How to understand CSS

@rachelandrew

If you work in web development
learning CSS **is not optional.**

Revisit the things you already know

```
 function MM_reloadPage(init) {  //reloads the window if Nav4 resized
  if (init==true) with (navigator) {if ((appName==&quot;Netscape&quot;)&amp;&amp;(parseIn
t(appVersion)==4)) {
    document.MM_pgW=innerWidth; document.MM_pgH=innerHeight; onresize=MM_reloadPage; }}
  else if (innerWidth!=document.MM_pgW || innerHeight!=document.MM_pgH) location.reload()
;
}
MM_reloadPage(true);
```

It’s true! Some things in CSS have weird names, strange casing, and odd rules.

We can’t break the web

Incomplete List of Mistakes in the Design of CSS

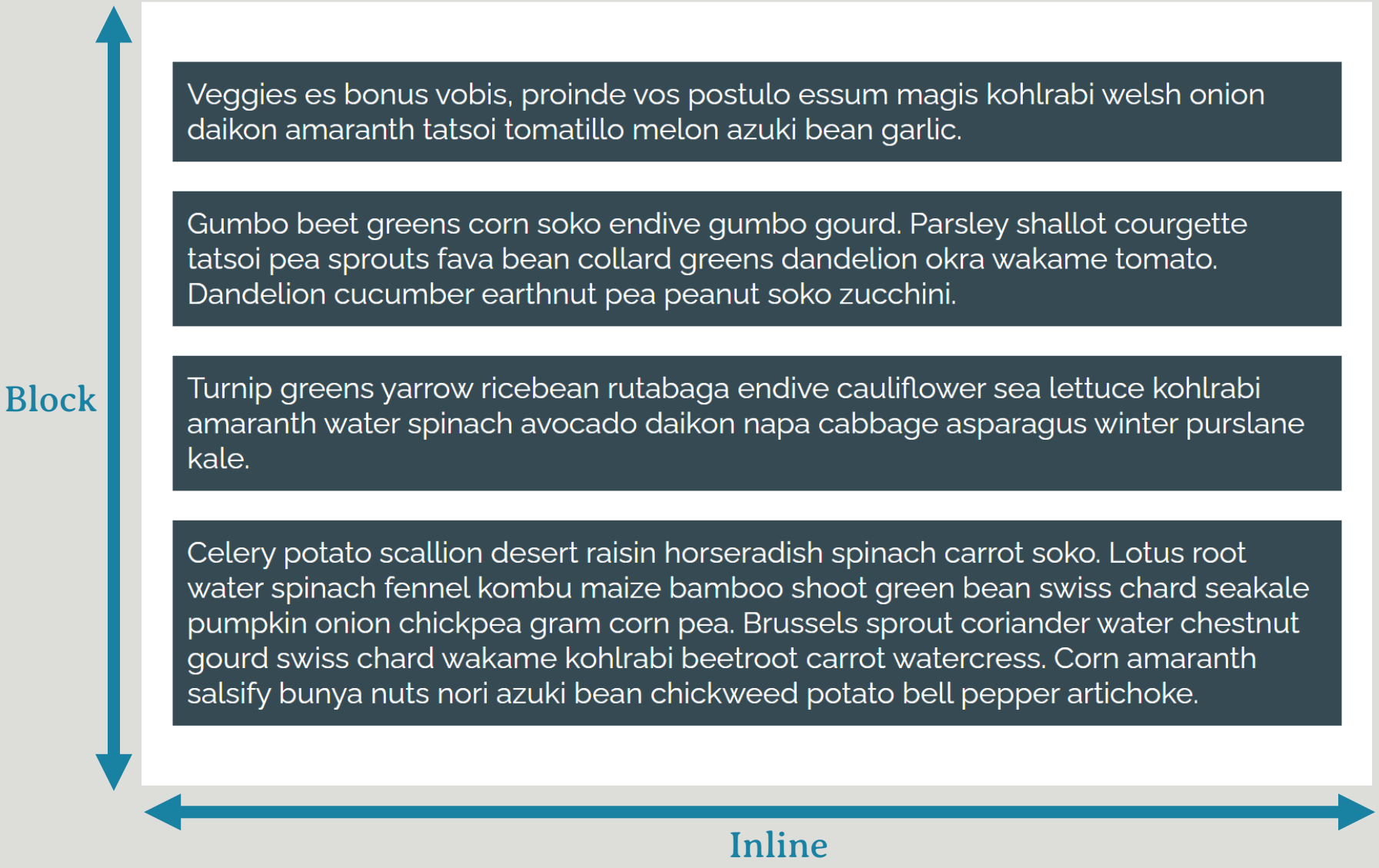
That should be corrected if anyone invents a time machine. :P

- `white-space: nowrap` should be `white-space: no-wrap`
  - and line wrapping behavior should not have been added to `white-space`
- `vertical-align` should not apply to table cells. Instead the CSS3 alignment properties should exist in Level 1.
- `vertical-align: middle` should be `text-middle` or `x-middle` because it's not really in the middle, and such a name would better describes what it does.
- Percentage heights should be calculated against `fill-available` rather than being undefined in auto situations.
- Table layout should be sane.
- Box-sizing should be `border-box` by default.
- `background-size` with one value should duplicate its value, not default the second one to `auto`. Ditto `translate()`.
- `background-position` and `border-spacing` (all 2-axis properties) should take *vertical* first, to match with the 4-direction properties like `margin`.
- Not quite a mistake, because it was a reasonable default for the 90s, but it would be more helpful since then if `background-repeat` defaulted to `no-repeat`.
- The 4-value shorthands like `margin` should go counter-clockwise (so that the inline-start value is before the block-start value).
- `z-index` should be called `z-order` or `depth` and should Just Work on all elements (like it does on flex items).
- `word-wrap`/`overflow-wrap` should not exist. Instead, `overflow-wrap` should be a keyword on 'white-space', like `nowrap` (`no-wrap`).
- The top and bottom margins of a single box should never have been allowed to collapse together automatically as this is the **root of all margin-collapsing evil**.
- Partial collapsing of margins instead of weird rules to handle min/max-heights?
- Tables (like other non-blocks, e.g. flex containers) should form pseudo-stacking contexts.
- The `currentColor` keyword should have retained the dash, `current-color`, as originally specified. Likewise all other color multi-word keyword names.
- There should have been a predictable color naming system (like CNS) instead of the arbitrary X11 names which were eventually adopted.
- `border-radius` should have been `corner-radius`.
- Absolutely-positioned replaced elements should stretch when opposite offset properties (e.g. left+right) are set, instead of being start-aligned.
- The `hyphens` property should be called `hyphenate`. (It's called `hyphens` because the XSL:FO people objected to `hyphenate`.)
- `rgba()` and `hsla()` should not exist, `rgb()` and `hsl()` should have gotten an optional fourth parameter instead (and the alpha value should have used the same format as R, G, and B or S and L).
- descendant combinator should have been » and indirect sibling combinator should have been ++, so there's some logical relationships among the selectors' ascii art
- the `*-blend-mode` properties should've just been `*-blend`
- The syntax of unicode ranges should have consistent with the rest of CSS, like `u0001-u00c8`.
- Unicode ranges should not have had a separate microsyntax with their own tokenization or token handling. The

Naming things is hard

Writing Modes

A writing-mode agnostic way of working.

