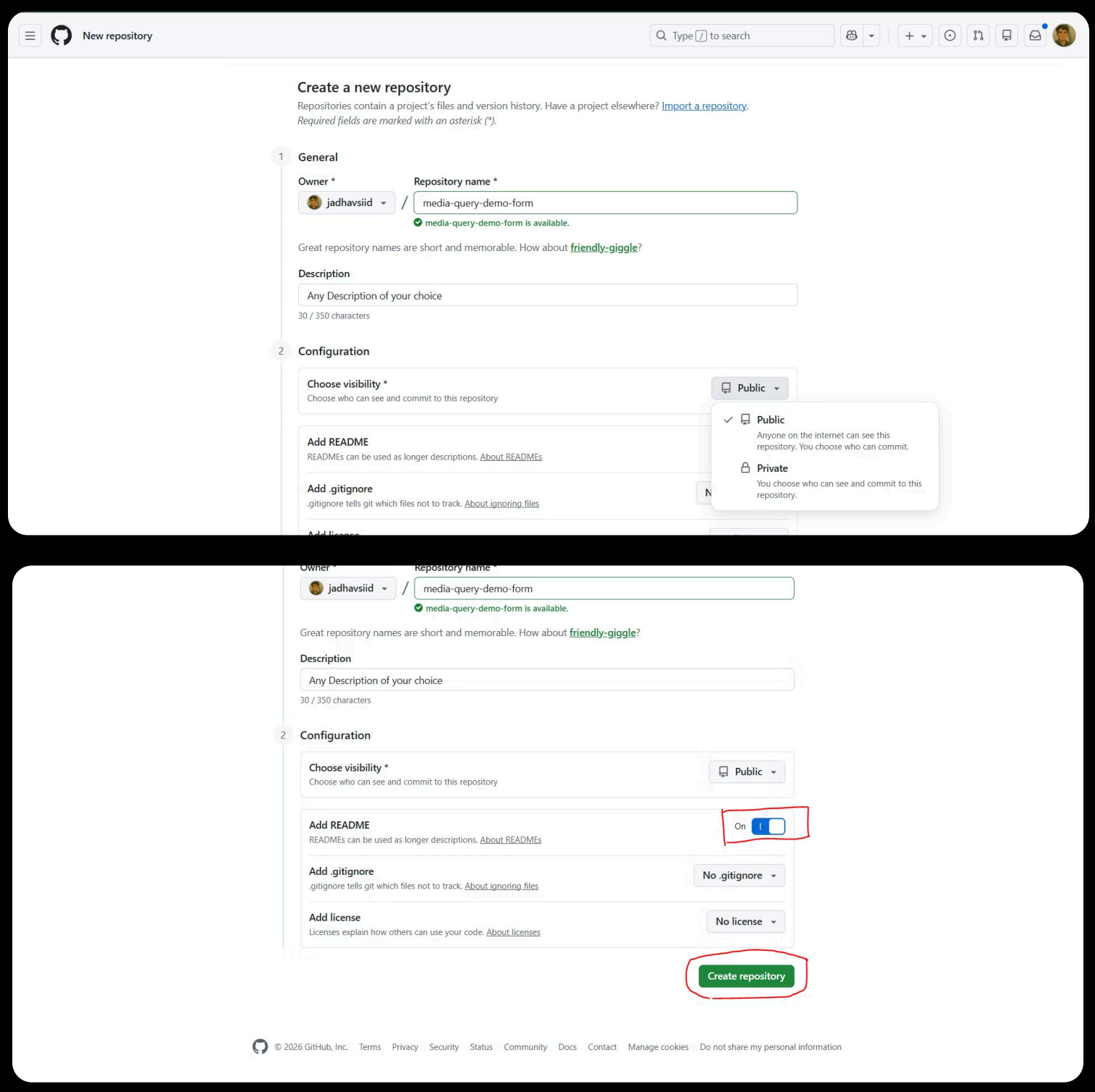


How To Host Website using GitHub Pages ?

