

Front-End Development

Labo 4 – CSS 3
Flexbox & Position

J. Pieck, L. Van Houdt, S. Van Peborgh, T. Van Nuland

Inhoud

1 INTRODUCTIE 3

2 EXPLORING THE STARTER FILES 3

2.1 DOWNLOAD STARTER FILES 3
2.2 EXPLORE THE HTML AND CSS 3
2.3 THE END GOAL 4

3 HEADER NAVIGATION 5

4 DE HERO SECTION 7

5 DE USP SECTION 8

6 PRODUCT SECTION 9

7 COOKIE-POLICY 12

8 STICKY HEADER 13

9 NEXT STEPS 14

9.1 EXTRA CHALLENGE 1 14
9.2 EXTRA CHALLENGE 2 14
9.3 EXTRA CHALLENGE 3 14

10 ZIP-FILE INLEVEREN 15

11 REFLECTION 15

1 introductie

Documentatie	Learning materials on Toledo
Goal	In this lab, you will learn to work **with CSS Flexbox and positioning** (relative, absolute, sticky). Step by step, you will build up a web page. By the end, you should **understand how elements behave inside a flex container** and **how positioning influences page layout**.
Learning Goals	By the end of this lab, you will be able to: • Construct various Flexbox layouts • Explain the difference between **absolute** and **relative** positioning • Evaluate how **different flexbox properties** behave within a responsive layout • Implement a **"sticky" navigation bar** that remains visible while scrolling • Use **Font Awesome icons** via a CDN

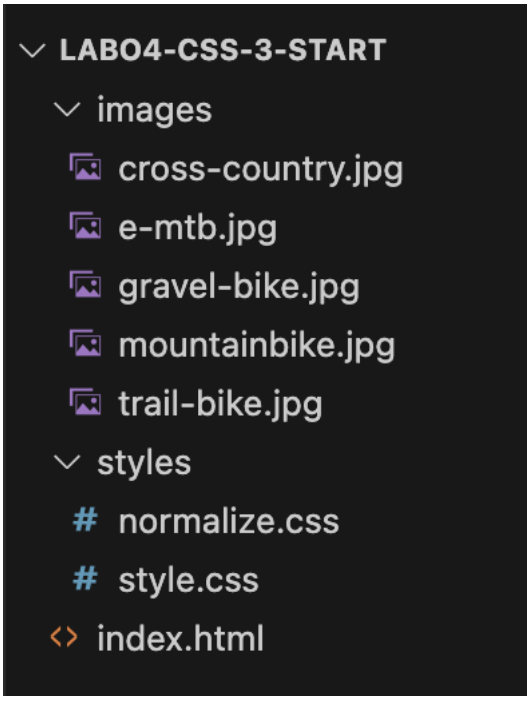
2 Exploring the Starter Files

2.1 Download Starter Files

Open the starter files in VS Code

Rename the folder to labo4-CSS-3.

The folder structure should resemble the example provided.