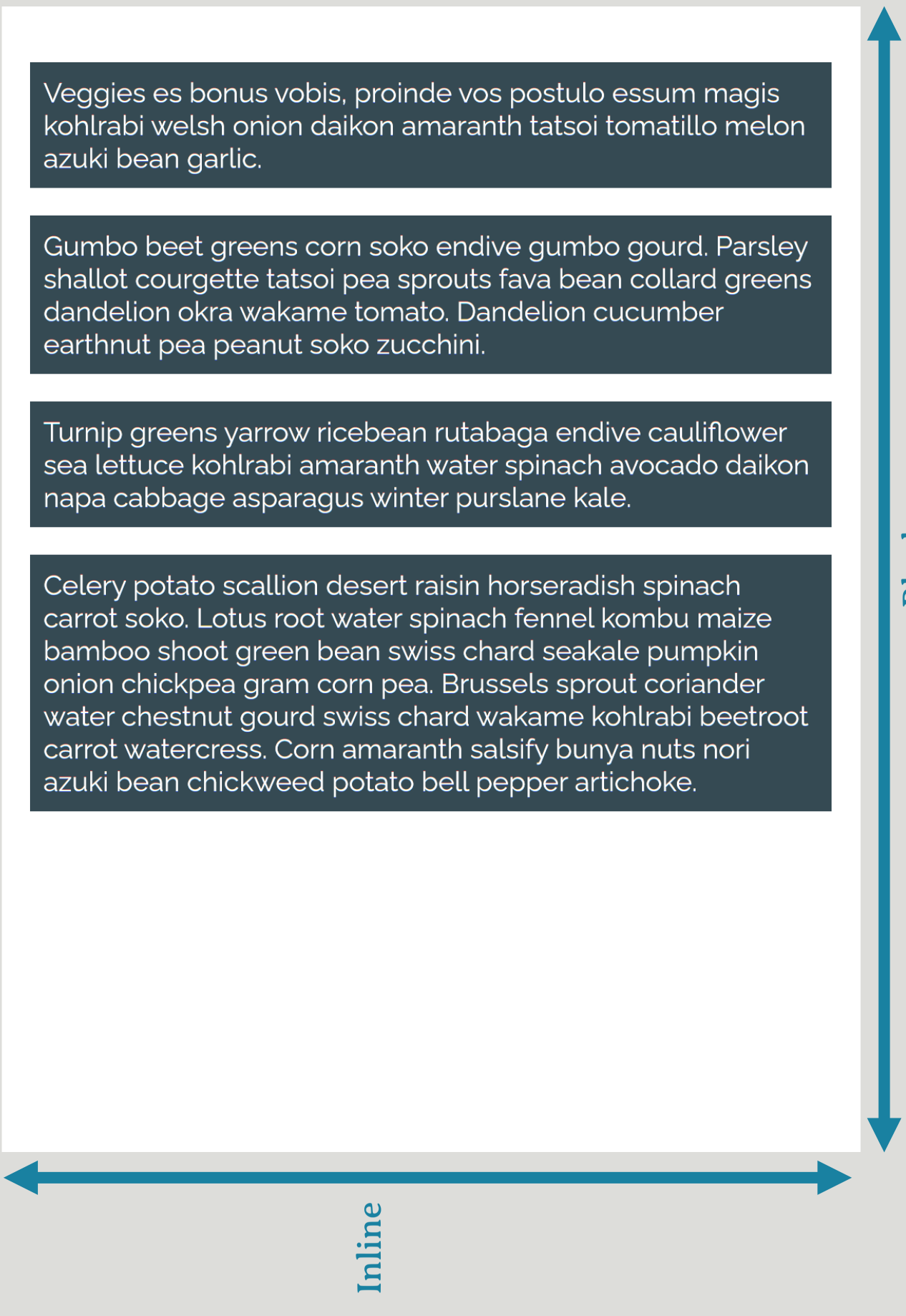

Block
Veggies es bonus vobis, proinde vos postulo essum magis kohlrabi welsh onion daikon amaranth tatsoi tomatillo melon azuki bean garlic.
Gumbo beet greens corn soko endive gumbo gourd. Parsley shallot courgette tatsoi pea sprouts fava bean collard greens dandelion okra wakame tomato. Dandelion cucumber earthnut pea peanut soko zucchini.
Turnip greens yarrow ricebean rutabaga endive cauliflower sea lettuce kohlrabi amaranth water spinach avocado daikon napa cabbage asparagus winter purslane kale.
Celery potato scallion desert raisin horseradish spinach carrot soko. Lotus root water spinach fennel kombu maize bamboo shoot green bean swiss chard seakale pumpkin onion chickpea gram corn pea. Brussels sprout coriander water chestnut gourd swiss chard wakame kohlrabi beetroot carrot watercress. Corn amaranth salsify bunya nuts nori azuki bean chickweed potato bell pepper artichoke.
Inline

Veggies es bonus vobis, proinde vos postulo essum magis kohlrabi welsh onion daikon amaranth tatsoi tomatillo melon azuki bean garlic.

Gumbo beet greens corn soko endive gumbo gourd. Parsley shallot courgette tatsoi pea sprouts fava bean collard greens dandelion okra wakame tomato. Dandelion cucumber earthnut pea peanut soko zucchini.

Turnip greens yarrow ricebean rutabaga endive cauliflower sea lettuce kohlrabi amaranth water spinach avocado daikon napa cabbage asparagus winter purslane kale.

Celery potato scallion desert raisin horseradish spinach carrot soko. Lotus root water spinach fennel kombu maize bamboo shoot green bean swiss chard seakale pumpkin onion chickpea gram corn pea. Brussels sprout coriander water chestnut gourd swiss chard wakame kohlrabi beetroot carrot watercress. Corn amaranth salsify bunya nuts nori azuki bean chickweed potato bell pepper artichoke.



Block start

Inline start

Veggies es bonus vobis, proinde vos postulo essum magis kohlrabi welsh onion daikon amaranth tatsoi tomatillo melon azuki bean garlic.

Gumbo beet greens corn soko endive gumbo gourd. Parsley shallot courgette tatsoi pea sprouts fava bean collard greens dandelion okra wakame tomato. Dandelion cucumber earthnut pea peanut soko zucchini.

Turnip greens yarrow ricebean rutabaga endive cauliflower sea lettuce kohlrabi amaranth water spinach avocado daikon napa cabbage asparagus winter purslane kale.

Celery potato scallion desert raisin horseradish spinach carrot soko. Lotus root water spinach fennel kombu maize bamboo shoot green bean swiss chard seakale pumpkin onion chickpea gram corn pea. Brussels sprout coriander water chestnut gourd swiss chard wakame kohlrabi beetroot carrot watercress. Corn amaranth salsify bunya nuts nori azuki bean chickweed potato bell pepper artichoke.

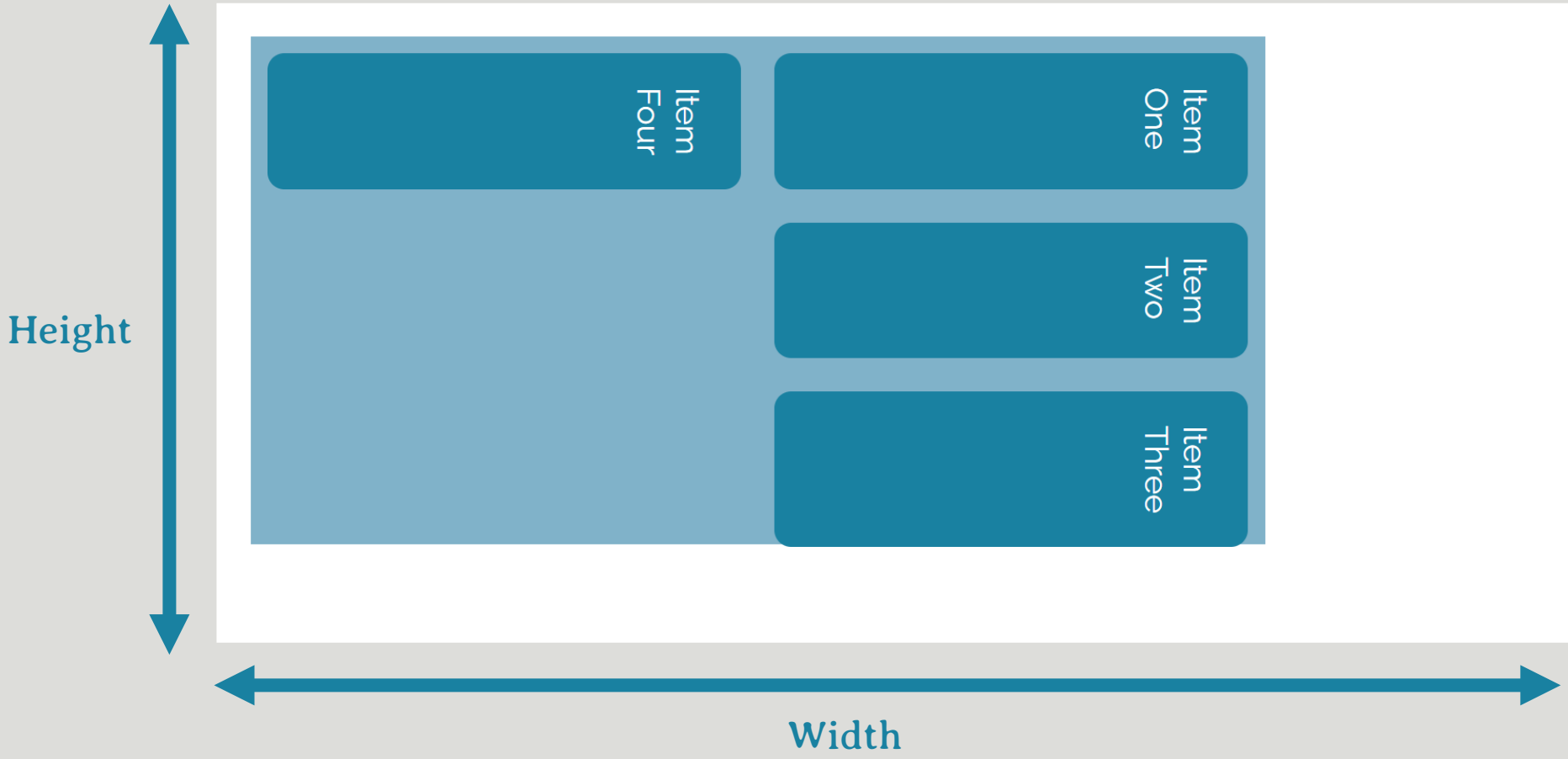
Inline end

Block end

Web layout is tied to physical dimensions

We think in top, right, bottom, left, width, height.


```
.example {
  width: 600px;
  height: 300px;
}
```


Item One
Item Two
Item Three
Item Four
Height
Width

Logical Properties & Values

Mapping the physical to the flow-relative.

```
.example {
  inline-size: 600px;
  block-size: 300px;
}
```

inline-size
Item Four
Item One
Item Two
Item Three
block-size

Physical vs. Logical

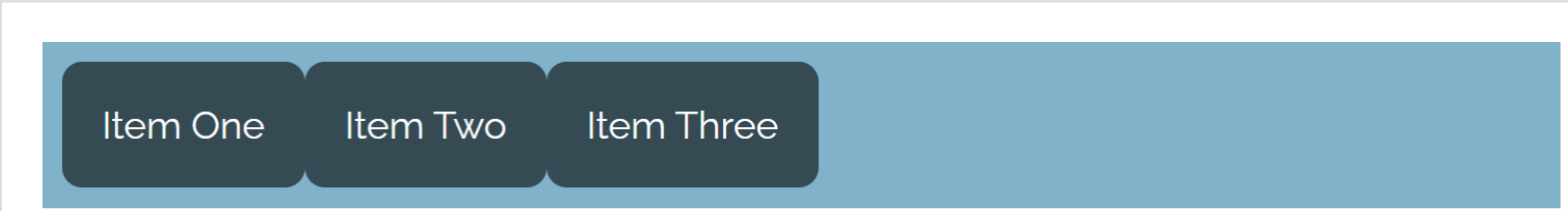
```
.example {
  padding-top: 10px;
  padding-right: 2em;
  margin-bottom: 2em;
}
```

```
.example {
  padding-block-start: 10px;
  padding-inline-end: 2em;
  margin-block-end: 2em;
}
```

Block and inline, start and end

Initial Values

Every property has a value.




```
.example {
  display: flex;
}
```

- Understanding Success Criterion 1.3.2 | W3C Understanding WCAG 2.0

Formal definition

Initial value	`row`
Applies to	flex containers
Inherited	no
Computed value	as specified
Animation type	discrete

Formal syntax

```
row | row-reverse | column | column-reverse
```

Examples

Reversing flex container columns and rows

HTML

```
<h4>This is a Column-Reverse</h4>
<div id="content">
  <div class="box" style="background-color:red;">A</div>
  <div class="box" style="background-color:lightblue;">B</div>
  <div class="box" style="background-color:yellow;">C</div>
</div>
<h4>This is a Row-Reverse</h4>
```

Normal flow

Block and inline layout

Just some HTML and content

Veggies es bonus vobis, proinde vos postulo essum magis kohlrabi welsh onion daikon amaranth tatsoi tomatillo melon azuki bean garlic.

