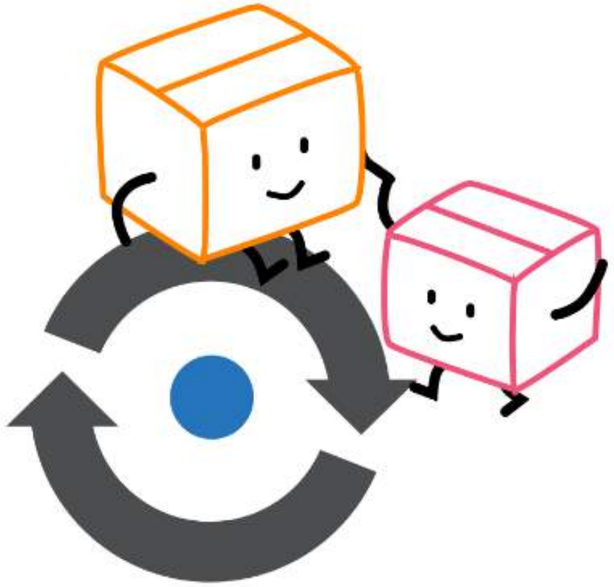
# Frontend Crash Course: CSS

Chen Hui Jing / @hj_chen



# Cascading

# Style

# Sheets

# Structure of a CSS rule

```
selector {
  property1: value;
  property2: value;
  property3: value;
}
```

- The **selector** identifies which HTML elements the rule will be applied to
- The **curly braces** contain the property-value pairs, separated with semi-colons
- The **properties** define the style of the selected element
- The **values** are dependent on the property, and indicate the value of the properties to be set

# CSS Selectors

- Type selector
- Attribute selectors
- Pseudo-classes
- Pseudo-elements
- Class selectors
- ID selectors
- Combinators

# Type selector

Matches all the **elements** of that type on the page

```
main {
  /* some properties */
}
```

```
div {
  /* some properties */
}
```

```
p {
  /* some properties */
}
```

# Attribute selectors

## <attr>

Targets elements that contain the <attr> attribute, regardless of its value.

```
<div data-colour="green"></div>
<div data-colour="blue"></div>
<div data-colour="yellow"></div>
```

```
[data-colour] {
/* some properties */
}
```

## <attr>=<val>

Targets elements where the <attr>'s value is exactly <val>.

```
<div data-colour="green"></div>
```

```
[data-colour="green"] {
/* some properties */
}
```

## <attr>~=<val>

Targets elements with the <attr> attribute whose value is a list of white space separated words, one of which must be <val>. <val> itself cannot contain white space, and neither can it be an empty string.

```
<div data-colour="green yellow blue"></
```

```
[data-colour~="green"] {
  /* some properties */
}
```

## <attr>|=<val>

Targets elements with the <attr> attribute whose value is exactly <val> or starts with <val> immediately followed by a “-”.
Primary use-case being for language subcode matching, like “en”, “en-US” and “en-UK”.

```
<div data-colour="green-table"></div>
<div data-colour="green-chair"></div>
<div data-colour="green-bottle"></div>
```

```
[data-colour|="green"] {
  /* some properties */
}
```

## <attr>^=<val>

A substring matching selector. Targets elements with the <attr> attribute whose value starts with <val>.
<val> cannot be an empty string.

```
<div data-colour="greenish-yellow"></di
<div data-colour="greengoblin"></div>
```

```
[data-colour^="green"] {
  /* some properties */
}
```

## <attr>$=<val>

A substring matching selector. Targets elements with the <attr> attribute whose value ends with <val>.
<val> cannot be an empty string.

```
<div data-colour="yellowish-green"></di
<div data-colour="seagreen"></div>
```

```
[data-colour$="green"] {
  /* some properties */
}
```

## <attr>*=<val>

A substring matching selector. Targets elements with the <attr> attribute whose value contains an instance of <val>.
<val> cannot be an empty string.

```
<div data-colour="goblingreenish"></div
```

```
[data-colour*="green"] {
  /* some properties */
}
```

# Pseudo-classes

## Structural pseudo-classes

- `:root`
- `:nth-child(n)`
- `:nth-last-child(n)`
- `:nth-of-type(n)`
- `:nth-last-of-type(n)`
- `:first-child`
- `:last-child`
- `:first-of-type`
- `:last-of-type`
- `:only-child`
- `:only-of-type`
- `:empty`

## Link pseudo-classes

- `:link`
- `:visited`

## User action pseudo-classes

- `:active`
- `:hover`
- `:focus`

## UI element pseudo-classes

- `:enabled`
- `:disabled`
- `:checked`

## Negation pseudo-class

- `:not(selector)`

## Language pseudo-class

- `:lang(zh)`

## Target pseudo-class

- `:target`

# Pseudo-elements

`::first-line`, `::first-letter`, `::before`, `::after`

D🐒 DO SOMETHING TO THE FIRST LETTER, OR FIRST line. Just the first though. Not the others. Leave them lines alone. 🦄

```
.pseudoelem p::first-line {
  text-transform: uppercase
}

.pseudoelem p::first-letter {
  font-size: 2em;
  color: seagreen;
}

.pseudoelem p::before {
  content: 'D🐒 '
}
```

# Class selector

Matches all the elements with the specified class attribute

```
<div class="example">
  // some random markup
</div>
```

```
.example {
  /* some properties */
}
```

# ID selector

Matches matches the element with the specified id attribute

```
<div id="example">
  // some random markup
</div>
```

```
#example {
  /* some properties */
}
```

# Combinators

## Descendant

Can be any level of descendant, multiple nesting

```
<div>
  <p>This is a <p> element in a <div> e
</div>
```

```
div span {
  /* some properties */
}
```

## Child

Immediate child element only

```
<div>
  <p>This is <p> element is a direct ch
</div>
```

```
div > p {
  /* some properties */
}
```

## Next sibling

Same parent, immediately following

```
<p>This is a <p> element.</p>
<ul>
  <li>This is a list item</li>
  <li>This is a list item</li>
</ul>
```

```
p + ul {
  /* some properties */
}
```

## Subsequent sibling

Same parent, can be anywhere following

```
<p>This is a <p> element.</p>
<p>This is another <p> element.</p>
<ul>
  <li>This is a list item</li>
  <li>This is a list item</li>
</ul>
```

```
p ~ ul {
  /* some properties */
}
```

# Who wins?

1. Importance
2. Specificity
3. Source order

# !important

Don't use it if you can help it.

This paragraph has a class of cookie and an id of butterscotch.

This paragraph has a class of cookie.

```
.important #butterscotch {
  background-color: crimson;
  color: whitesmoke;
  border: 0.25em inset teal;
}

.important .cookie {
  background-color:
mediumslateblue;
  color: lavender;
  border: none;
}
```

# CSS Specificity

| 0–∞ | 0–∞ | 0–∞ | 0–∞ |
| --- | --- | --- | --- |
| Inline styles | IDs | Classes, attributes and pseudo-classes | Elements and pseudo-elements |

```
ul {
// CSS properties
}
```

0, 0, 0, 1

```
.class-1 .class-2 p {
// CSS properties
}
```

0, 0, 2, 1

```
#id-1 .class-3 div {
// CSS properties
}
```

0, 1, 1, 1

# Source order

The tie-breaker for equal specificity values

This paragraph has a class of cookie and an id of butterscotch.

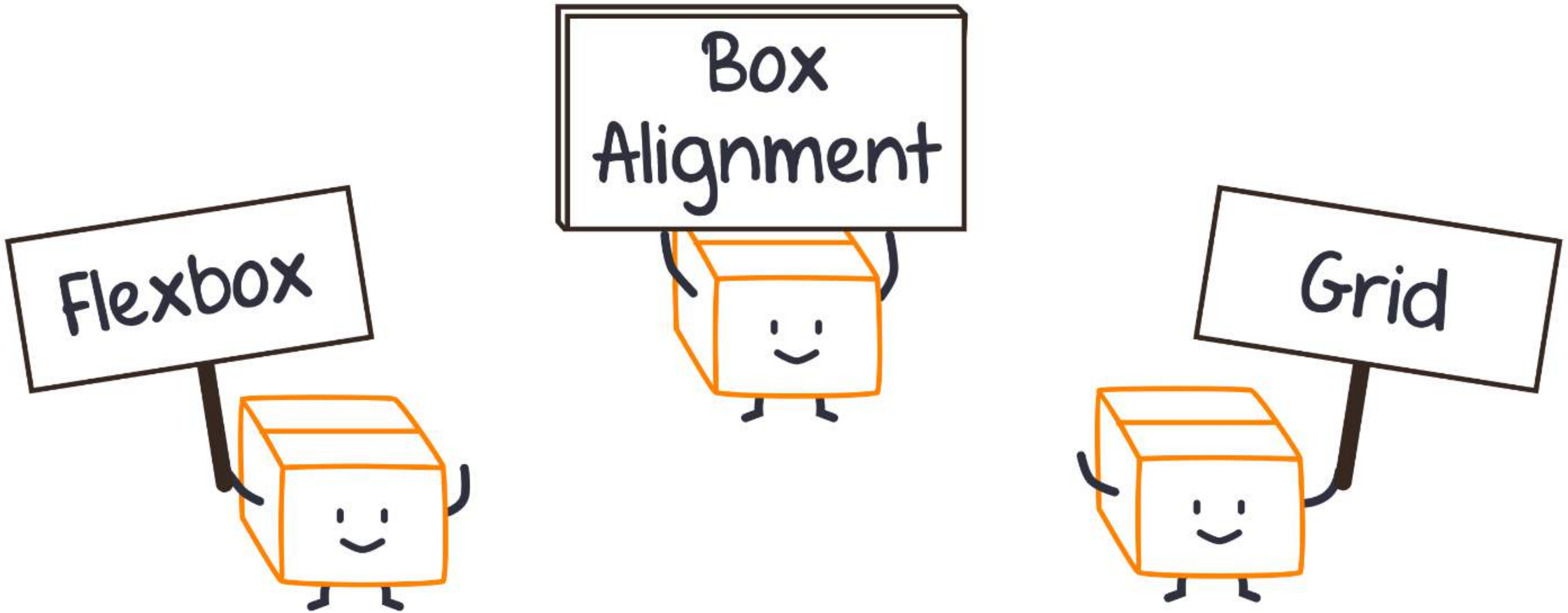
This paragraph has a class of cookie.

```
.sourceorder .cookie.cookie {
  background-color: crimson;
  color: whitesmoke;
  border: 0.25em inset teal;
}

.sourceorder .cookie:first-
child {
  background-color:
mediumslateblue;
  color: lavender;
  border: none;
```

# General guidelines for writing CSS

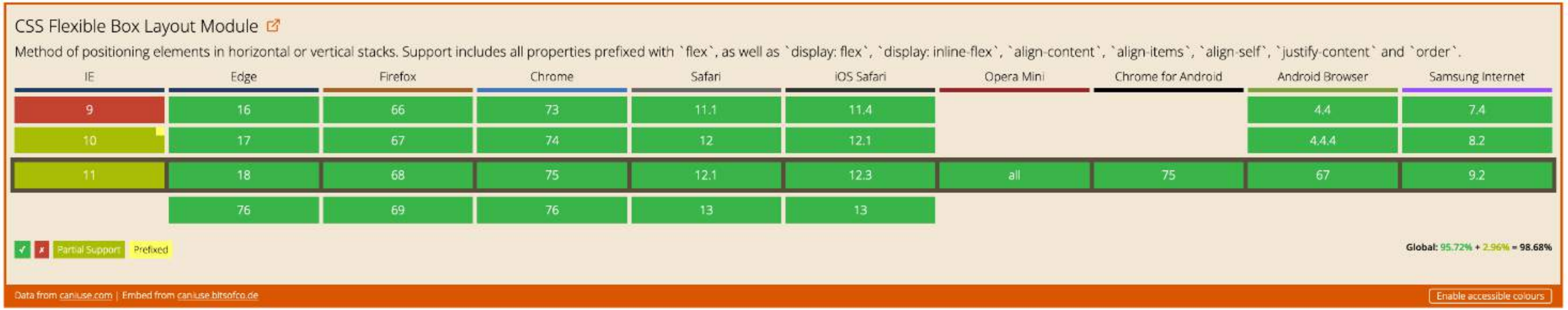
- Declare your styles from lowest specificity then move up
- Keep your specificity as low as possible
- Name your classes sensibly
- Never style IDs
- Don’t write inline styles

# Layout



# Flexbox support

98.68% of internet users are using a browser that supports Flexbox



CSS Flexible Box Layout Module

Method of positioning elements in horizontal or vertical stacks. Support includes all properties prefixed with `flex`, as well as `display: flex`, `display: inline-flex`, `align-content`, `align-items`, `align-self`, `justify-content` and `order`.

