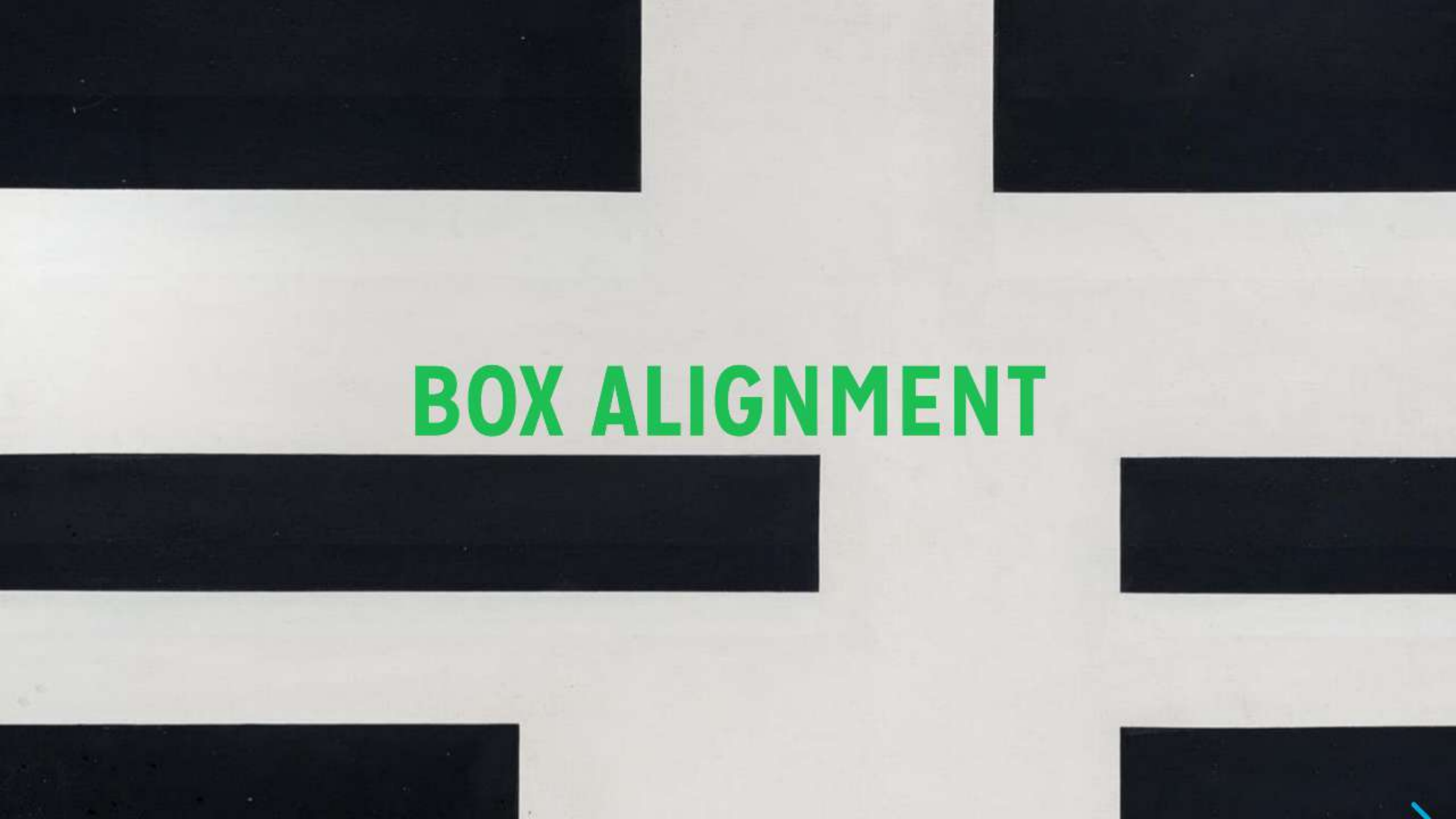
BOX ALIGNMENT



Surname First name

陈 慧 晶

Chen Hui Jing

@hj_chen

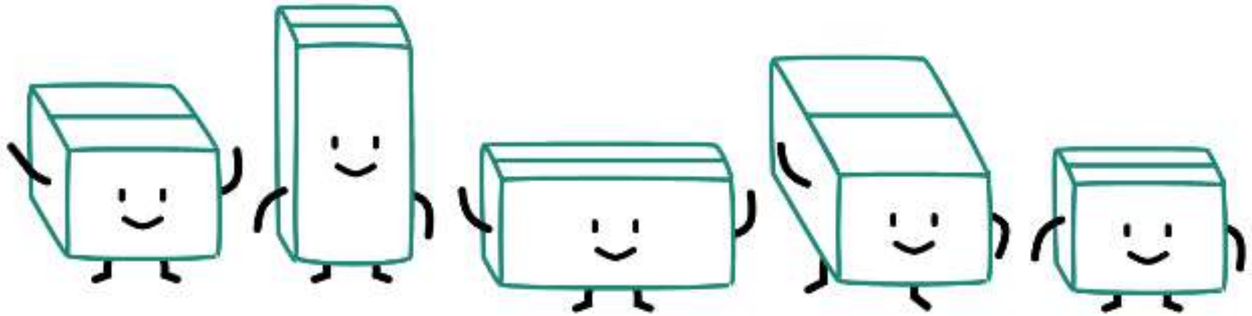
https://events.mozilla.org/techspeakers

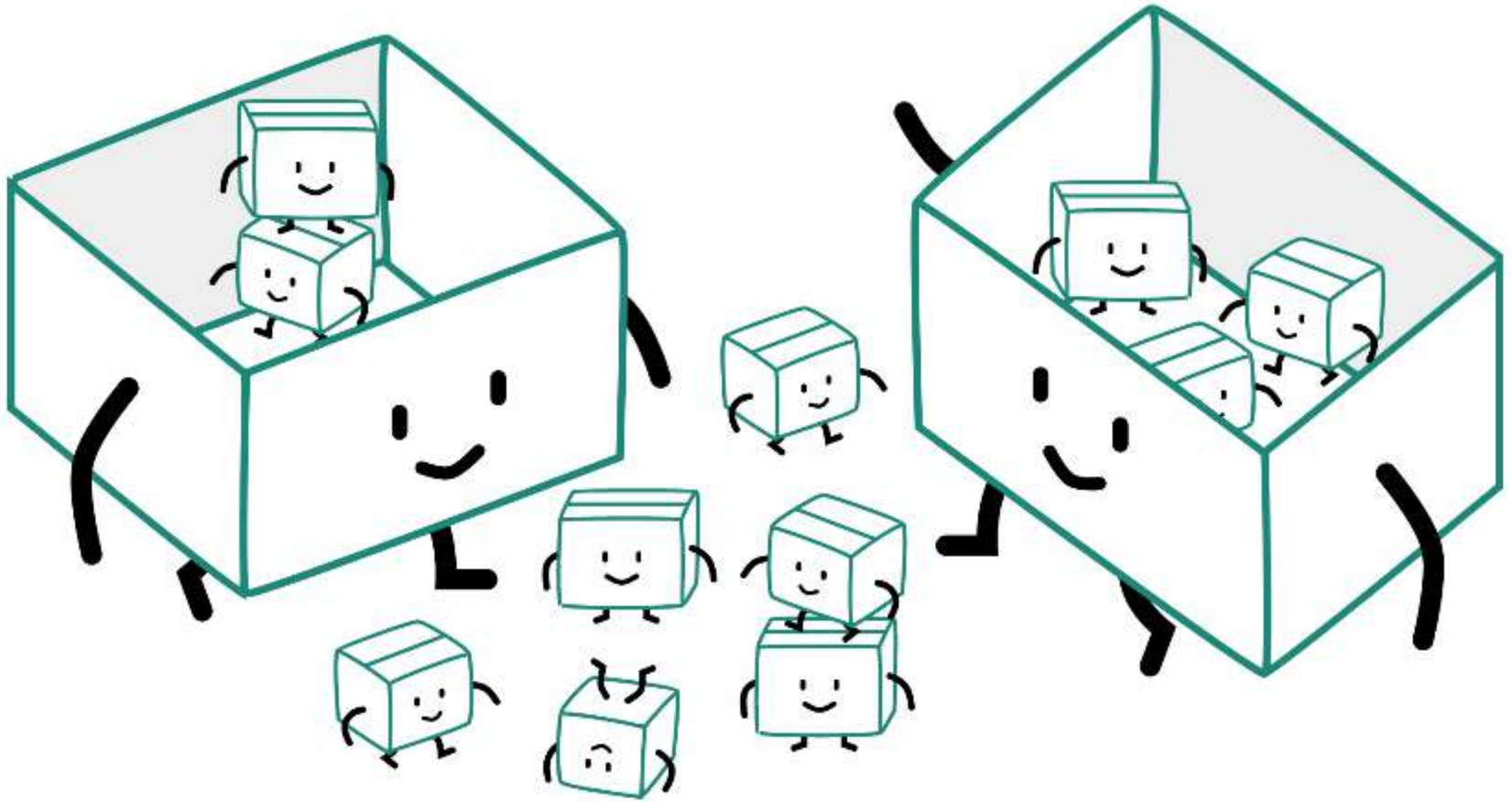
Developer Advocate

nexmo®

The Vonage® API Platform


“As CSS developers, we are programmers of boxes.”
—Lara Schenck
Lara Schenck @laras126
CSS
CSS





Where do boxes come from?

Cool colour font, Playbox by Matt Lyon

Evolution of CSS Specifications

CSS1
Recommendation:
17 Dec 1996

CSS2
Recommendation:
12 May 1998

CSS2.1
Recommendation:
7 Jun 2011

CSS2.2
Working draft:
12 Apr 2016

CSS3
Decision to modularise: 14 Apr 2000
(26 modules)

CSS Snapshot 2017
(88 modules)

Completed
- CSS Snapshot 2017
- CSS Snapshot 2015
- CSS Snapshot 2010
- CSS Snapshot 2007
- CSS Color Level 3
- CSS Namespaces
- Selectors Level 3
- CSS Level 2 Revision 1
- CSS Level 1
- CSS Print Profile
- Media Queries
- CSS Style Attributes

Stable
- CSS Backgrounds and Borders Level 3
- CSS Conditional Rules Level 3
- CSS Multi-column Layout Level 1
- CSS Values and Units Level 3
- CSS Cascading and Inheritance Level 3
- CSS Fonts Level 3
- CSS Writing Modes Level 3
- CSS Counter Styles Level 3

Rewriting
- CSS Basic Box Model Level 3
- CSS Generated Content Level 3

Refining
- CSS Animations
- Web Animations 1.0
- CSS Text Level 3
- CSS Transforms
- CSS Transitions
- CSS Box Alignment Level 3
- CSS Display Level 3
- Preview of CSS Level 2
- CSS Timing Functions Level 1

Testing
- CSS Image Values and Replaced Content Level 3
- CSS Speech
- CSS Flexible Box Layout Level 1
- CSS Text Decoration Level 3
- CSS Shapes Level 1
- CSS Masking Level 1
- CSS Fragmentation Level 3
- CSS Cascading Variables
- Compositing and Blending Level 1
- CSS Syntax Level 3
- CSS Grid Layout Level 1
- CSS Basic User Interface Level 3
- CSS Will Change Level 1
- Media Queries Level 4
- Geometry Interfaces Level 1
- CSS Cascading and Inheritance Level 4
- CSS Scroll Snap Level 1
- CSS Containment Level 1