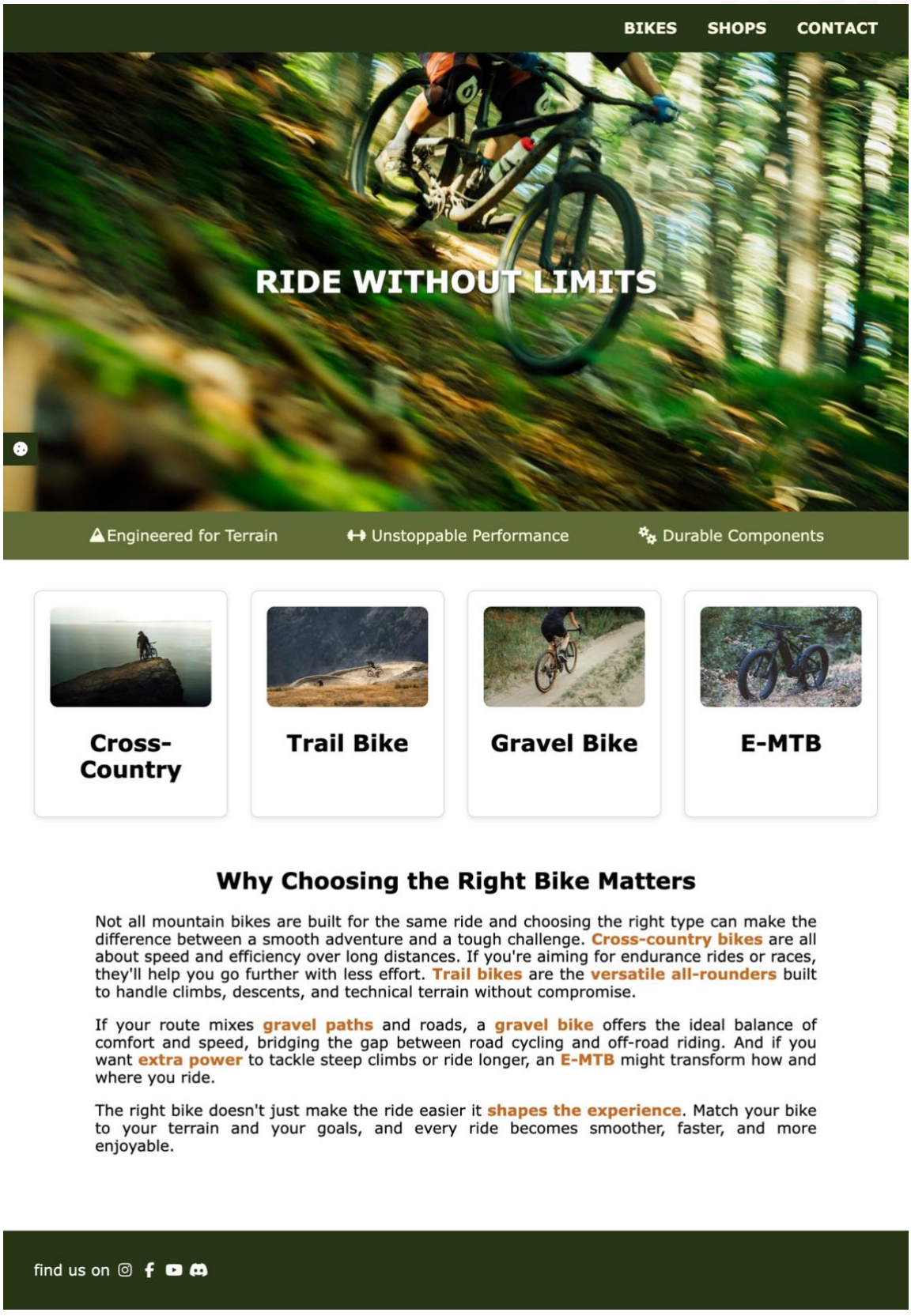


2.2 Explore the HTML and CSS

Open the page in Chrome and use the Inspector to explore the existing page.
Pay attention to the structure of the HTML, the available classes and IDs, and examine the style rules applied to each element to understand what the existing CSS is doing.

2.3 The End Goal

Carefully examine the screenshot of the final result. This is what we'll build step by step throughout the lab.



Compare it side-by-side with the browser view of the starter file.



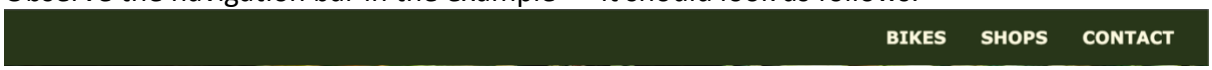
What elements will still need to be added or modified?



Based on your current knowledge, can you already identify where to use Flexbox?
Where might the position property come in handy?

3 Header Navigation

Observe the navigation bar in the example — it should look as follows.



Add the following CSS rules for `nav` in `style.css`

Maak van de `nav` een flexcontainer en voeg een `gap` toe

```
nav {
    display: flex;
    gap: 2rem;
}
```

Increase the value of gap and observe what happens.