- **Step-4:** On your Profile page Click on the **Repositories** tab and after migrating on it click on **New**



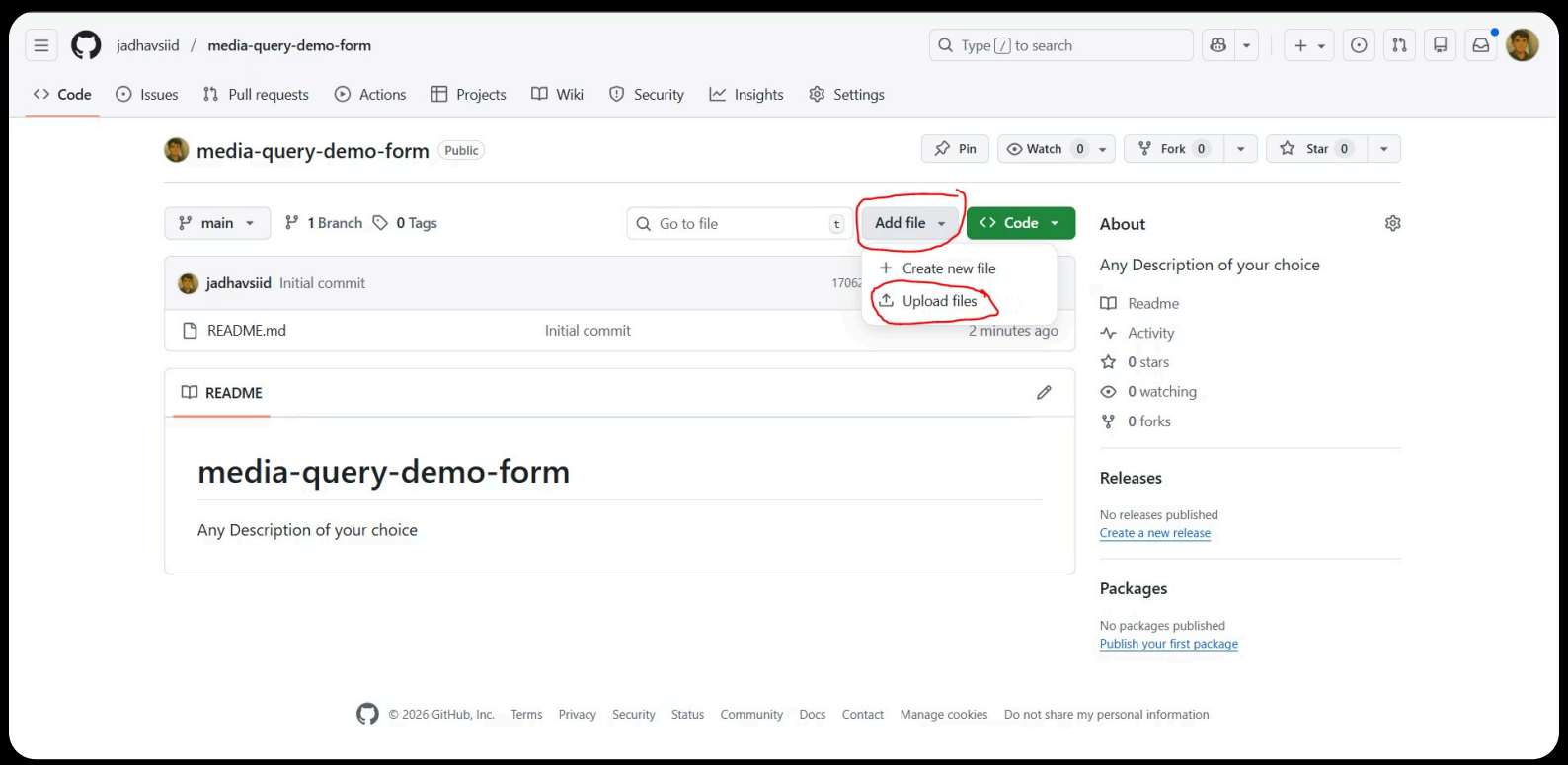
- **Step-5:** On next page, give your repository **name of your choice** then give **any description** of your choice set the **visibility of repository as public,** next you can add **README.md** file to your repository as per your wish and then click on **Create repository.**