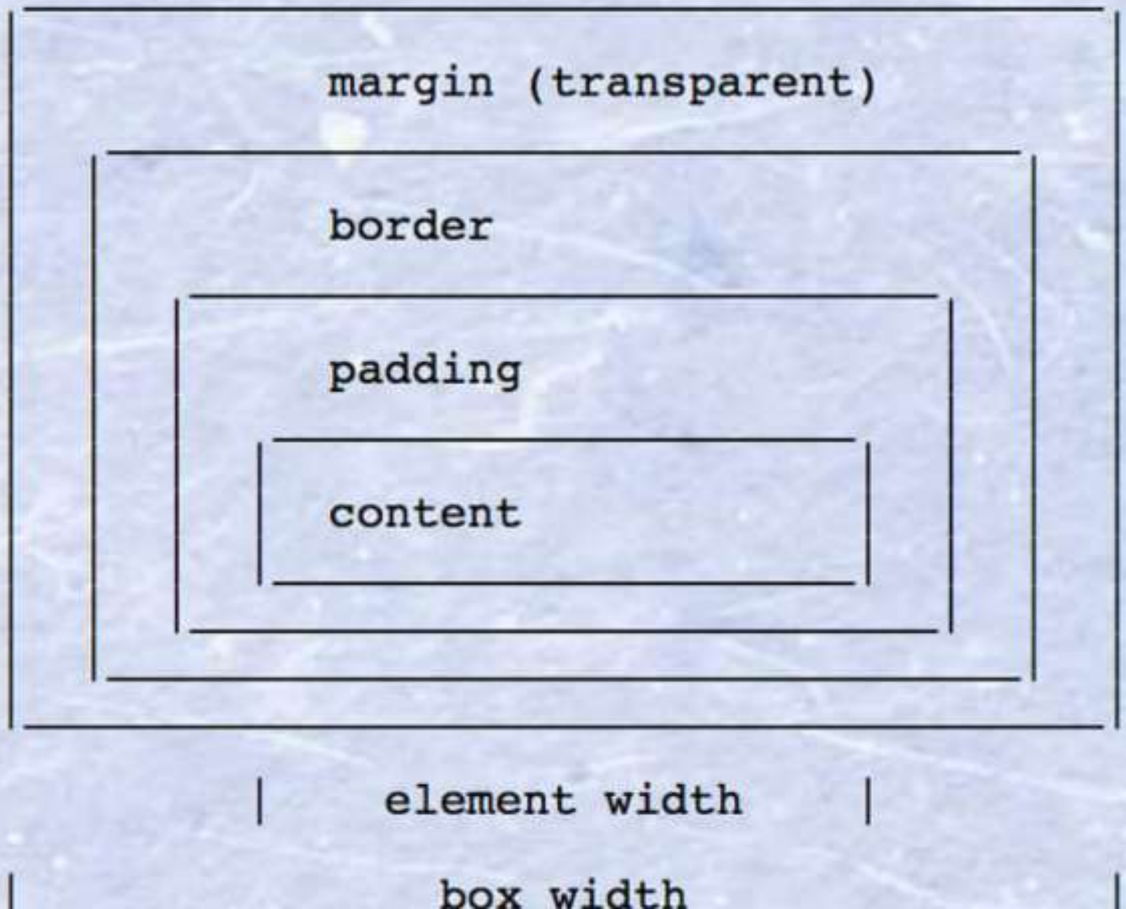
Exploring
- CSS Backgrounds and Borders Level 4
- CSS Device Adaptation
- CSS Exclusions
- Filter Effects
- CSS Generated Content for Paged Media
- CSS Page Floats
- CSS Template Layout
- CSS Line Grid
- CSS Lists Level 3
- CSS Positioned Layout Level 3
- CSS Regions
- CSS Table Level 3
- CSS Object Model
- CSS Font Loading
- CSS Scoping Level 1
- Non-element Selectors
- CSS Inline Layout Level 3
- Motion Path Level 1
- CSS Round Display Level 1
- CSS Basic User Interface Level 4
- CSS Text Level 4
- CSS Painting API Level 1
- CSS Properties and Values API Level 1
- CSS Typed OM Level 1
- Worklets Level 1
- CSS Color Level 4
- CSS Fonts Level 4
- CSS Rhythmic Sizing Level 1
- CSS Image Values and Replaced Content Level 4
- CSS Fill and Stroke Level 3
- CSS Logical Properties and Values Level 1
- CSS Overflow Level 4

“If we hadn’t developed CSS, we could have ended up with the web being a giant fax machine”

—Håkon Wium Lie

4 Formatting model

CSS1 assumes **a simple box-oriented formatting model** where each element results in **one or more rectangular boxes.** (Elements that have a 'display' value of 'none' are not formatted and will therefore not result in a box.) All boxes have a core content area with optional surrounding padding, border and margin areas.



- per pixel control: CSS1 values simplicity over level of control, and although the combination of background images and styled HTML is powerful, control to the pixel level is not possible.
- author control: the author cannot enforce the use of a certain sheet, only suggest
- a layout language: CSS1 does not offer multiple columns with text-flow, overlapping frames etc.
- a rich query language on the parse tree: CSS1 can only look for ancestor elements in the parse tree, while other style sheet languages (e.g. DSSSL [6]) offers a full query language.

We expect to see extensions of CSS in several directions:

- paper: better support for printing HTML documents
- support for non-visual media: work is in the process to add a list of properties and corresponding values to support speech and braille output
- color names: the currently supported list may be extended
- fonts: more precise font specification systems are expected to complement existing CSS1 font properties.
- values, properties: we expect vendors to propose extensions to the CSS1 set of values and properties. Extending in this direction is trivial for the specification, but interoperability between different UAs is a concern
- layout language: support for two-dimensional layout in the tradition of desktop publishing packages.
- other DTDs: CSS1 has some HTML-specific parts (e.g. the special status of the 'CLASS' and 'ID' attributes) but should easily be extended to apply to other DTDs as well.

We do not expect CSS to evolve into:

- a programming language

9 Visual formatting model

9.1 Introduction to the visual formatting model

This chapter and the next describe the visual formatting model: how user agents process the document tree for visual media.

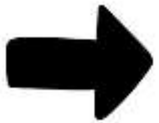
In the visual formatting model, each element in the document tree generates zero or more boxes according to the box model. The layout of these boxes is governed by:

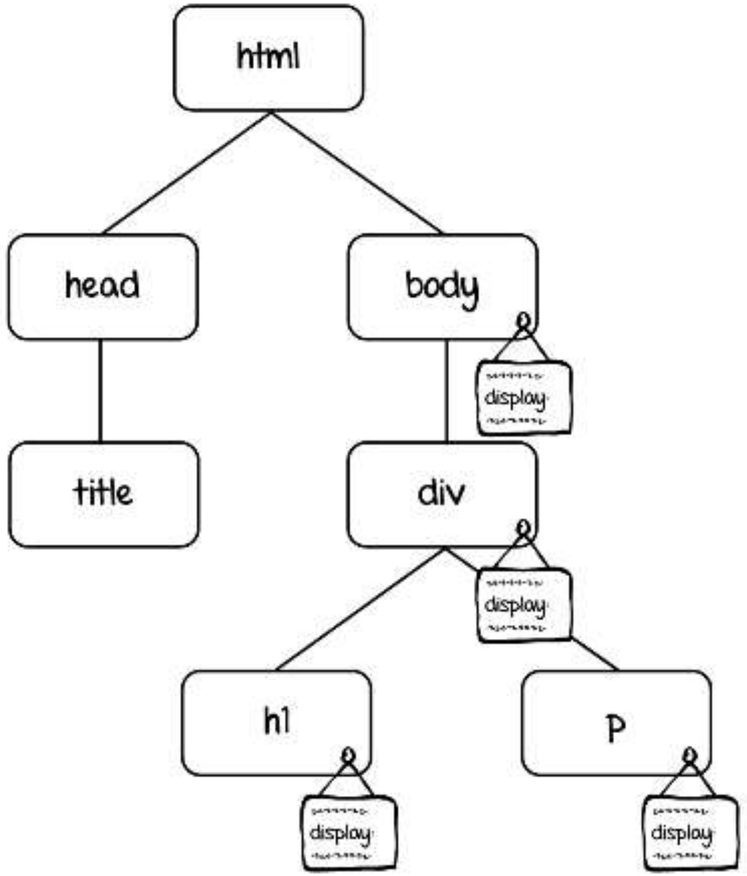
- box dimensions and type.
- positioning scheme (normal flow, float, and absolute positioning).
- relationships between elements in the document tree.
- external information (e.g., viewport size, intrinsic dimensions of images, etc.).

The properties defined in this chapter and the next apply to both continuous media and paged media. However, the meanings of the margin properties vary when applied to paged media (see the page model for details).

Wait, what?

```
<!doctype html>
<html lang="en">
  <head>
    <title>CSS rocks</title>
  </head>
  <body>
    <div>
      <h1>Title</h1>
      <p>Flying rabbits, whoa</p>
    </div>
    ...
    ...
  </body>
</html>
```





📦 Box dimensions and type

📦 Positioning scheme
(normal flow / float / absolute positioning)

Layout of boxes

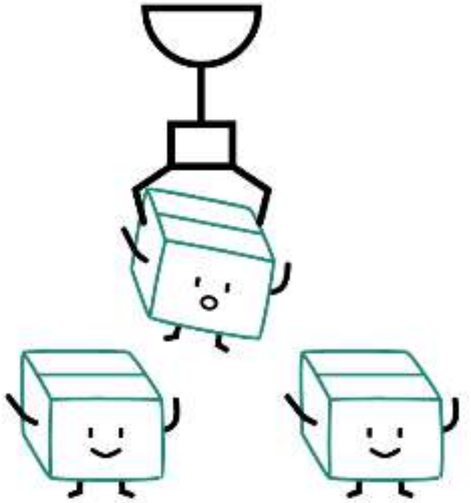
📦 Relationships between elements in the document tree

📦 External information
(e.g. viewport size, intrinsic dimensions of images etc.)

Positioning schemes

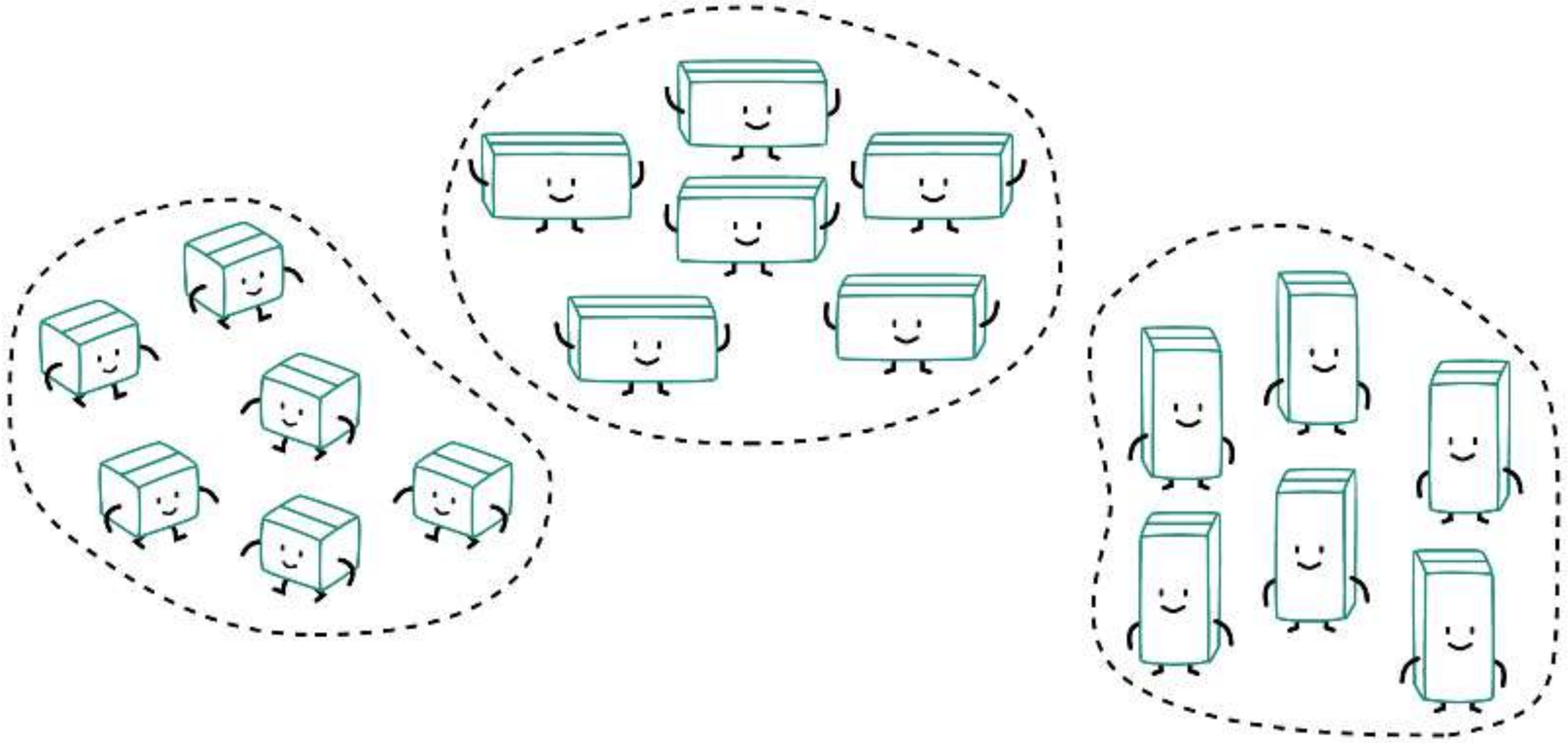
Normal flow

Floats



Absolute positioning

What is a formatting context?