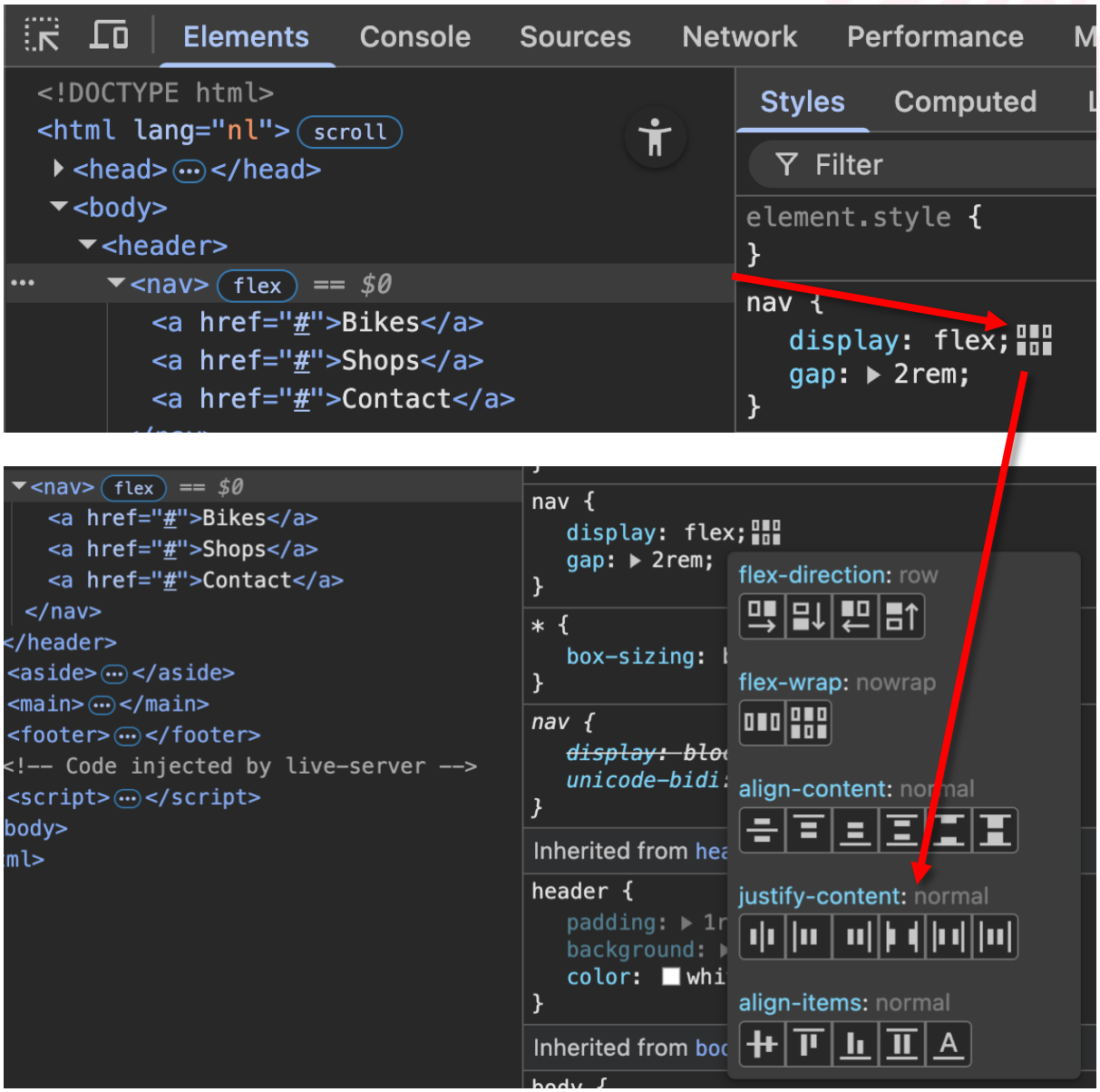
What exactly does the `gap` property do?

Currently, the elements are aligned on the left. To align them to the right, you need to add another flex property.

Consult this article to find the correct property: https://css-tricks.com/snippets/css/a-guide-to-flexbox/

You can also experiment with different flexbox settings in Chrome DevTools.

In the Elements tab, select the `nav` element, then click the flexbox icon next to `display: flex;` to explore different options interactively.



Try several settings until you achieve the desired layout.



Which flex property do you need, and what value should it take?

Once identified, add it to your `nav` selector in `style.css`

4 De Hero Section

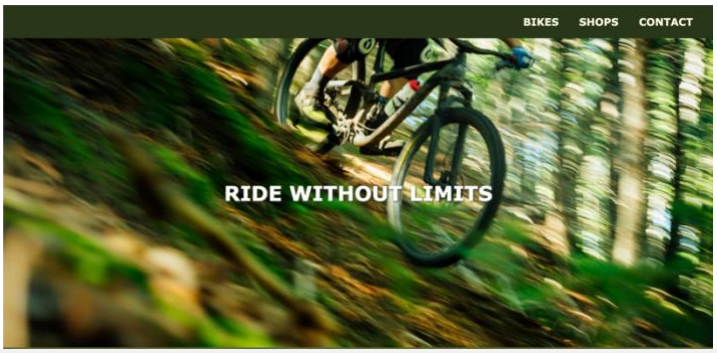
Inspect the following HTML within the `<aside>` for the section with class hero

```
<section class="hero">
  <h1>Ride Without Limits</h1>
</section>
```

This section already has a background image and some base styling in CSS

```
.hero {
    height: 50vh;
    background-image: url('../images/mountainbike.jpg');
    background-size:cover;
    background-position: center;
    color:#fff;
    text-align: center;
}
.hero h1{
    text-transform: uppercase;
    text-shadow:1px 1px 3px #283618;
    margin:0;
}
```

dd flex properties to `.hero` so the title is centred both vertically and horizontally



Which element should become the flex container?