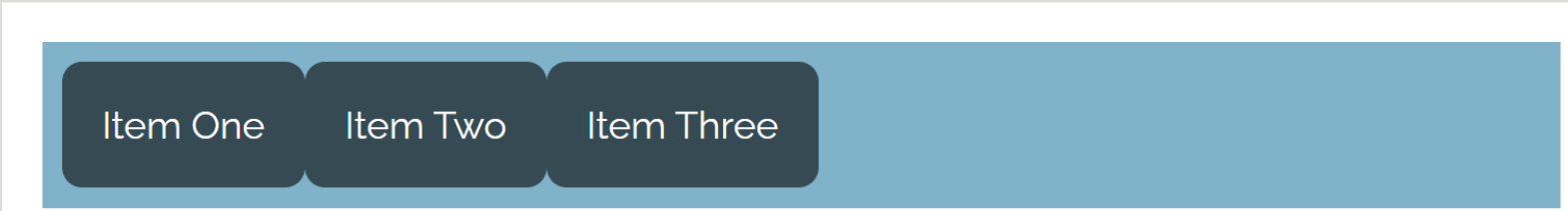
Gumbo beet greens corn soko endive gumbo gourd. Parsley shallot courgette tatsoi pea sprouts fava bean collard greens dandelion okra wakame tomato. Dandelion cucumber earthnut pea peanut soko zucchini.

Turnip greens yarrow ricebean rutabaga endive cauliflower sea lettuce kohlrabi amaranth water spinach avocado daikon napa cabbage asparagus winter purslane kale.

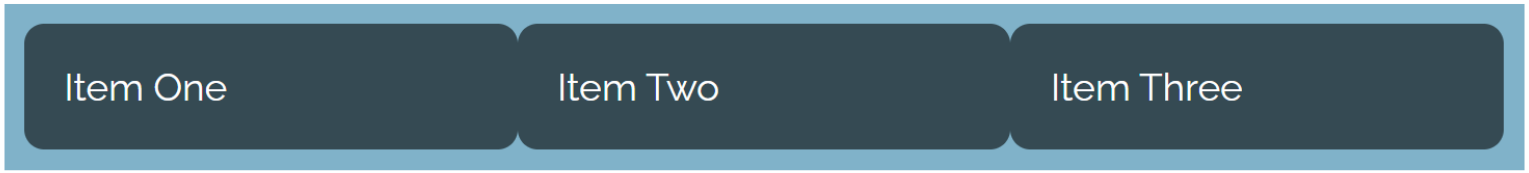
Formatting contexts

Switching from block to flex or grid.


Item One
Item Two
Item Three




```
.example {
  display: flex;
}
```




```
.example {
  display: grid;
  grid-template-columns: 1fr 1fr 1fr;
}
```

Changing the value of display changes the formatting context of the direct children of the element.

Inside those children we return to normal flow.

Item One

Item Two

Item Three

Paragraph 1.

Paragraph 2.

Generated content

The strange world of ::before and ::after

::before and ::after are pseudo-elements

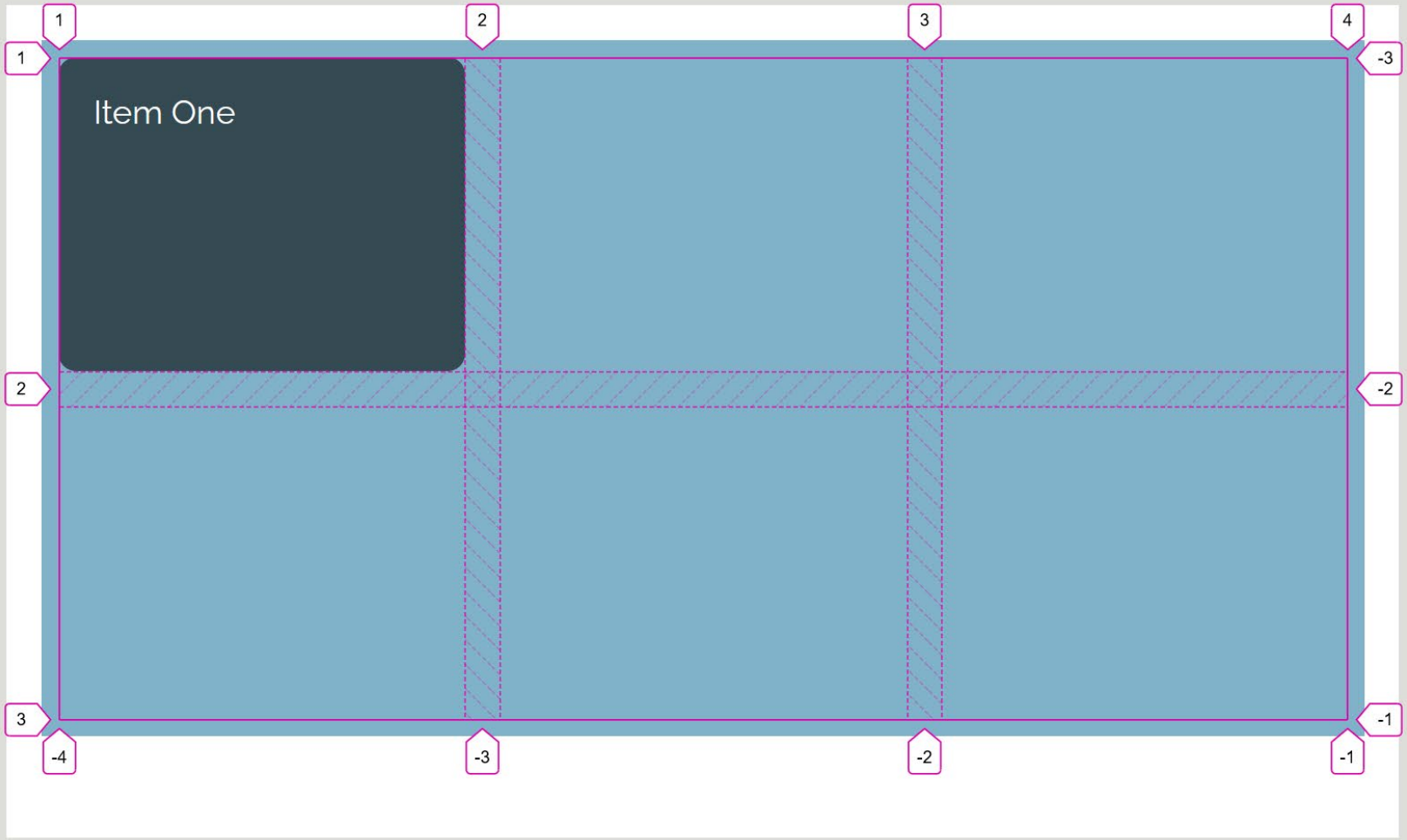
They use two colons :: to distinguish them from pseudo-classes (one colon).

In the past they were defined with one colon :before and :after.

So browsers maintain that syntax for backwards compatibility.

::before and ::after add a first and last child

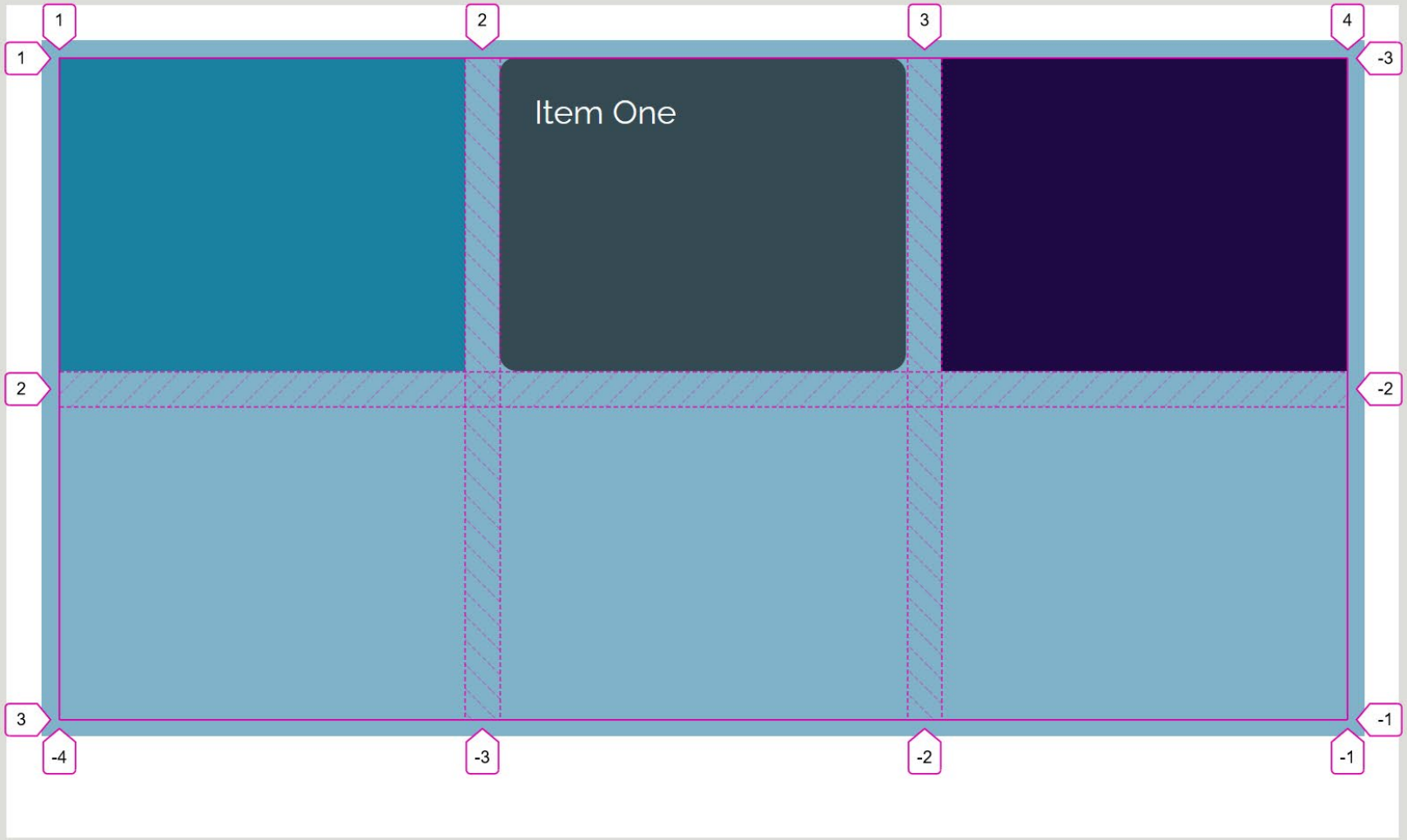
Before the other children and after the other children of the element.


