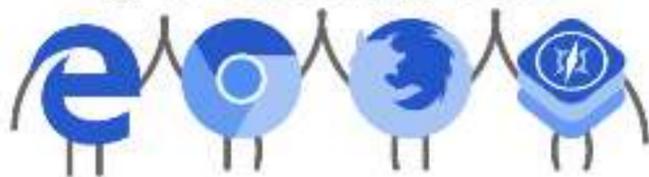


BUILDING LAYOUTS WITH MODERN CSS

#TeamWeb



Original image by the amazing Lin Clark

Link to slides:

<http://bit.ly/detech-css>

Surname First name

陈	慧	晶
Chen	Hui	Jing



@hj_chen



<https://events.mozilla.org/techspeakers>



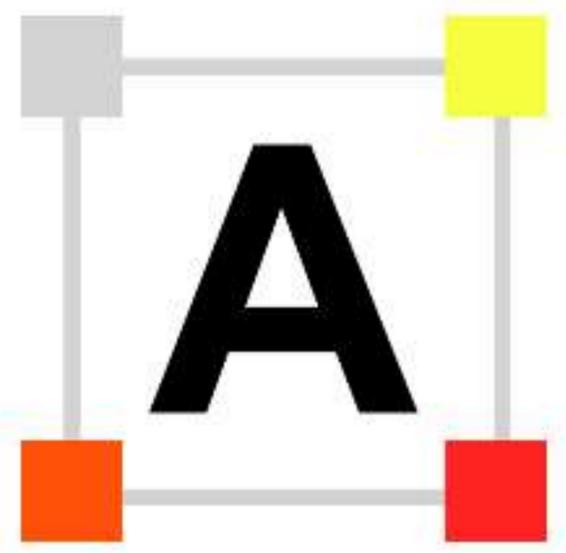
🥑 Developer Advocate 🥑

nexmo[®]
The Vonage[®] API Platform

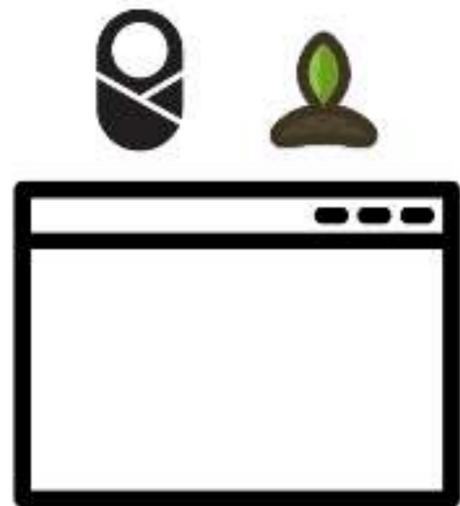
General agenda

- Introduction and set up
- Pre-grid techniques and concepts
- Flexbox basics
 - Flex syntax
 - Aligning flex items
- Real-world application: Image gallery
- Grid basics
 - Auto-placement
 - Flexible sizing
 - Manual placement
- Real-world application: Responsive dashboard