| IE | Edge | Firefox | Chrome | Safari | iOS Safari | Opera Mini | Chrome for Android | Android Browser | Samsung Internet |
|---|---|---|---|---|---|---|---|---|---|
| 9 | 16 | 66 | 73 | 11.1 | 11.4 | | | 4.4 | 7.4 |
| 10 | 17 | 67 | 74 | 12 | 12.1 | | | 4.4.4 | 8.2 |
| 11 | 18 | 68 | 75 | 12.1 | 12.3 | all | 75 | 67 | 9.2 |
| | 76 | 69 | 76 | 13 | 13 | | | | |

✓ ✗ Partial Support Prefixed

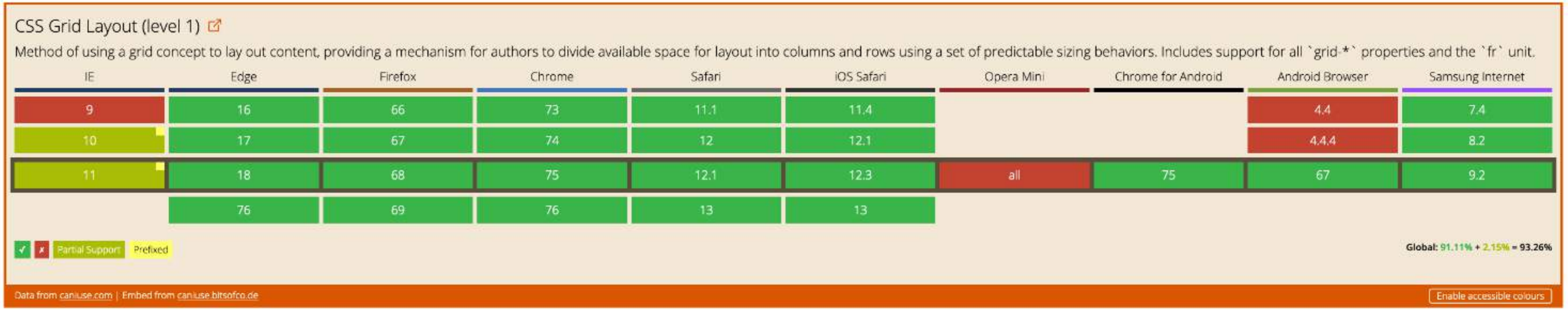
Global: 95.72% + 2.96% = 98.68%

Data from caniuse.com | Embed from caniuse.bitsofco.de

Enable accessible colours

# Grid support

93.26% of internet users are using a browser that supports Grid



CSS Grid Layout (level 1)

Method of using a grid concept to lay out content, providing a mechanism for authors to divide available space for layout into columns and rows using a set of predictable sizing behaviors. Includes support for all `grid-*` properties and the `fr` unit.

| IE | Edge | Firefox | Chrome | Safari | iOS Safari | Opera Mini | Chrome for Android | Android Browser | Samsung Internet |
|---|---|---|---|---|---|---|---|---|---|
| 9 | 16 | 66 | 73 | 11.1 | 11.4 | | | 4.4 | 7.4 |
| 10 | 17 | 67 | 74 | 12 | 12.1 | | | 4.4.4 | 8.2 |
| 11 | 18 | 68 | 75 | 12.1 | 12.3 | all | 75 | 67 | 9.2 |
| | 76 | 69 | 76 | 13 | 13 | | | | |