1
2
3
4
1
-3
Item One
2
-2
3
-1
-4
-3
-2
-1

```
.example {
    display: grid;
}

.example::before {
  content: "";
  background-color: #1981a1;
}

.example::after {
  content: "";
  background-color: #1f0945;
}
```

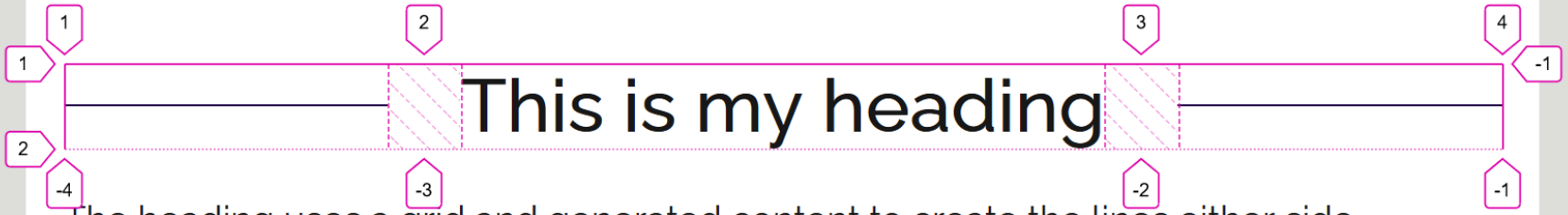

1
2
3
4
1
-3
Item One
2
-2
3
-1
-4
-3
-2
-1

```css
h1 {
  display: grid;
  grid-template-columns: 1fr auto 1fr;
  gap: 1em;
}

h1::before, h1::after {
  content: "";
  align-self: center;
  border-bottom: 2px solid #1f0945;
}
```

This is my heading

The heading uses a grid and generated content to create the lines either side.



The heading uses a grid and generated content to create the lines either side.

Busting out of flow

Position and float

Veggies es bonus vobis, proinde vos postulo essum magis kohlrabi welsh onion daikon amaranth tatsoi tomatillo melon azuki bean garlic.

Gumbo beet greens corn soko endive gumbo gourd. Parsley shallot courgette tatsoi pea sprouts fava bean collard greens dandelion okra wakame tomato. Dandelion cucumber earthnut pea peanut soko zucchini.

Turnip greens yarrow ricebean rutabaga endive cauliflower sea lettuce kohlrabi.

display: flow-root

Creates a new Block Formatting Context

```
.box {
  background-color: rgb(43,91,128);
  display: flow-root;
}
```

Veggies es bonus vobis, proinde vos postulo essum magis kohlrabi welsh onion daikon amaranth tatsoi tomatillo melon azuki bean garlic.

Gumbo beet greens corn soko endive gumbo gourd. Parsley shallot courgette tatsoi pea sprouts fava bean collard greens dandelion okra wakame tomato. Dandelion cucumber earthnut pea peanut soko zucchini.

Turnip greens yarrow ricebean rutabaga endive cauliflower sea lettuce kohlrabi.

Margin collapsing

The rules around combining margins.

Margins collapse in the block direction

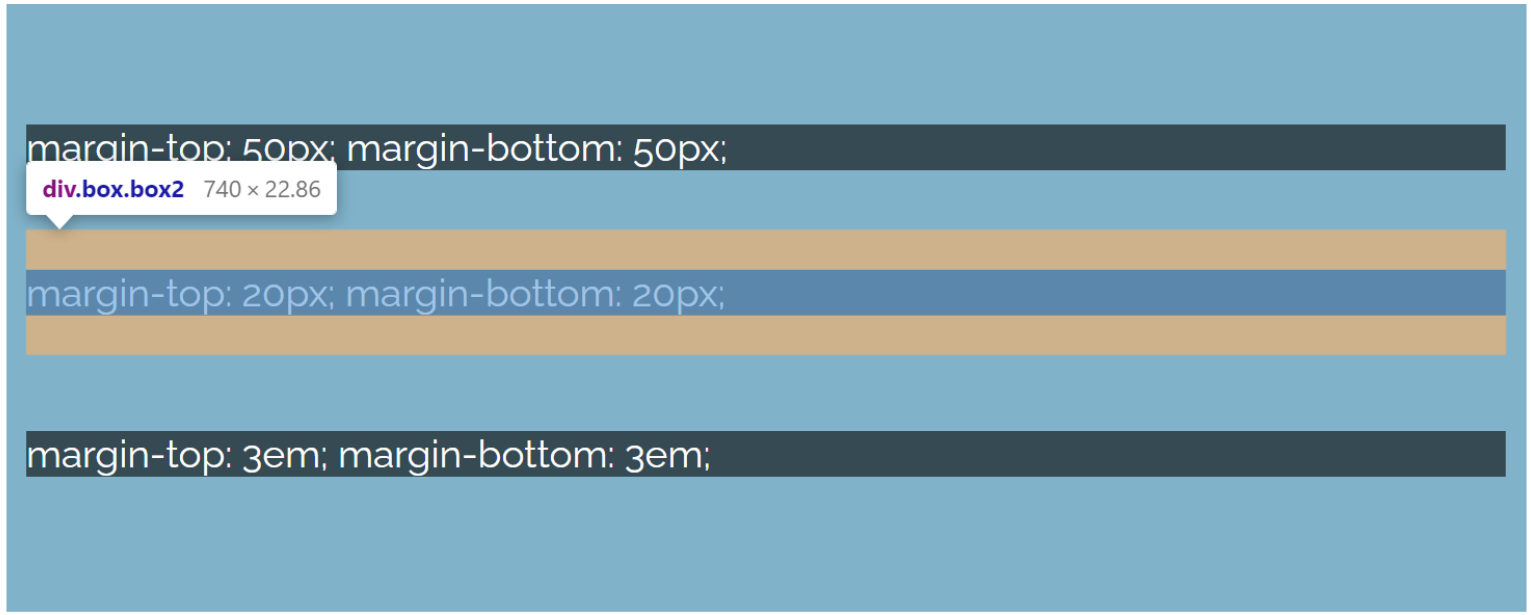
For example, between paragraphs.

Margins only collapse on items participating in a block formatting context.

Flex and Grid items do not collapse margins.

Adjacent children

The margin-bottom of a paragraph will combine with the margin-top of a following paragraph.


margin-top: 50px; margin-bottom: 50px;
div.box.box2 740 × 22.86
margin-top: 20px; margin-bottom: 20px;
margin-top: 3em; margin-bottom: 3em;

Completely empty boxes.

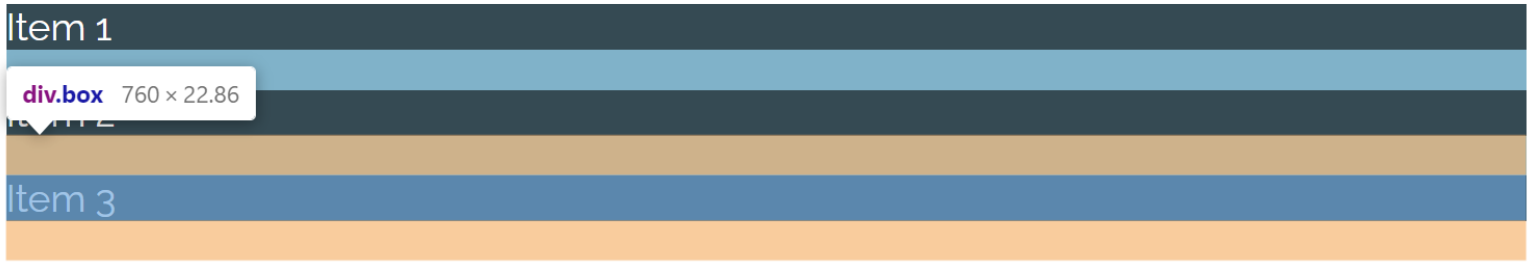
The top and bottom margin will combine.