Which flex properties do you need?

5 De USP Section

Inspect the HTML and CSS for the `.usp` section:

```
<section class="usp">
    <ul>
        <li><i class="fa-solid fa-mountain"></i>Engineered for Terrain</li>
        <li><i class="fa-solid fa-dumbbell"></i> Unstoppable Performance</li>
       <li><i class="fa-solid fa-gears"></i> Durable Components</li>
    </ul>
</section>
```

```
.usp {
    background:#606C38;
    color: #FEFAE0;
}
.usp ul {
    padding:1rem;
    margin:0;
}
.usp li{
    list-style-type: none;
}
```



Distribute the items evenly across the width of the page.



Which element should be the flex container, and why not another one?

Use `justify-content` to control distribution, testing each of the following values:

- `space-between`
- `space-around`
- `space-evenly`

How do these values behave differently?

How does the layout respond if you add another `<li>`?

The icons in this section come from font-awesome.
Notice that the `<head>` of your page includes the link to Font Awesome

To use a specific icon, simply lookup and in include the corresponding class.
`<i class="fa-solid fa-house"></i>`

This only works if the `<link>` tag to Font Awesome is present in the `<head>`.

6 Product Section

Inspect the HTML and CSS for the product section. Add the following to your `style.css` under the section with class products.

```
flex-wrap: wrap;
gap: 1.5rem;
justify-content: center;
```

Then locate the selector for class product and add:

```
.product {
    flex-basis: 220px;
     ...
}
```

Notice these are two different classes: products and product!

Resize the browser window, reduce and then enlarge it again.

What happens to the flex items due to the `flex-wrap` property?

What does `flex-basis` do?

Now add `flex-grow` and repeat the resizing:

```
.product {
    flex-basis: 220px;
    flex-grow: 1;
    ...
}
```



What just happened?



What does `flex-grow` control?

Set `flex-grow` to 0 to disable it, then add `flex-shrink`.

```
.product {
    flex-basis: 220px;
    flex-grow: 0;
    flex-shrink: 1;
    ...
}
```

Change the `flex-wrap` property to `nowrap`

```
section.products {
    display: flex;
    flex-wrap: nowrap;
    gap: 1.5rem;
    justify-content: center;
}
```

Resize the window again.



What do you observe?



What does `flex-shrink` do?



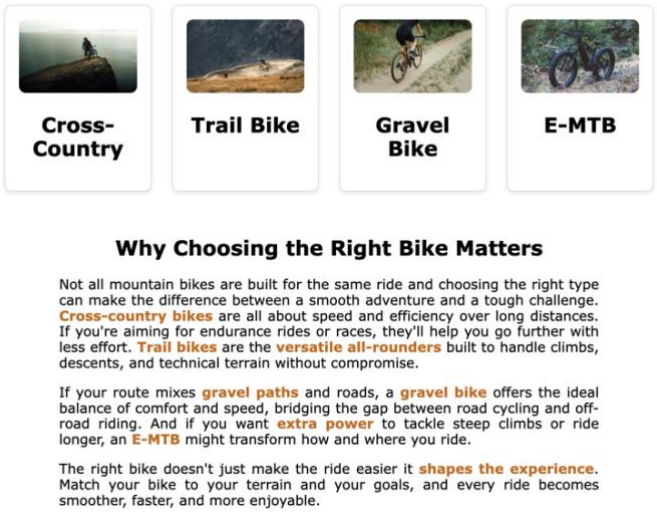
The shorthand for `flex-grow`, `flex-shrink`, and `flex-basis` is

```
flex: 1 1 220px;
```

Since these properties are often used together, this shorthand is considered best practice.

