**Gemini**

**CSS Grid System and Breakpoints**

**CSS Grid System**

The CSS Grid Layout module is a powerful tool for creating complex layouts on the web. It provides a flexible grid-based system for arranging items on a page. Unlike traditional layout methods, Grid allows you to position items both horizontally and vertically using a grid container and grid tracks.

**Implementing CSS Grid**

To implement a CSS grid layout, you define a grid container and place items within it. You then use grid properties to position the items within the grid.

**Example:**

HTML

<div class="container">

<div class="item">Item 1</div>

<div class="item">Item 2</div>

<div class="item">Item 3</div>

</div>

Use code [with caution.](/faq#coding)

CSS

.container {

display:

grid;

grid-template-columns: repeat(auto-fit, minmax(200px, 1fr));

grid-gap: 20px;

}

.item {

background-color:

lightblue;

padding: 20px;

text-align: center;

}

Use code [with caution.](/faq#coding)

In this example:

* The .container class defines the grid container.
* grid-template-columns sets the number and size of columns.
* grid-gap defines the spacing between grid items.

**Breakpoints and Media Queries**

**Breakpoints** are specific screen sizes at which the layout of your website changes. Media queries allow you to define different styles for different screen sizes, ensuring your website is responsive.

**Example:**

CSS

@media only screen and (max-width: 768px) {

.container {

grid-template-columns: 1fr;

}

}

Use code [with caution.](/faq#coding)

This media query will change the grid layout to a single column when the screen width is less than or equal to 768 pixels.

**Responsive Layout Using CSS Grid and Media Queries**

HTML

<div class="container">

<div class="item">Item 1</div>

<div class="item">Item 2</div>

<div class="item">Item 3</div>

<div class="item">Item 4</div>

</div>

Use code [with caution.](/faq#coding)

CSS

.container {

display: grid;

grid-template-columns: repeat(auto-fit, minmax(200px, 1fr));

grid-gap: 20px;

}

.item {

background-color:

lightblue;

padding: 20px;

text-align: center;

}

@media only screen and (max-width: 768px) {

.container {

grid-template-columns: 1fr;

}

}

Use code [with caution.](/faq#coding)

This example creates a responsive grid layout that adapts to different screen sizes. On larger screens, the items will be arranged in multiple columns. On smaller screens, the items will stack on top of each other.

By combining CSS Grid with media queries, you can create flexible and responsive layouts that look great on a variety of devices.