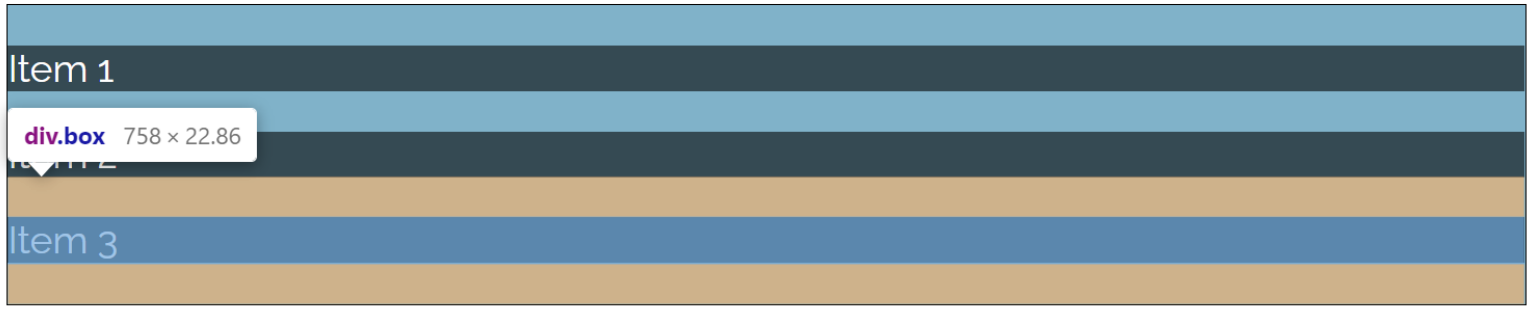

A box
Another box


A box
Another box

First and last child and parent

The margin on these children can be combined with the margin on their parent.


Item 1
div.box 760 × 22.86
Item 3


Item 1
div.box 758 × 22.86
Item 3

Box Alignment

https://drafts.csswg.org/css-align/

Aligning in the block and inline dimensions

Distribution of space and
alignment of items within their space

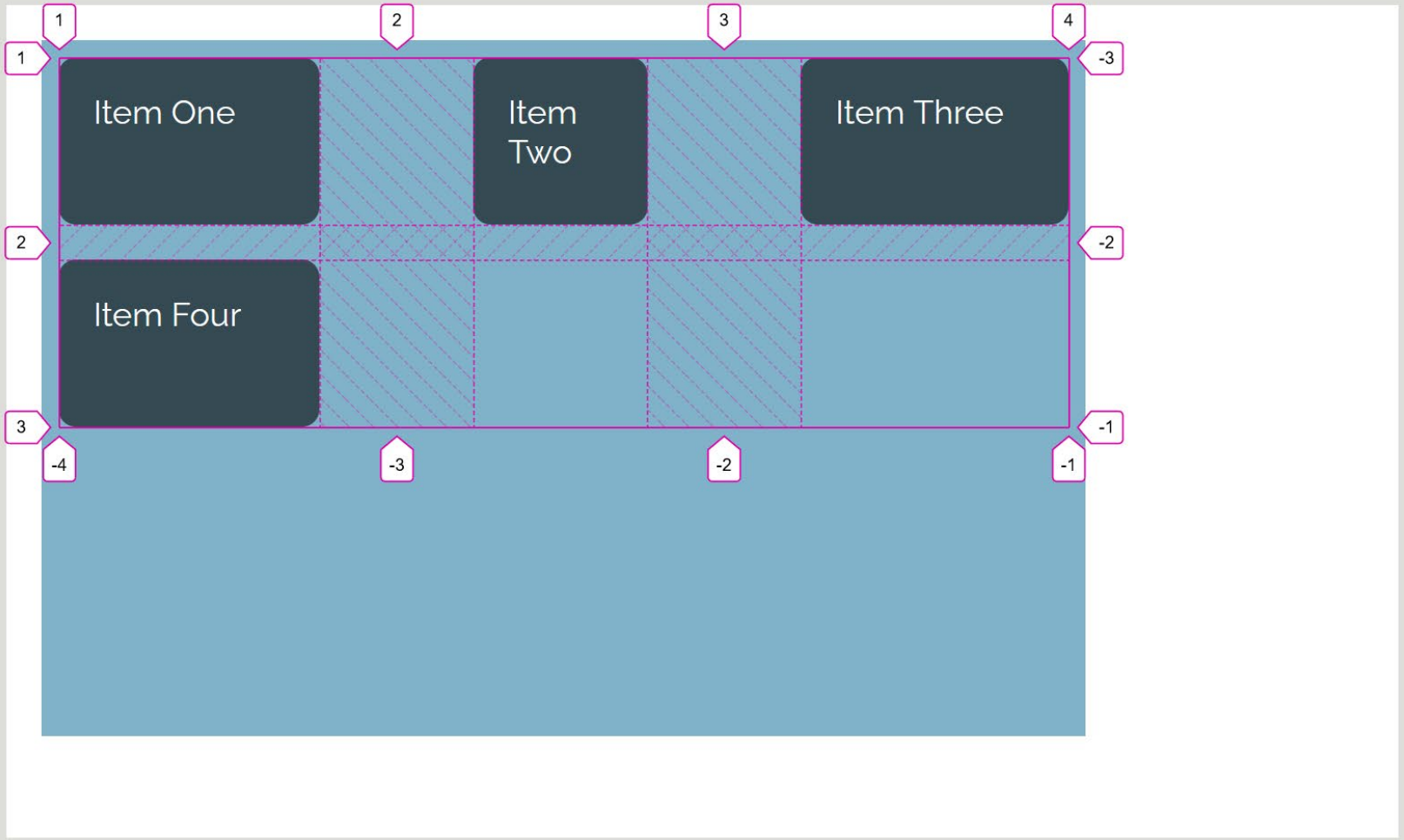
Block start

Inline start

justify-content

In grid, inline space distribution between tracks.

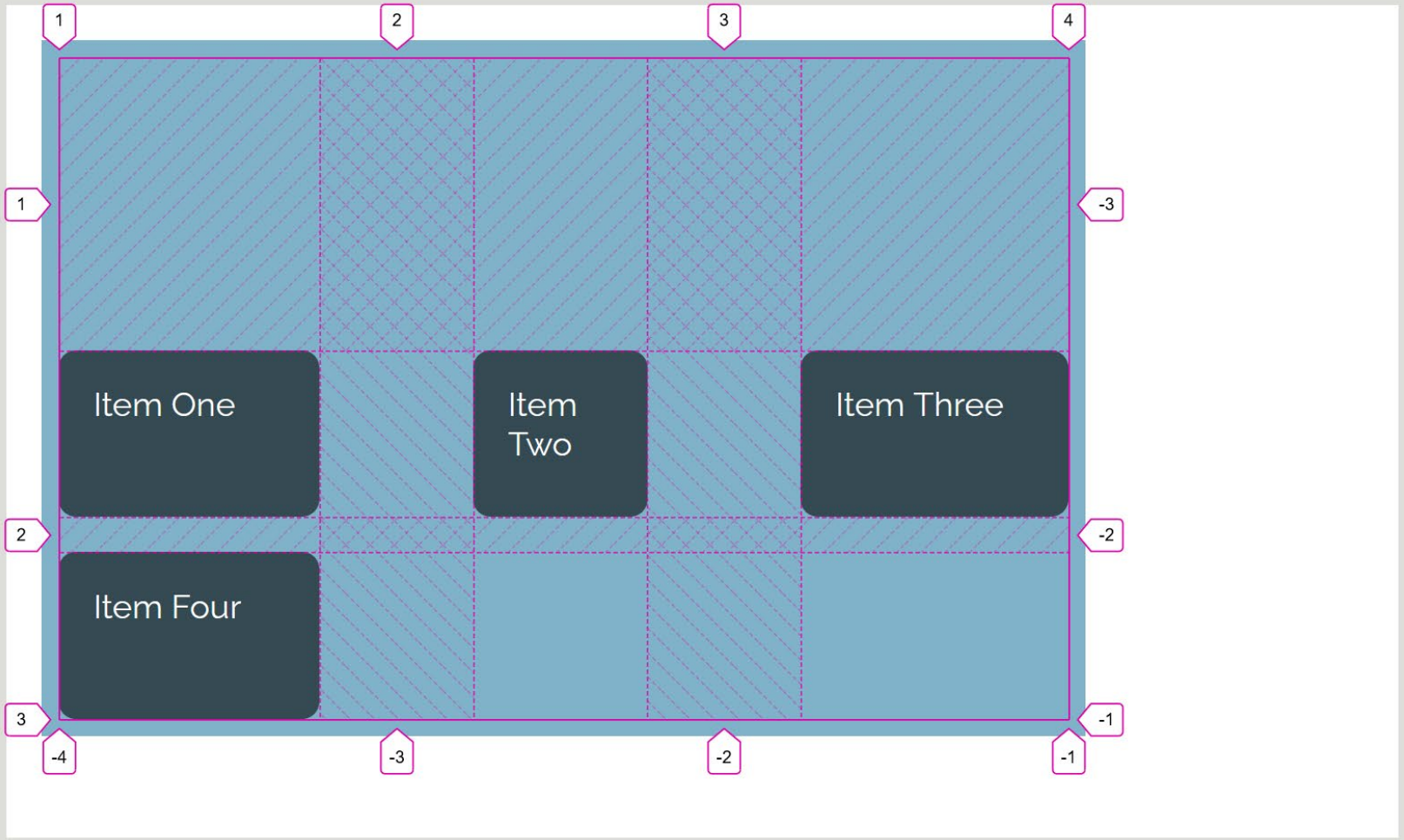

```
.example {
  justify-content: space-between;
}
```




```
.example {
  justify-content: space-between;
}
```

align-content

In Grid, block dimension space distribution between tracks

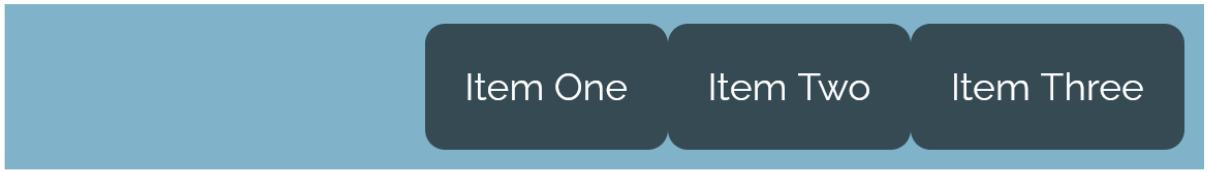



```
.example {
  align-content: end;
}
```