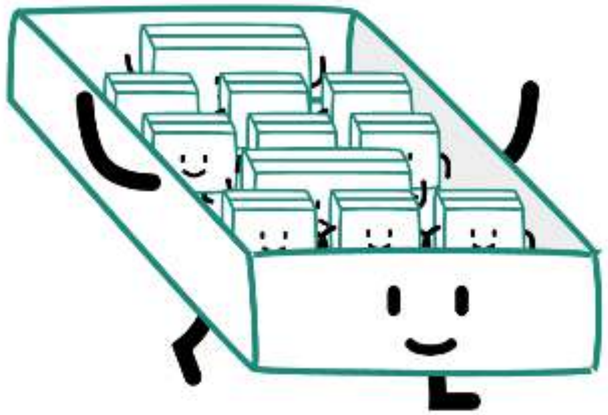
The `display` property

Defines an element's **display type**,which consists of the two basic qualities of how an element generates boxes

CSS Display Module Level 3

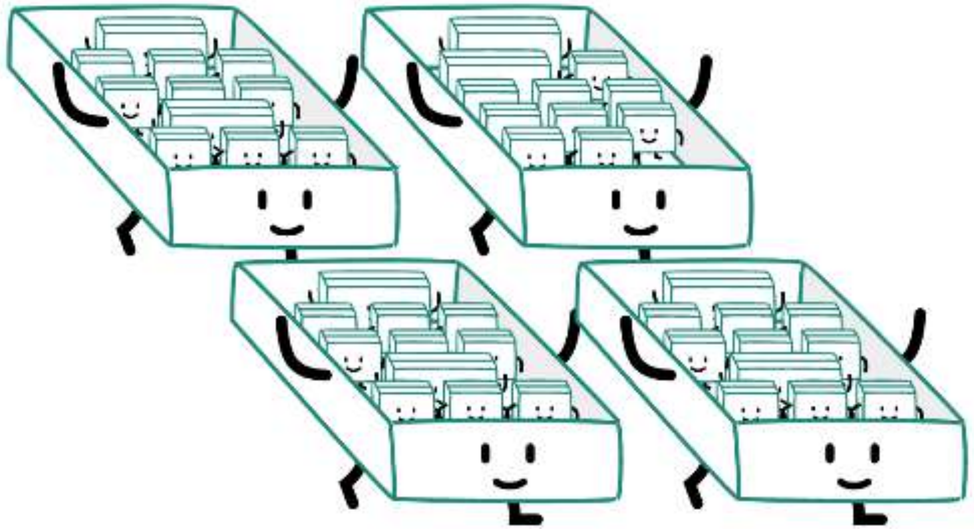
Inner display type

Defines the generated formatting context for descendant boxes



Outer display type

Dictates a principal box's own participation in flow layout



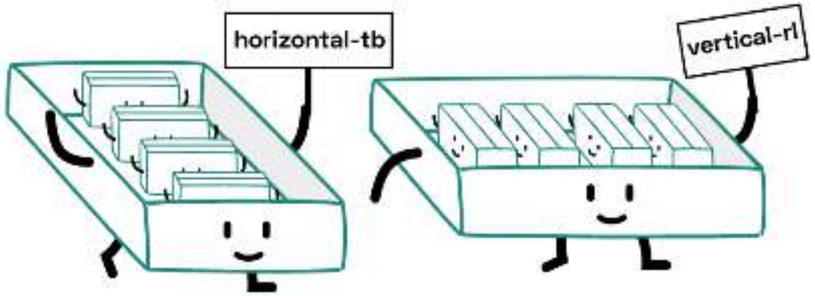
Short `display`	Full `display`	Generated box
none	–	subtree omitted from box tree
contents	–	element replaced by content in box tree
ruby	inline ruby	inline-level ruby container
block ruby	block ruby	block box containing ruby container
table	block table	block-level table wrapper box containing table box
inline-table	inline table	inline-level table wrapper box containing table box
list-item	block flow list-item	block box with additional marker box
inline list-item	inline flow list-item	inline box with additional marker box
run-in	run-in flow	run-in box (inline box with special box-tree-munging rules)

Short `display`	Full `display`	Generated box
flex	block flex	block-level flex container
inline-flex	inline flex	inline-level flex container
grid	block grid	block-level grid container
inline-grid	inline grid	inline-level grid container
block	block flow	block-level block container
flow-root	block **flow-root**	block-level block container that establishes a new block formatting context (BFC)
inline	inline flow	inline box
inline-block	inline **flow-root**	inline-level block container

Block formatting context

The context that **block-level** boxes participate in

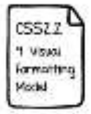
Boxes are laid out one after another, in the block flow direction, from the start of the containing block



Margins along the block flow direction between **adjacent block-level** boxes in the **same** block formatting context collapse

Who establishes new block formatting contexts?

- Floats
- Absolutely positioned elements
- Block containers that are **not** block boxes
- Block boxes with `overflow` **other than** `visible`
- Boxes with `display` set to `flow-root`



We need a new BFC because...?

1. Prevent collapsing margins

This is a line of text in a p tag.

I'm a box with margins.

I'm another box with margins.

```
<p>This is a line of text in a p tag.</p>
<div class="block-wrapper">
  <div class="box1">I'm a box with margins.</div>
  <div class="box2">I'm another box with margins.
</div>
```

```
.collapse .block-wrapper {
  overflow: auto;
}

.collapse .box1 {
  margin: 0.5em;
}
```

2. Stop text from flowing around the float

I'm a floated box!

This is just a bunch of text that is going on and on so it's long enough to wrap around the float, line boxes yo!

```
<div class="block-wrapper">
  <div class="box1">I'm a floated box!</div>
  <p class="box2">This is just a bunch of text th
</div>
```

```
.stop-flow .box1 {
  float: left;
}

.stop-flow .box2 {
  overflow: auto;
}
```

3. Contains floats

Floaty! ^_^ Floaty too! :)

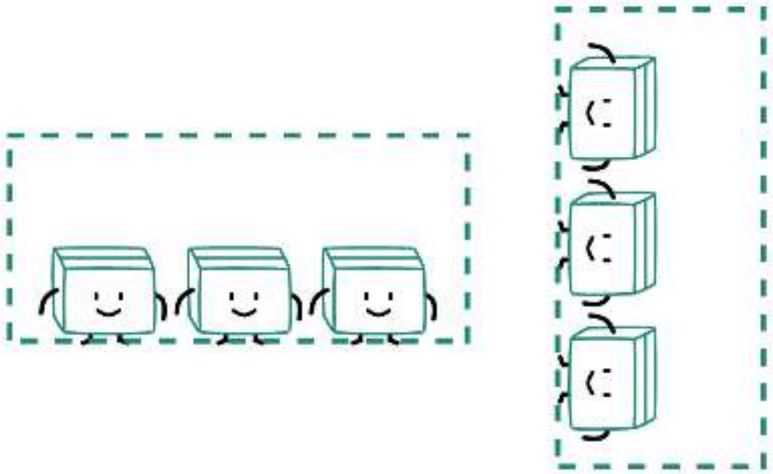
```
<div class="block-wrapper">
  <p class="box1">Floaty! ^_^</p>
  <p class="box2">Floaty too! :)</p>
</div>
```

```
.contain .block-wrapper {
  border: 3px solid indigo;
  display: flow-root;
}

.contain .box1 {
  float: left;
}
```

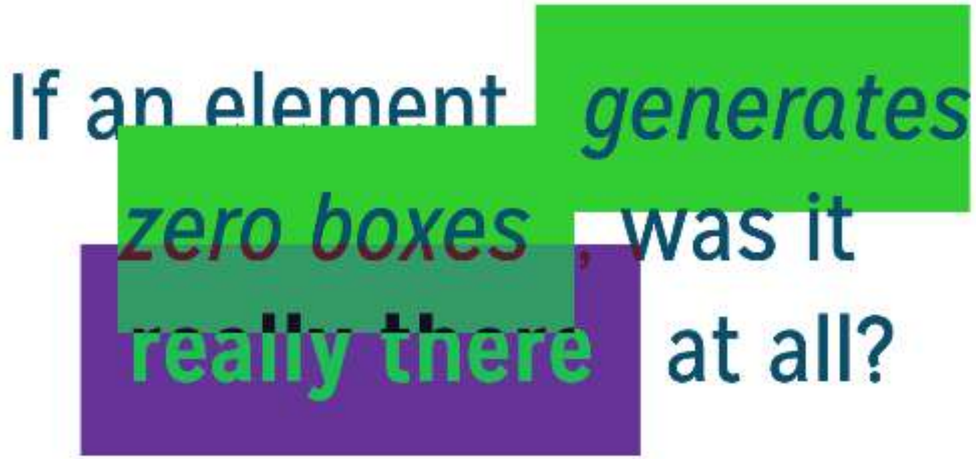
Inline formatting contexts

Established by a block container box that contains **no block-level** boxes

Boxes are laid out one after another, in the inline direction, from the start of the containing block



Inline box construction

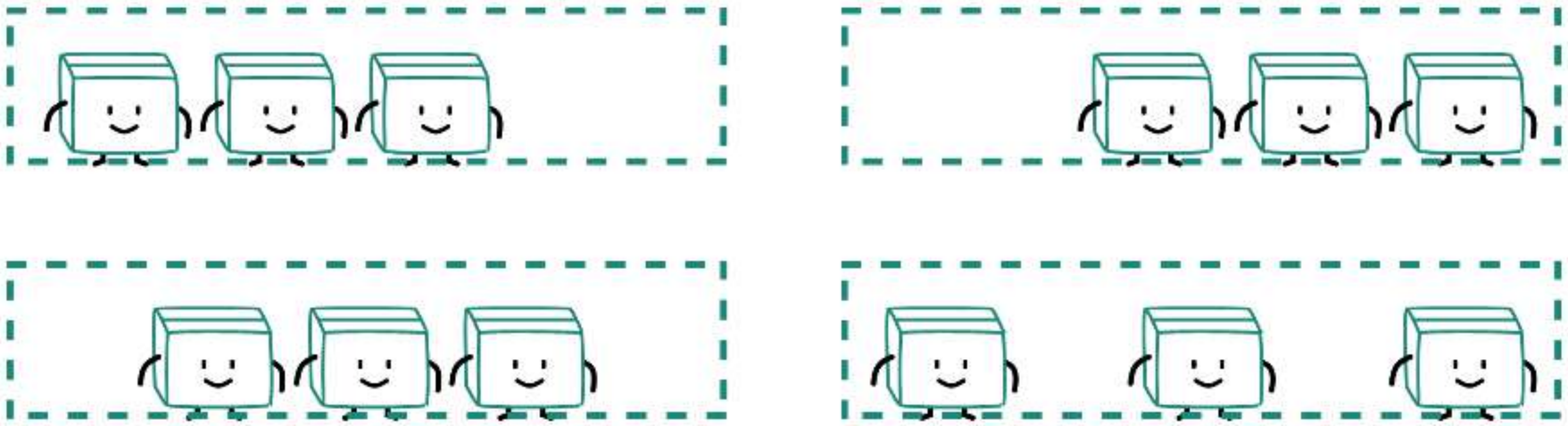



```
<p class="line-container">If an element
<em>generates zero boxes</em>, was it
<strong>really there</strong> at all?</p>
```

```
  padding: 0.5em;
  background-color: limegreen;
}

.linebox .line-container strong {
  padding: 0.5em;
  background-color: rebeccapurple;
  mix-blend-mode: difference;
}
```