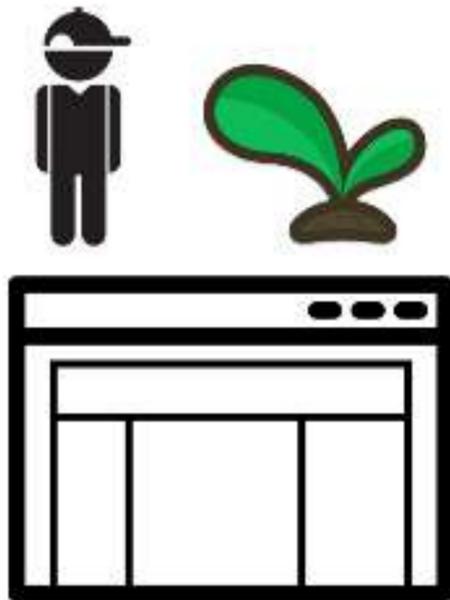
Evolution of browsers



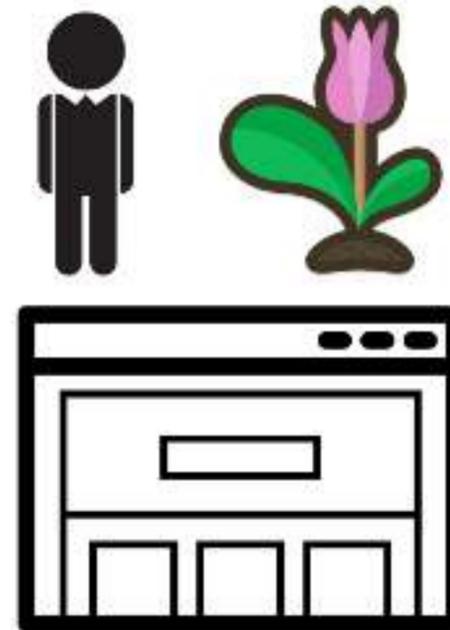
No layout



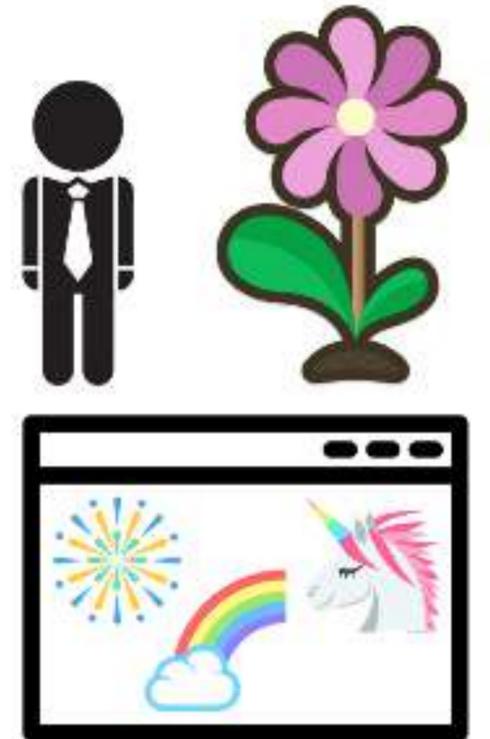
HTML Tables



CSS Floats

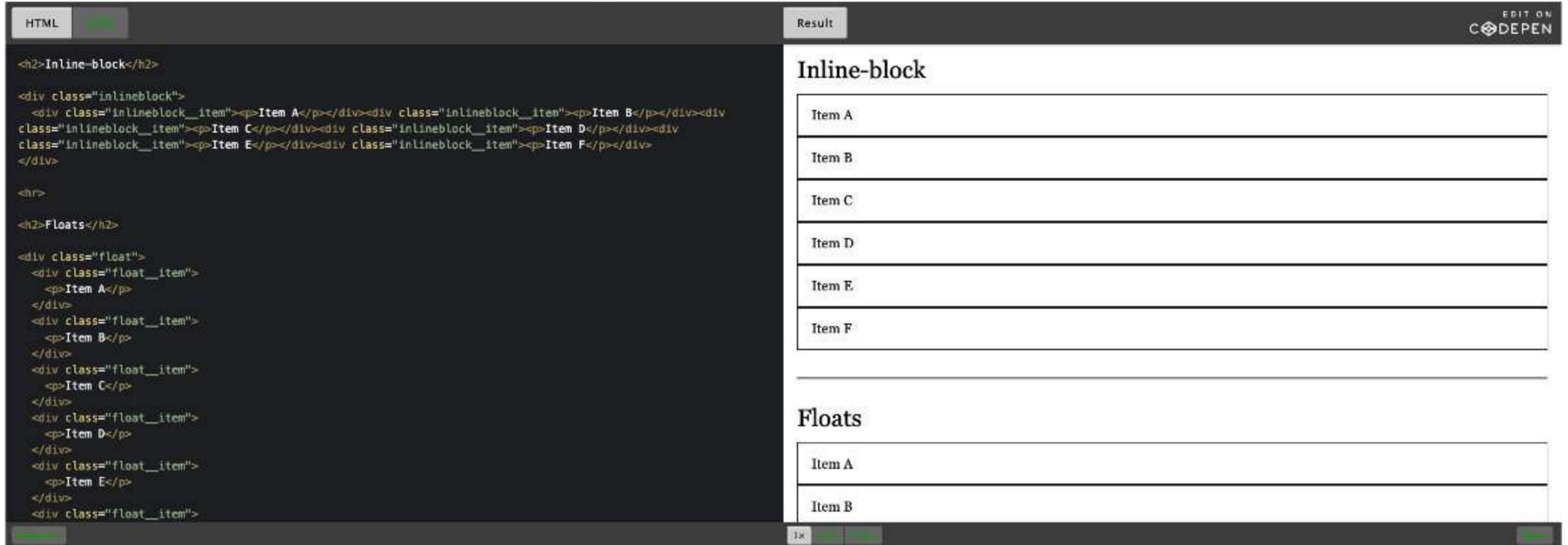


Frameworks



Grid and beyond

Pre-grid techniques



The screenshot shows a CodePen editor with two tabs: 'HTML' and 'Result'. The 'HTML' tab contains the following code:

```
<h2>Inline-block</h2>

<div class="inlineblock">
  <div class="inlineblock_item"><p>Item A</p></div><div class="inlineblock_item"><p>Item B</p></div><div
class="inlineblock_item"><p>Item C</p></div><div class="inlineblock_item"><p>Item D</p></div><div
class="inlineblock_item"><p>Item E</p></div><div class="inlineblock_item"><p>Item F</p></div>
</div>

</div>

<h2>Floats</h2>

<div class="float">
  <div class="float_item">
    <p>Item A</p>
  </div>
  <div class="float_item">
    <p>Item B</p>
  </div>
  <div class="float_item">
    <p>Item C</p>
  </div>
  <div class="float_item">
    <p>Item D</p>
  </div>
  <div class="float_item">
    <p>Item E</p>
  </div>
  <div class="float_item">
    <p>Item F</p>
  </div>
</div>
```

The 'Result' tab shows the rendered output. Under the heading 'Inline-block', there is a vertical stack of six rectangular boxes, each containing one of the items from A to F. Under the heading 'Floats', there is a vertical stack of two rectangular boxes, the top one containing 'Item A' and the bottom one containing 'Item B'. The 'Codepen' logo and 'EDIT ON CODEPEN' text are visible in the top right corner of the editor interface.

<https://codepen.io/huijing/pen/KGOmLZ/>

The flex syntax

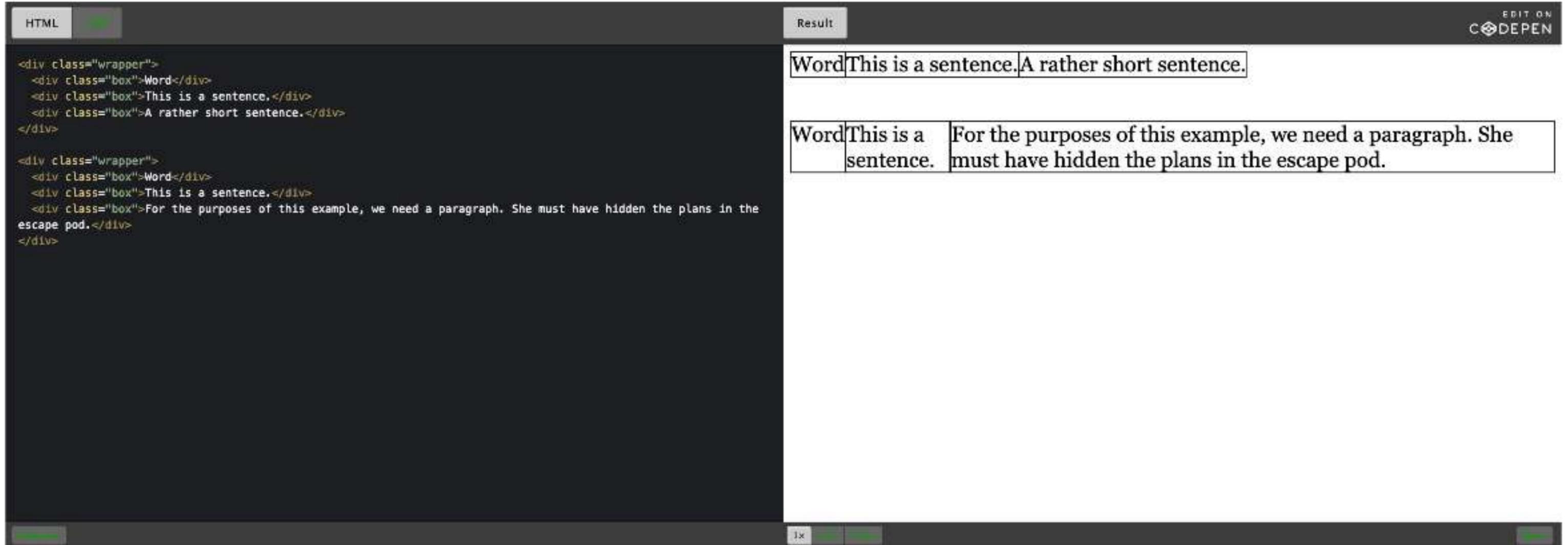
brackets are for grouping

[<'flex-grow'> <'flex-shrink'>? || <'flex-basis'>]

preceding type/
word/group is
optional

seperates 2 options,
one or more must occur,
order doesn't matter

Flexbox basics



The screenshot shows a CodePen editor with two tabs: 'HTML' and 'Result'. The 'HTML' tab contains the following code:

```
<div class="wrapper">
  <div class="box">Word</div>
  <div class="box">This is a sentence.</div>
  <div class="box">A rather short sentence.</div>
</div>

<div class="wrapper">
  <div class="box">Word</div>
  <div class="box">This is a sentence.</div>
  <div class="box">For the purposes of this example, we need a paragraph. She must have hidden the plans in the escape pod.</div>
</div>
```

The 'Result' tab shows the rendered output. The first example shows three boxes in a row: 'Word', 'This is a sentence.', and 'A rather short sentence.'. The second example shows three boxes in a row: 'Word', 'This is a sentence.', and 'For the purposes of this example, we need a paragraph. She must have hidden the plans in the escape pod.'. The 'EDIT ON CODEPEN' logo is visible in the top right corner of the editor.

<https://codepen.io/huijing/pen/NEPwoj/>

Flex shorthand

`flex: initial`

`flex: 0 1 auto`, cannot grow but can shrink when there isn't enough space

`flex: auto`

`flex: 1 1 auto`, can grow and shrink to fit available space

`flex: none`

`flex: 0 0 auto`, cannot grow or shrink, AKA inflexible

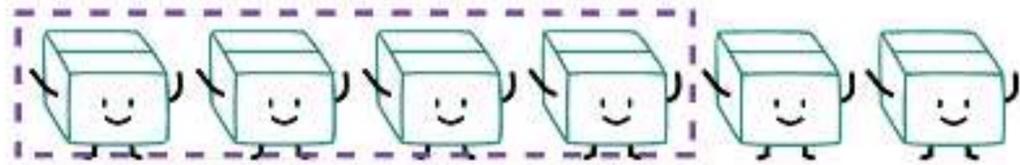
`flex: <positive-number>`

`flex: <positive-number> 1 0`, can grow and shrink, extent of growth depends on flex factor