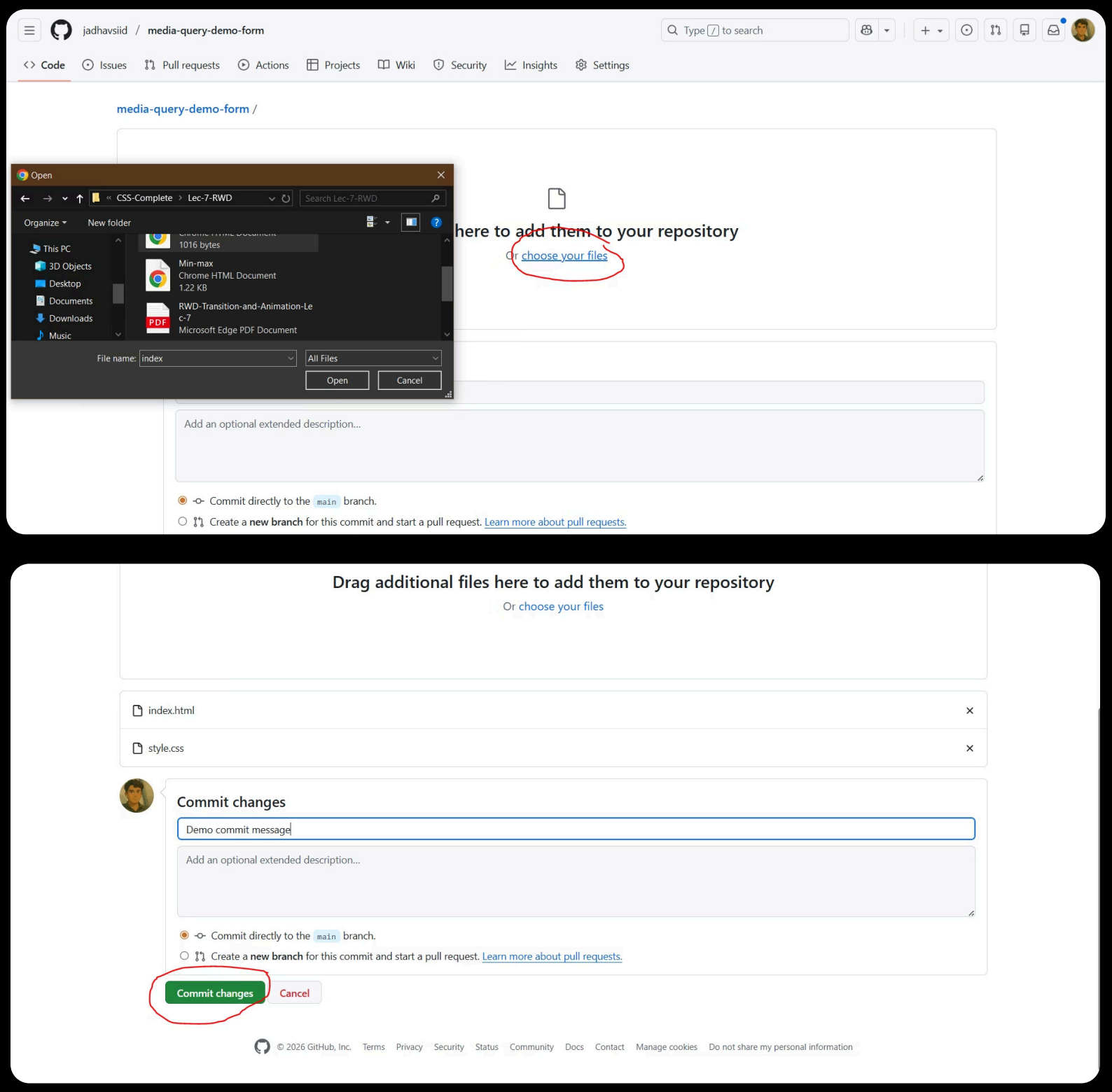
This Creates your Repository Succesfully !!

How To Host Website using GitHub Pages ?