Alignment along the inline-axis

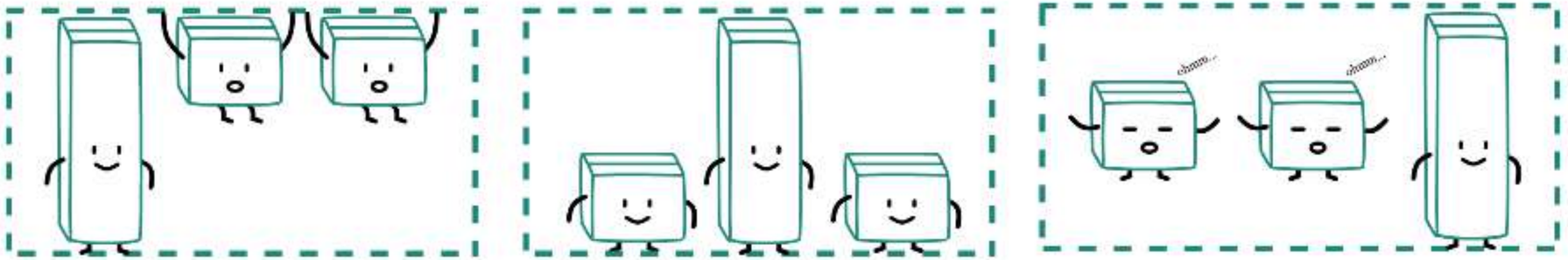
The `text-align` property aligns inline boxes along the inline-axis



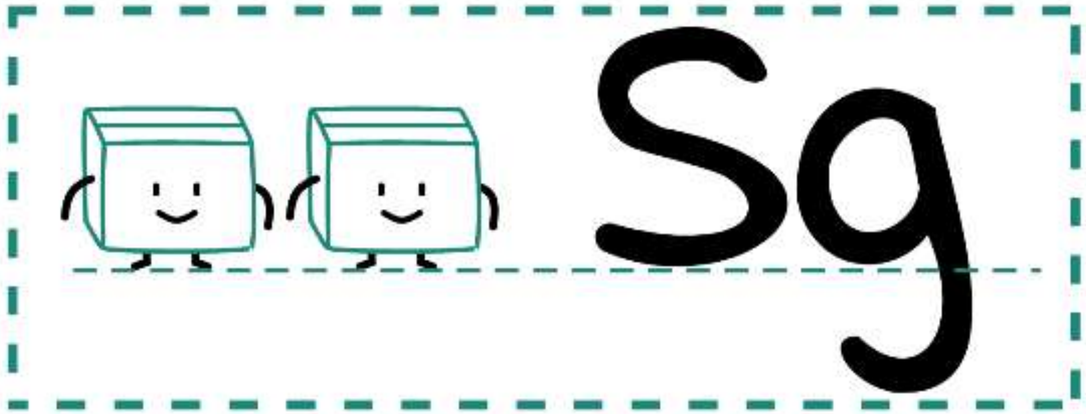
Applicable only when there is extra space available in the **line box**

Alignment along the block-axis

Boxes may be aligned along the block-axis in different ways, with the `vertical-align` property



The height of the line box is based on its font, and its line-height



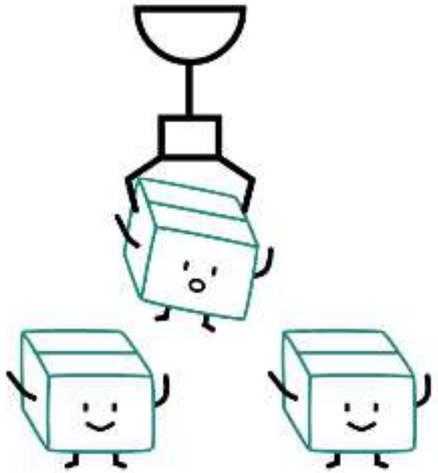
Explaining the inline-block centring technique


```
.centring .wrapper {
  height: 100%;
}

.centring .wrapper::after {
  content: '';
  display: inline-block;
  height: 100%;
  vertical-align: middle;
  background-color: limegreen;
  width: 6px;
}
```

Centering in the Unknown by Chris Coyier

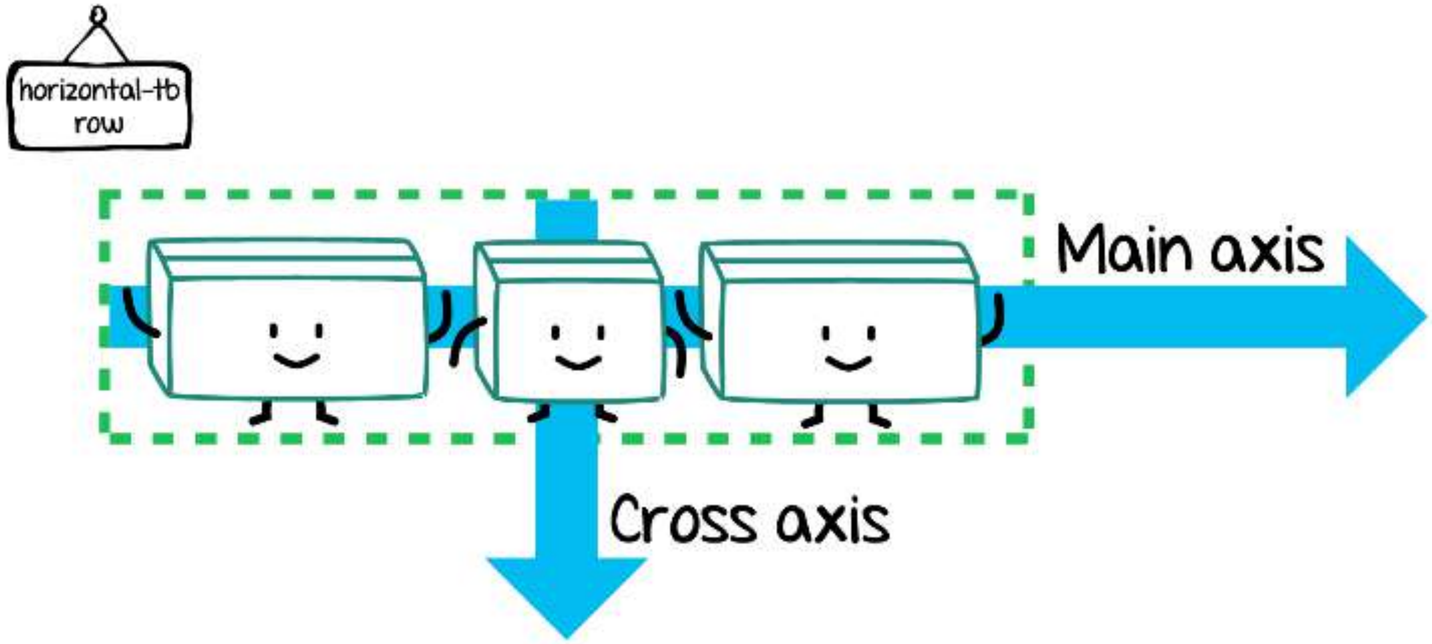
Flex formatting context

Established by a block-level or inline-level **flex** container box

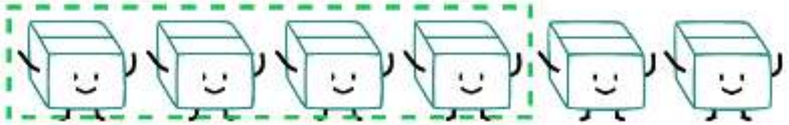


CSS
Flexbox
Box Layout
Module
Level 1

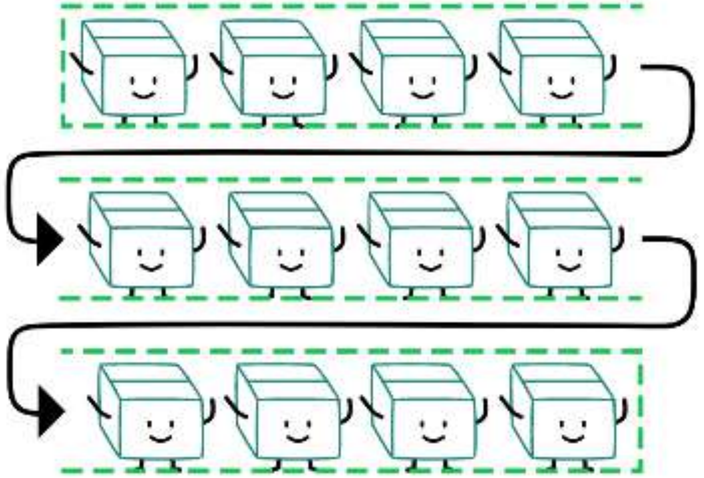
Flex axes



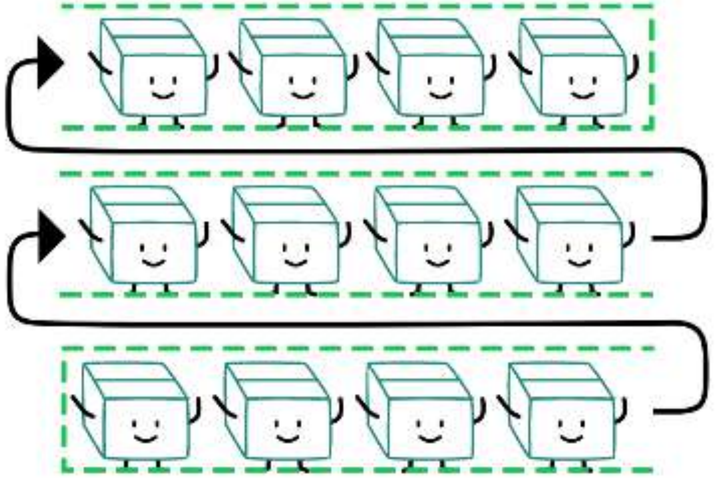
Flex lines



nowrap



wrap



wrap-reverse

Flex directions

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

```
.directions .wrapper {
  display: flex;
  flex-wrap: wrap;
  writing-mode: horizontal-tb;
  flex-direction: row;
  width: 100%;
  height: 100%;
}

.directions .box {
  height: 4em;
  width: 4em;
```