✓ ✗ Partial Support Prefixed

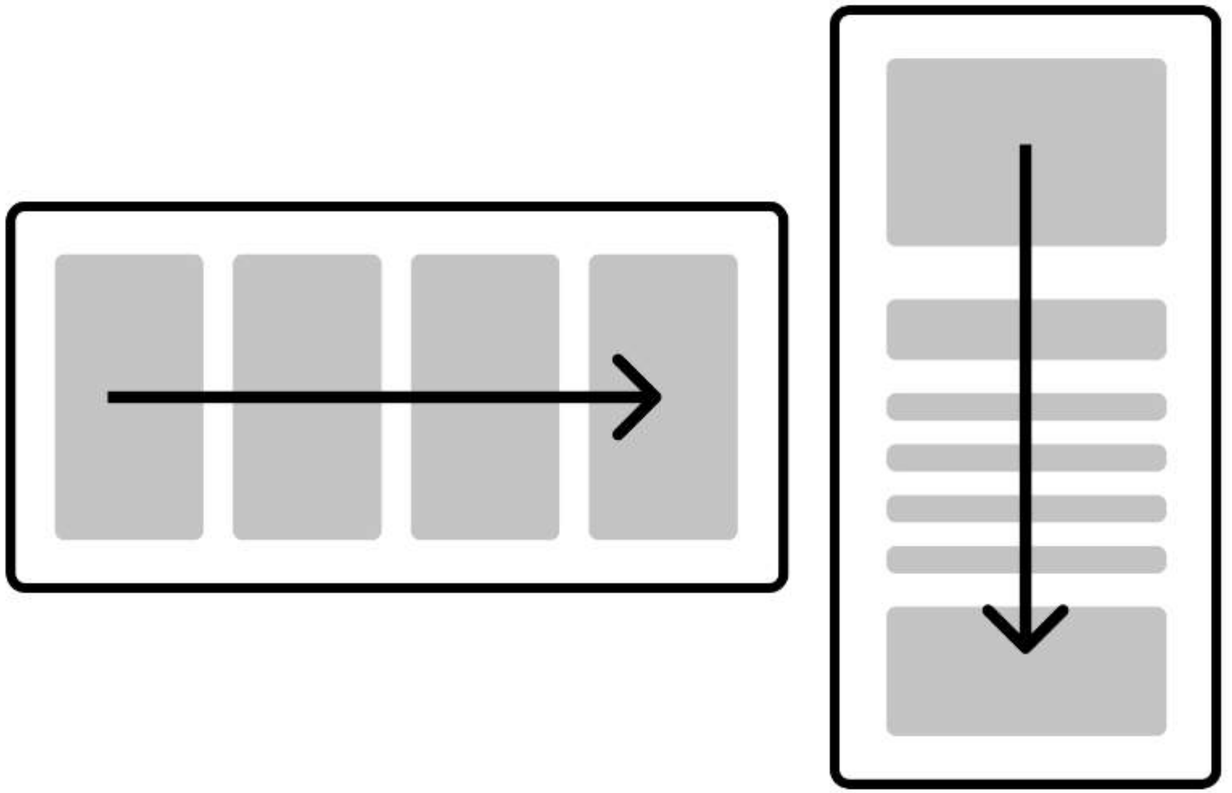
Global: 91.11% + 2.15% = 93.26%

Data from caniuse.com | Embed from caniuse.bitsofco.de

Enable accessible colours

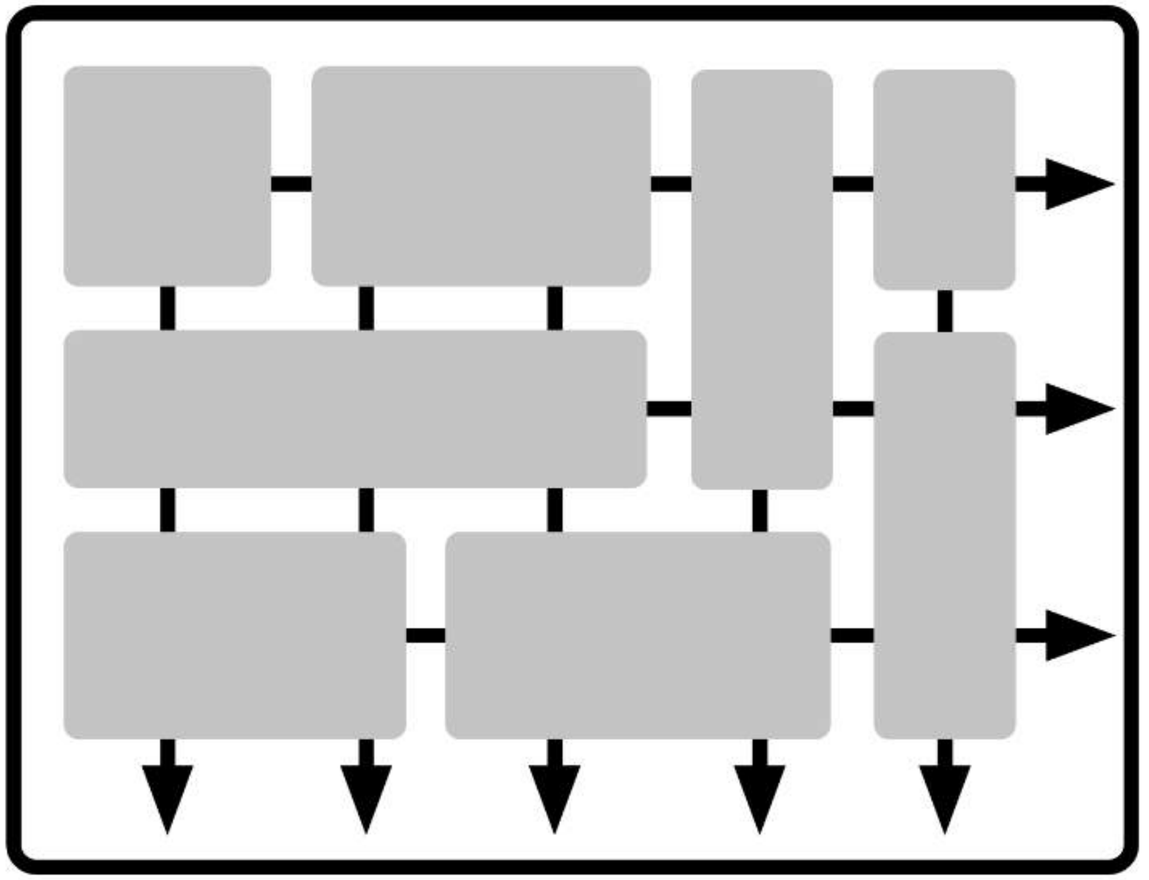
# Flexbox

For single dimension components

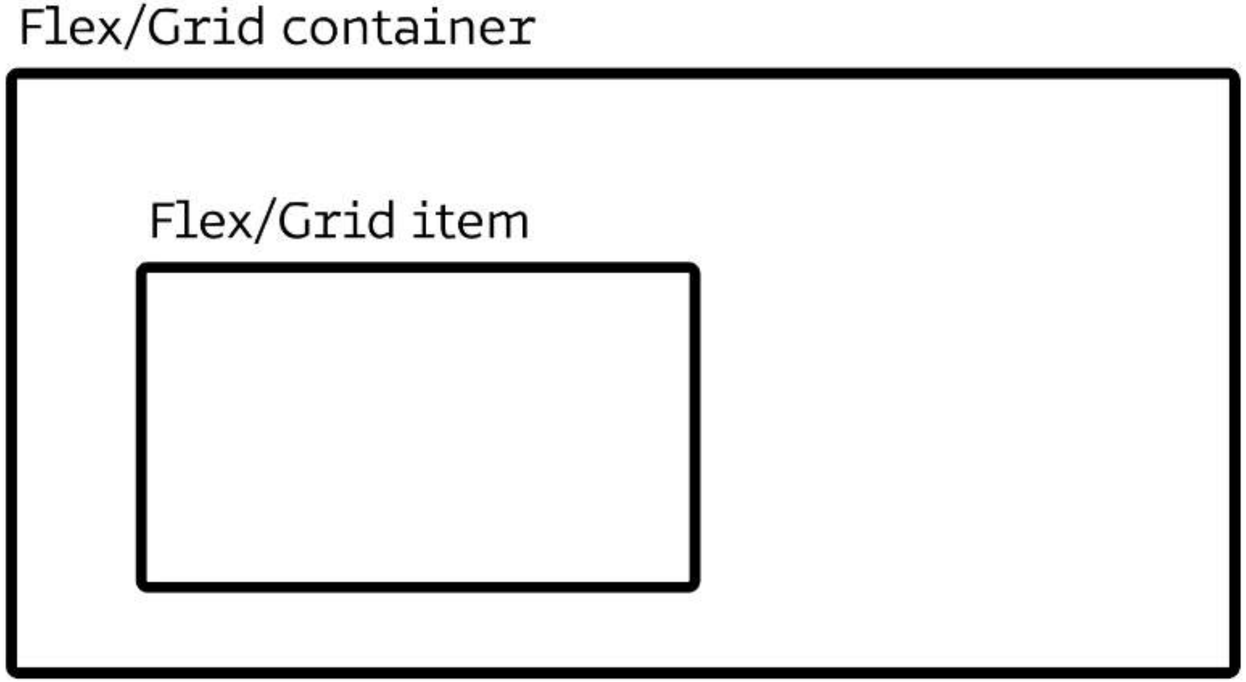


# Grid

For two-dimensional layouts



# Container-child relationship

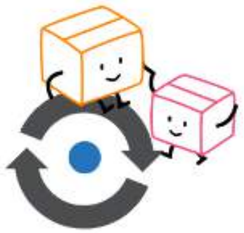


# Flex shorthand

| | |
|---|---|
| `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 |

# Flexbox use-cases

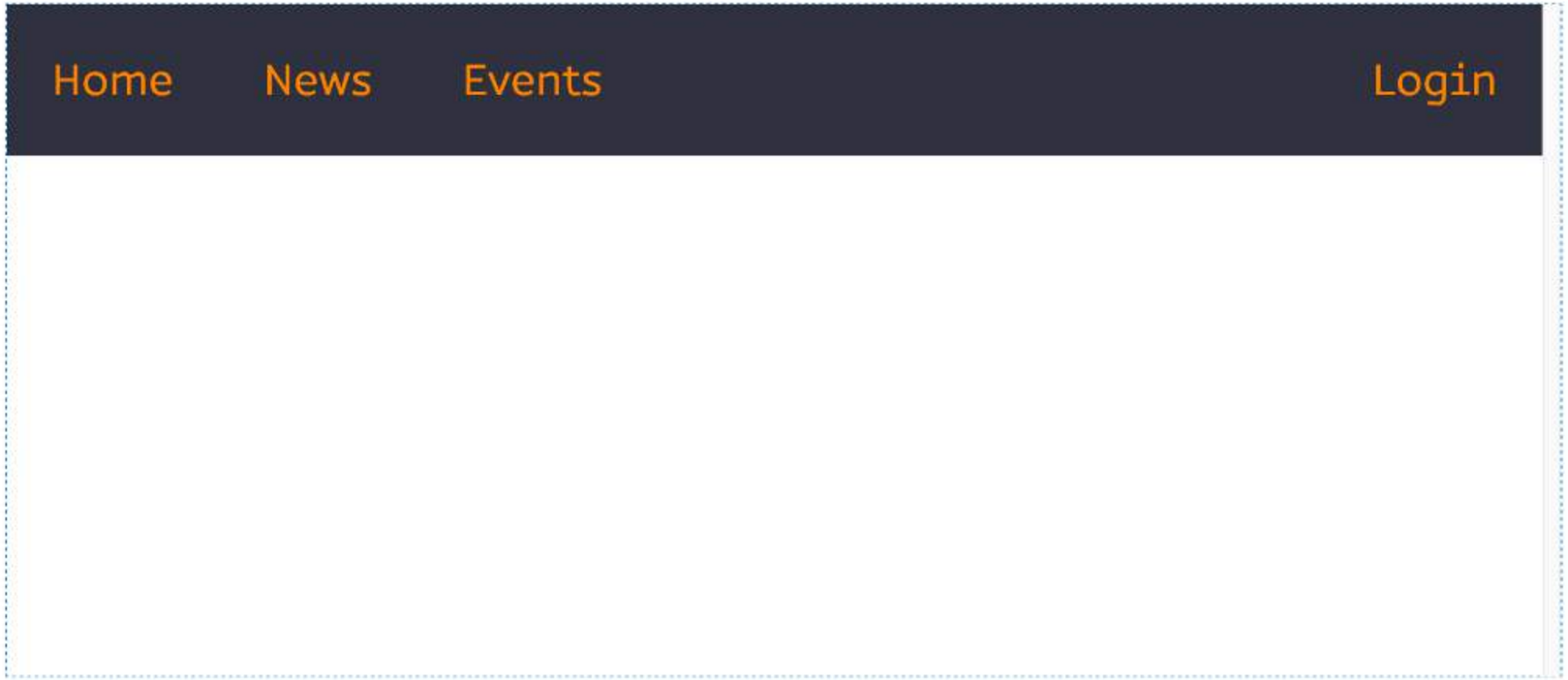
Centre stuff in their parent container



```
.automargin {
  display: flex;
}

.automargin div {
  margin: auto;
}
```

# Flexbox use-cases

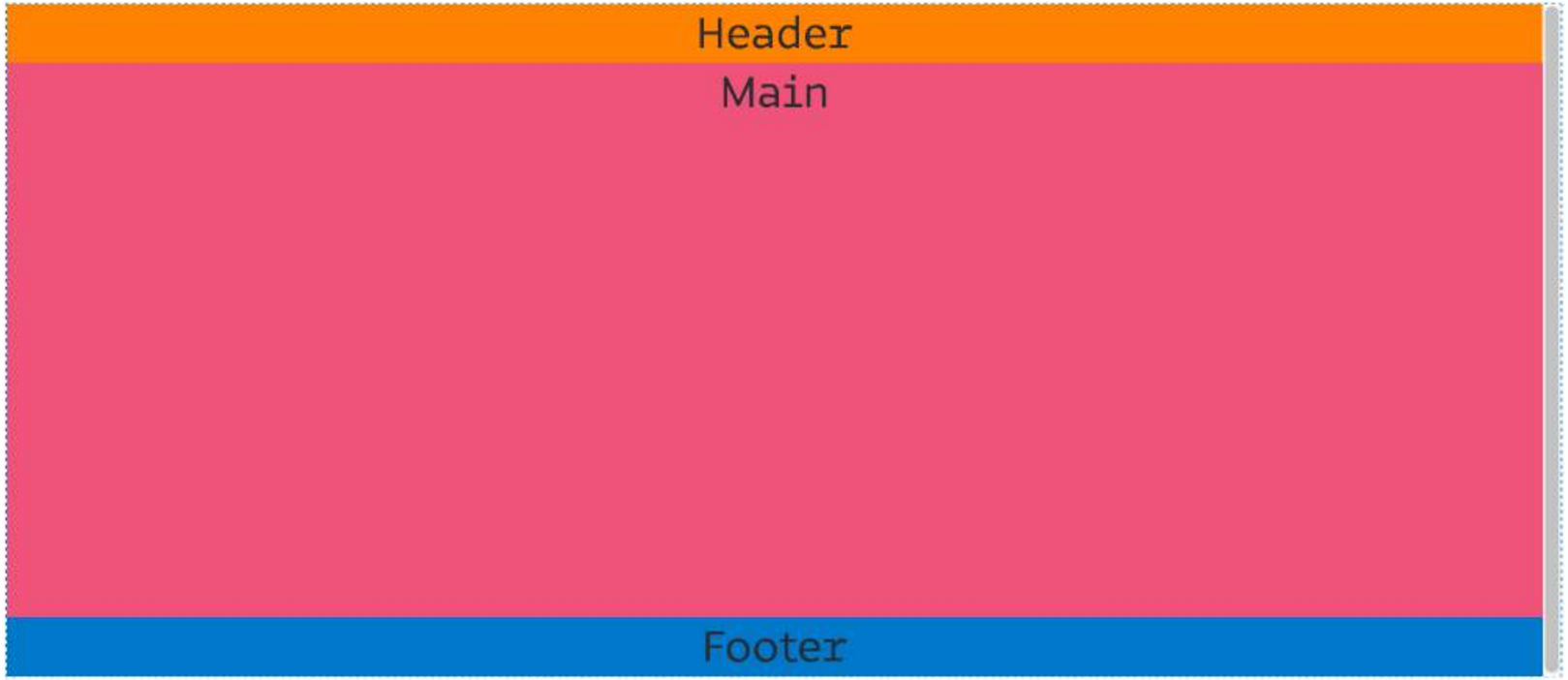
When you need 1 thing by itself on the opposite side




```
.navbar ul {
  display: flex;
}

.navbar li:last-child {
  margin-left: auto;
}
```

# Flexbox use-cases

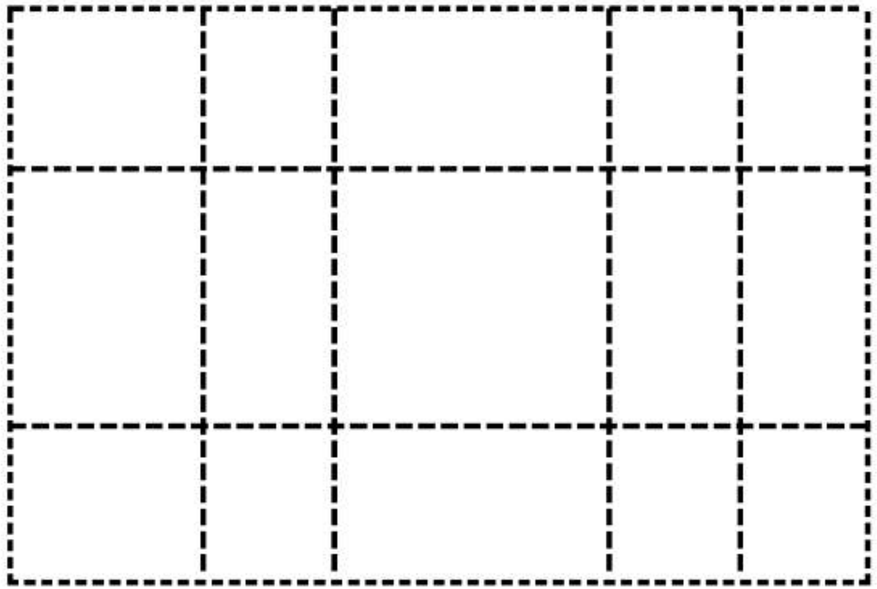
To stick the header and footer at the top and bottom



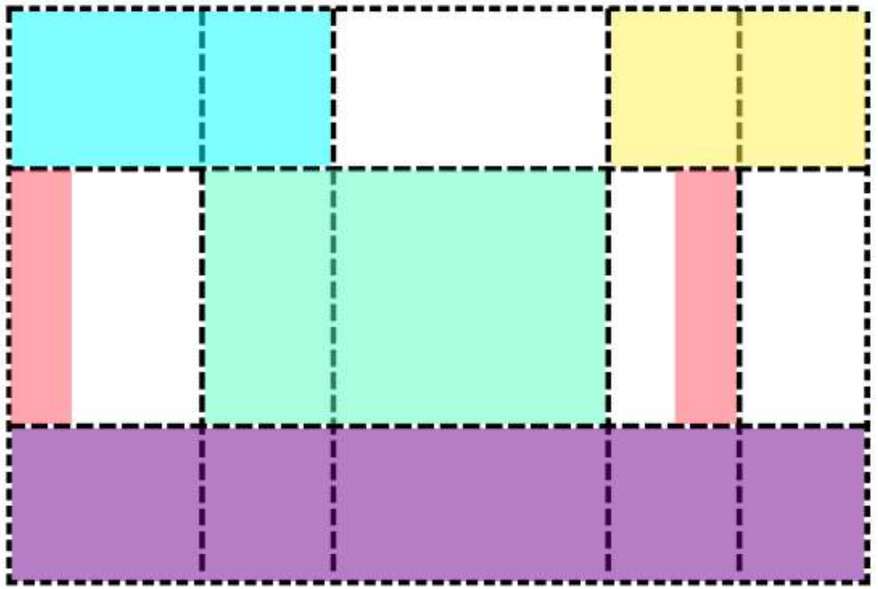

```
.flex1 {
  display: flex;
  flex-direction: column
}

.flex1 .main {
   flex: 1;
}
```

# CSS grid basics

Define your grid.