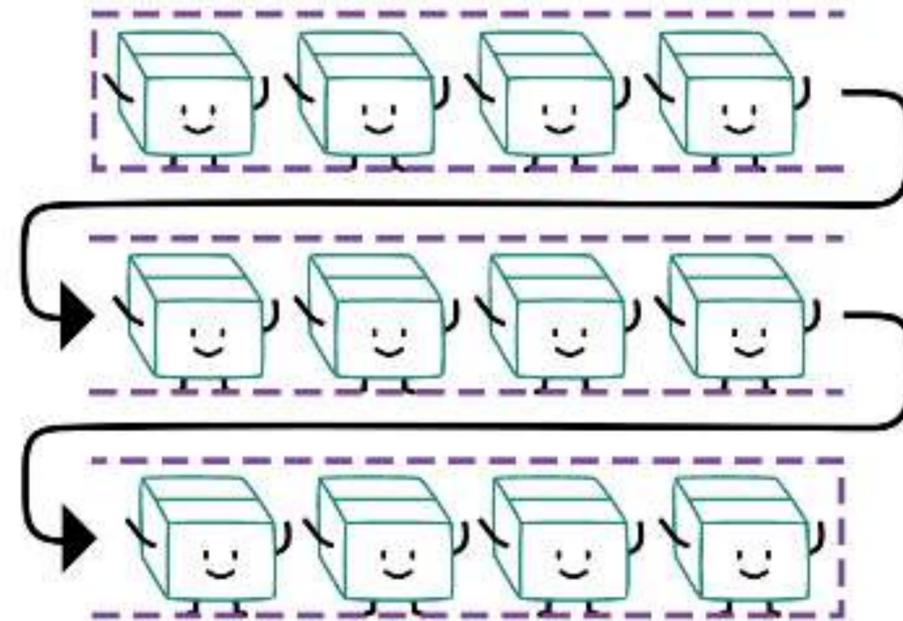
“ Authors are encouraged to control flexibility using the flex shorthand rather than with its longhand properties directly, as the shorthand *correctly resets any unspecified components to accommodate common uses.* ”

–CSS Flexible Box Layout Module Level 1

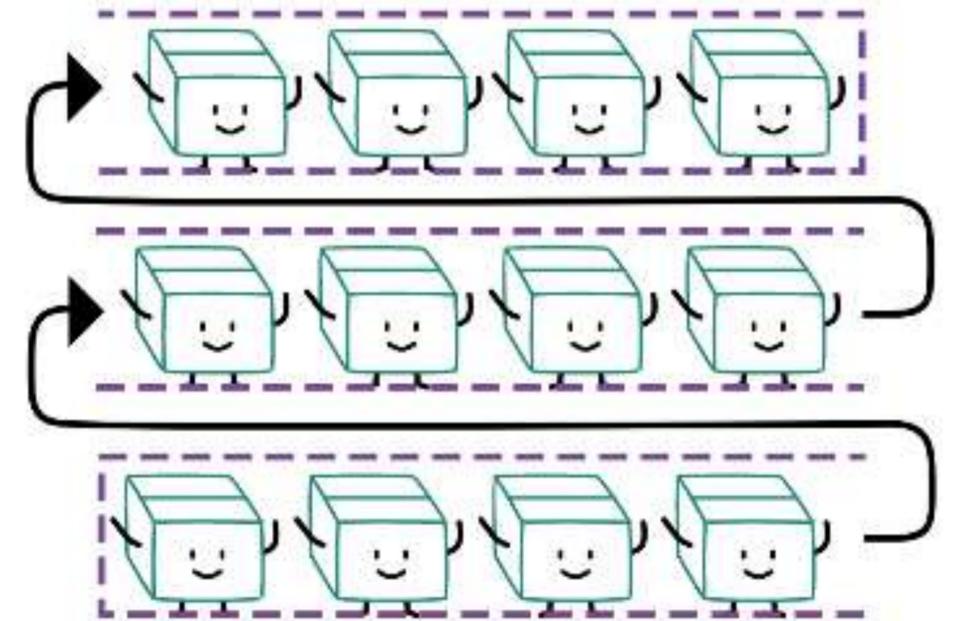
Flex lines



nowrap



wrap



wrap-reverse

Auto-margins with Flexbox

SCSSResultEDIT ON CODEPEN

```
.flex {
  display: flex;
  height: 100%;
}

.centre .flex__item {
}

nav {
  box-sizing: border-box;
  background-color: black;
  display: flex;
}

nav a {
  padding: 1em;
  color: white;
  display: inline-block;
}

nav a:last-child {
}

// General styles
html {
  height: 100%;
}
```



The image shows a CodePen editor interface. On the left, there is a dark-themed code editor with SCSS code. The code defines a flex container and a centered flex item, along with a navigation bar with white text on a black background. On the right, the 'Result' tab shows a white background with a small, centered image of a kitten. The kitten is a tabby with blue eyes, looking directly at the camera. The CodePen logo and 'EDIT ON CODEPEN' text are visible in the top right corner of the editor.

<https://codepen.io/huijing/pen/EOWedx/>

Aligning flex items

The screenshot shows a CodePen editor with the following HTML code in the left pane:

```
<div class="wrapper">
  <div class="box">1一</div>
  <div class="box">2二</div>
  <div class="box">3三</div>
  <div class="box">4四</div>
  <div class="box">5五</div>
  <div class="box">6六</div>
  <div class="box">7七</div>
  <div class="box">8八</div>
  <div class="box">9九</div>
  <div class="box">10十</div>
  <div class="box">11十一</div>
  <div class="box">12十二</div>
  <div class="box">13十三</div>
  <div class="box">14十四</div>
  <div class="box">15十五</div>
  <div class="box">16十六</div>
  <div class="box">17十七</div>
  <div class="box">18十八</div>
  <div class="box">19十九</div>
  <div class="box">20二十</div>
</div>
```

The right pane, labeled "Result", shows the rendered output. It features a flex container with 20 items, each containing a number and its corresponding Chinese numeral. The items are arranged in three rows, demonstrating flex alignment. The first row contains items 1 through 7. The second row contains items 8 through 13. The third row contains items 14 through 18. The remaining items (19 and 20) are not visible in the screenshot. The items are aligned to the center of the container.

1一	2二	3三	4四	5五	6六	7七
8八	9九	10十	11十一	12十二	13十三	
14十四	15十五	16十六	17十七	18十八		

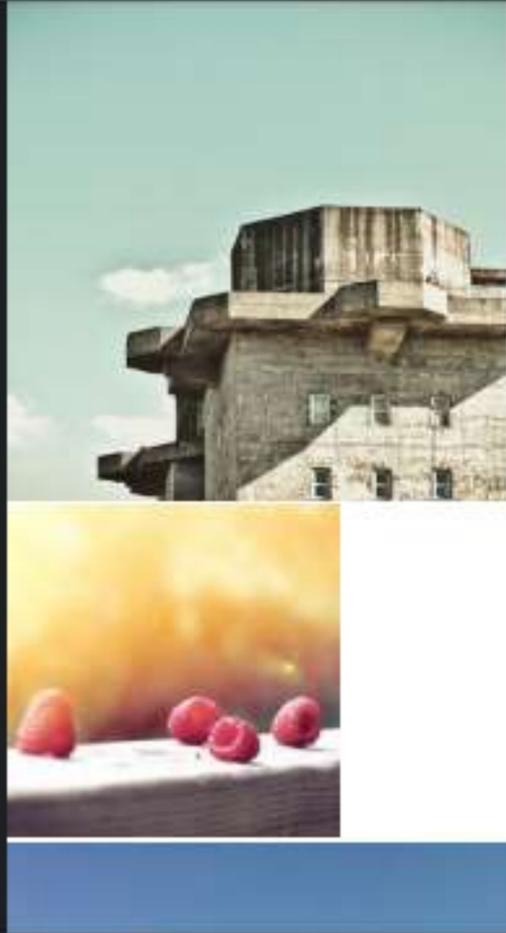
<https://codepen.io/huijing/pen/zMNqOo/>