CSS Flexbox Box Layout Module Level 1

All the directions

- LTR
- RTL

- horizontal-tb
- vertical-rl
- vertical-lr
- sideways-lr

- wrap
- wrap-reverse

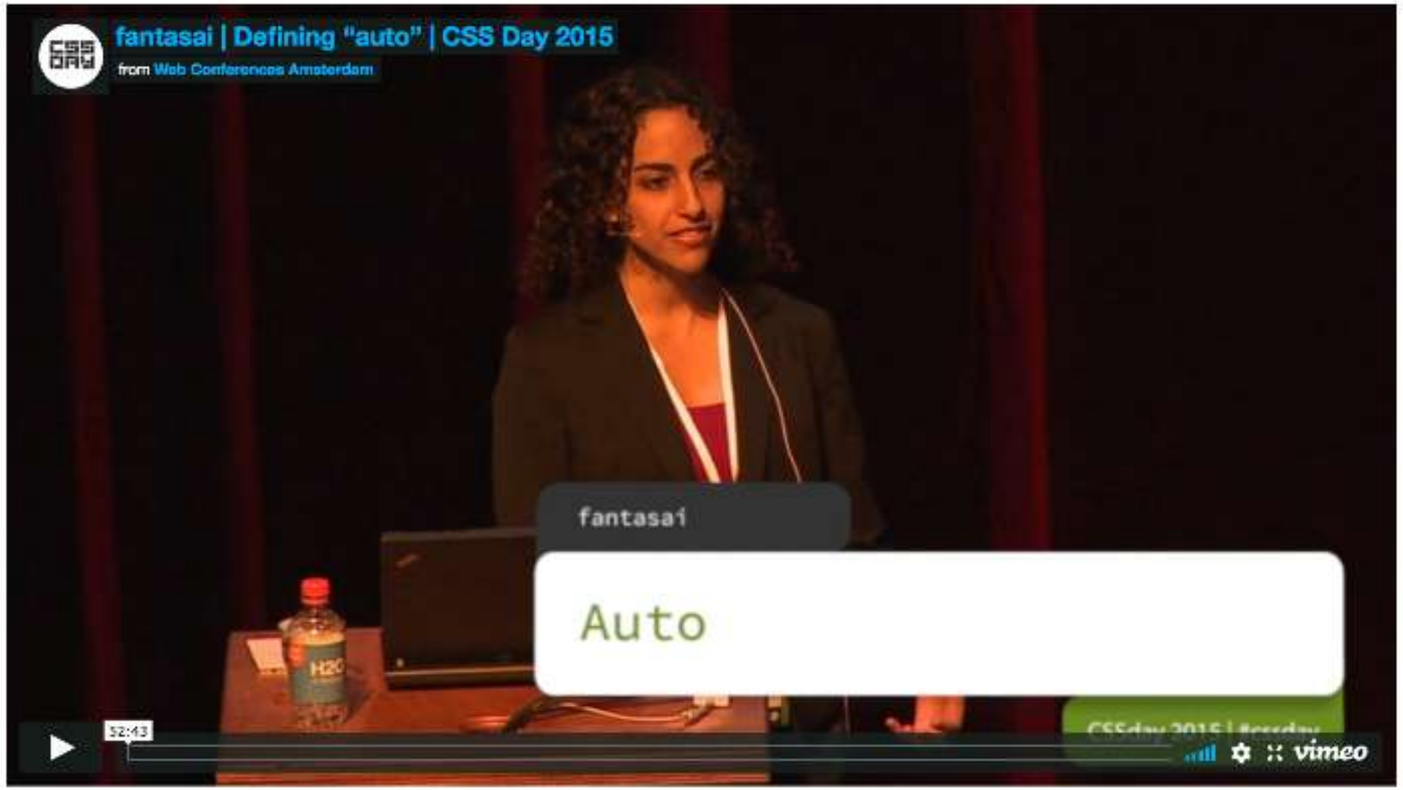
- row
- row-reverse
- column
- column-reverse

2 * 4 * 2 * 4 = 

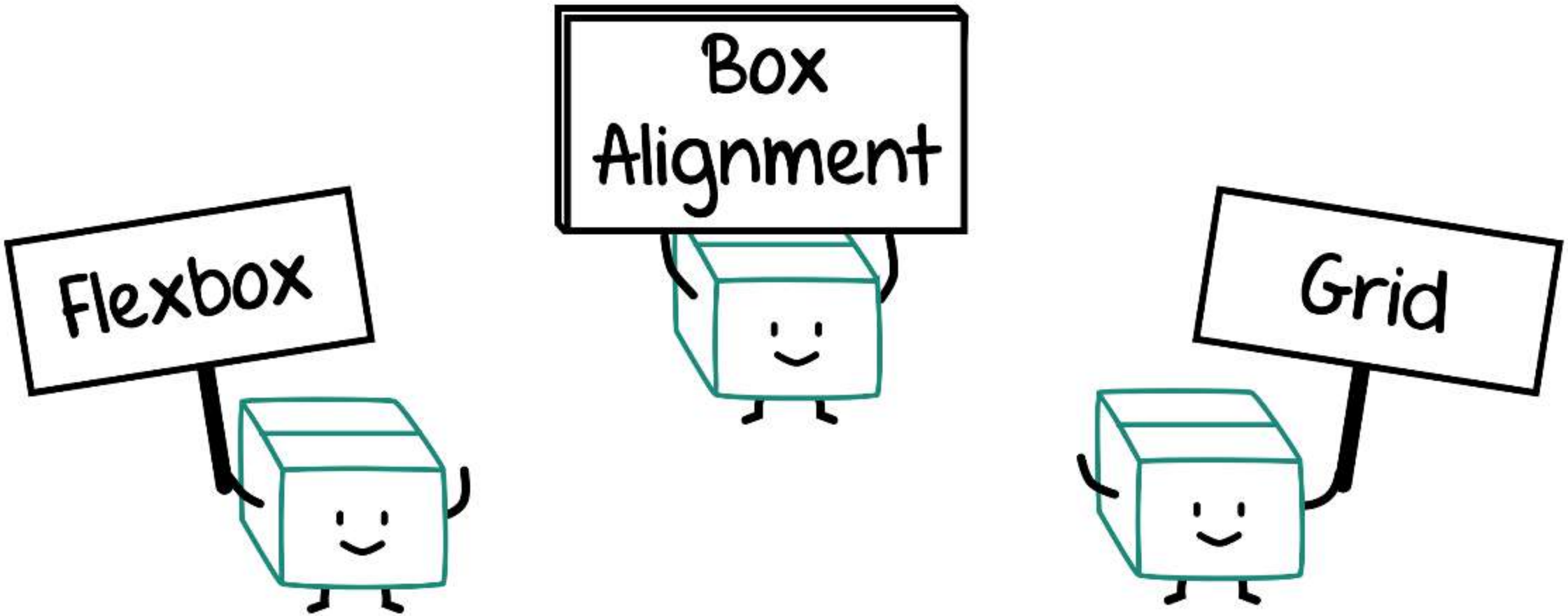
Aligning with `auto` margins

```
.automargin{
  display: flex;
}

.automargin div {
  border: 1px solid;
  margin: auto;
}
```

Defining “auto” by Elika Etemad (AKA fantasai)



fantasai | Defining “auto” | CSS Day 2015 from Web Conferences Amsterdam on Vimeo.


Box
Alignment
Flexbox
Grid

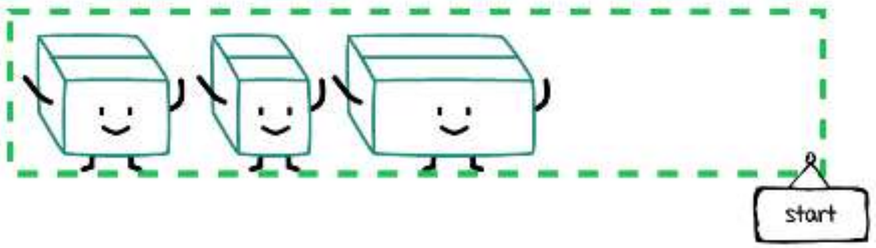
Aligning along the main axis

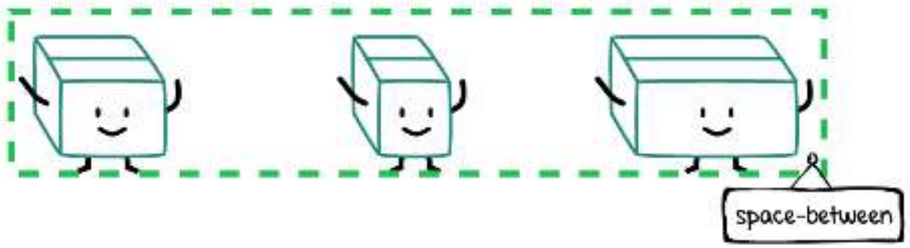
`justify-content` helps distribute extra free space left over **after** flexible lengths and auto margins are resolved.

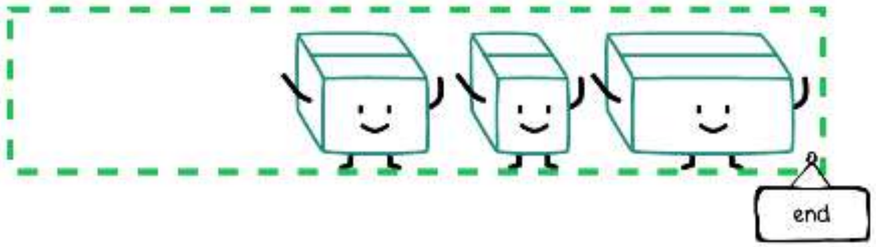
1一	2二	3三	4四	5五	6六
7七	8八	9九	10十	11十一	12十二
13十三	14十四	15十五	16十六	17十七	18十八
19十四	20二十				

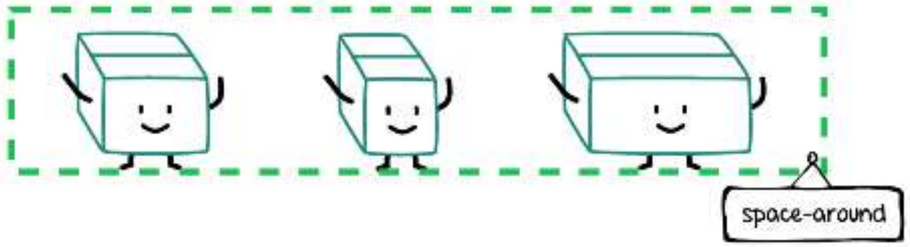
```
.mainaxis .wrapper {
  display: flex;
  flex-wrap: wrap;
  justify-content: flex-start;
}

.mainaxis .box {
  height: 5em;
  width: 5em;
  border: 1px solid;
}
```