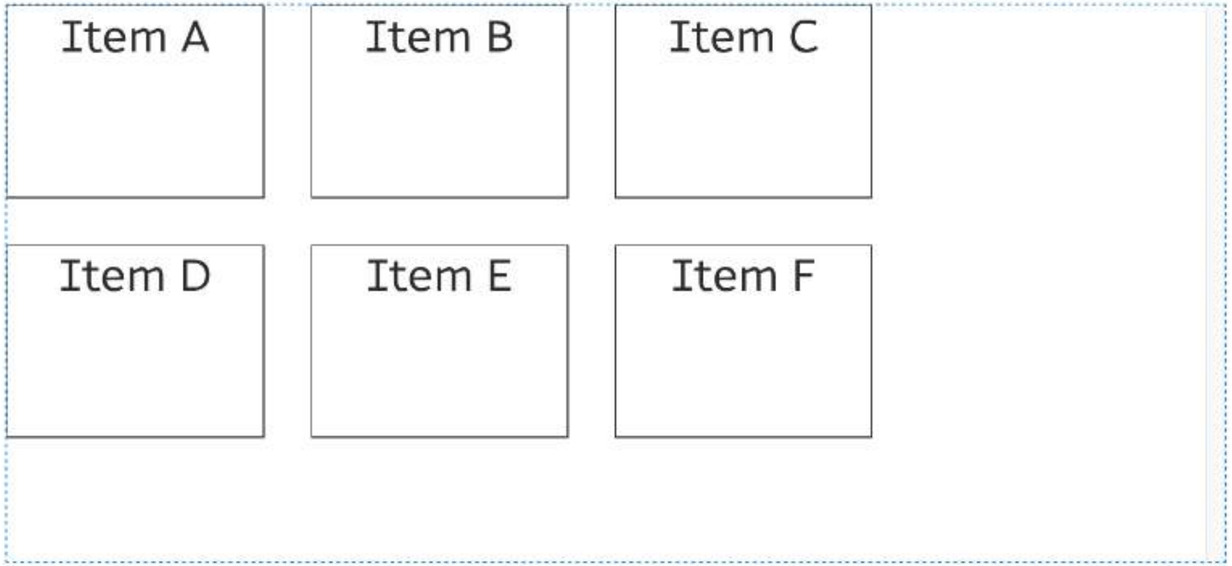


Place items in the grid.



# Defining a grid

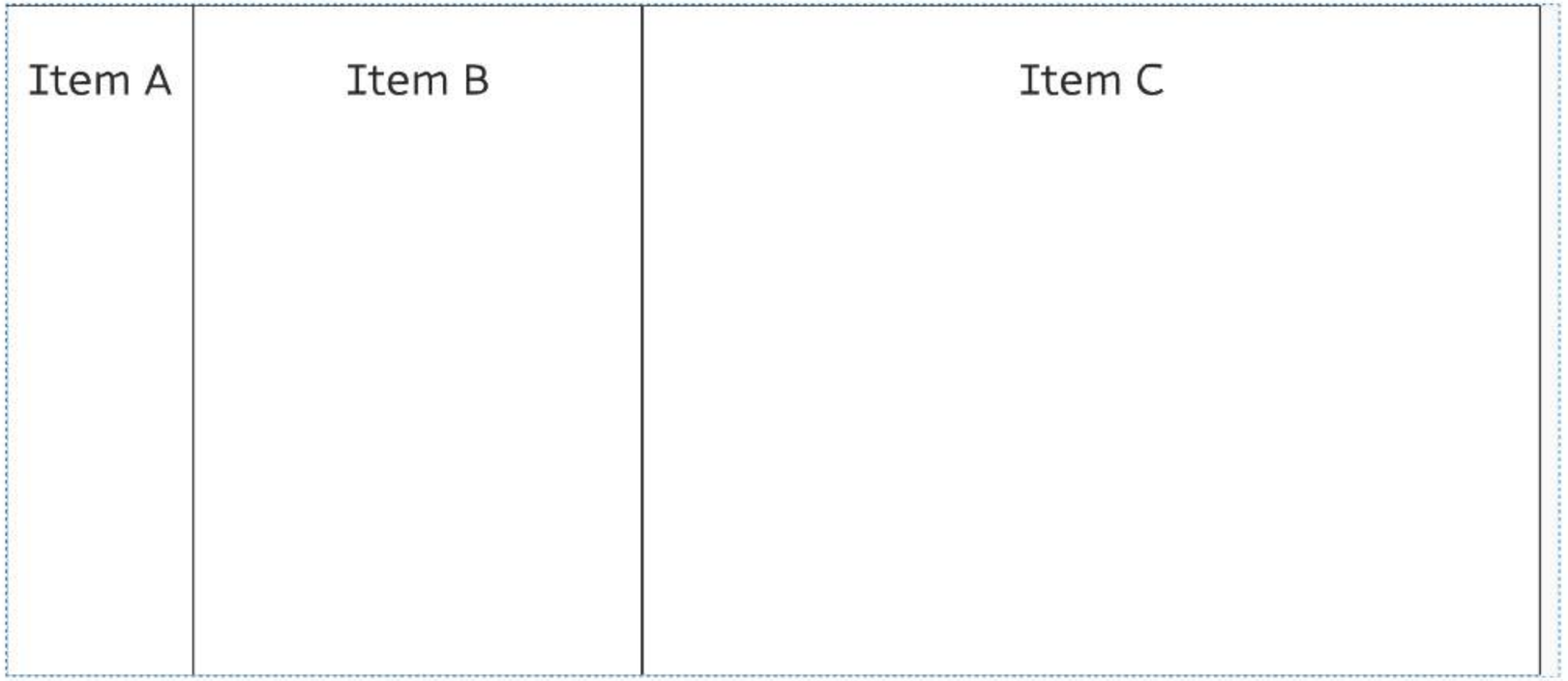
```
<div class="grid1">
  <div class="grid1__item">
    <p>Item A</p>
  </div>
  <div class="grid1__item">
    <p>Item B</p>
  </div>
  <div class="grid1__item">
    <p>Item C</p>
  </div>
  <div class="grid1__item">
    <p>Item D</p>
  </div>
```




```
.grid1 {
  display: grid;
  grid-template-columns: 200px 200px 200px;
  grid-template-rows: 150px 150px;
  gap: 1em;
```

# The `fr` unit

Represents a **fraction** of the **free space** in the grid container.



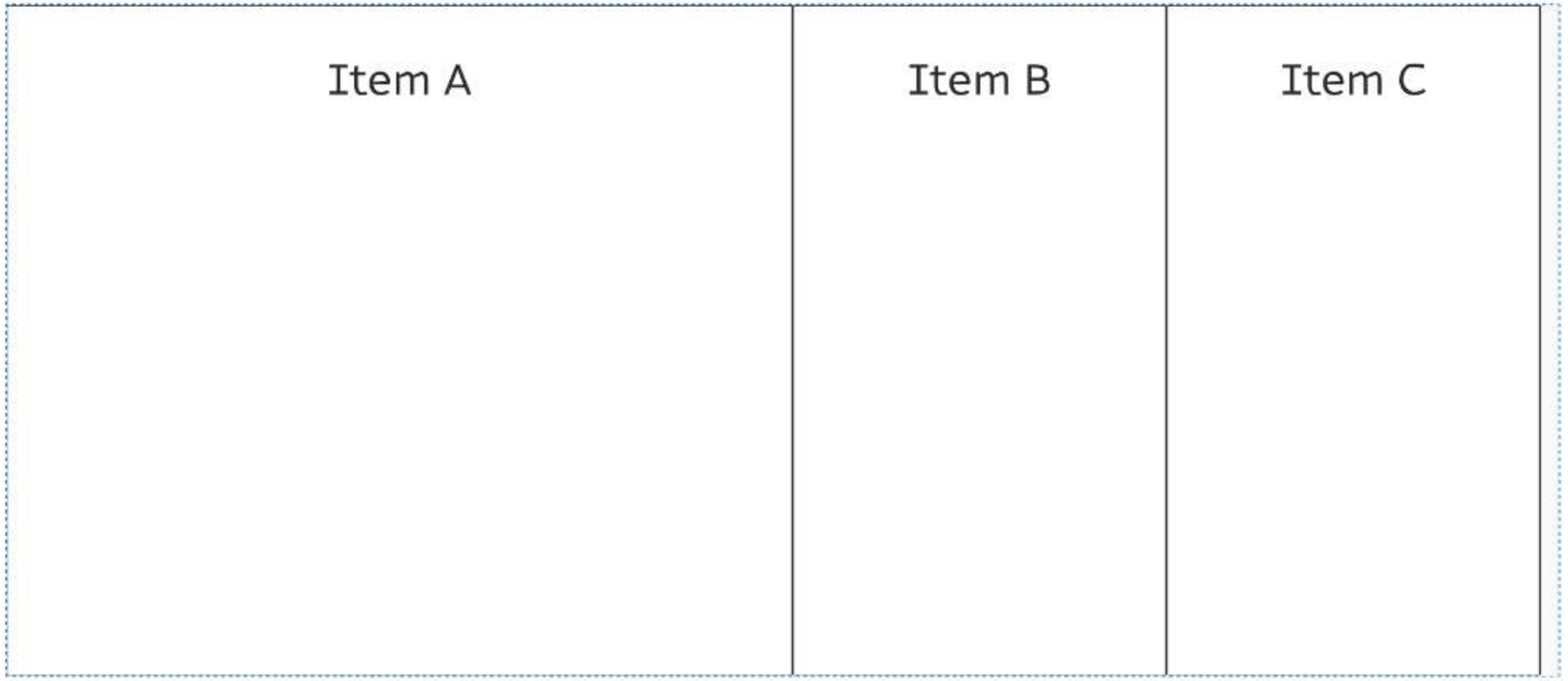

```
.grid4 {
  display: grid;
  grid-template-columns: 150px
1fr 2fr;
}
```

# Fluid CSS grid

```
.container {
  display: grid;
  grid-template-columns: repeat(3, 3fr 2fr);
}
```

# The `minmax()` function

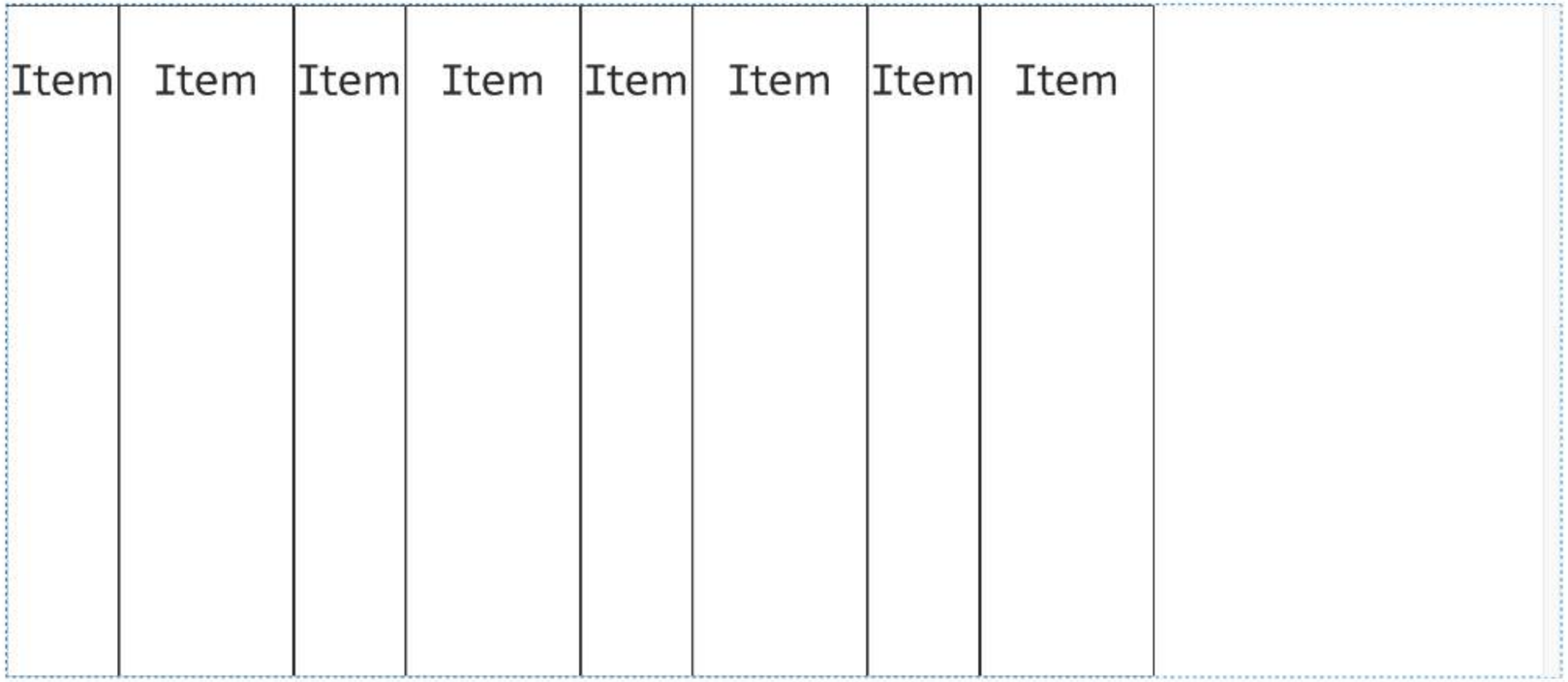
Defines a size range for columns or rows in the grid.




```
.grid5 {
  display: grid;
  grid-template-columns:
minmax(200px, 1fr) 300px
300px;
}
```

# The `repeat()` function

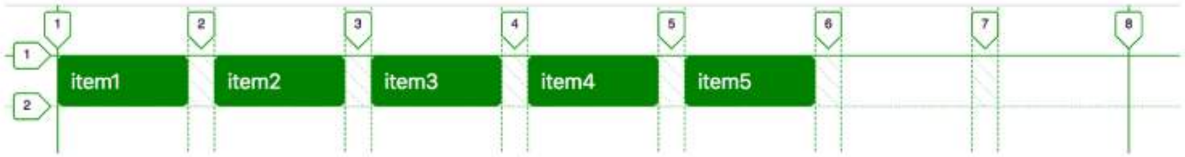
To specify a large number of columns or rows that follow a similar pattern




```
.grid6 {
  display: grid;
  grid-template-columns:
repeat(4, 90px 140px);
}
```

# auto-fill vs. auto-fit

Allow browser to determine how many tracks to create depending on track size.

