CSS Flexbox Layout

Flexible, Robust Line-Based Layout

R. Scott Granneman & Jans Carton

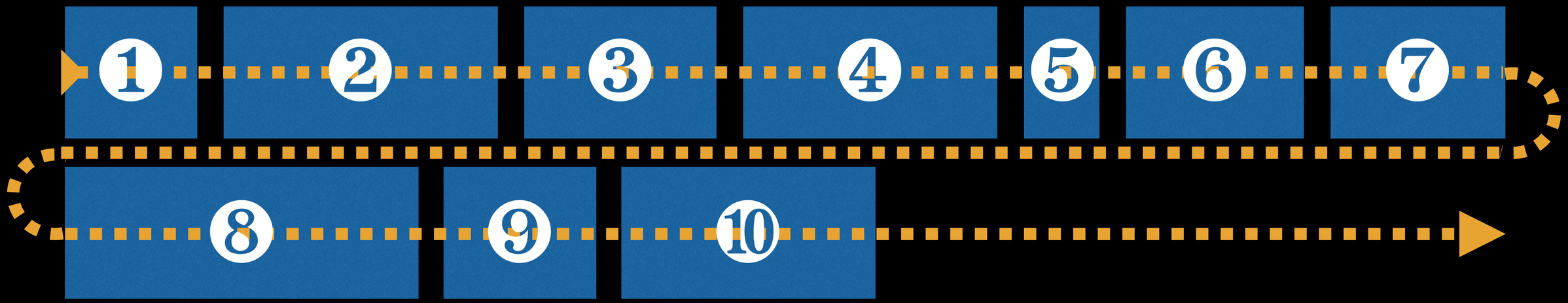
W3C Working Draft

CSS Flexible Box Layout Module Level 1

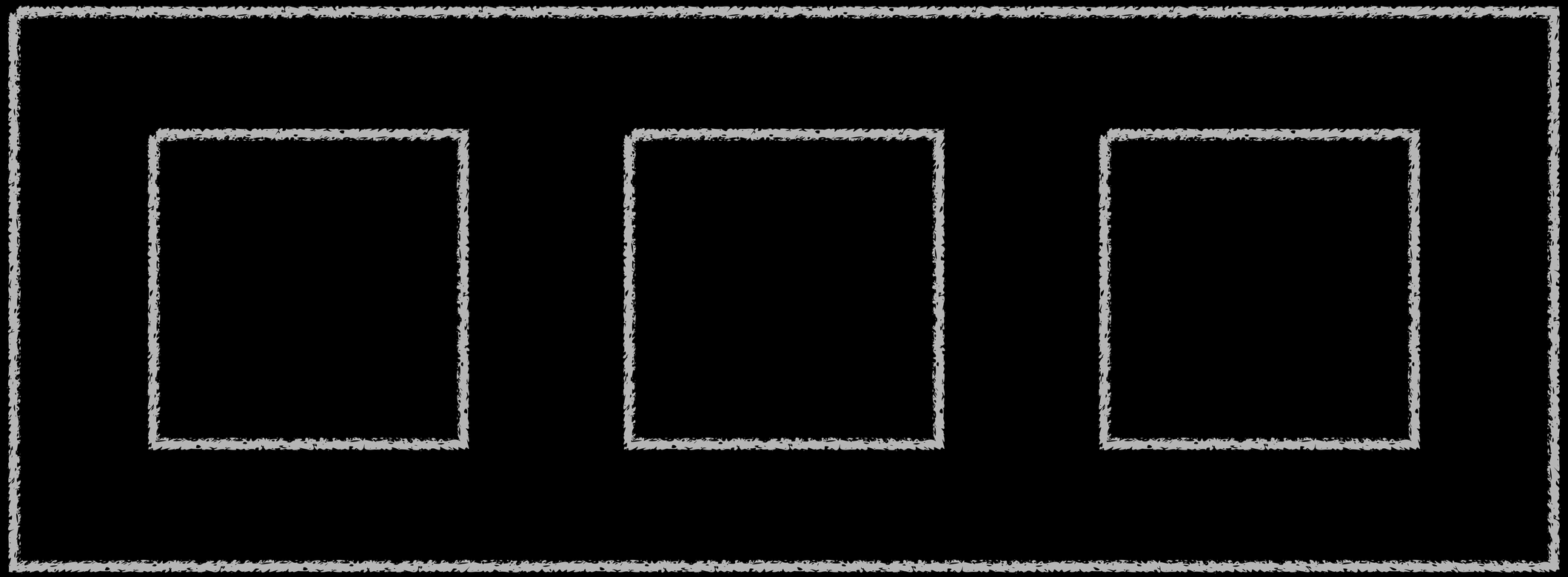
W3C Last Call Working Draft 14 May 2015

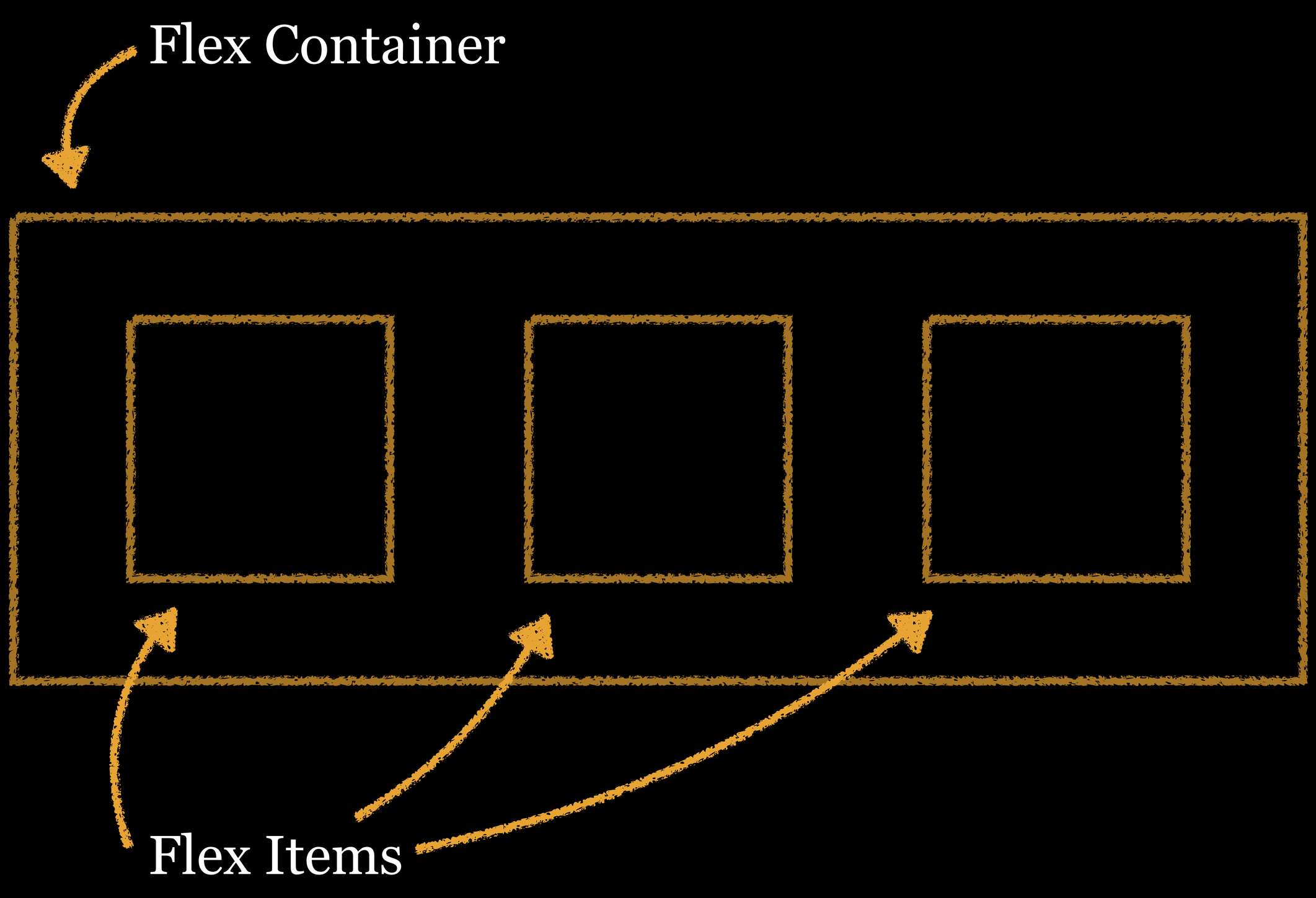
This version:
http://www.w3.org/TR/2015/WD-css-flexbox-1-20150514/
Latest version:
http://www.w3.org/TR/css-flexbox-1/
Previous Versions:
http://www.w3.org/TR/2014/WD-css-flexbox-1-20140925/
http://www.w3.org/TR/2014/WD-css-flexbox-1-20140325/
http://www.w3.org/TR/2012/CR-css3-flexbox-20120918/
http://www.w3.org/TR/2012/WD-css3-flexbox-20120612/
http://www.w3.org/TR/2012/WD-css3-flexbox-20120322/
http://www.w3.org/TR/2011/WD-css3-flexbox-20111129/
http://www.w3.org/TR/2011/WD-css3-flexbox-20110322/
http://www.w3.org/TR/2009/WD-css3-flexbox-20090723/
Editors draft:
http://dev.w3.org/csswg/css-flexbox/
Feedback:
www-style@w3.org with subject line "`[css-flexbox] … message topic …`" (archives)
Test Suite:
http://test.csswg.org/suites/css-flexbox-1_dev/nightly-unstable/