- **Step-6:** After Creating your repository, inside your repository page. Click on **Add file** and then on **Upload files**

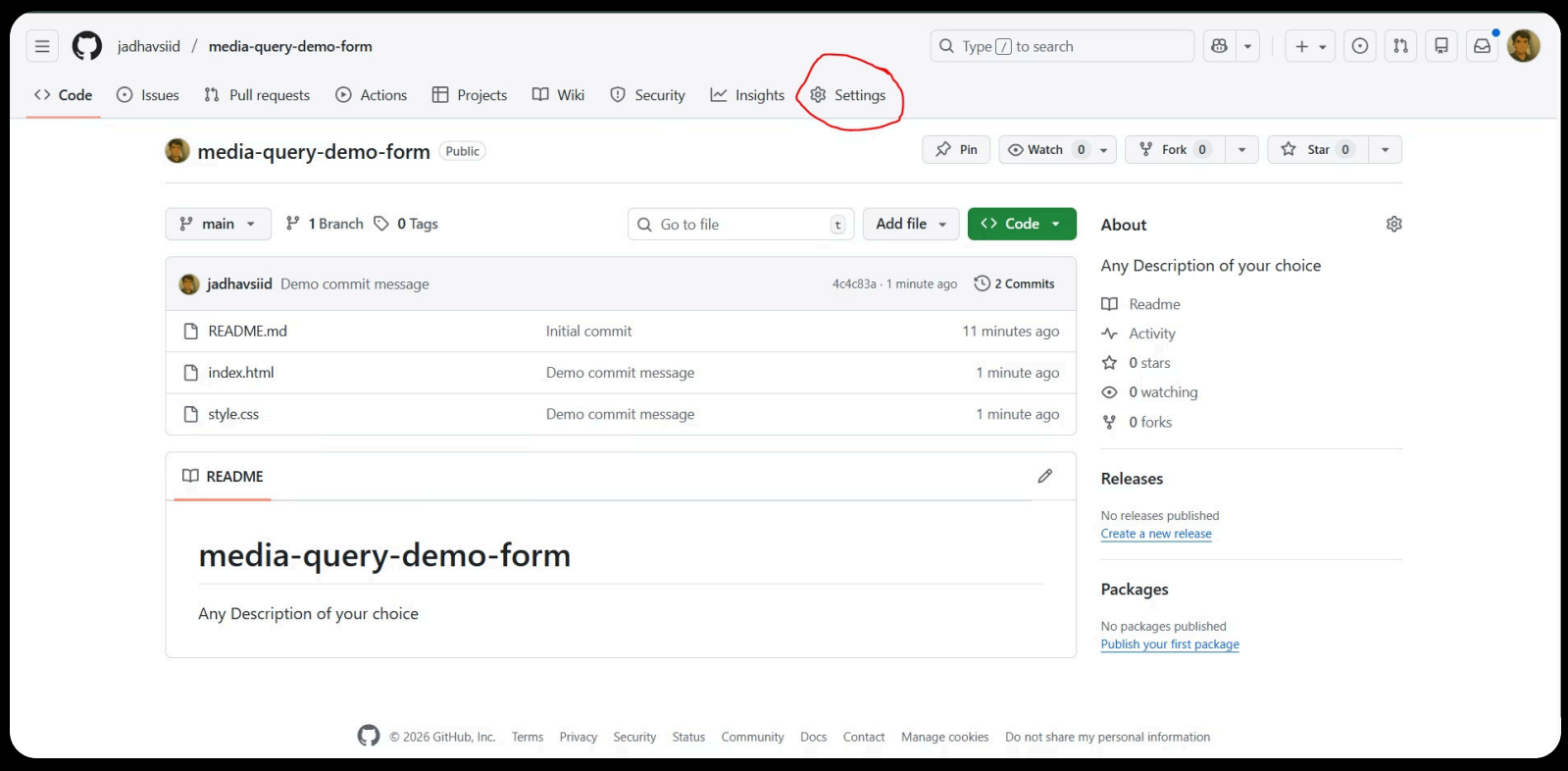


- **Step-7:** Click on **Choose your files** and add files from your local storage, when your files will be added it will get shown like the second image, you can also write your commit message shown and click on **Commit changes.**