In flexbox, we justify on the main axis and align on the cross axis

Justify-content

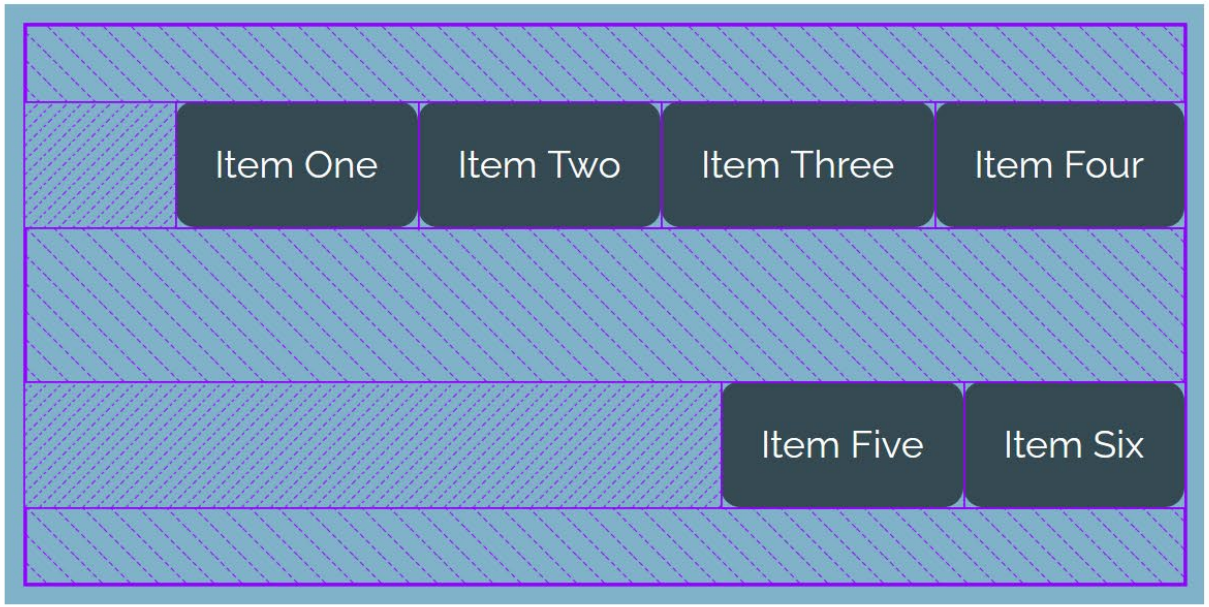
In flex layout, main axis space distribution between flex items




```
.example {
    justify-content: flex-end;
}
```

align-content

In flex layout, cross-axis space distribution between lines

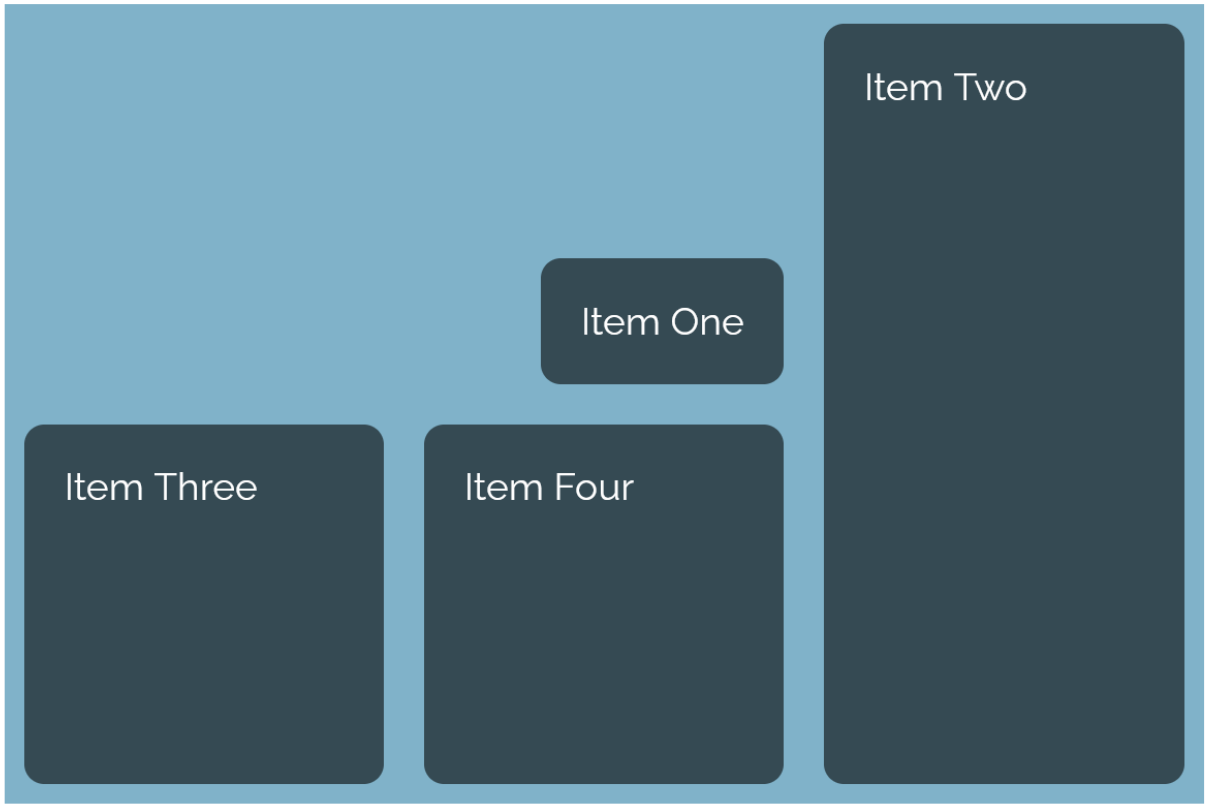



```
.example {
  align-content: space-around;
}
```

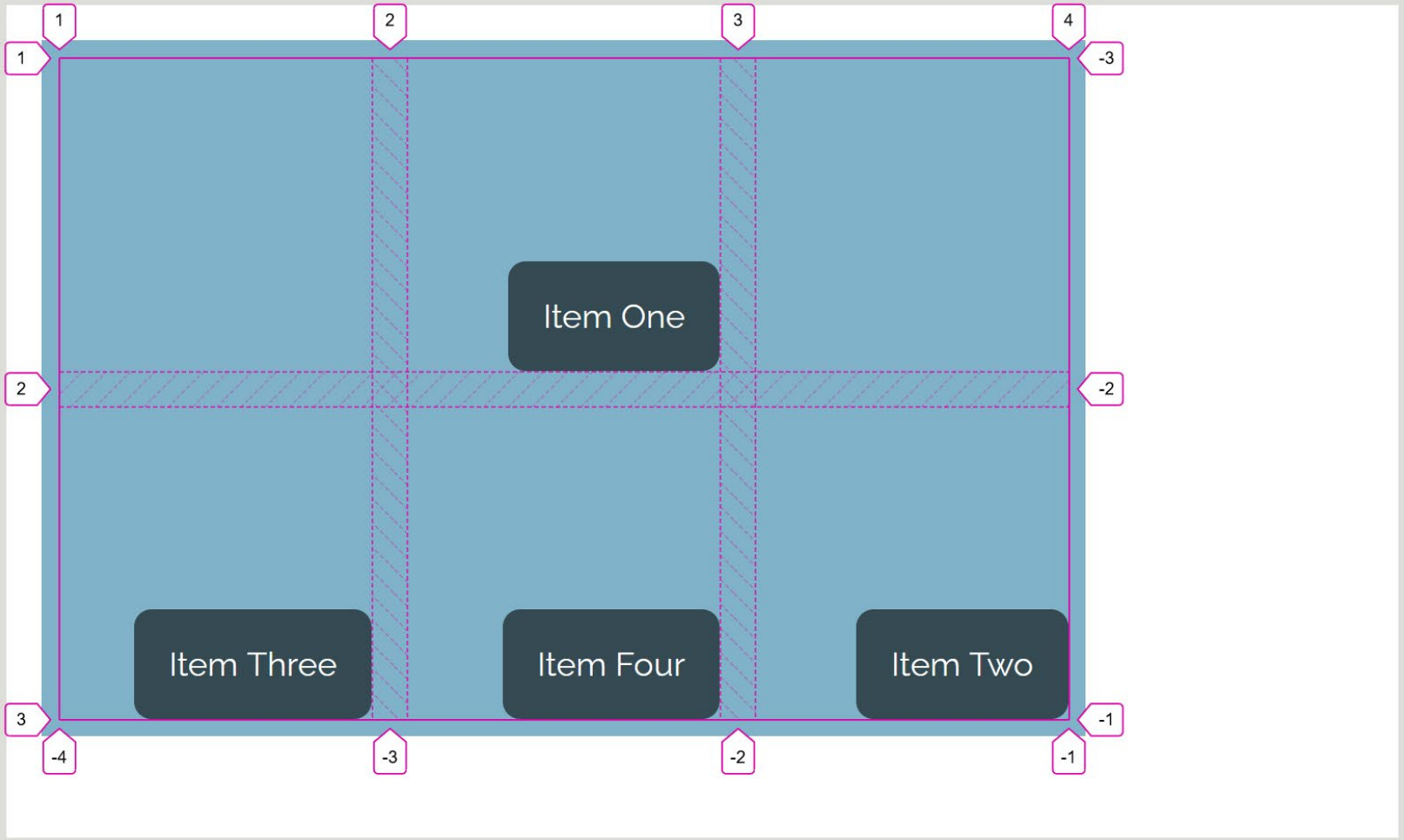
For –content properties to do anything, you must have spare space to distribute!