Editors:
Tab Atkins Jr. (Google)
fantasai (Mozilla)
Rossen Atanassov (Microsoft)
Former Editors:

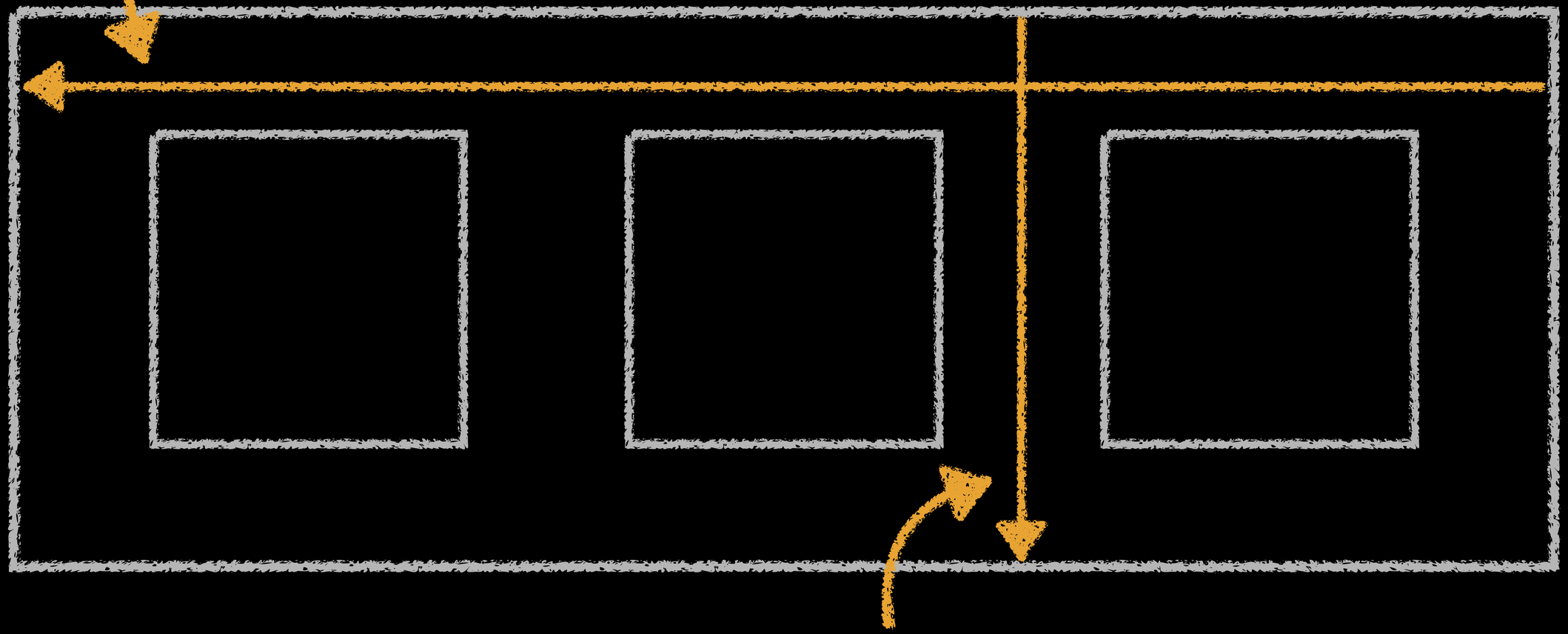


Flexbox is for laying out elements in a particular direction along a (sometimes wrapped) line