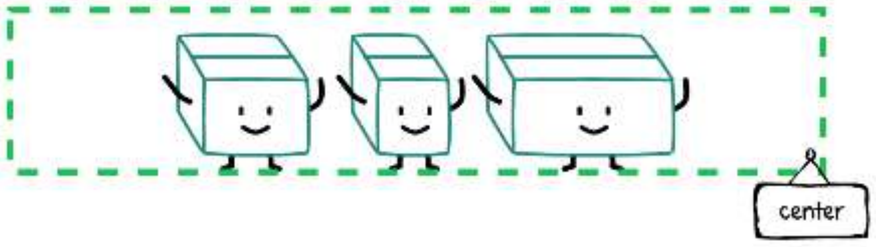
CSS
Flexbox
Box Layout
Module
Level 1

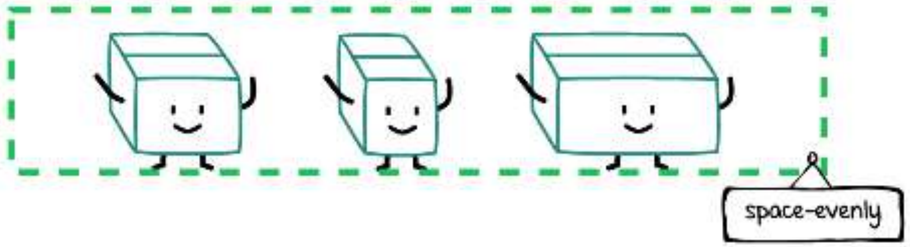








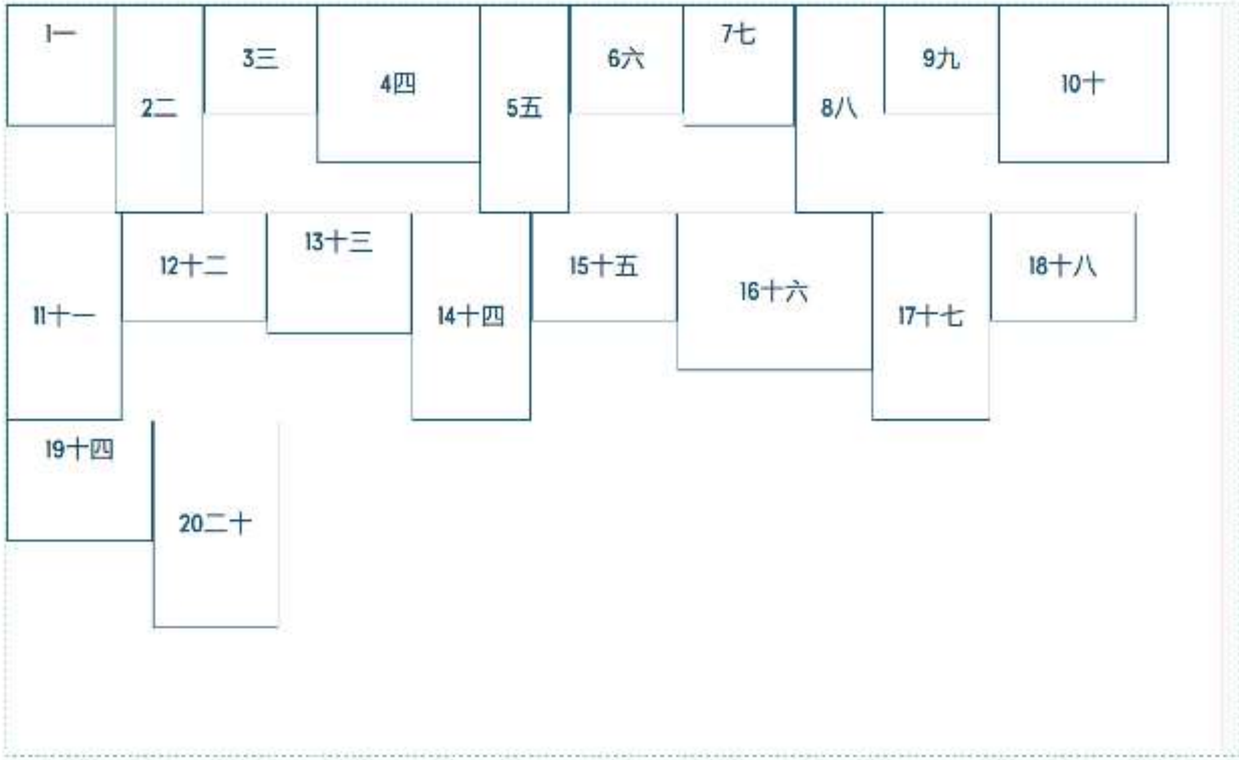




CSS Flexible Box Layout Module Level 1

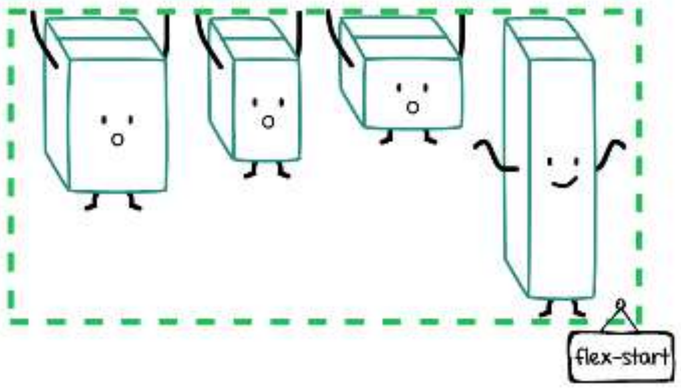
Aligning along the cross axis

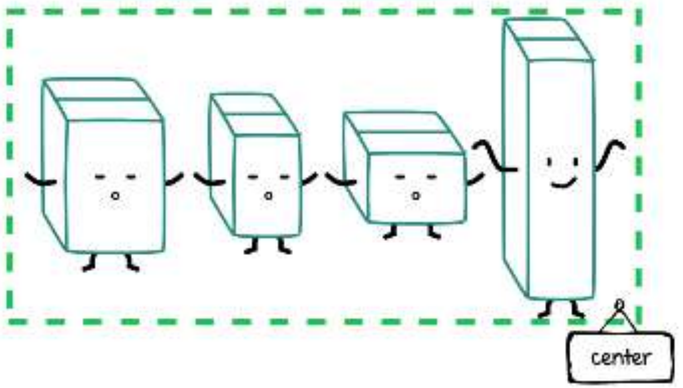
`align-items` sets the default alignment for all flex items along the cross axis of the flex line.
Over-ridable by `align-self`.

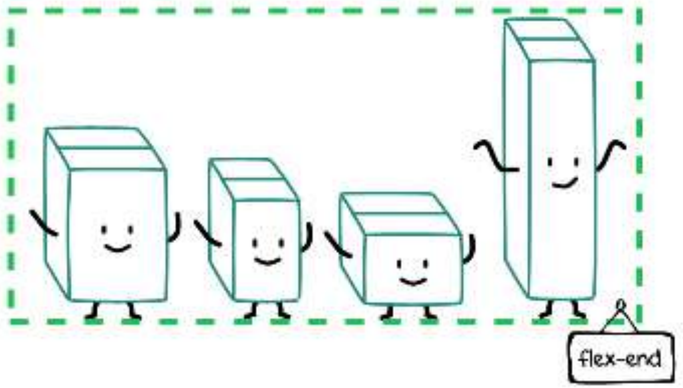


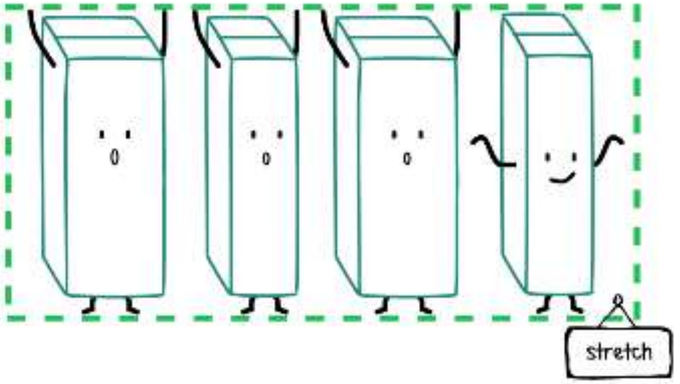

```
.crossaxis .wrapper {
  display: flex;
  flex-wrap: wrap;
  align-items: flex-start;
}

.crossaxis .box:nth-child(2n+1) {
  padding: 0.5em 1.5em 3em;
  align-self: ;
}
.crossaxis .box { border: 1px solid }
.crossaxis .box:nth-child(2n) { padding:
```

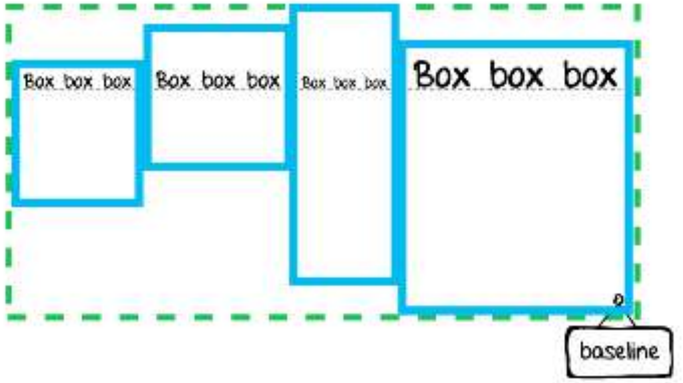
CSS
Flexbox
Box Layout
Module
Level 1











Packing flex lines

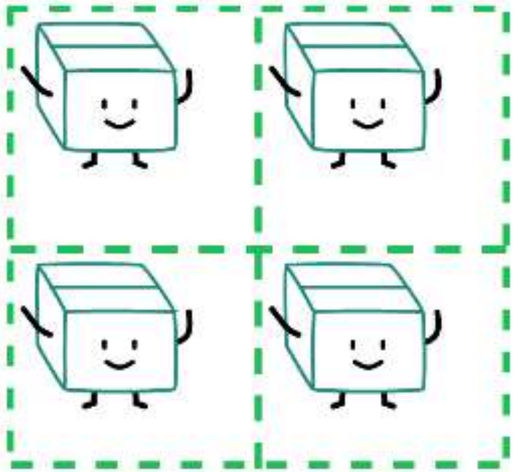
`align-content` aligns flex lines within the flex container if there is extra space along the **cross-axis**.


```
.packaxis .wrapper {
  display: flex;
  flex-wrap: wrap;
  align-content: stretch;
}

.packaxis .box {
  height: 5em;
  width: 5em;
  border: 1px solid;
}
```

CSS Grid Layout Module Level 1

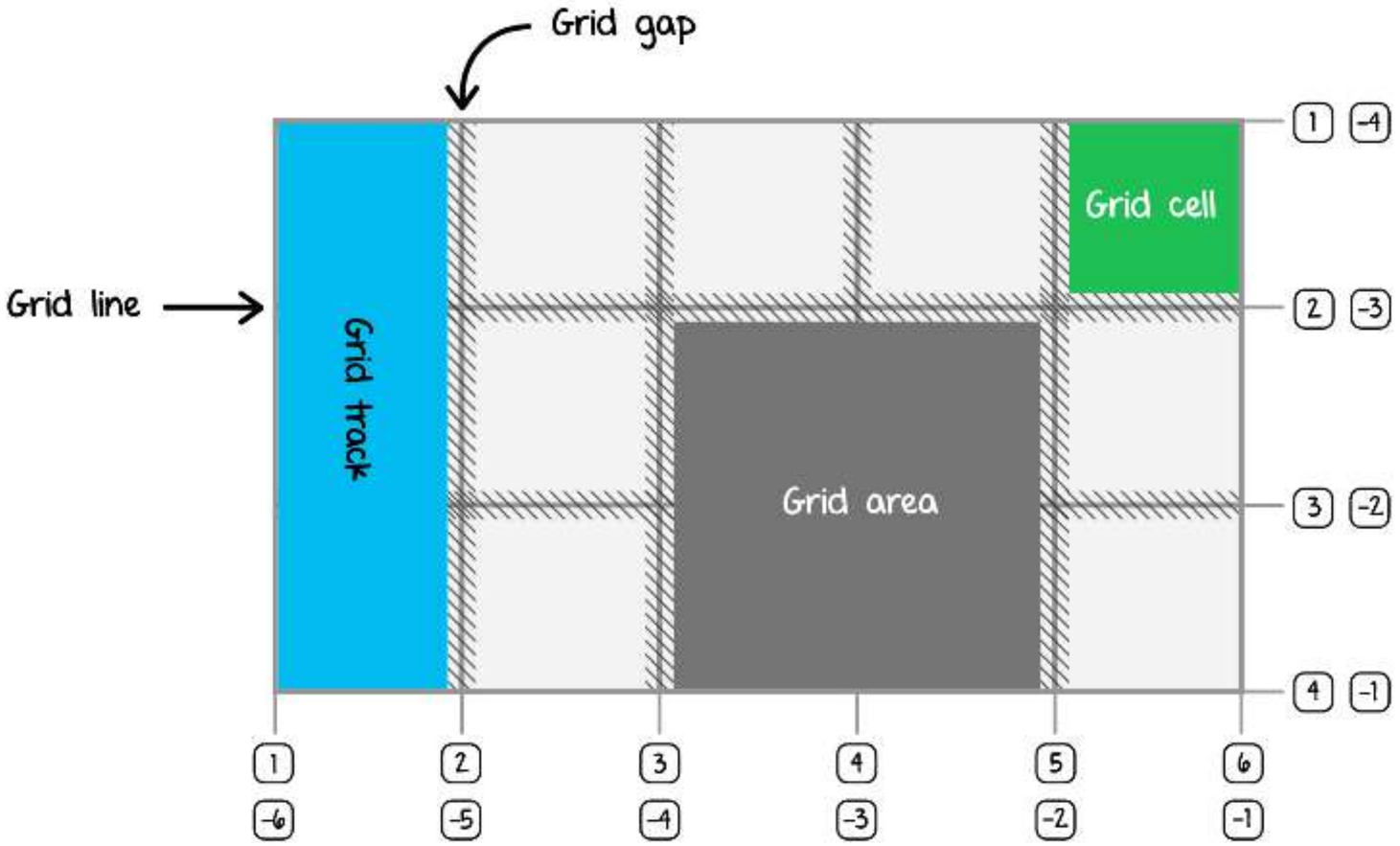
Grid formatting context

Established by a block-level or inline-level **grid** container box



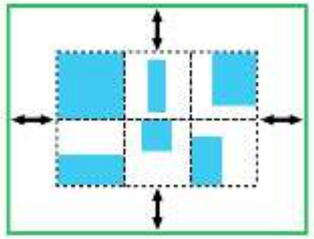
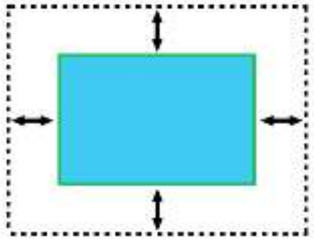
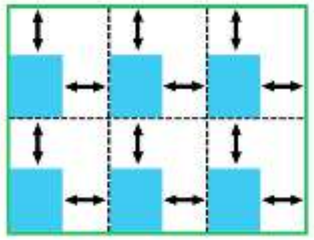
CSS Grid Layout Module Level 1