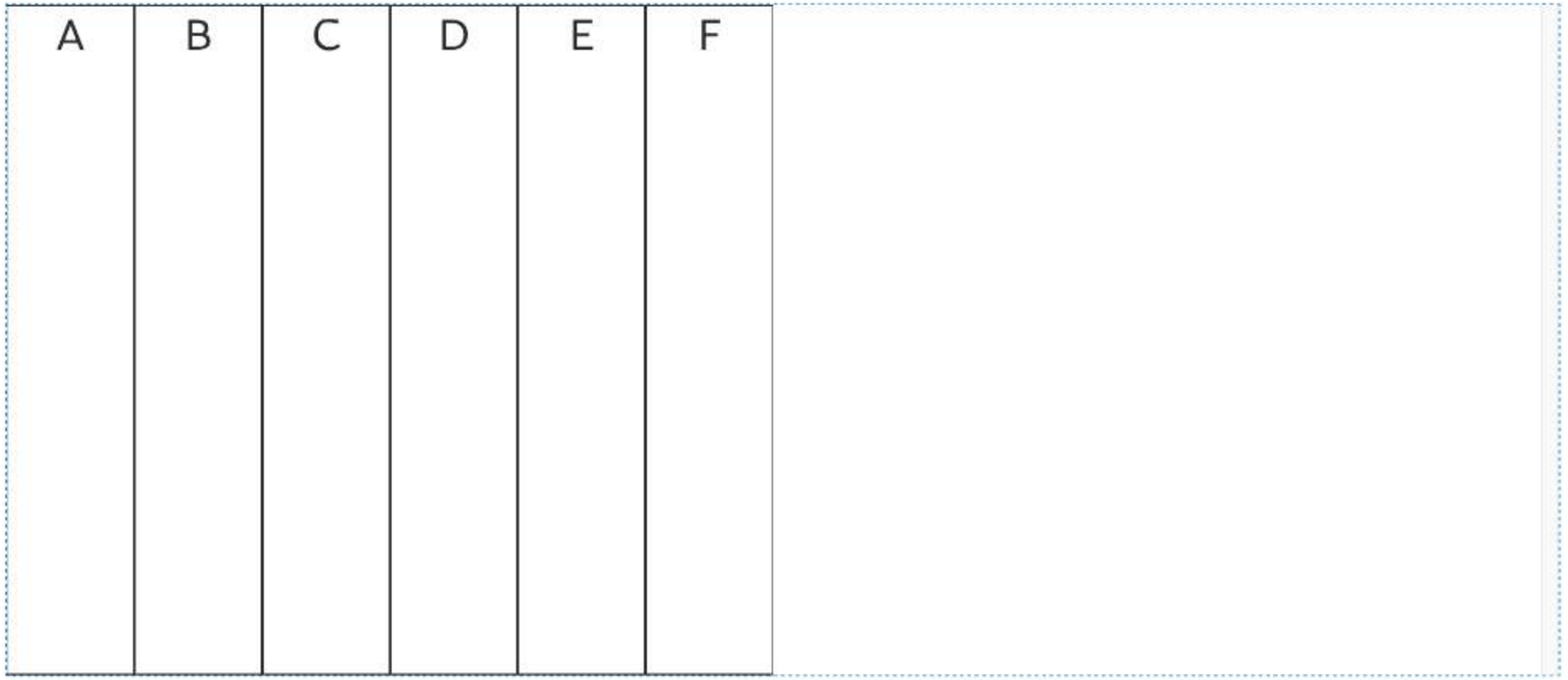


```
repeat(auto-fill, 100px);
```

```
repeat(auto-fit, 100px);
```

auto-fit collapses empty tracks.

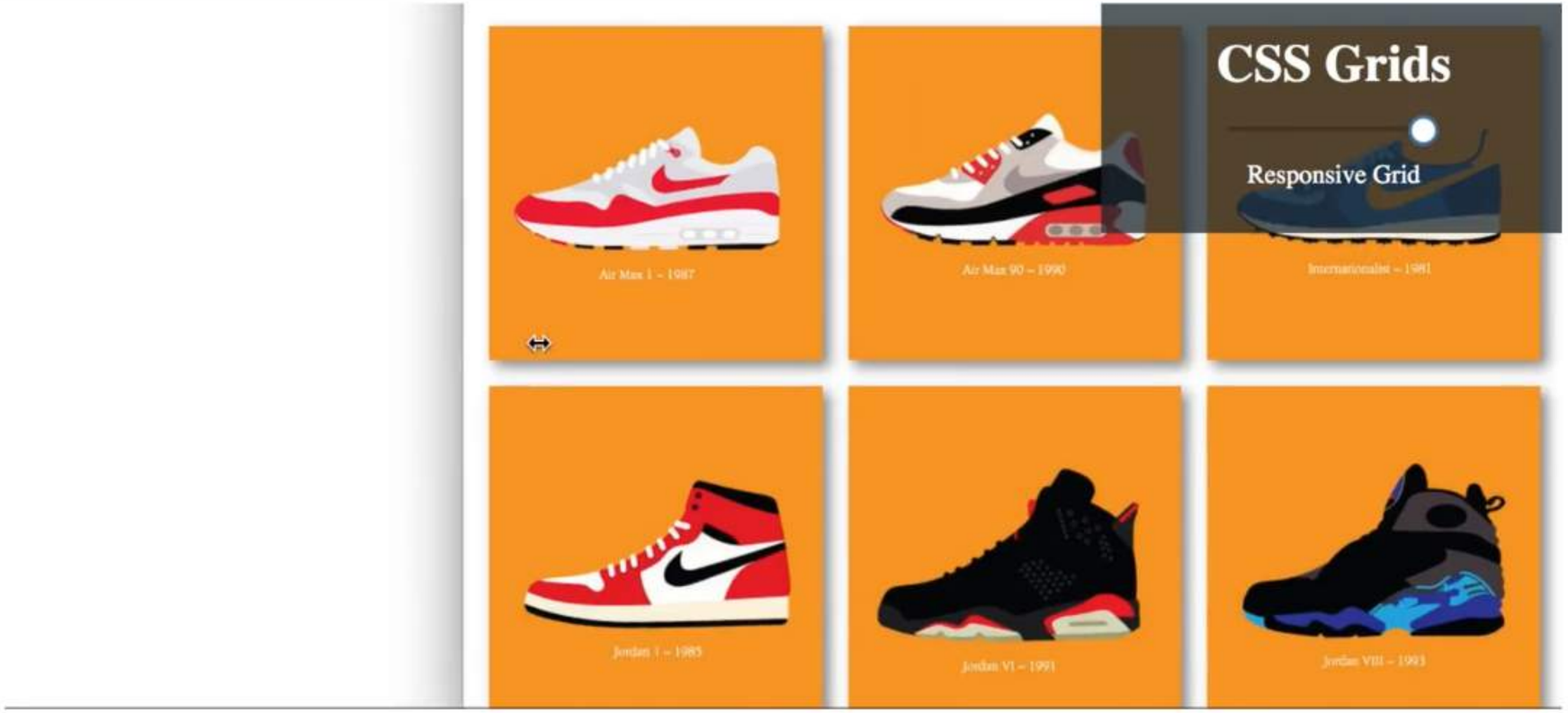
# auto-fill **versus** auto-fit



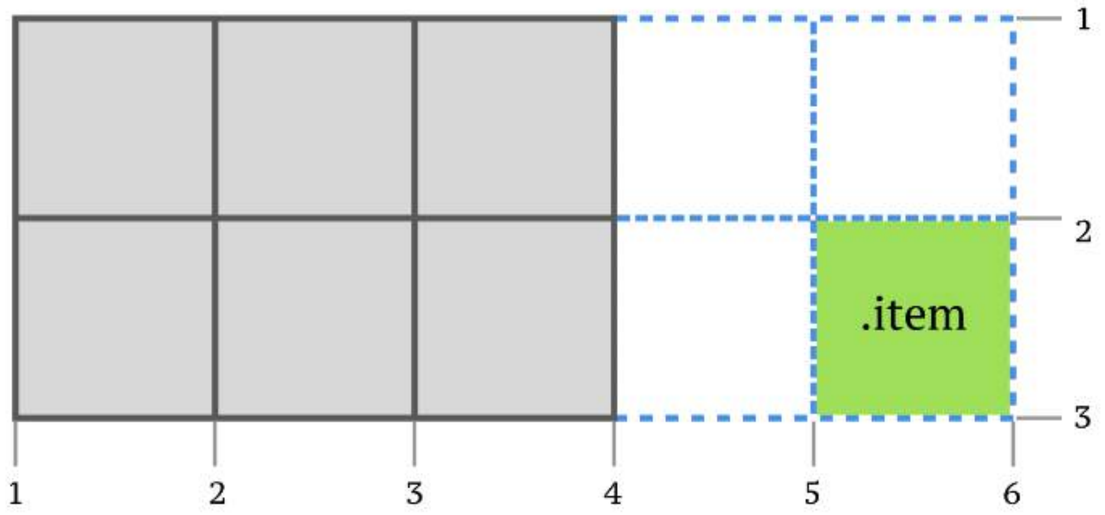

```
.keyword {
  display: grid;
  grid-template-columns:
repeat(auto-fill,
minmax(100px, 1fr));
}
```

# Responsive grid without media queries

```
.container {
  display: grid;
  grid-template-columns: repeat(auto-fill, minmax(10em, 1fr));
}
```



# Explicit Grid and Implicit Grid

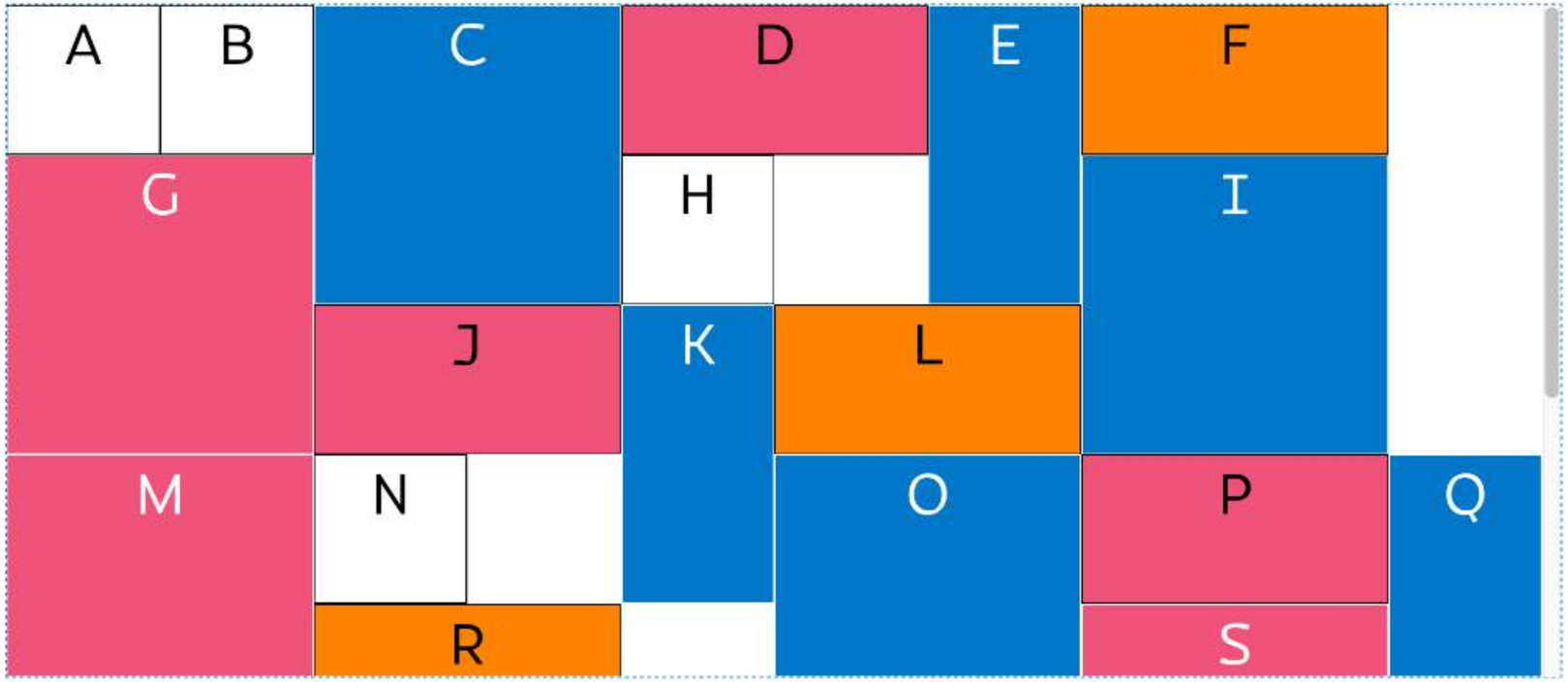



```
.grid-container {
  display: grid;
  grid-template-columns: 150px 150px 150px
  grid-template-rows: 150px 150px; /* two
}

.item {
  grid-column: 5 / 6; /* grid-column-start
  grid-row:  2 / 3; /* grid-row-start: 2;
}
```

The browser will create an implicit grid to hold items placed outside the grid.