Aligning items inside their areas


Item One
Item Two
Item Three
Item Four




```
.example {
  justify-self: end;
  align-self: end;
}
```



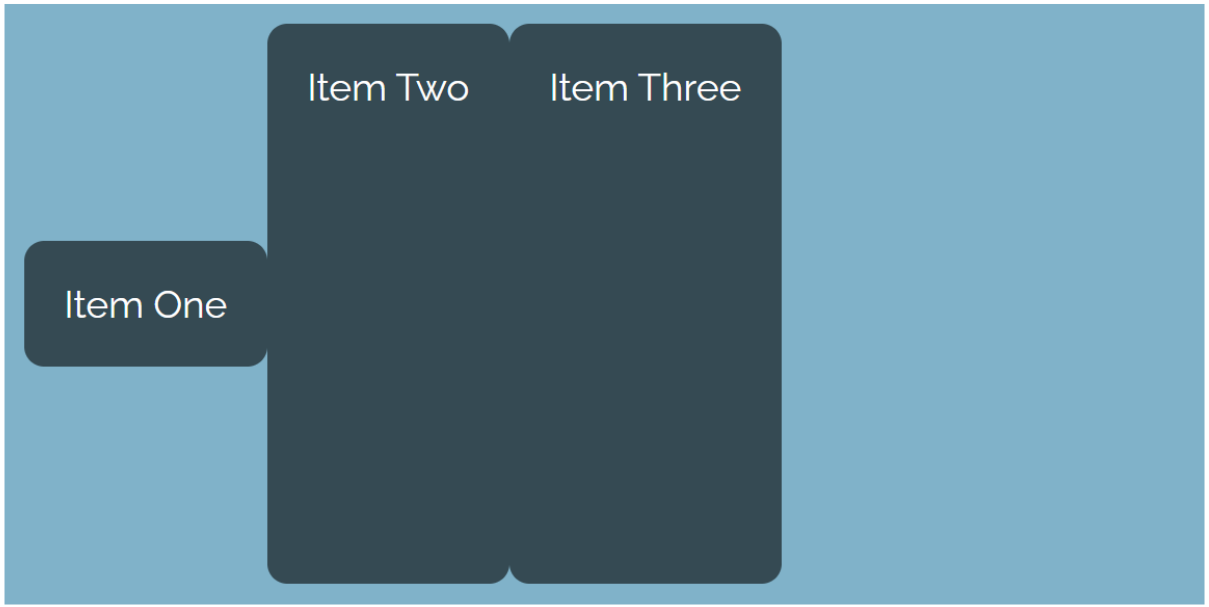

```
.example {
  justify-items: end;
  align-items: end;
}
```

“

[justify-self] does not apply to flex items, because there is more than one item in the main axis.”

”

https://drafts.csswg.org/css-align/#justify-flex




```
.example {
  align-self: center;
}
```

justify-

- Main axis alignment in flexbox (the direction of flex-direction)
- Flexbox only supports justify-content (not justify–items or justify–self)
- Inline axis alignment in grid

align-

- Cross axis alignment in flexbox
- Block axis alignment in grid

-content

- Space distribution between flex items or grid tracks
- No spare space and these properties do nothing

-items, -self

- Alignment within the grid area
- Alignment against other flex items on the cross axis

Box sizing

https://drafts.csswg.org/css-sizing-3/

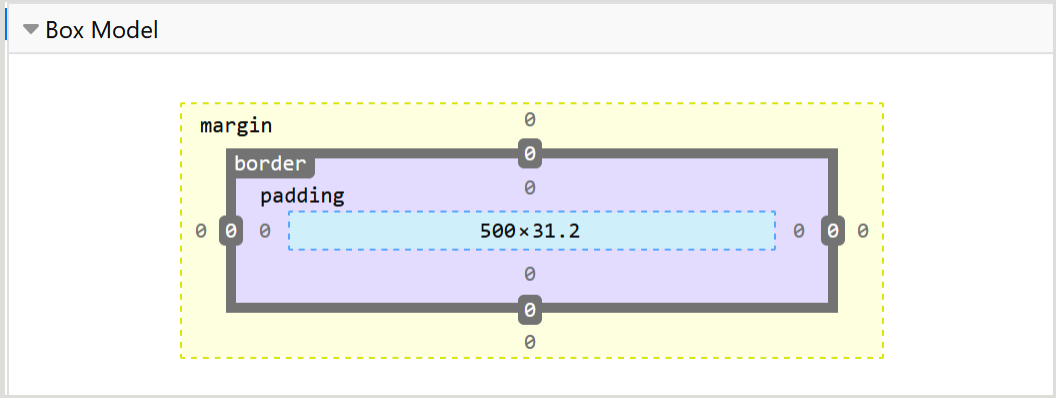
What about the Box Model?

Isn't there a rule that anyone talking about CSS must show a box model diagram?

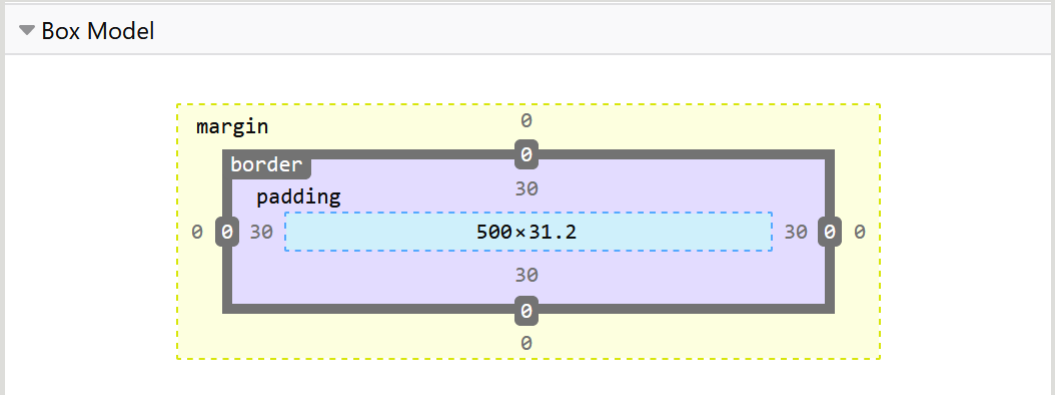
When we had to control the size of each item in a layout, the Box Model was very important.

I am a box with content

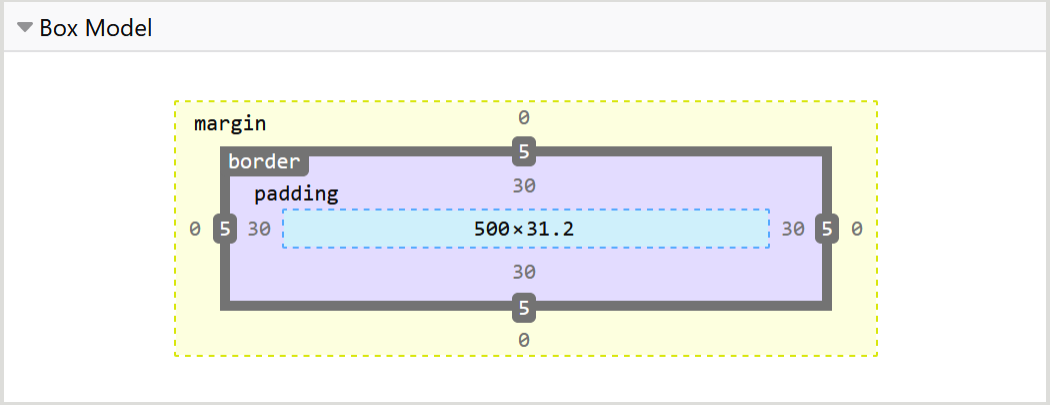
I am a box with content



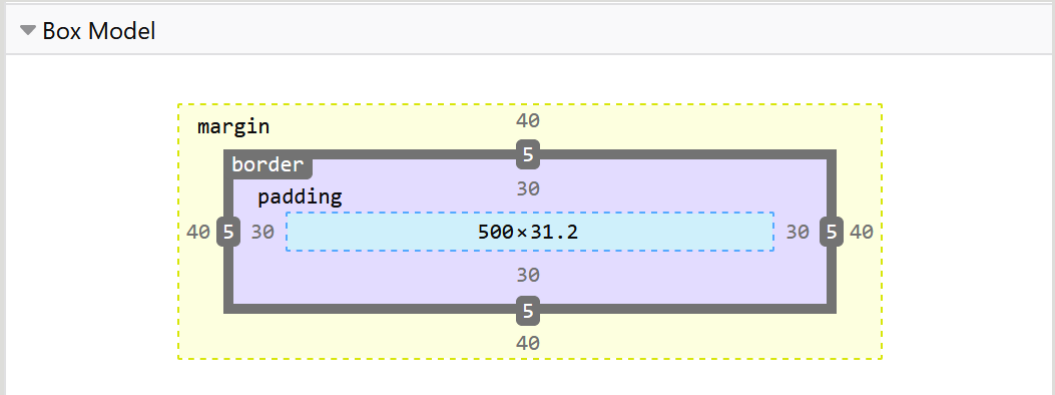
I am a box with content



I am a box with content



I am a box with content



What is the inline-size or width of the box?

