How to use container queries in CSS

CSS container queries allow you to apply styles based on the element's container size, rather than the entire viewport. Here is a detailed guide to using container queries in CSS.

CSS container queries allow you to apply styles based on the element's container size, rather than the entire viewport. Here is **a detailed guide on how to use container queries in CSS** .

For a long time, using media queries was the only way to create responsive UI designs on different screen sizes. But it still has limitations. One of the biggest problems with using media queries is that you can only style an element to respond to changes in screen size, not its closest container element.

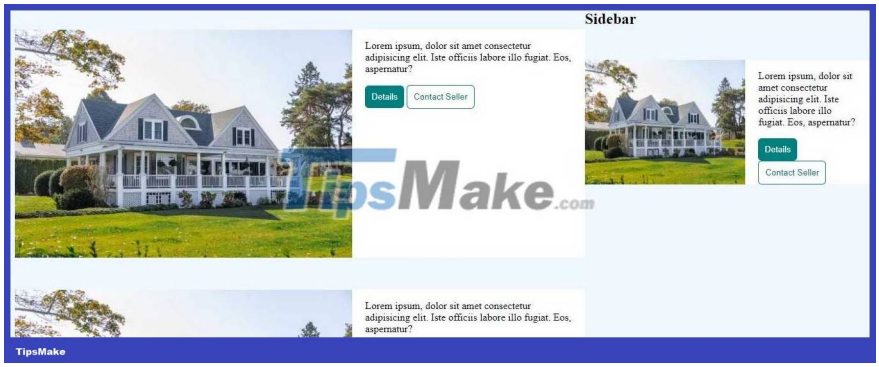
Container queries are introduced to solve the above problem. They are increasingly used in frameworks like React, Vue.js.

Learn how to use container queries to make your **CSS responsive with TipsMake.com.com.**

Why use CSS Container queries?

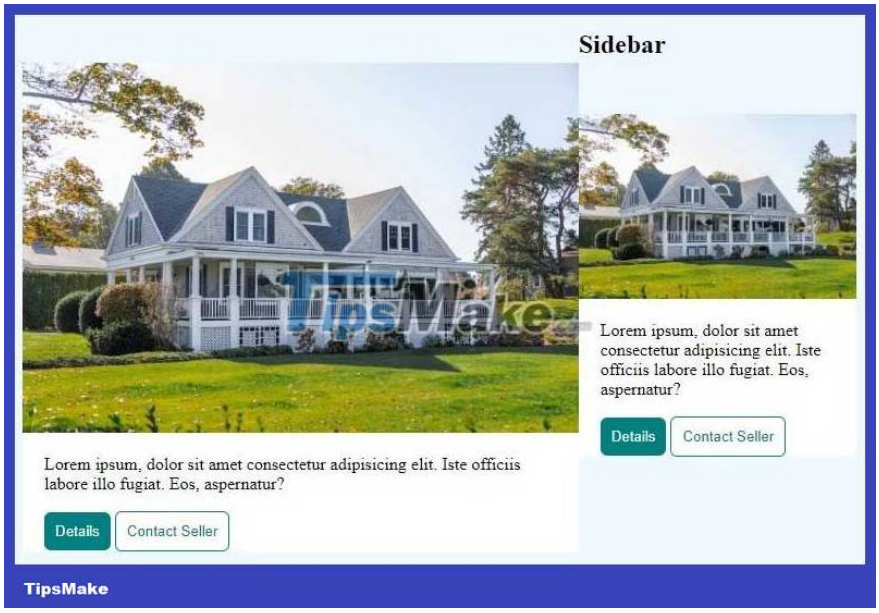
To understand the importance of container queries, consider the following example. First, you need to copy the starter code from the GitHub repository.

After successfully cloned the repo, run this code. You will find a web page similar to the following image:



Here you have a grid layout consisting of 2 columns: main section and sidebar. The main body looks good, but the sidebar looks out of proportion to the main content.

This layout is responsive. When you narrow the browser, you can see the tag transform into a vertical column:



In other words, when the content cannot be read, the layout transforms into a vertical column, where the image is superimposed on the content. This affects the results of media queries. This is the only way that you can decide the size of the elements across the entire screen size.

In this case, whenever the screen is smaller than 800px, you stack everything on top of each other using Flexbox alignment. On larger screens, put content side by side:

```
@media(max-width: 800px) { .card { flex-direction: column; } .card-header { widt
```

For the longest time, media queries have been one of the main web design principles for creating responsive layouts with CSS. But you will inevitably run into situations where media queries are not enough or not the most convenient solution.