Image gallery

HTML Result EDIT ON CODEPEN

```
<ul>
<li></li>

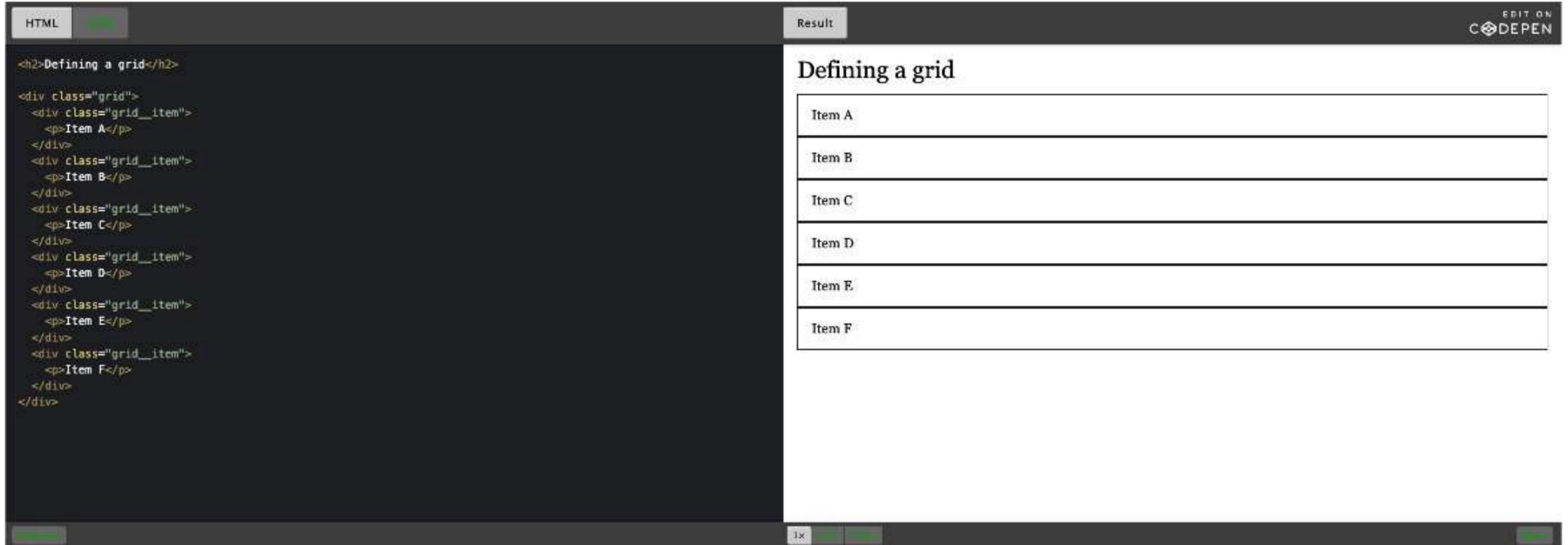
```



1x

<https://codepen.io/huijing/pen/dQPZmL/>

Defining a grid



The screenshot shows a CodePen editor with two tabs: "HTML" and "Result". The "HTML" tab is active, displaying the following code:

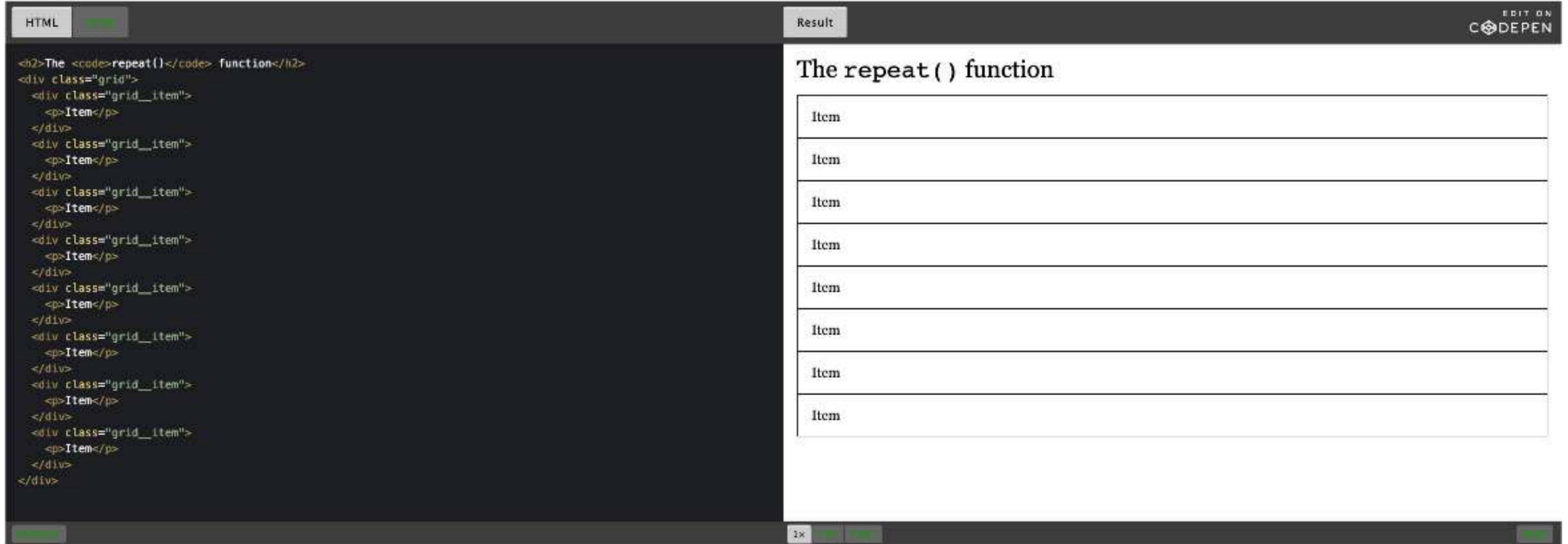
```
<h2>Defining a grid</h2>

<div class="grid">
  <div class="grid_item">
    <p>Item A</p>
  </div>
  <div class="grid_item">
    <p>Item B</p>
  </div>
  <div class="grid_item">
    <p>Item C</p>
  </div>
  <div class="grid_item">
    <p>Item D</p>
  </div>
  <div class="grid_item">
    <p>Item E</p>
  </div>
  <div class="grid_item">
    <p>Item F</p>
  </div>
</div>
```

The "Result" tab shows the rendered output, which is a vertical stack of six rectangular boxes, each containing one of the items from the code: "Item A", "Item B", "Item C", "Item D", "Item E", and "Item F".

<https://codepen.io/huijing/pen/mzKoNj/>

The repeat() function



The screenshot shows a CodePen editor with two tabs: 'HTML' and 'Result'. The 'HTML' tab contains the following code:

```
<h2>The repeat() function</h2>
<div class="grid">
  <div class="grid_item">
    <p>Item</p>
  </div>
  <div class="grid_item">
    <p>Item</p>
  </div>
</div>
```

The 'Result' tab shows the rendered output: a heading 'The repeat () function' followed by a vertical stack of ten rectangular boxes, each containing the text 'Item'.

<https://codepen.io/huijing/pen/XxveJe/>

grid-auto-row and grid-auto-column



The screenshot shows a CodePen editor with two tabs: 'HTML' and 'Result'. The 'HTML' tab contains the following code:

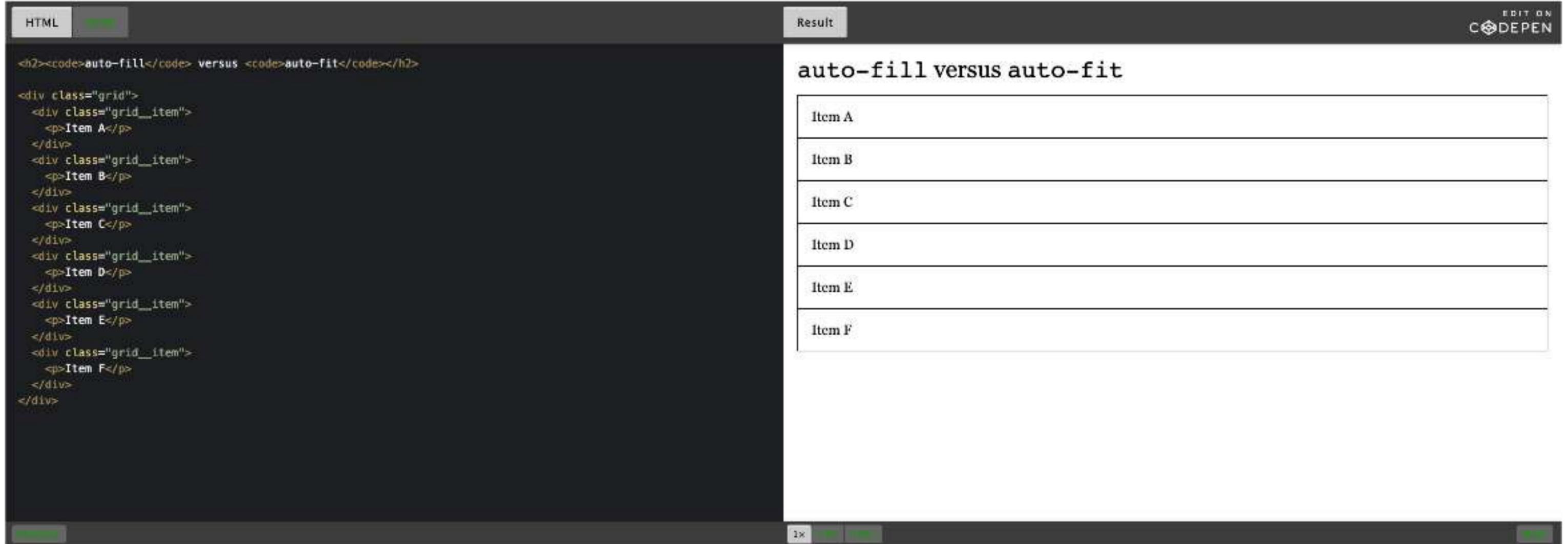
```
<h2><code>grid-auto-row</code> and <code>grid-auto-column</code></h2>

<div class="grid">
  <div class="grid__item">
    <p>A</p>
    <p>A</p>
  </div>
  <div class="grid__item">
    <p>B</p>
  </div>
  <div class="grid__item">
    <p>C</p>
  </div>
  <div class="grid__item">
    <p>D</p>
  </div>
  <div class="grid__item">
    <p>E</p>
  </div>
  <div class="grid__item">
    <p>F</p>
  </div>
  <div class="grid__item">
    <p>G</p>
  </div>
  <div class="grid__item">
    <p>H</p>
  </div>
  <div class="grid__item">
    <p>I</p>
  </div>
  <div class="grid__item">
    <p>J</p>
  </div>
</div>
```

The 'Result' tab shows the rendered output, which is a vertical stack of ten rectangular boxes, each containing a letter from A to J. The boxes are separated by thin horizontal lines, and the letters are centered within each box.

<https://codepen.io/huijing/pen/ePqyMz/>

auto-fill versus auto-fit



The screenshot shows a CodePen editor with two tabs: 'HTML' and 'Result'. The 'HTML' tab contains the following code:

```
<h2><code>auto-fill</code> versus <code>auto-fit</code></h2>

<div class="grid">
  <div class="grid_item">
    <p>Item A</p>
  </div>
  <div class="grid_item">
    <p>Item B</p>
  </div>
  <div class="grid_item">
    <p>Item C</p>
  </div>
  <div class="grid_item">
    <p>Item D</p>
  </div>
  <div class="grid_item">
    <p>Item E</p>
  </div>
  <div class="grid_item">
    <p>Item F</p>
  </div>
</div>
```

The 'Result' tab shows the rendered output, which is a vertical list of six items, each in a separate box:

Item A
Item B
Item C
Item D
Item E
Item F

The interface includes a '1x' zoom level indicator and a 'EDIT ON CODEPEN' button in the top right corner.

<https://codepen.io/huijing/pen/GYVyMX/>

The grid-auto-flow property

The screenshot shows a CodePen editor with the following HTML code in the left pane:

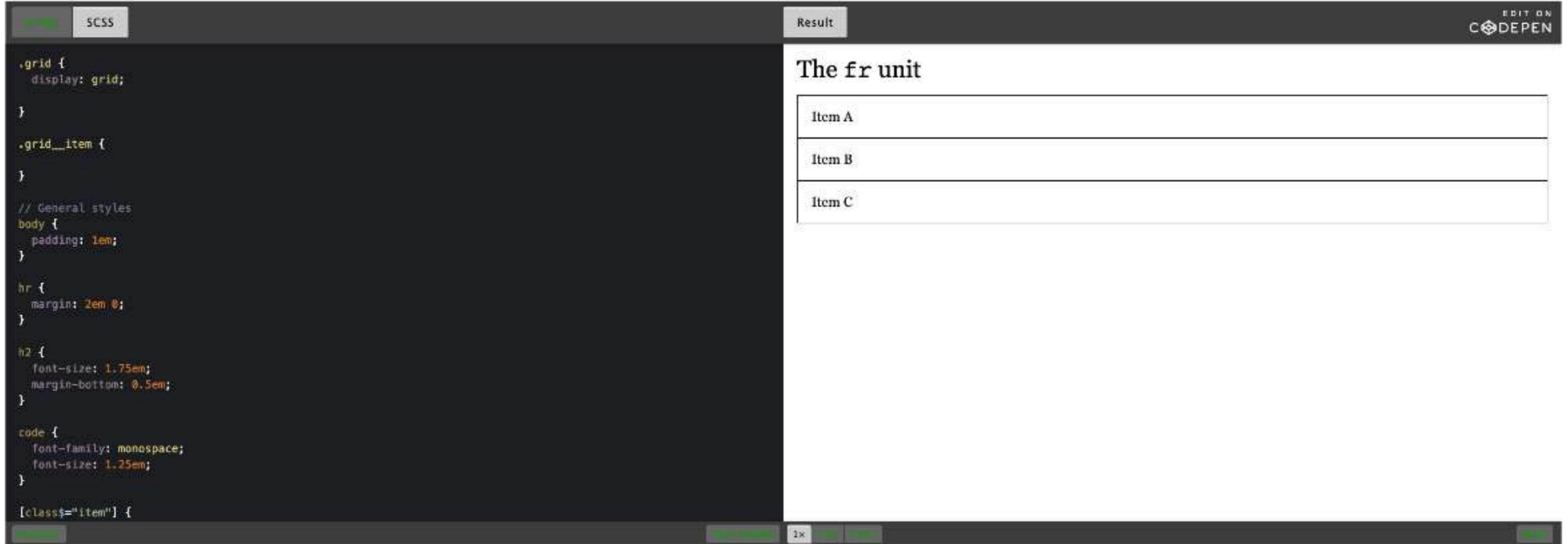
```
<h2>The grid-auto-flow property</h2>
<div class="grid">
  <div class="grid_item">
    <p>A</p>
  </div>
  <div class="grid_item">
    <p>B</p>
  </div>
  <div class="grid_item">
    <p>C</p>
  </div>
  <div class="grid_item">
    <p>D</p>
  </div>
  <div class="grid_item">
    <p>E</p>
  </div>
  <div class="grid_item">
    <p>F</p>
  </div>
  <div class="grid_item">
    <p>G</p>
  </div>
  <div class="grid_item">
    <p>H</p>
  </div>
  <div class="grid_item">
    <p>I</p>
  </div>
```

The right pane shows the rendered result, titled "The grid-auto-flow property". It displays a grid with 10 rows and 2 columns. The items are arranged as follows:

A	B
C	D
E	
F	G
H	
I	J
K	
L	M
N	
O	P

<https://codepen.io/huijing/pen/LgwegQ/>

The fr unit



The screenshot shows a CodePen editor with two tabs: 'SCSS' and 'Result'. The 'SCSS' tab contains the following code:

```
.grid {
  display: grid;
}

.grid__item {
}

// General styles
body {
  padding: 1em;
}

hr {
  margin: 2em 0;
}

h2 {
  font-size: 1.75em;
  margin-bottom: 0.5em;
}

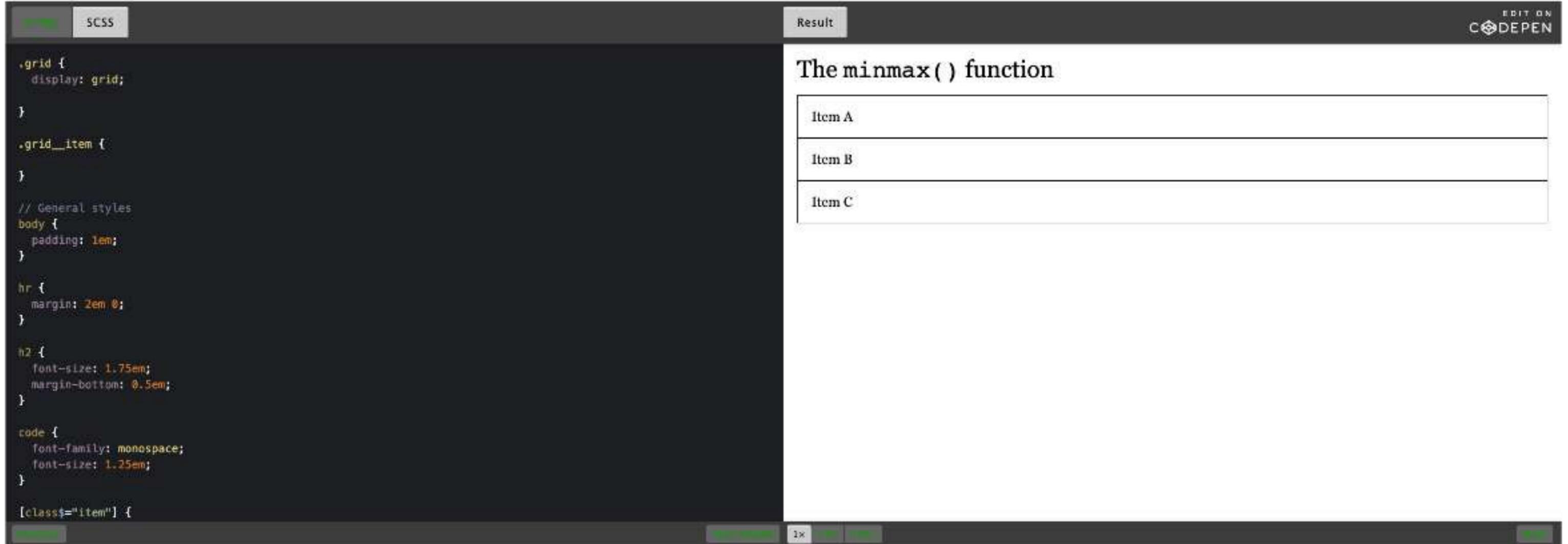
code {
  font-family: monospace;
  font-size: 1.25em;
}

[class$="item"] {
```

The 'Result' tab shows the rendered output, which is a vertical stack of three items, each in a separate box with a border and padding. The items are labeled 'Item A', 'Item B', and 'Item C' from top to bottom. The title of the result is 'The fr unit'. In the top right corner of the editor, there is a link to 'EDIT ON CODEPEN'.

<https://codepen.io/huijing/pen/WaVdLN/>

The minmax() function



The image shows a CodePen editor interface. On the left, the SCSS code is displayed in a dark theme. On the right, the rendered result is shown in a light theme. The rendered result displays the text 'The minmax() function' followed by three horizontal lines, each containing the text 'Item A', 'Item B', and 'Item C' respectively. The CodePen logo and 'EDIT ON CODEPEN' text are visible in the top right corner of the editor.

```
.grid {
  display: grid;
}

.grid__item {
}

// General styles
body {
  padding: 1em;
}

hr {
  margin: 2em 0;
}

h2 {
  font-size: 1.75em;
  margin-bottom: 0.5em;
}

code {
  font-family: monospace;
  font-size: 1.25em;
}

[class$="item"] {
```

Result

EDIT ON CODEPEN

The minmax() function

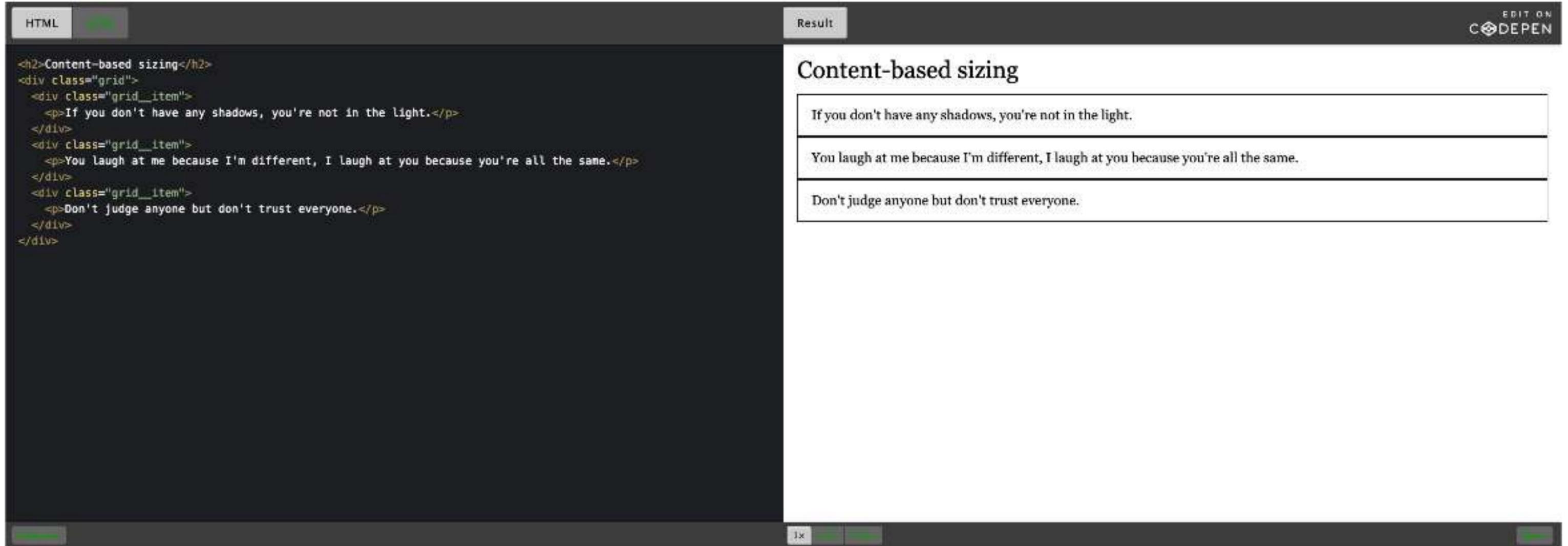
Item A

Item B

Item C

<https://codepen.io/huijing/pen/jegYob/>

Content-based sizing



The image shows a CodePen editor interface. On the left, the 'HTML' tab is active, displaying the following code:

```
<h2>Content-based sizing</h2>
<div class="grid">
  <div class="grid_item">
    <p>If you don't have any shadows, you're not in the light.</p>
  </div>
  <div class="grid_item">
    <p>You laugh at me because I'm different, I laugh at you because you're all the same.</p>
  </div>
  <div class="grid_item">
    <p>Don't judge anyone but don't trust everyone.</p>
  </div>
</div>
```

On the right, the 'Result' tab shows the rendered output:

Content-based sizing

If you don't have any shadows, you're not in the light.

You laugh at me because I'm different, I laugh at you because you're all the same.

Don't judge anyone but don't trust everyone.

The rendered output consists of three paragraphs, each contained within a rectangular box with a thin black border. The boxes are stacked vertically and their widths are determined by the length of the text they contain, demonstrating content-based sizing.