By default, the content-box

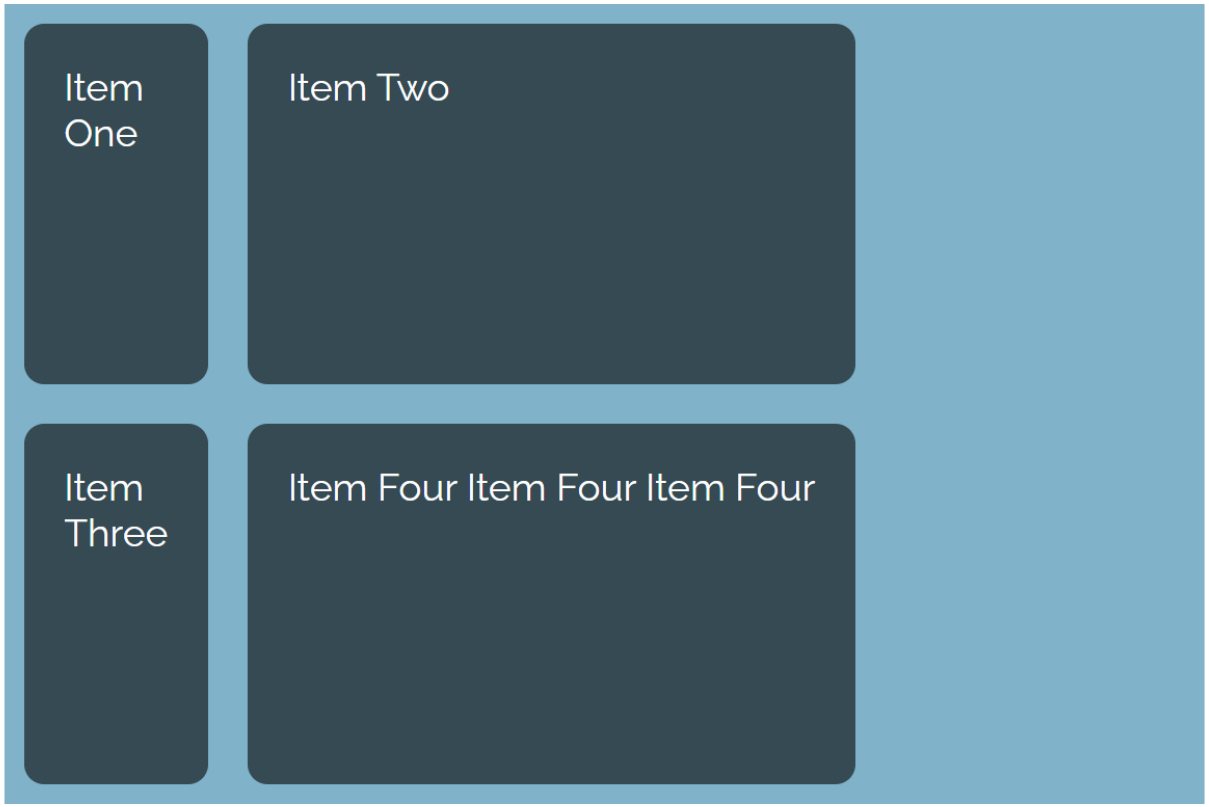
If you want your specified width to include padding and border...

... set the box-sizing property to border-box.

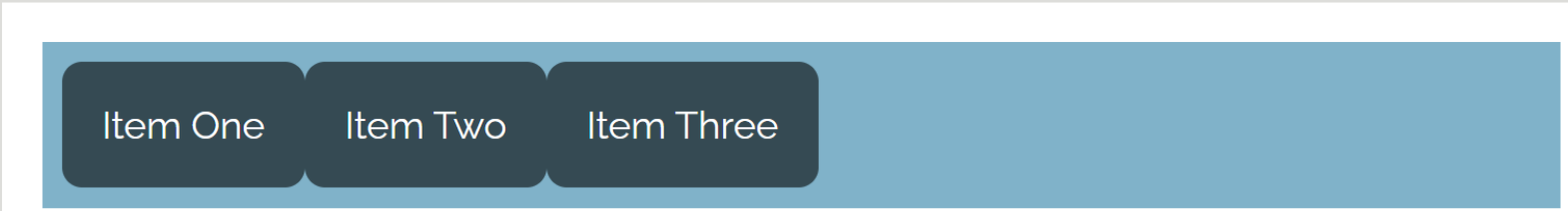
```
.example {
  box-sizing: border-box;
}
```

In the past everything was a length or a percentage.

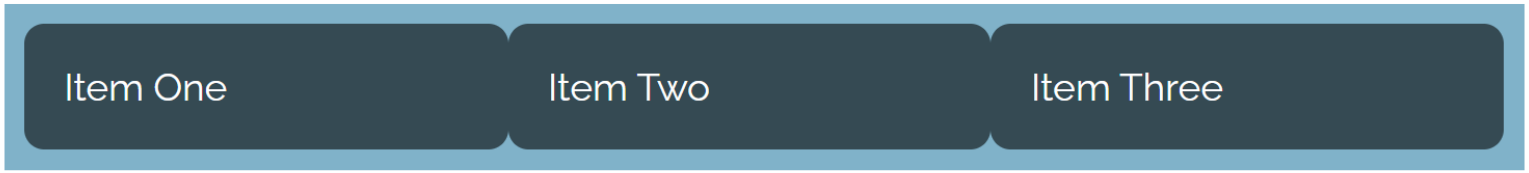
What is the minimum and maximum size of this thing


Item One
Item Two
Item Three
Item Four Item Four Item Four

Any content-based sizing is worked out based on these min and max content sizes.

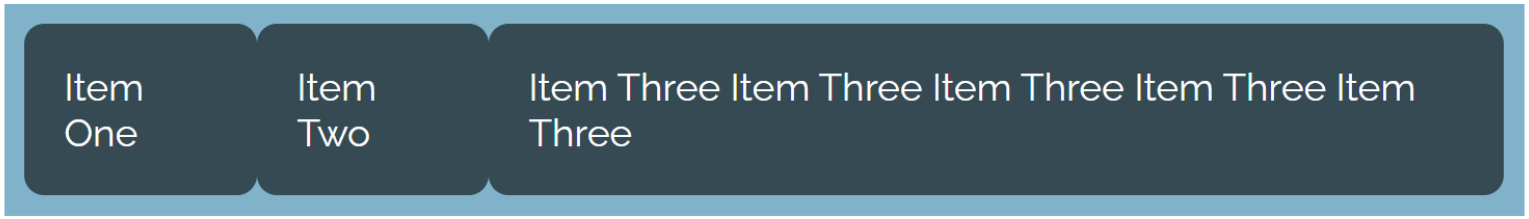



```
.example {
  display: flex;
}
```



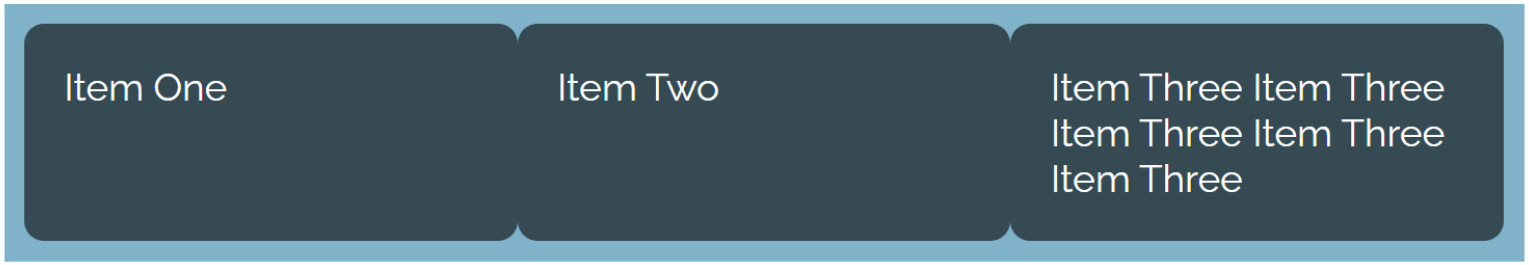

```
.example {
  display: flex;
}

.example > * {
  flex: auto;
}
```



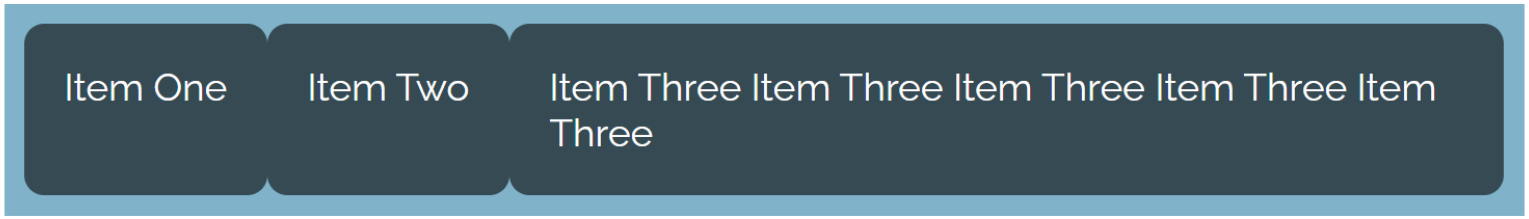

```
.example {
  display: flex;
}

.example > * {
  flex: auto;
}
```




```
.example {
  display: flex;
}

.example > * {
  flex: 1;
}
```




```
.example {
  display: grid;
  grid-template-columns: auto auto auto;
}
```

Do not be afraid of the specifications!

Interesting information lives there.

Researching a Property in the CSS Specifications

Rachel Andrew

14 December 2018

Published in Code

1 comment

I frequently joke that I'm "reading the specs so you don't have to", as I unpack some detail of a CSS spec in a post on my blog, some documentation for MDN, or an article on Smashing Magazine. However waiting for someone like me to write an article about something is a pretty slow way to get the information you need. Sometimes people like me get things wrong, or specifications change after we write a tutorial.

What if you could just look it up yourself? That's what you get when you learn to read the CSS specifications, and in this article my aim is to give you the basic details you need to grab quick information about any CSS property detailed in the CSS specs.

Where are the CSS Specifications?

The easiest way to see all of the CSS specs is to take a look at the Current Work page in the CSS section of the W3C Website. Here you can see all of the specifications listed, the level they are at and their status. There is also a link to the specification from this page. I explained CSS Levels in my article Why there is no CSS 4.

Thank you!

@rachelandrew