One of the main scenarios here is when you have a sidebar. In this case you have to directly select the sidebar tag and try to make it smaller.

```
.sidebar .card { /* Làm th? sidebar nh? h?
n */ } @media(max-width: 800px) { /*T?o ki?u trang khi màn hình nh? h?
n 800px */ }
```

The problem can be quite complicated if you are working with a lot of elements because you have to manually select them all and resize them. You will end up with more code and more complexity.

This is where container queries can come in handy. Instead of being forced to use the viewport when resizing, you can use any element on the page as a container. The container will then have its own width based on your media queries.

How to create a container query

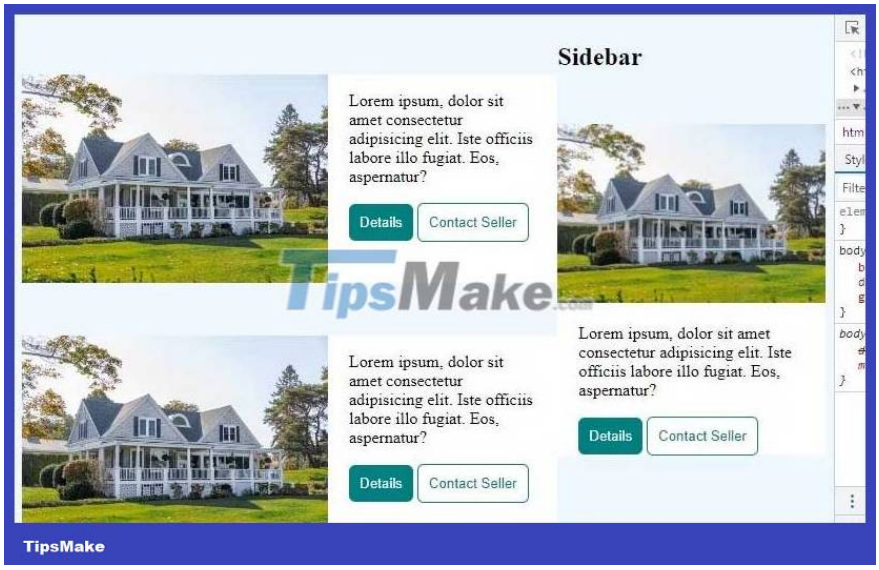
To create a container query, you start by targeting the element you want to use as the container. Inside this block, you need to set **the container-type** to **inline-size** :

```
main, .sidebar { container-type: inline-size }
```

When saving the CSS file, nothing changes on the page. But how you can use a container query to resize and restyle the tag nested in the main body and sidebar. In the following container query, you are changing the tag to a vertical column on a screen that is 500px wide or less:

```
@container(max-width: 500px) { .card { flex-direction: column; } .card-header { w
```

This will work like a normal media query. But instead of measuring the size of the screen, you are measuring the size of the container (main and sidebar). So now when the container is at 500px or more, you use the horizontal viewer. Otherwise, you use vertical (default for flexbox).



From the image above, you can see that the sidebar is vertical because it is less than 500px. Meanwhile, the main content keeps the horizontal layout because it's 500px larger. If the browser is minimized, the sidebar and main content will both use vertical alignment when it reaches 500px or less.

This container query is extremely powerful because it allows to resize everything based on the container instead of the full width of the browser. This is especially useful when dealing with components (like React or Vue).

With container queries, you can easily resize container-based UI elements, allowing you to create completely independent components.

Name the container

When creating the @container query , it first looks for the container of the element you are targeting in this query. Here is the main container and the sidebar.

Even if the tags are in another container, it will just ignore the other container and select only the nearest container. This is part of a broader concept of CSS aka CSS selectors.

The following example turns the body component into a container:

```
body { container-type: inline-size; }
```

Now we have 3 separate containers: body, main and aside. By default, the tag being targeted in the container query is closer to the main body or sidebar than to the body. Therefore, it uses the main sections and sidebar as containers to query the container.

To override this behavior, you need to do two things. First, you need to name the container for the content.

```
body { container-type: inline-size; container-name: body; }
```

Then when creating the container query, you need to define the container name after **@container** .

```
@container body (max-width: 1000px){ /* Các quy t?c CSS nh?m m?
c tiêu container ph?n n?i dung */ }
```

This is useful if you want to use a specific element as the container instead of the closest container for the element you are targeting. In other words, you can choose any container and tweak your layout.

Container unit

Another great feature about containers is that you can use container units. They act like viewport units. However, the container unit uses cqw (to set the width) and cqh (to set the height). These units determine the exact height and width of the container.

Here's what you need to know about using container queries in CSS. Hope the article is useful to you.