Grid terminology

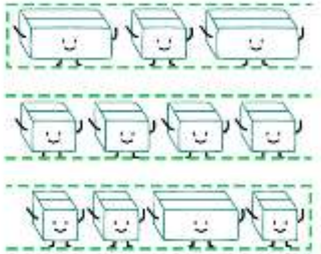


CSS Box Alignment Module Level 3

Box alignment properties

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

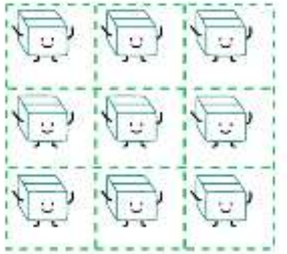
Flexbox



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

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

Grid

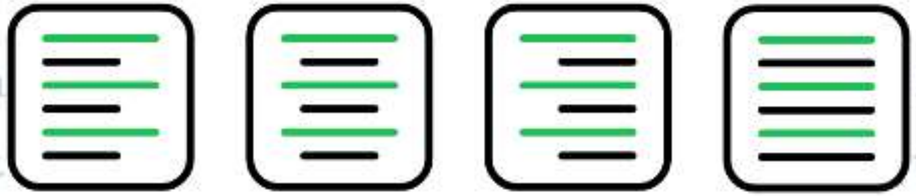


- `align-content`
- `justify-content`
- `align-items`
- `align-self`
- `justify-items`
- `justify-self`

Flexbox

- align-conten
- justify-cont
- align-items
- align-self

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

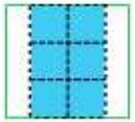
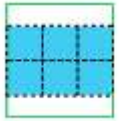
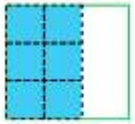
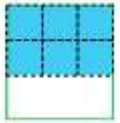
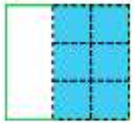
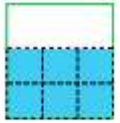
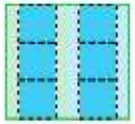
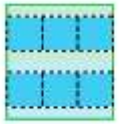
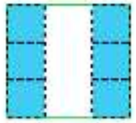
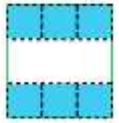
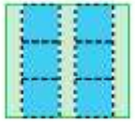
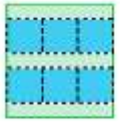


Justify

Grid

- lign-content
- ustify-content
- align-items
- align-self
- justify-items
- justify-self

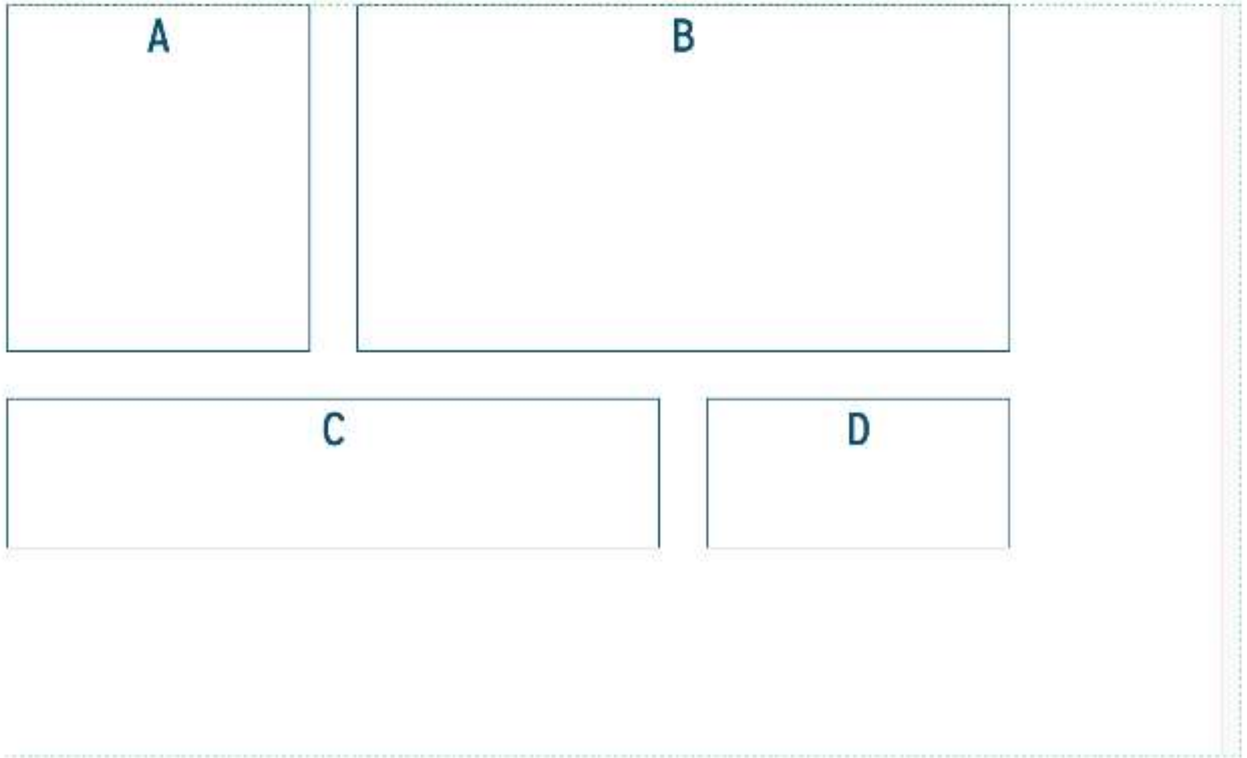
CSS Box Alignment Module Level 3

Values	justify-content	align-content
center		
start		
end		
space-around		
space-between		
space-evenly		

CSS Box Alignment Module Level 3

justify/align-content

content-distribution properties

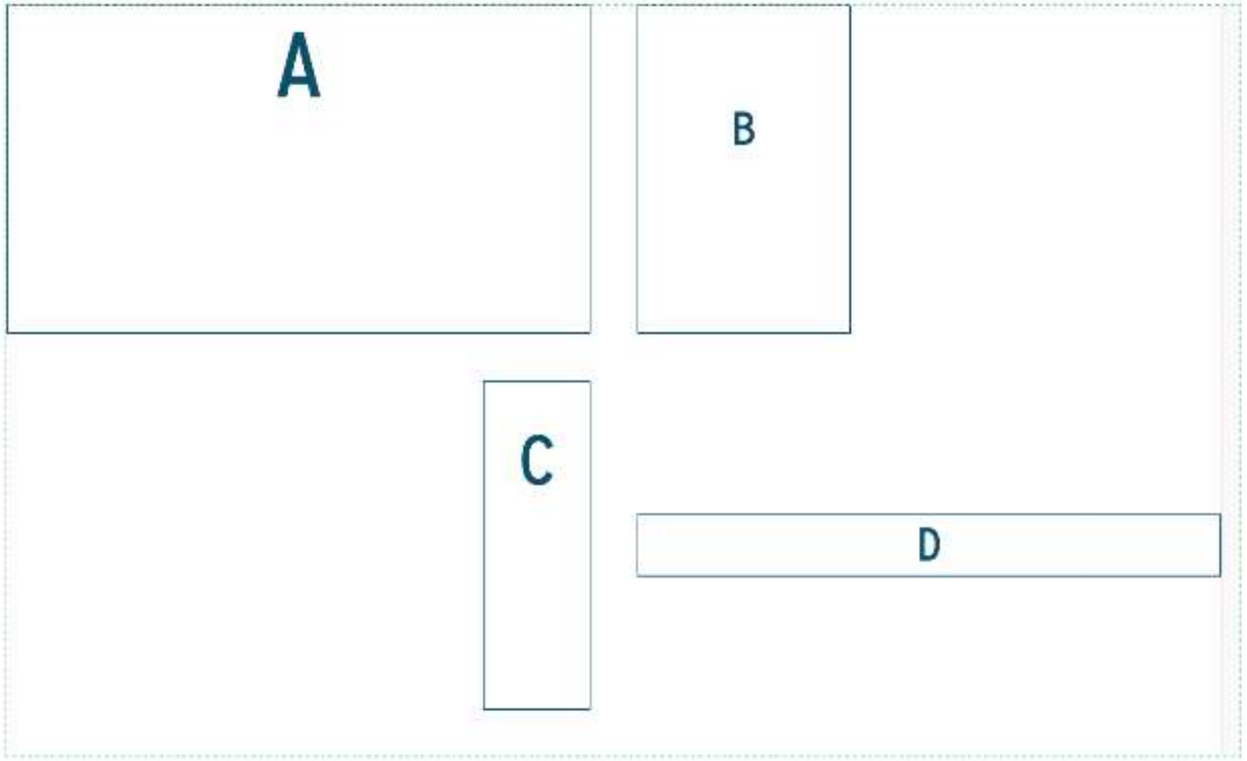



```
.content {
  justify-content: normal;
  align-content: normal;

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

CSS Box Alignment Module Level 3

justify/align-self

self-alignment properties

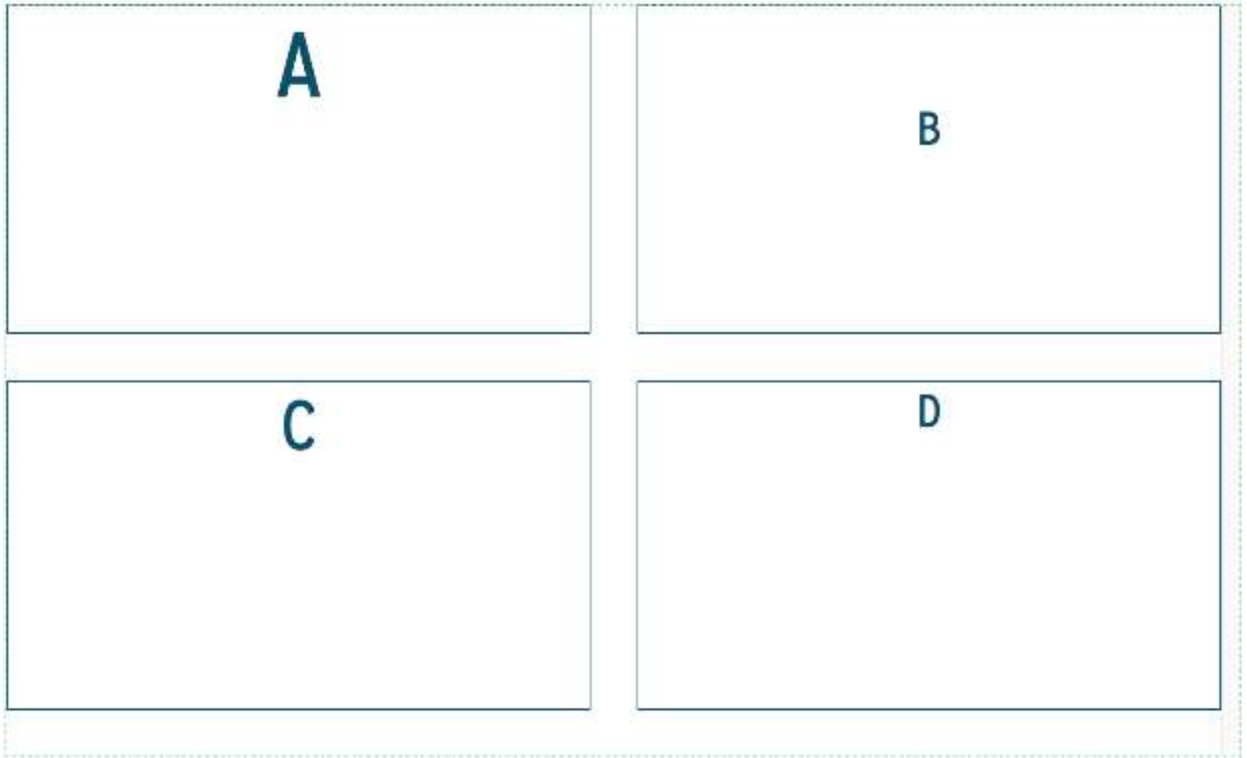