We'll allow flex-grow and flex-shrink. Thus, your final CSS should be this:

```
section.products {
    display: flex;
    flex-wrap: nowrap;
    gap: 1.5rem;
    justify-content: center;
}
.product {
    flex: 1 1 220px;
    background: #fff;
    border: 1px solid #ddd;
    border-radius: 8px;
    text-align: center;
    padding: 1rem;
    box-shadow: 0 2px 5px rgba(0,0,0,0.1);
}
```



7 Cookie-Policy

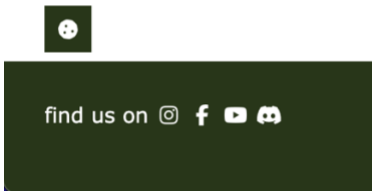
In the footer, you'll find the following HTML

```
<div class="cookies">
    <i class="fa-solid fa-cookie"></i>
    <a href="#" id="cookie-policy">cookie-policy</a>
</div>
```

Add this CSS to the `div` with class `cookies`

```
position: relative;
bottom: 100px;
```

This moves the div slightly upwards, just above the footer.



What does position relative do?

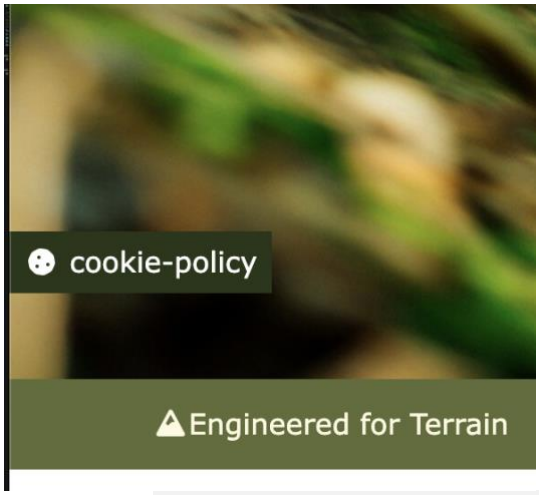
Why doesn't the footer move along with it?

What happens to the space the div previously occupied in the footer?

This is not what we are aiming for in the final result, we want the cookie icon to appear mid-page, above the fold.

Modify the CSS for the div with class cookies to use absolute positioning.

```
position: absolute;
top: 50vh;
left: 0;
z-index: 1;
```



The z-index: 1 ensures the cookie policy appears above all content.
You'll notice the effect in the next chapter.

The hyperlink to the cookie policy appears only when hovering over the cookie icon. Normally, this would require JavaScript but here it's done purely with CSS selectors.
Examine the CSS carefully: do you understand how this trick works?

8 Sticky Header

We want the header to stay visible while scrolling.
Add the `position` property to the `header` so it remains fixed at the top.

```
header{
    position: sticky;
    top: 0;
    z-index: 2;
    ...
}
```

Scroll down, you'll see the header now "sticks" to the top.

The z-index controls the depth of an element, it controls which elements will slide above or below others.
Try setting it to 0 and observe what happens to the cookie policy.

What happens? Why does this occur?

Set the z-index to -1



What happens? Why does this occur?

Finally, set it back to 2.



Remember: you can experiment with these adjustments in Chrome DevTools without modifying your source files.

9 Next Steps

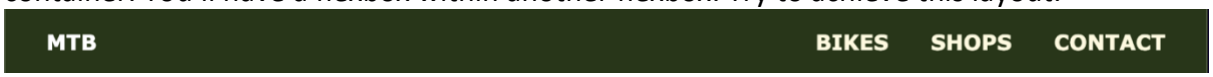
9.1 Extra challenge 1

In the header's HTML, there's a commented-out element

```
<!-- <div id="logo">MTB</div> -->
```

Uncomment this line.

Ensure it appears alongside the nav flexbox, meaning the header itself must also be a flex container. You'll have a flexbox within another flexbox. Try to achieve this layout.



9.2 Extra challenge 2

https://flexboxfroggy.com/

9.3 Extra challenge 3

Try setting the `.cookies` div to `position: fixed;`.

10 ZIP-file Inleveren

- Rename your folder as follows: “lastname_firstname_labo4_css3”
  e.g. Doe_John_labo4_css3
- In Windows Explorer (or Finder on Mac), compress the folder into a ZIP file.
- Do not use .gzip, .7zip, or .rar!
- You should now have a file lastname_firstname_labo4_css3.zip
- Upload this file to the Toledo assignment page

11 Reflection



What are the main differences between relative, absolute, sticky, and fixed positioning?



What distinguishes justify-content from align-items?



What is a flex-container?



What are the values flex-start, flex-end, center, space-evenly, space-around, space-between?



Where do you encounter Flexbox or positioning in real-world websites? Find a few examples.



Why is Flexbox so valuable in designing responsive layouts