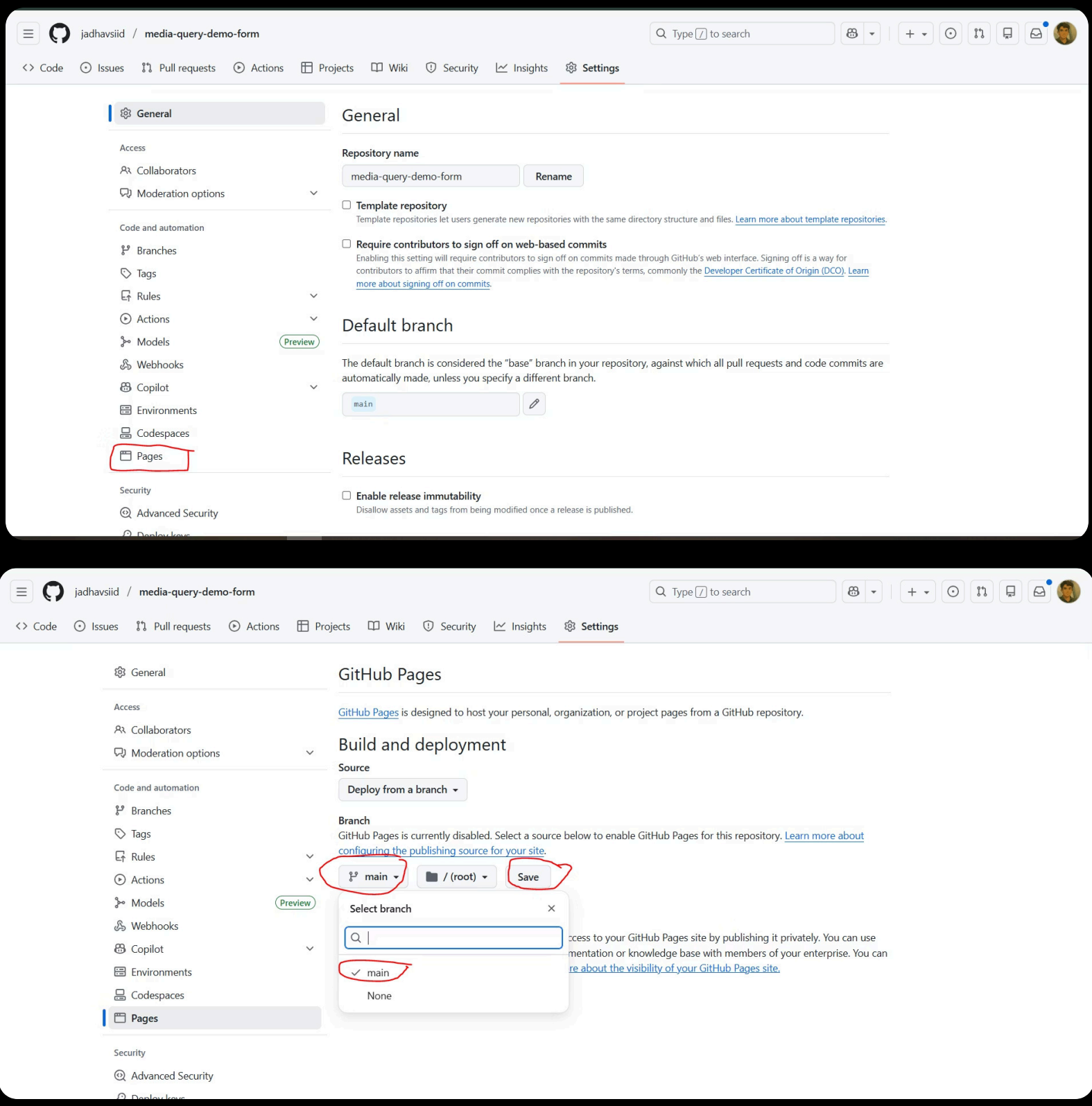


How To Host Website using GitHub Pages ?

Step-8: Now within your repository migrate to **Settings** tab.