<https://codepen.io/huijing/pen/bmXLpd/>

Image gallery (part 2)

HTML Result EDIT ON CODEPEN

```
<ul>
<li></li>

```



1x

<https://codepen.io/huijing/pen/dQPZmL/>

Line-based placement

SCSSEDIT ON CODEPEN

```
.grid {
  display: grid;
  grid-template-columns: repeat(3, calc(100% / 3));
  grid-template-rows: repeat(3, calc(100% / 3));
}

.grid__item {
}

img {
  max-width: 100%;
  max-height: 100%;
}

// General styles
html {
  height: 100%;
}

body {
  padding: 1em;
  height: 100%;
}

hr {
  margin: 2em 0;
}
```

Result

Line-based placement



1x

<https://codepen.io/huijing/pen/pxMaYp/>

grid-column and grid-row

SCSS Result EDIT ON CODEPEN

```
.grid {  
  display: grid;  
  grid-template-columns: repeat(6, calc(100% / 6));  
  grid-template-rows: repeat(6, calc(90vh / 6));  
}  
  
.grid__item:nth-child(1) {  
}  
  
.grid__item:nth-child(2) {  
}  
  
.grid__item:nth-child(3) {  
}  
  
.grid__item:nth-child(4) {  
}  
  
.grid__item:nth-child(5) {  
}  
  
.grid__item:nth-child(6) {  
}
```

grid-column and grid-row

					
---	---	---	---	---	---

1x

<https://codepen.io/huijing/pen/zmgRgM/>

The span keyword

SCSSEDIT ON CODEPEN

```
.grid {
  display: grid;
  grid-template-columns: repeat(5, 20%);
  grid-template-rows: repeat(5, 20vh);
}

.grid_item {
}

img {
  max-width: 100%;
  max-height: 100%;
}

// General styles
html {
  height: 100%;
}

body {
  padding: 1em;
  height: 100%;
}

hr {
  margin: 2em 0;
}
```

Result

The span keyword

A screenshot of a CodePen editor showing SCSS code and its rendered result. The code defines a grid with 5 columns and 5 rows. The first cell of the grid contains a kitten image, while the other cells are empty. The kitten image is stretched to fit the grid cell.

1xEDIT ON CODEPEN

<https://codepen.io/huijing/pen/NOQYMM/>

Using the grid-area shorthand

SCSSEDIT ON CODEPEN

```
.grid {
  display: grid;
  grid-template-columns: repeat(5, 20%);
  grid-template-rows: repeat(3, 30vh);
}

.grid_item {
}

img {
  max-width: 100%;
  max-height: 100%;
}

// General styles
html {
  height: 100%;
}

body {
  padding: 1em;
  height: 100%;
}

hr {
  margin: 2em 0;
}
```

Result

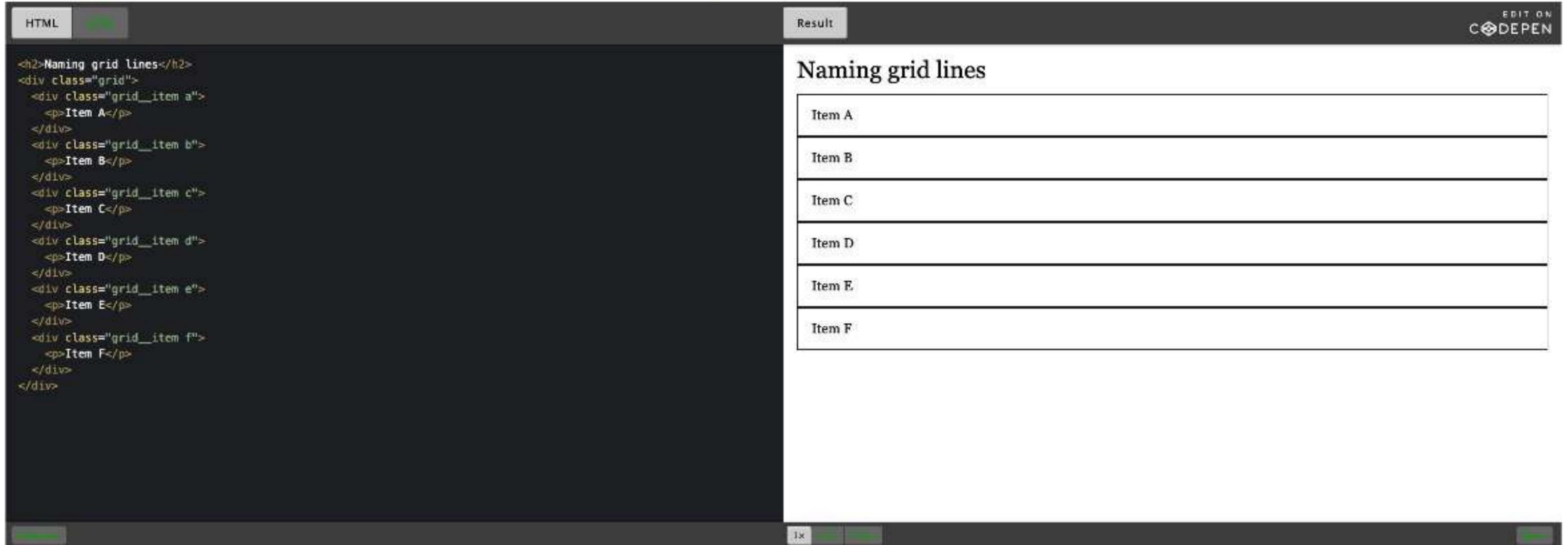
Using the grid-area shorthand



1x

<https://codepen.io/huijing/pen/vVoRzx/>

Naming grid lines



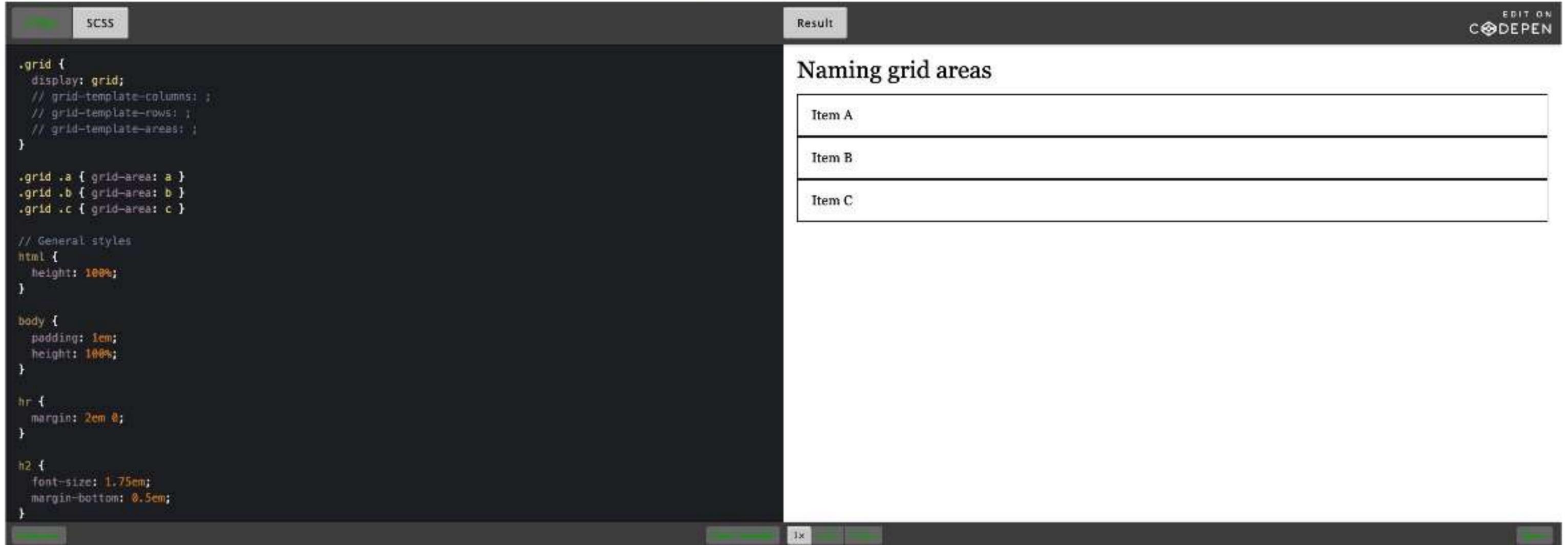
The screenshot shows a CodePen editor with two tabs: 'HTML' and 'Result'. The 'HTML' tab is active, displaying the following code:

```
<h2>Naming grid lines</h2>
<div class="grid">
  <div class="grid__item a">
    <p>Item A</p>
  </div>
  <div class="grid__item b">
    <p>Item B</p>
  </div>
  <div class="grid__item c">
    <p>Item C</p>
  </div>
  <div class="grid__item d">
    <p>Item D</p>
  </div>
  <div class="grid__item e">
    <p>Item E</p>
  </div>
  <div class="grid__item f">
    <p>Item F</p>
  </div>
</div>
```

The 'Result' tab shows the rendered output, which is a vertical stack of six horizontal lines, each containing text from top to bottom: Item A, Item B, Item C, Item D, Item E, and Item F. The CodePen logo and 'EDIT ON CODEPEN' text are visible in the top right corner of the editor interface.