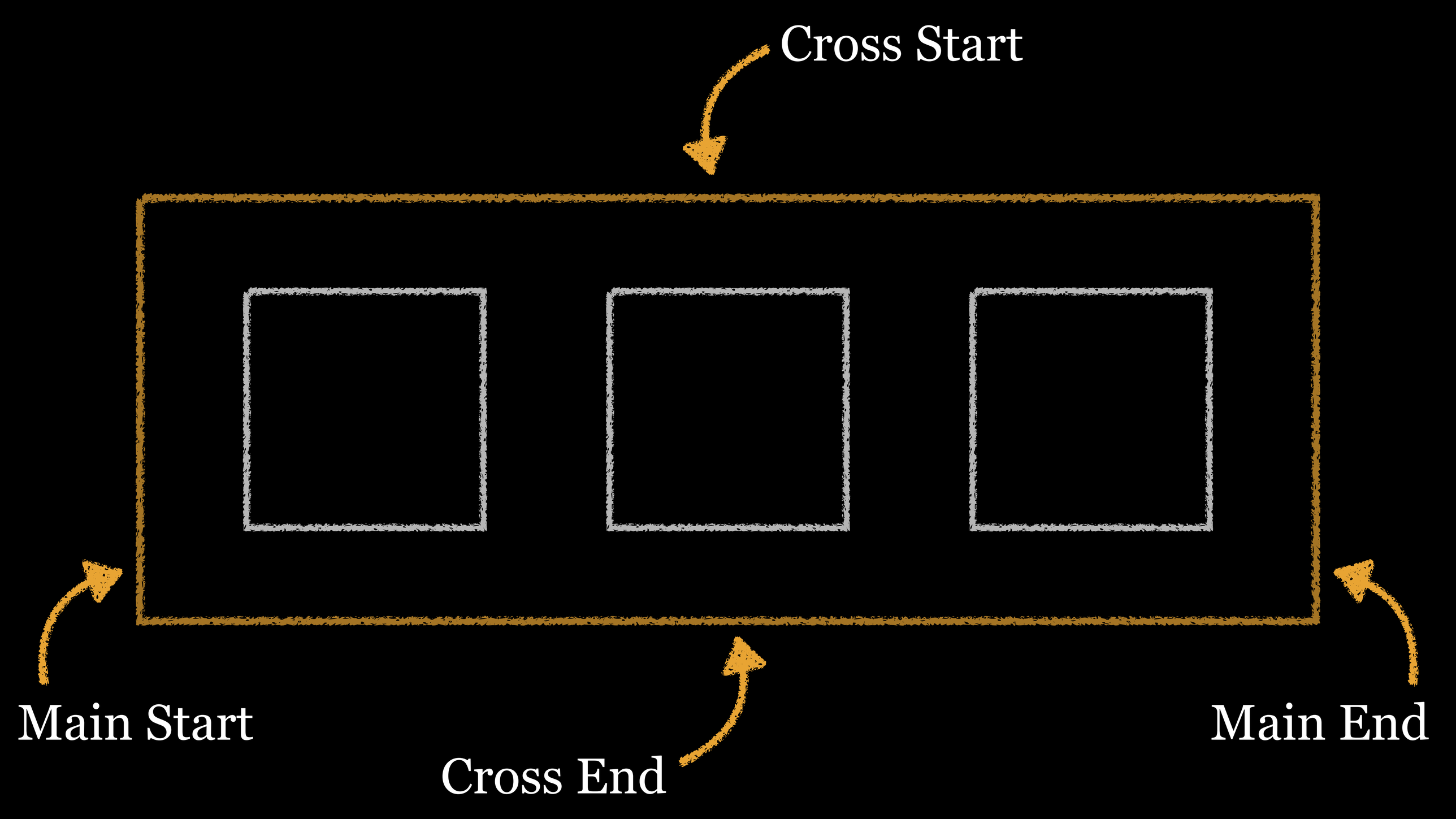


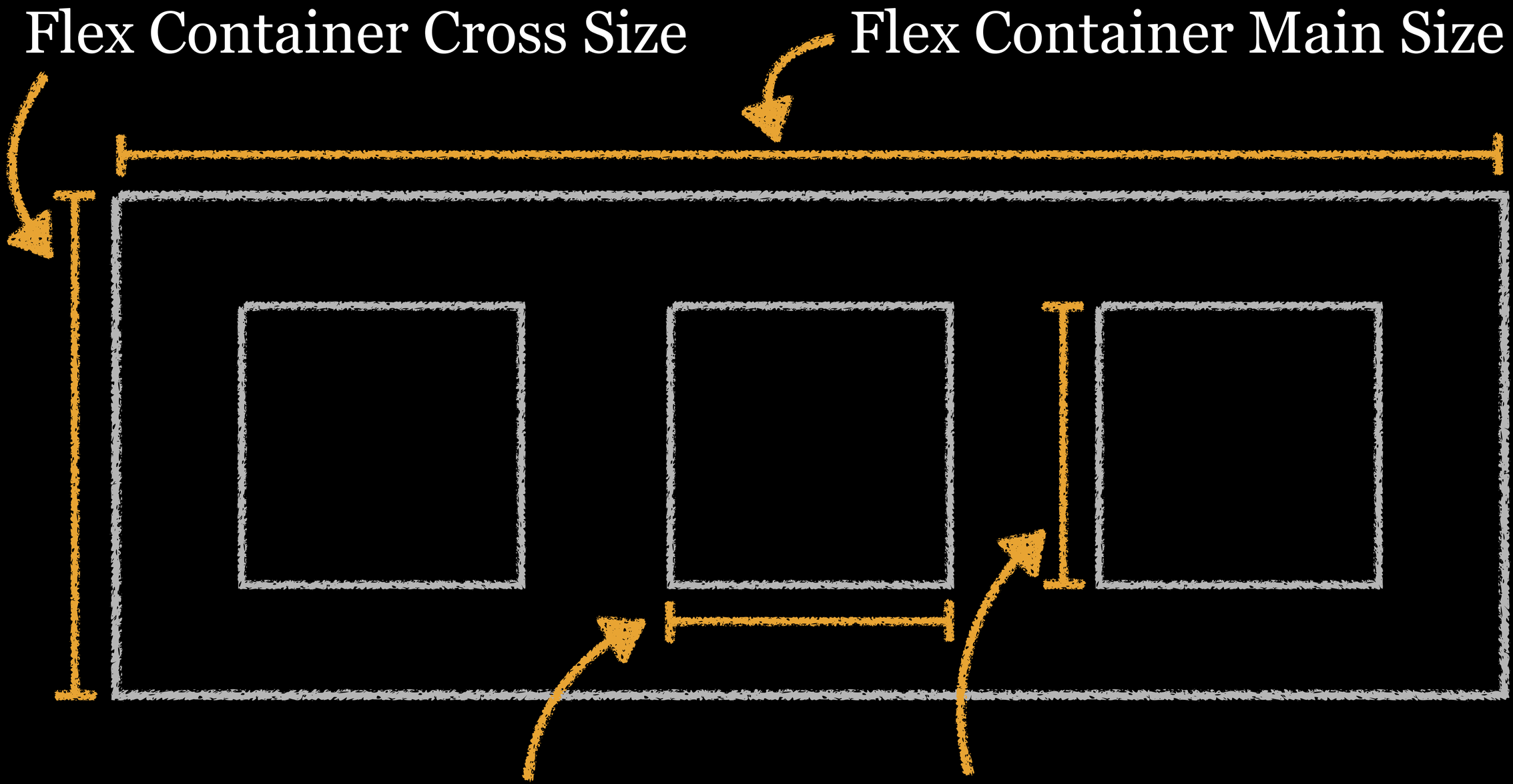

Flex Container
Flex Items

Main Axis: primary axis on which flex items are laid out (↔ or ↕)



Cross Axis: perpendicular to the main axis


Cross Start
Main Start
Cross End
Main End


Flex Container Cross Size
Flex Container Main Size
Flex Item Main Size
Flex Item Cross Size

Flex container

Flex items

Main axis

Cross axis

On the following slides, you'll see these symbols

```
display: flex|inline-flex

flex-direction
flex-wrap
flex-flow
order

justify-content
align-content
place-content

align-items
align-self
```

```
column-gap
row-gap
gap

flex-grow
flex-shrink
flex-basis

flex
```

Triggering Flexbox Layout

`display: flex` (`<inside>`)
`display: block flex` (`<outside>` `<inside>`)

Triggers flex layout inside the box:

» Flex box stacks (because `block`)
» *Immediate* children become *flex items*

HTML

```
<ul class="flex-container">
  <li>The odd mechanism of the hooked fastener</li>
  <li>was perfectly well known to me</li>
  <li>and I snapped up the still rustless and workable lid and drew out the book within.</li>
  <li>The latter, as expected, was some twenty by fifteen inches in area</li>
  <li>and two inches thick; the thin metal covers opening at the top.</li>
</ul>
```

CSS (SCSS) Compiled

```
.flex-container {
  display: flex;
}
```

JS

`display: inline-flex`

Generates an *inline box that behaves according to the flexbox model*:

» Generates atomic inline box — an inline box that *does not break across lines* (like `display: inline-block`)
» I*mmediate* children become *flex items*
» Changes the layout mode inside it

HTML

```
The Shadow
<ul class="flex-container">
  <li>The odd mechanism of the hooked fastener</li>
  <li>was perfectly well known to me</li>
  <li>and I snapped up the still rustless and workable lid and drew out the book within.</li>
  <li>The latter, as expected, was some twenty by fifteen inches in area</li>
  <li>and two inches thick; the thin metal covers opening at the top.</li>
</ul>
Out of Time
```

CSS (SCSS) Compiled

```
.flex-container {
  display: inline-flex;
}
.flex-container > * {
  width: 70px;
}
```

JS

The Shadow

1. The odd mechanism of the hooked fastener
2. was perfectly well known to me
3. and I snapped up the still rustless and workable lid and drew out the book within.
4. The latter, as expected, was some twenty by fifteen inches in area
5. and two inches thick; the thin metal covers opening at the top.