# The `grid-auto-flow` property

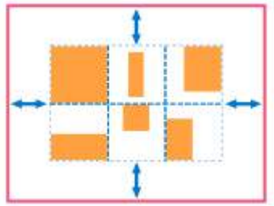
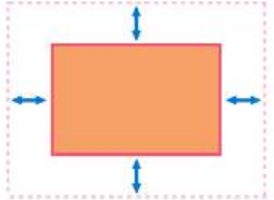
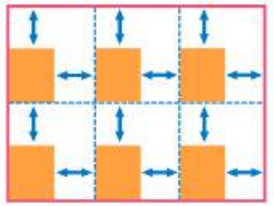
Adjusting the direction and density of grid items




```
.autoflow {
  display: grid;
  grid-template-columns:
repeat(auto-fit, minmax(120px,
1fr));
  grid-auto-rows: 120px;
  grid-auto-flow: row;
}

.autoflow__item:nth-child(3n)
{
  background-color: #ff8200;
```

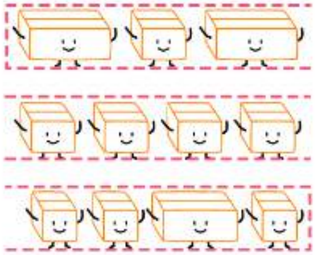
# Box Alignment

To create a cohesive and common box alignment model to share among all of CSS

| Property | Axis | Aligns | | Applies to |
|---|---|---|---|---|
| `justify-content` | main/inline | content within element (effectively adjusts padding) |  | block containers, flex containers and grid containers |
| `align-content` | cross/block | | | |
| `justify-self` | inline | element within parent (effectively adjusts margins) |  | block-level boxes, absolutely-positioned boxes and grid items |
| `align-self` | cross/block | | | absolutely-positioned boxes, flex items and grid items |
| `justify-items` | inline | items inside box (controls child items) |  | block containers and grid containers |
| `align-items` | cross/block | | | flex-containers and grid-containers |

Source: CSS Box Alignment Module Level 3

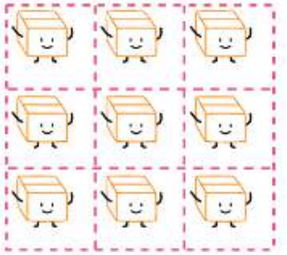
# Flexbox



- `align-content`
- `justify-content`
- `align-items`
- `align-self`

The `justify-items`/`justify-self` properties do **not** apply to flex items

# Grid



- `align-content`
- `justify-content`
- `align-items`
- `align-self`
- `justify-items`
- `justify-self`

# Flexbox



- align-content
- justify-content
- align-items
- align-self

The justify-items/justify-self properties do **not** apply to flex items

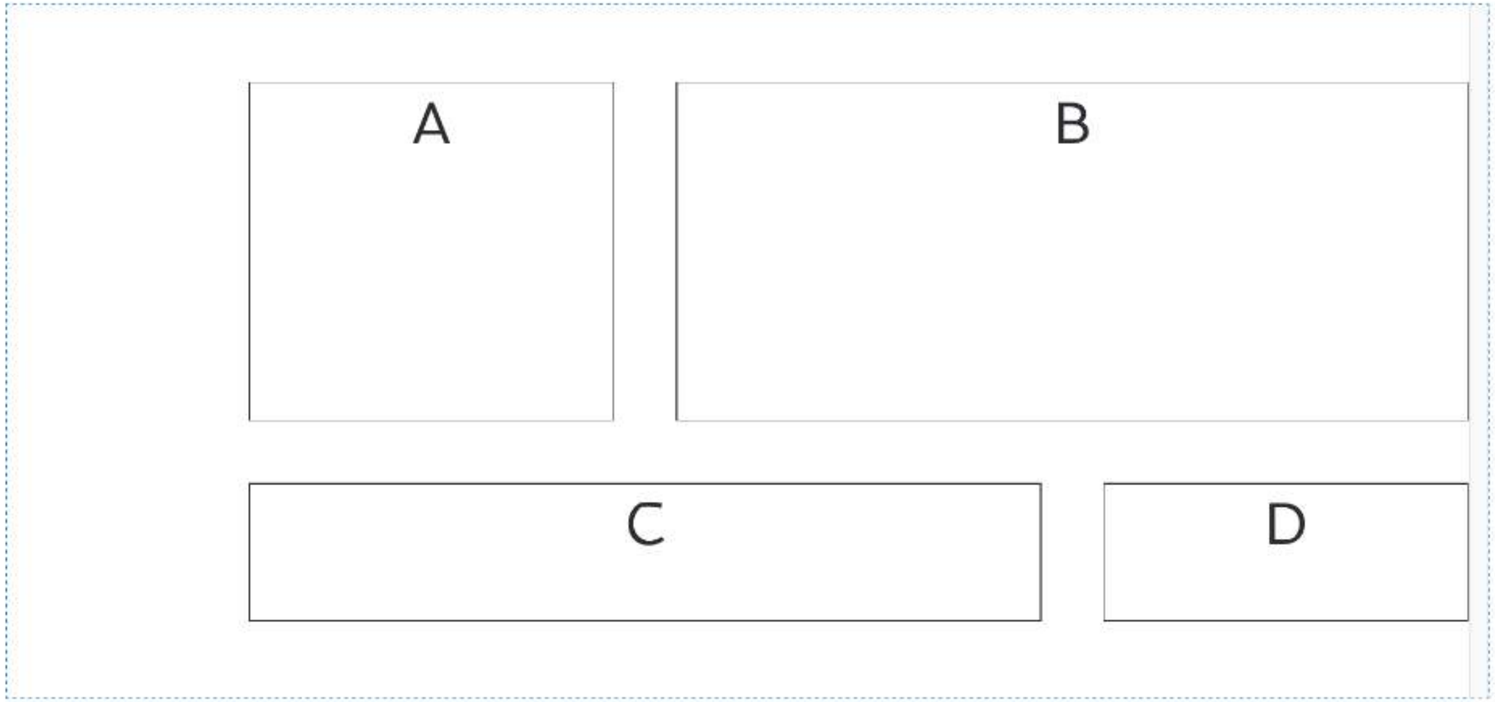
Justify

# Grid



- align-content
- justify-content
- align-items
- align-self
- justify-items
- justify-self

# justify-content & align-content

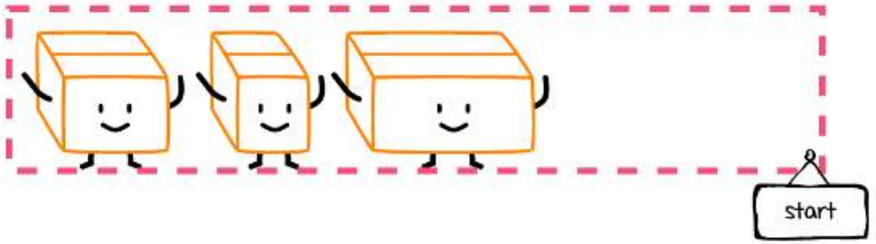


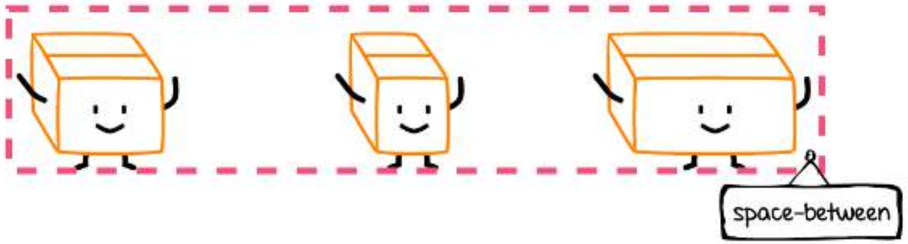

```
.gridcontent {
  justify-content: end;
  align-content: center;

  display: grid;
  grid-template-columns: repeat(3,
25%);
  grid-template-rows: repeat(3,
20%);
  grid-gap: 1em;
  grid-template-areas:
    "a b b"
```

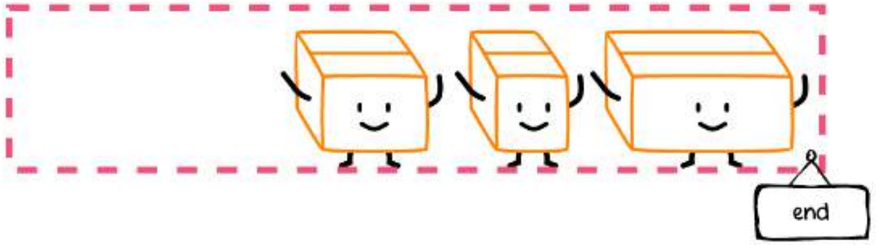
# justify-content & align-content

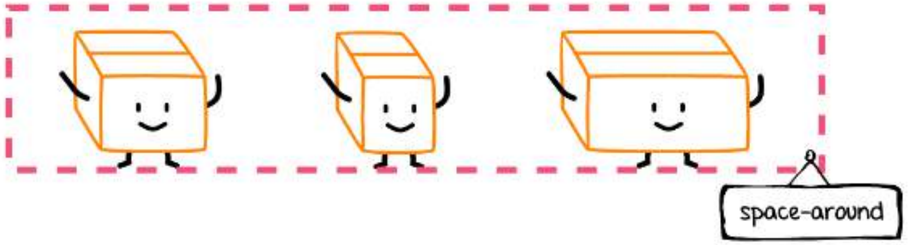
| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1一 | 2二 | 3三 | 4四 | 5五 | 6六 | 7七 | 8八 |
| 9九 | 10十 | 11十一 | 12十二 | 13十三 | 14十四 | 15十五 | 16十六 |
| 17十七 | 18十八 | 19十四 | 20二十 | | | | |

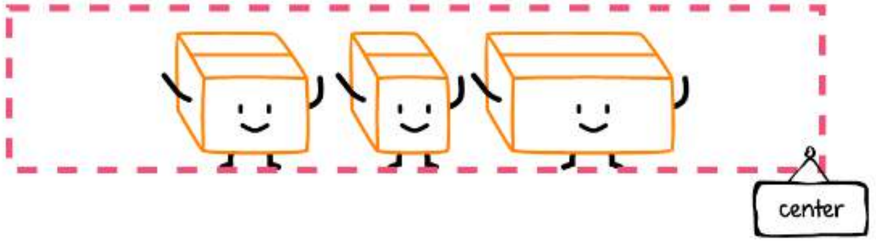
```
.flexcontent .wrapper {
  display: flex;
  flex-wrap: wrap;
  justify-content: flex-start;
  align-content: end;
  height: 70vh;
}

.flexcontent .box {
  height: 3.5em;
  width: 4.5em;
  border: 1px solid;
```