```
.self_itemA {
  grid-area: a;
}

.self_itemB {
  grid-area: b;
  //align-self: start;
  justify-self: start;
}

.self_itemC {
  grid-area: c;
```

CSS Box Alignment Module Level 3

justify/align-items

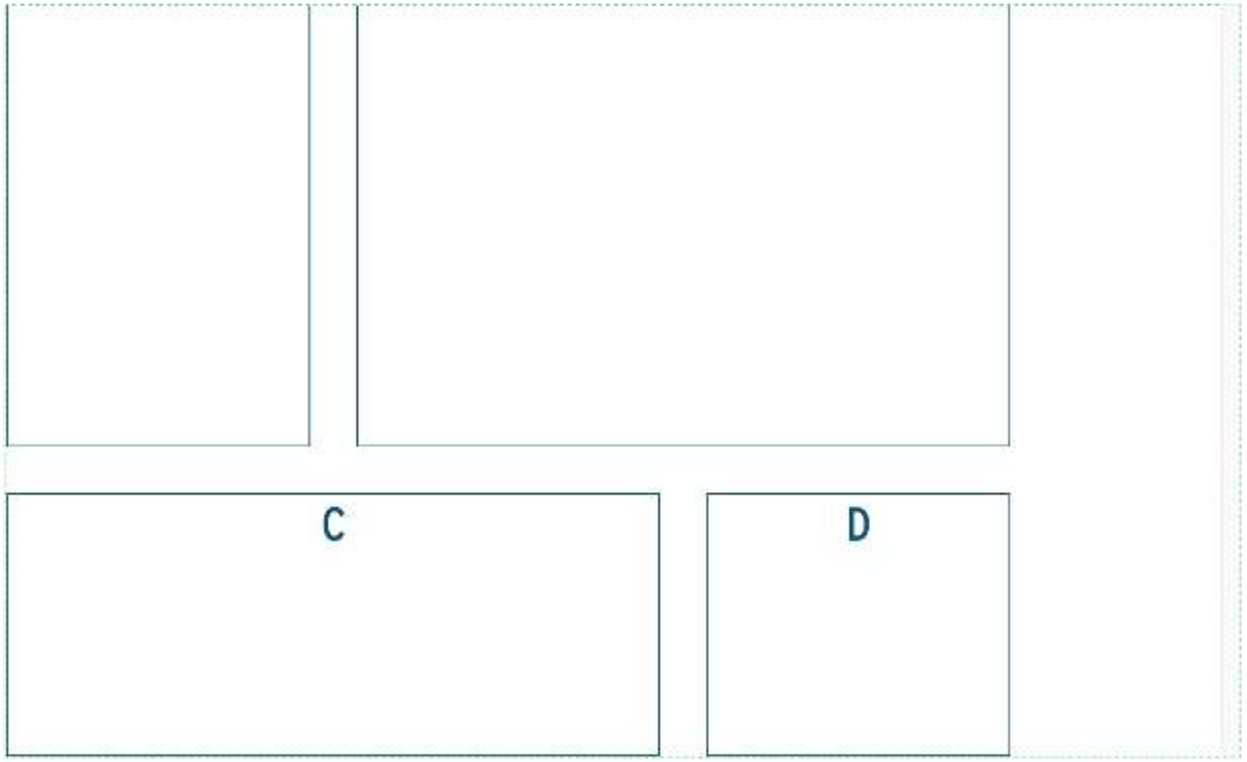
defaults for justify/align-self



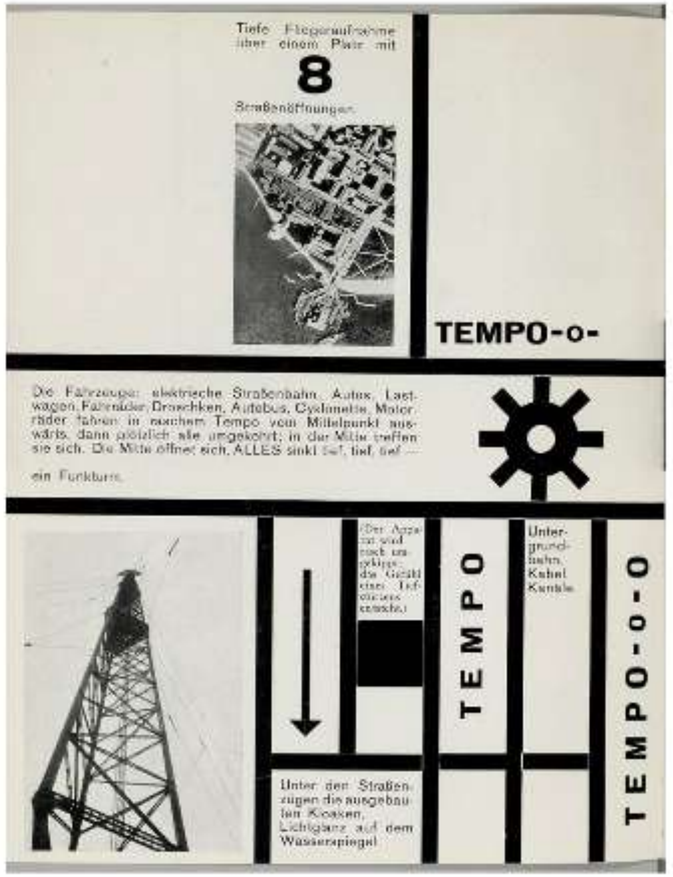

```
.items {
  justify-items: normal;
  align-items: normal;

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

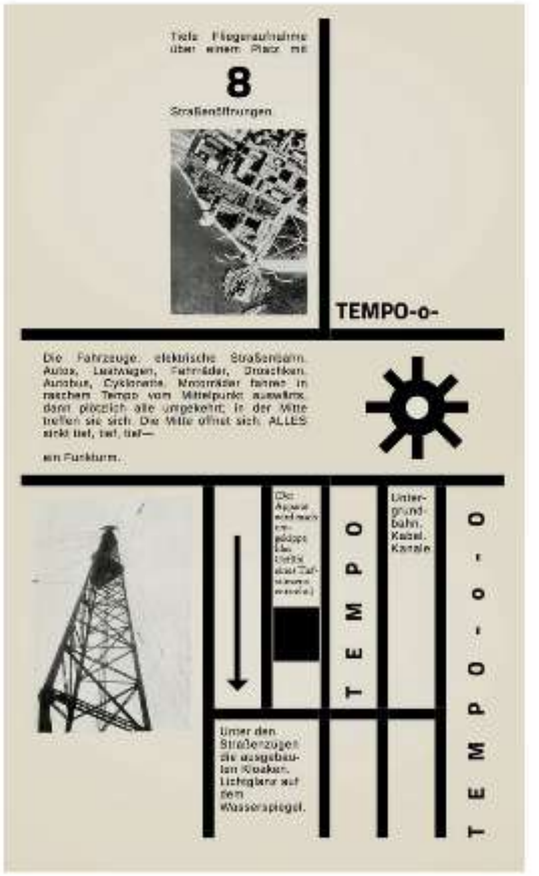
CSS Box Alignment Module Level 3

Overflow alignment keywords




```
.overflow {
  justify-content: normal;
  align-content: end;

  display: grid;
  grid-template-columns: repeat(3, 25%);
  grid-template-rows: repeat(3, 35%);
  grid-gap: 1em;
  grid-template-areas:
    "a b b"
    "a b b"
    "c c d";
```





Original image from Moholy-Nagy, Malerei, Fotografie, Film

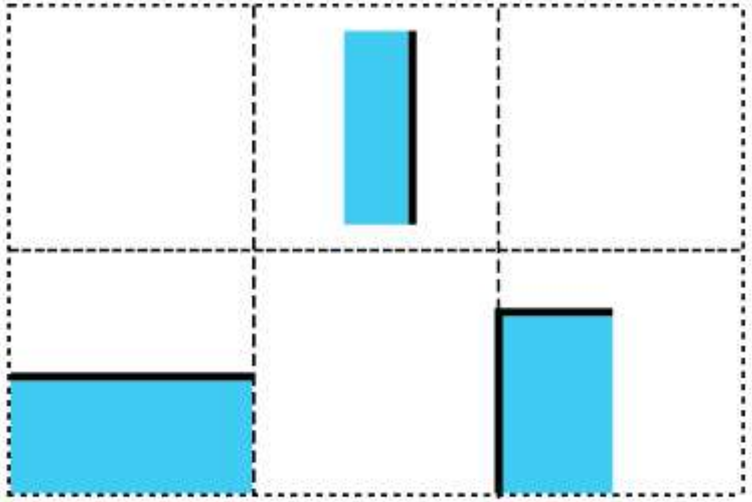
Bauhaus in the browser

Featuring...

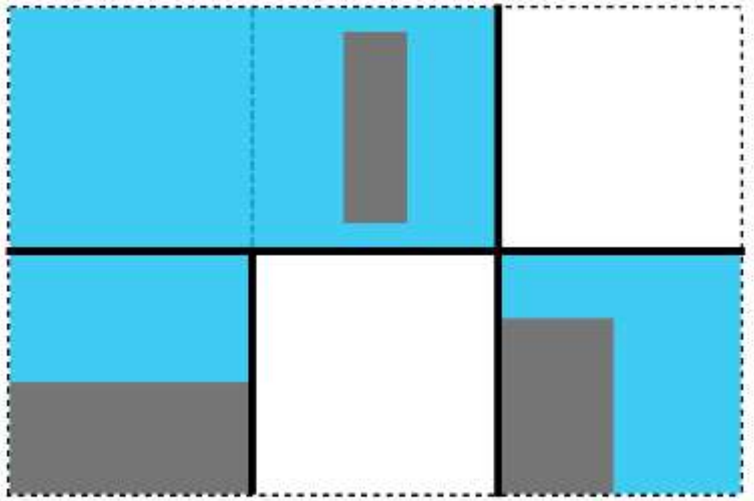
- Grid
- Flexbox
- Writing mode
- Transforms
- Box alignment

https://codepen.io/huijing/pen/PpqomV | Full page demo

```
.grid {
  display: grid;
  grid-template-columns: 30% 9% 9% 9% 9% 9%;
  justify-content: center;
}

.grid__item:nth-child(1) {
  grid-column: span 3;
  justify-self: end;
  border-right: 1em solid;
  padding: 1em;
  text-align: justify;
  text-align-last: justify;
```



These are not the borders you are looking for



That's more like it

References and resources

- Inside a super fast CSS engine: Quantum CSS (aka Stylo)
- CSS2.2 Visual formatting model
- CSS Display Module Level 3
- CSS2.2 Tables
- CSS Box Alignment Module Level 3
- CSS Writing Modes Level 3
- Bug 1038294 - [css-display] Implement the multi-keyword syntax for the 'display' property
- Learn CSS the pedantic way
- Vertical-Align: All You Need To Know
- Understanding CSS Layout And The Block Formatting Context
- The New Layout Standard For The Web: CSS Grid, Flexbox And Box Alignment
- Demystifying CSS alignment
- Powerful New Additions to the CSS Grid Inspector in Firefox Nightly
- Use cases for Flexbox

Thank you!

 https://www.chenhuijing.com

 @hj_chen

 @hj_chen

 @huijing

Font used is Mission Gothic, by James T. Edmondson and Trevor Baum