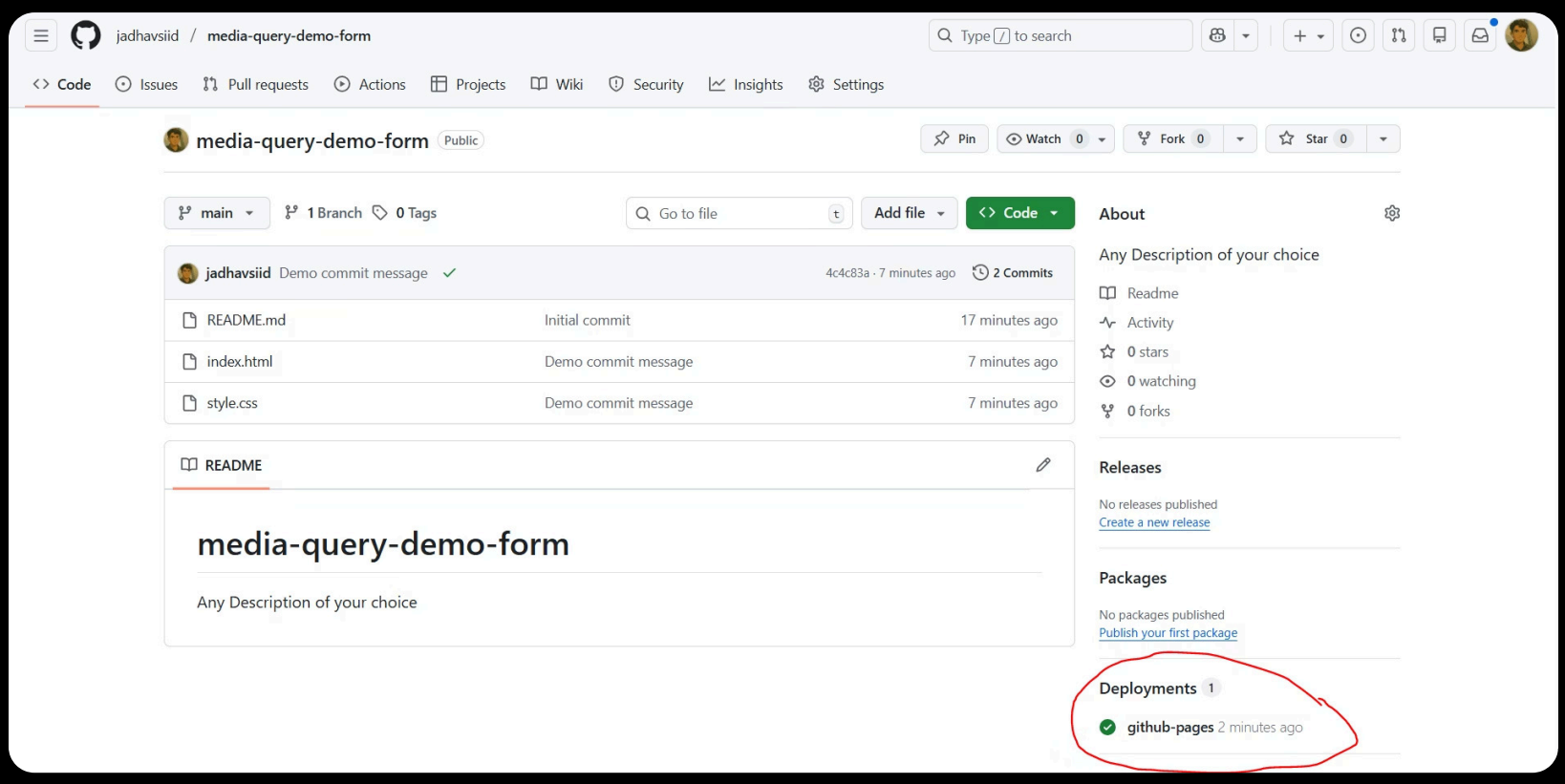


Step-9: On your left sidebar click on **Pages** and choose branch as **Main** and then click on **Save.**



How To Host Website using GitHub Pages ?

Step-10: After that come into your repository and refresh the page using your browser's refresh button, you'll find your deployments here after few minutes. **Click on it.**



- **Step-11: Click here** as shown below and your hosted website will be visible.