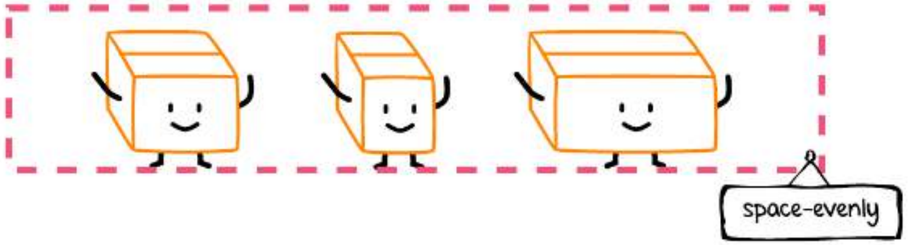

start


space-between

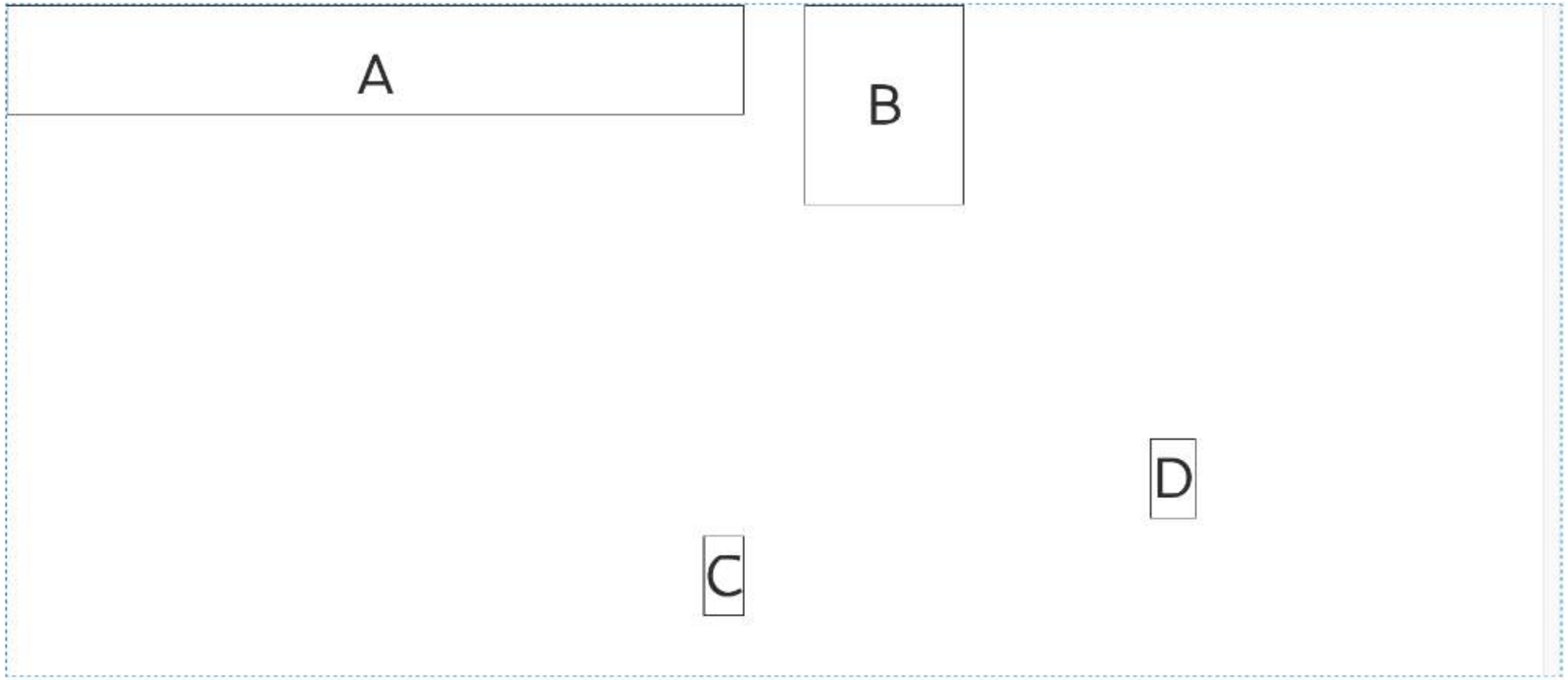

end


space-around


center

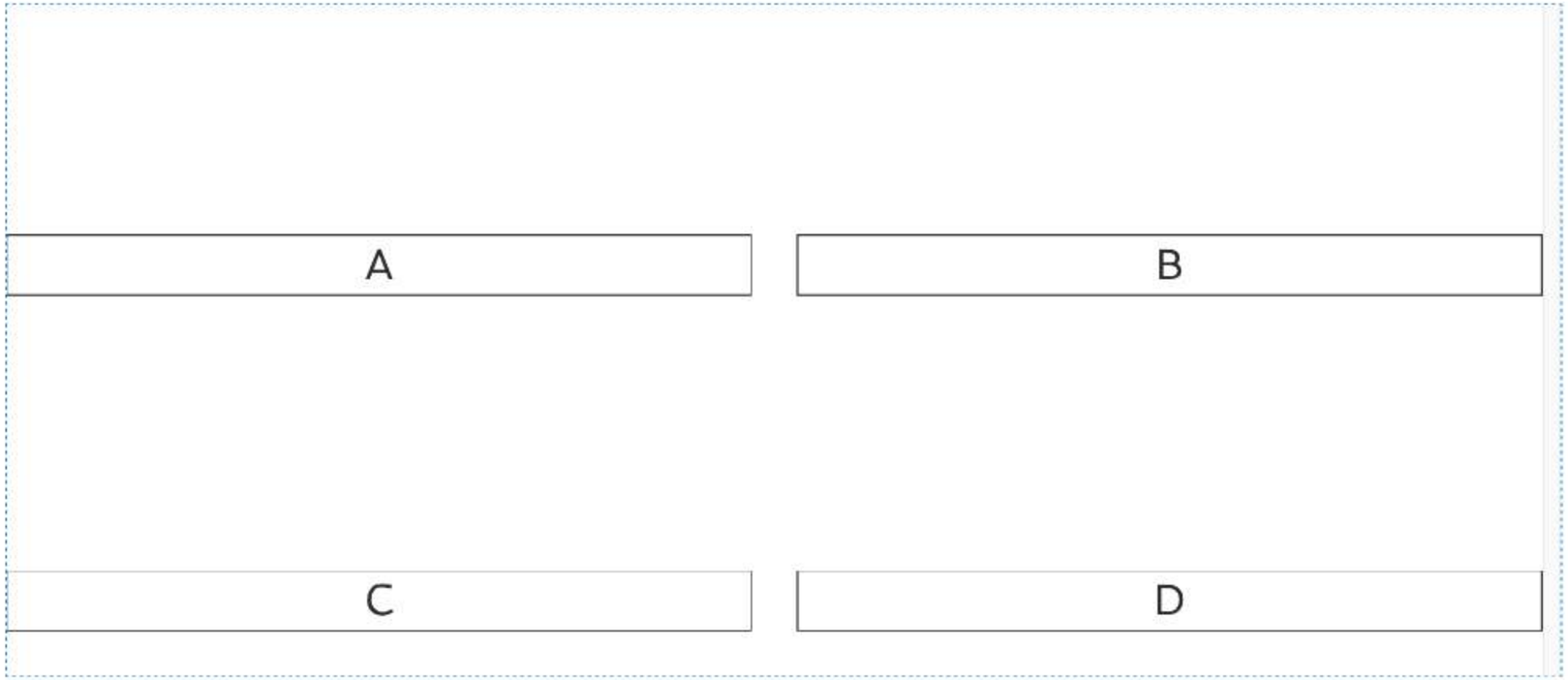

space-evenly

# justify-self & align-self



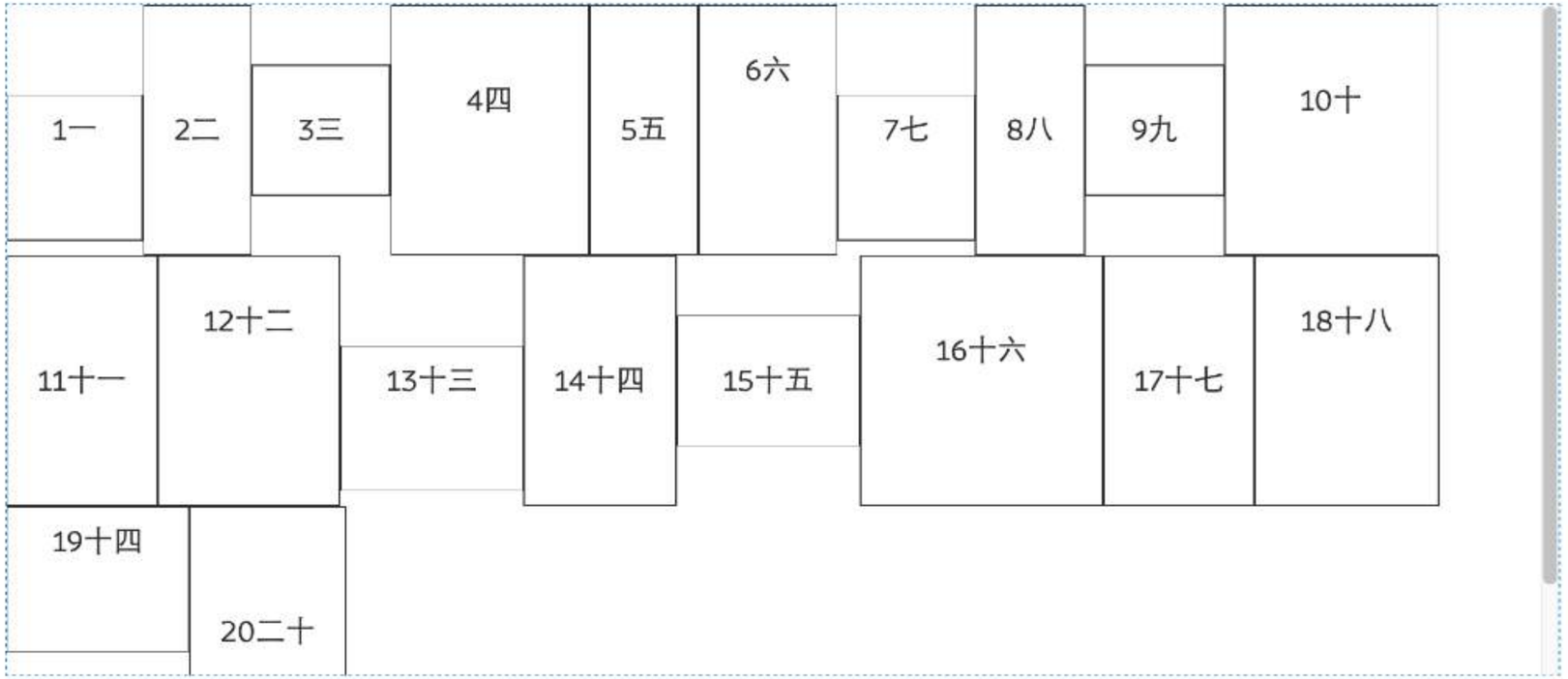

```
  grid-area: c;
  align-self: end;
  justify-self: end;
}

.gridself .box.d {
  grid-area: d;
  align-self: center;
  justify-self: center;
}

.gridself {
  display: grid;
```

# justify-items & align-items

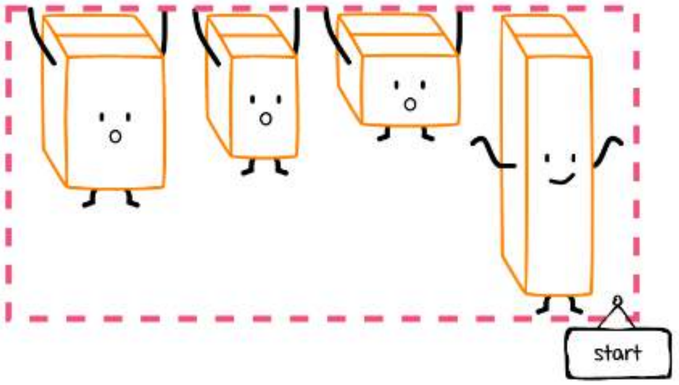


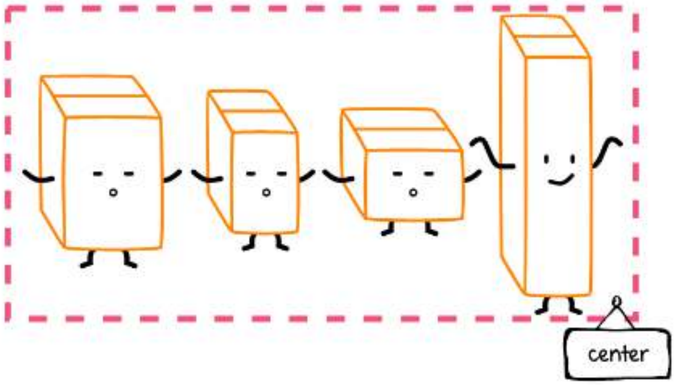

```
.griditems {
  justify-items: normal;
  align-items: end;

  display: grid;
  grid-template-columns:
repeat(4, 1fr);
  grid-gap: 1em;
  grid-auto-rows: calc(25% -
1em);
  grid-template-areas:
    "a a b b"
```