<https://codepen.io/huijing/pen/xyvWBZ/>

Naming grid areas



The screenshot shows a CodePen editor with two panels. The left panel, labeled 'SCSS', contains the following code:

```
.grid {  
  display: grid;  
  // grid-template-columns: ;  
  // grid-template-rows: ;  
  // grid-template-areas: ;  
}  
  
.grid .a { grid-area: a }  
.grid .b { grid-area: b }  
.grid .c { grid-area: c }  
  
// General styles  
html {  
  height: 100%;  
}  
  
body {  
  padding: 1em;  
  height: 100%;  
}  
  
hr {  
  margin: 2em 0;  
}  
  
h2 {  
  font-size: 1.75em;  
  margin-bottom: 0.5em;  
}
```

The right panel, labeled 'Result', shows the rendered output. It features a heading 'Naming grid areas' followed by three horizontal lines, each containing the text 'Item A', 'Item B', and 'Item C' respectively. The CodePen logo and 'EDIT ON CODEPEN' text are visible in the top right corner of the editor interface.

<https://codepen.io/huijing/pen/EdqLxP/>

Responsive dashboard

Some dashboard

Some statistics

- 🔥 rating: 65
- 👤 rating: 100
- 🏆 rating: 75

I dunno...use your imagination

Okay, fine...this is supposed to be a game board so let's centre the content of the board.
This is sort of a trick question, actually...😏
Also, let's vertically centralise the title, score and statistics. Then centre the controls both ways.

Up Up Down Down Left Right Left Right B A 0:00 / 0:06

Do we have enough time?

HTMLEDIT ON CODEPEN

```
<div class="title">
  <h1>Some dashboard</h1>
</div>

<div class="score">
  <p>Score: 100</p>
</div>

<div class="stats">
  <h2>Some statistics</h2>
  <p>👤 rating: 65</p>
  <p>👤 rating: 100</p>
  <p>👤 rating: 75</p>
</div>

<div class="board">
  <h2 class="board_title">I dunno...use your imagination</h2>
  <div class="board_text">
    <p>Okay, fine...this is supposed to be a game board so let's centre the content of the board.</p>
    <p>This is sort of a trick question, actually...😏</p>
    <p>Also, let's vertically centralise the title, score and statistics. Then centre the controls both
ways.</p>
  </div>
</div>

<div class="controls">
  <p><span>Up</span> <span>Up</span> <span>Down</span> <span>Down</span> <span>Left</span> <span>Right</span> <span>Left</span> <span>Right</span> B A</p>
</div>
```

Some dashboard

Score: 100

Some statistics

👤 rating: 65
👤 rating: 100
👤 rating: 75

I dunno...use your imagination

Okay, fine...this is supposed to be a game board so let's centre the content of the board.
This is sort of a trick question, actually...😏
Also, let's vertically centralise the title, score and statistics. Then centre the controls both ways.

Up Up Down Down Left Right Left Right B A

1x

<https://codepen.io/huijing/pen/xQbXKO/>

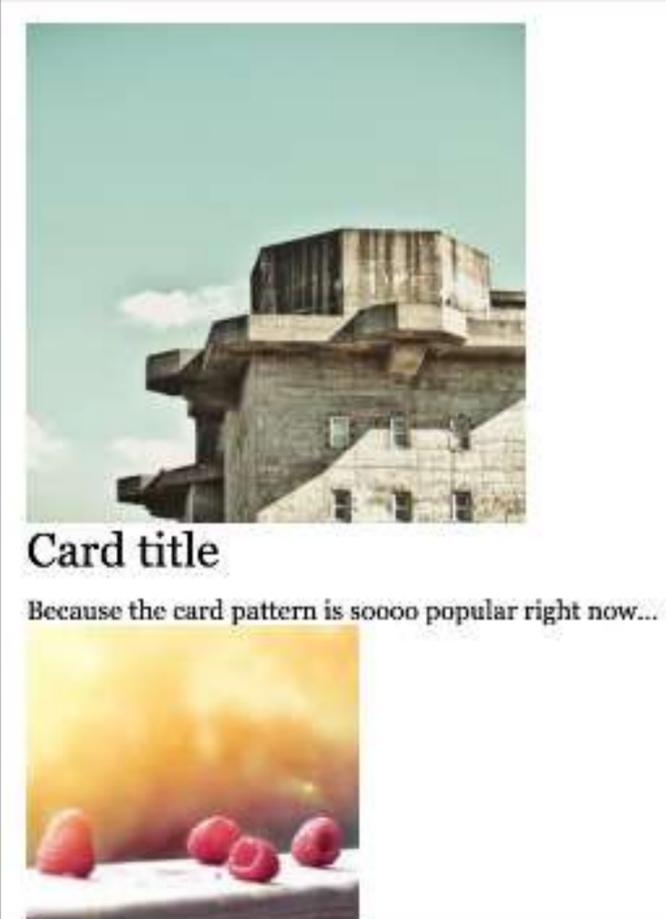
Feature queries

```
.selector {  
  /* Styles that are supported in old browsers */  
}  
  
@supports (property:value) {  
  .selector {  
    /* Styles for browsers that support the specified property */  
  }  
}
```

Card-based layout

HTML Result EDIT ON CODEPEN

```
<main class="cards">
  <div class="card">
    
    <h2>Card title</h2>
    <p>Because the card pattern is soooo popular right now...</p>
  </div>
  <div class="card">
    
    <h2>Card title</h2>
    <p>Because the card pattern is soooo popular right now...</p>
  </div>
  <div class="card">
    
    <h2>Card title</h2>
    <p>Because the card pattern is soooo popular right now...</p>
  </div>
  <div class="card">
    
    <h2>Card title</h2>
    <p>Because the card pattern is soooo popular right now...</p>
  </div>
  <div class="card">
    
    <h2>Card title</h2>
    <p>Because the card pattern is soooo popular right now...</p>
  </div>
  <div class="card">
    
    <h2>Card title</h2>
  </div>
</main>
```



Card title

Because the card pattern is soooo popular right now...

1x

<https://codepen.io/huijing/pen/VVPjBL/>

Useful references

- [What Happens When You Create A Flexbox Flex Container?](#)
- [Everything You Need To Know About Alignment In Flexbox](#)
- [Use Cases For Flexbox](#)
- [CSS Grid Layout Module Level 1](#)
- [Codrops CSS Grid reference](#)
- [Grid by Example](#)
- [Learn CSS Grid](#)
- [Grid Auto-Placement Is Ready](#)
- [Automatizing the Grid](#)
- [Deep Dive into Grid Layout Placement](#)
- [CSS Grid Layout and positioned items](#)
- [CSS Logical Properties and Values in Chromium and WebKit](#)
- [Changes on CSS Grid Layout in percentages and indefinite height](#)
- [The Story of CSS Grid, from Its Creators](#)
- [CSS Grid Layout is Here to Stay](#)
- [The New Layout Standard For The Web: CSS Grid, Flexbox And Box Alignment](#)
- [Grid “fallbacks” and overrides](#)
- [Cascading Web Design with Feature Queries](#)
- [Basic grid layout with fallbacks using feature queries](#)



Thank you!



<https://www.chenhuijing.com>



@hj_chen



@hj_chen



@huijing

Font used is [ABeeZee](#), by Anja Meiners