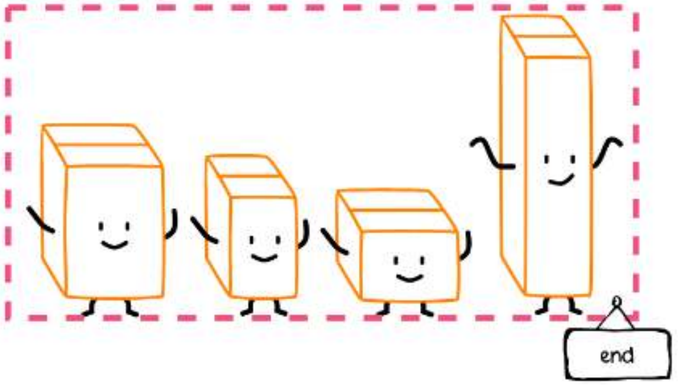
# align-items & align-self

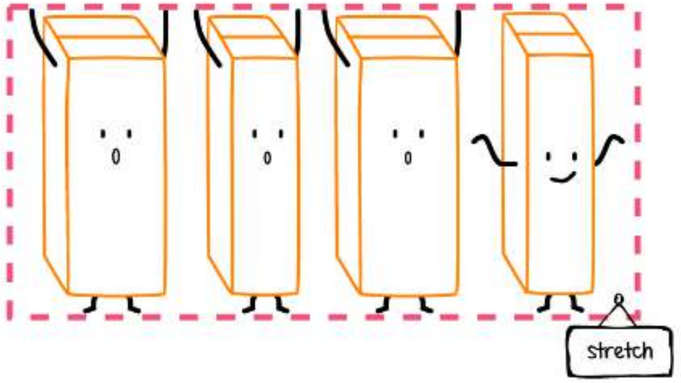


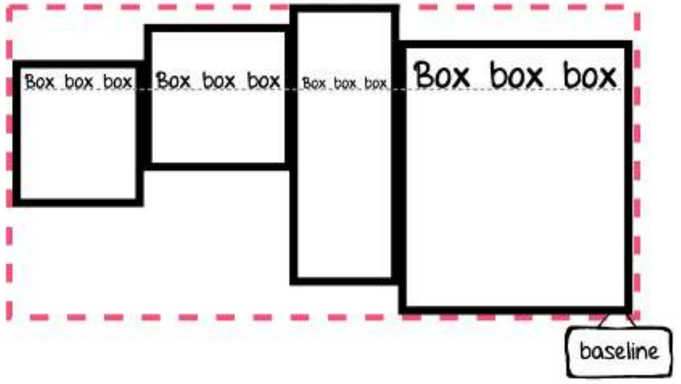

```
.flexitems .wrapper {
  display: flex;
  flex-wrap: wrap;
  align-items: stretch;
}

.flexitems .box:nth-
child(2n+1) {
  padding: 0.5em 1.5em 3em;
  align-self: baseline;
}
```


start


center


end


stretch


Box box box
Box box box
Box box box
Box box box
baseline

# Let's build a thing

Demo

# Cool stuff & pretty things

If time permits...

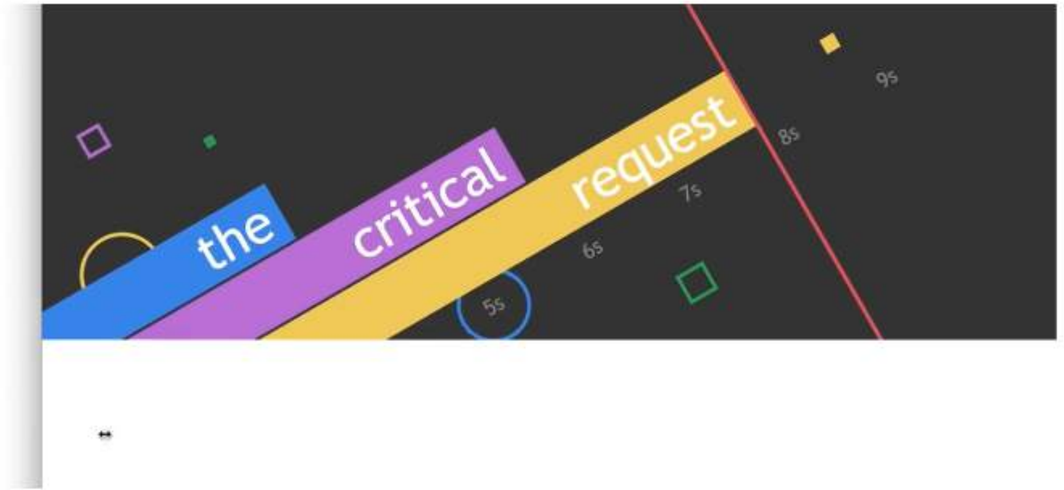
# CSS transforms

Allows us to change the **shape** and **position** of the affected content without disrupting the normal document flow

# 2D transform functions

| | |
|---|---|
| `rotate( <angle> )` | Performs a 2D rotation by the angle specified around the element's origin |
| `translate( <translation-value> [, <translation-value> ]? )` | Performs a 2D translation in the specified X and Y directions |
| `skew( <angle> [, <angle> ]? )` | Performs a 2D skew by the angles specified |
| `scale( <number> [, <number> ]? )` | Performs a 2D scale operation by the scaling vector specified |

# Diagonal text with rotate()




```
.trf2d-cont {
  display: grid;
  grid-template-columns: repeat(9, 1fr)
  grid-template-rows:  repeat(3, 2em) re
  grid-gap: 0.125em;
  position: relative;
}

@media screen and (min-width: 480px) {
  .trf2d-cont {
    transform: rotate(-30deg) translate
  }
}
```

# Perspective text

```
.trf3d-cont div {
  perspective: 45vw;
  perspective-origin: 50% 0%;
}

.trf3d-cont h1 {
  transform: rotateY(-45deg);
  background-color: #0077c8;
  color: #191919;
  padding: 0.25em 1em;
}
```

## CSS transitions

Lets you **create gradual transitions** between the values of specific CSS properties

## CSS animations

Lets you **animate the values** of CSS properties over time, using keyframes

*Note: not all CSS properties are animatable*

# Transition properties

`transition` is a shorthand property, covers the following (values are initial default values):

- `transition-delay: 0s`
- `transition-duration: 0s`
- `transition-property: all`
- `transition-timing-function: ease`

# Animation properties

`animation` is a shorthand property, covers the following (values are initial default values):

- `animation-name: none`
- `animation-duration: 0s`
- `animation-timing-function: ease`
- `animation-delay: 0s`
- `animation-iteration-count: 1`
- `animation-direction: normal`
- `animation-fill-mode: none`
- `animation-play-state: running`

Specified as one or more single animations, separated by commas.

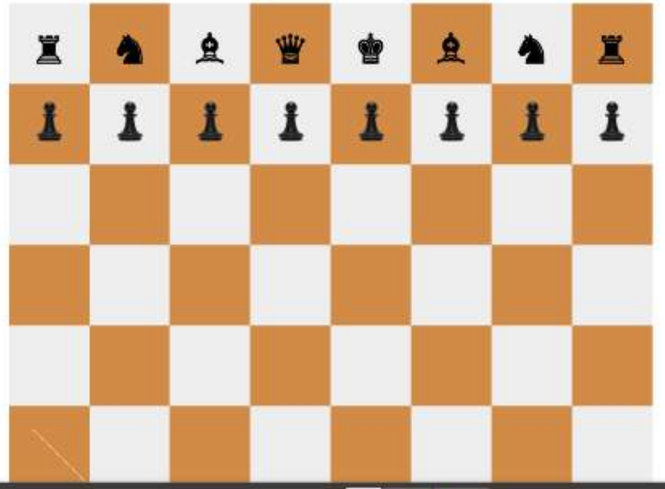
# Fool’s mate

Result

EDIT ON CODEPEN

## Fool's Mate

In chess, Fool's Mate, also known as the Two-Move Checkmate, is the checkmate in the fewest possible number of moves from the start of the game. This can be achieved only by Black, who can deliver checkmate on move 2 with the queen. Fool's Mate received its name because it can only occur if White commits an extraordinary blunder. Even among rank beginners, the mate almost never occurs in practice.



1×

# Magical kittencorn

HTML

```
<div>
  <svg class="kittencorn" xmlns="http://www.w3.org/2000/svg" viewBox="0 0 415 352" fill="none"><path
class="rainbow" stroke="#000" stroke-width="5" d="M123.877 230.922c3.406 2.175 6.516 6.728 8.244 12.418 1.715
5.643 1.948 12.021.031 17.572-1.888 5.468-5.891 10.25-12.963 12.885-7.165 2.669-17.684
3.205-32.61-.401-29.756-7.188-54.915-26.626-69.709-46.127-7.404-9.76-12.078-19.362-13.534-27.273-1.455-7.906.345-
5.04-16.585 2.651-1.719 4.958-2.45 7.019-2.553 2.056-.102 4.063.411 6.135 1.492 4.28 2.234 8.493 6.679 13.401
12.284.685.781 1.38 1.583 2.088 2.399 9.146 10.54 20.398 23.508 37.356 27.282 17.484 3.891 28.625 4.625 36.201
4.894 1.074.038 2.056.066 2.962.093 2.338.068 4.167.121 5.751.285 2.118.219 3.459.614 4.588 1.335z"/><path
class="rainbow" fill-rule="evenodd" stroke="#000" stroke-linecap="square" stroke-width="5" d="M151.901
58.517S86.599 28.76 71.819 42.603c-14.78 13.845 9.153 90.422 9.153 90.422s11.039-23.8
29.824-42.843c18.785-19.043 41.105-31.665 41.105-31.665z" clip-rule="evenodd"/><path fill="#000" fill-
rule="evenodd" d="M138 67.311S93.39 46.981 83.292 56.44c-10.097 9.458 6.253 61.771 6.253 61.771s7.54-16.259
20.374-29.268C122.752 75.933 138 67.31 138 67.31z" clip-rule="evenodd"/><path class="rainbow" fill-
rule="evenodd" stroke="#000" stroke-linecap="square" stroke-width="5" d="M327 58.517s65.302-29.758
80.082-15.914c14.78 13.845-9.152 90.422-9.152 90.422s-11.039-23.8-29.824-42.843C349.32 71.139 327 58.517 327
58.517z" clip-rule="evenodd"/><path fill="#000" fill-rule="evenodd" d="M340.451 67.311s44.611-20.33
54.707-10.871c10.097 9.458-6.252 61.771-6.252 61.771s-7.541-16.259-20.374-29.268c-
12.833-13.01-28.081-21.632-28.081-21.632z" clip-rule="evenodd"/><path class="rainbow" fill-rule="evenodd"
stroke="#000" stroke-width="5" d="M239.5 319c95.269 0 172.5-30.225 172.5-112.067C412 125.091 364.154 43 239.5
43S67 125.091 67 206.933 144.231 319 239.5 319z" clip-rule="evenodd"/><path fill="#fff" fill-rule="evenodd"
d="M269.421 85.475s-17.077-79.652-29.71-79.652C227.077 5.823 210 85.474 210 85.474s13.603 10.033 29.711
10.033c16.107 0 29.71-10.032 29.71-10.032z" clip-rule="evenodd"/><path stroke="#1CAFEF" stroke-linecap="round"
stroke-width="5" d="M220.57 43.788s10.301 9.215 18.976 12.264c8.675 3.05 23.971 4.846 23.971 4.846M227.126
24.256s7.002 7.088 13.172 9.746c6.17 2.658 16.23 2.453 16.23 2.453"/><path stroke="#000" stroke-
linecap="square" stroke-width="5" d="M269.421 85.475l1.484 2.012 1.299-.959-.338-1.578-2.445.525zm0 0l1.483
2.012-.002.002-.004.003-.012.008-.039.029-.136.097c-.116.083-.284.2-.5.347a49.67 49.67 0 0 1-1.856 1.198
```

Result

EDIT ON CODEPEN

1×

# Specifications

Read them first, tutorials later.

CSS specifications: https://www.w3.org/Style/CSS/specs.en.html

# MDN web docs

Go-to resource for all things web development.

MDN web docs: https://developer.mozilla.org/en-US/

# Thank you!

https://www.chenhuijing.com

@hj_chen

@hj_chen

@huijing

Font is Magnetic Pro by Olivier Gourvat