Out of Time

display:					iOS		
flex	8*	12	28	9	9	29	4.4
inline-flex	11†	12	28	9	9	29	4.4

* Uses `-ms-flexbox` † 8–10 use `-ms-inline-flexbox`

Direction, Wrapping, & Order

flex-direction

flex-wrap

flex-flow

order

`flex-direction`

Specifies *how flex items are laid out in the flex container* by setting…

» the main axis: ↔ ↕
» the direction of the flow along the main axis

`flex-direction`

Values:

» `row` (default) →
» `row-reverse` ←
» `column` ↓
» `column-reverse` ↑

`flex-direction: row`

Flex items are stacked in a row *from left-to-right*

(If the default for your locale is `direction: rtl`, then it's the opposite)

HTML

```
<ul class="flex-container">
  <li>The odd mechanism of the hooked fastener</li>
  <li>was perfectly well known to me</li>
  <li>and I snapped up the still rustless and workable lid and drew out the book within.</li>
  <li>The latter, as expected, was some twenty by fifteen inches in area</li>
  <li>and two inches thick; the thin metal covers opening at the top.</li>
</ul>
```

CSS (SCSS) Compiled

```
.flex-container {
  display: flex;
  flex-direction: row;
}

```

JS

1. The odd mechanism of the hooked fastener
2. was perfectly well known to me
3. and I snapped up the still rustless and workable lid and drew out the book within.
4. The latter, as expected, was some twenty by fifteen inches in area
5. and two inches thick; the thin metal covers opening at the top.

`flex-direction: row-reverse`

Flex items are stacked in a row *from right-to-left*

(If the default for your locale is `direction: rtl`, then it's the opposite)

HTML

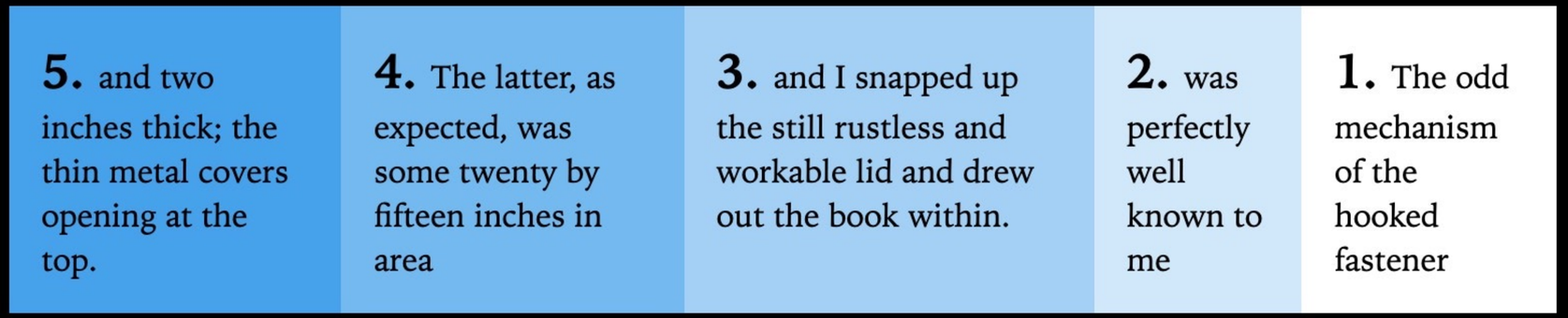
```
<ul class="flex-container">
  <li>The odd mechanism of the hooked fastener</li>
  <li>was perfectly well known to me</li>
  <li>and I snapped up the still rustless and workable lid and drew out the book within.</li>
  <li>The latter, as expected, was some twenty by fifteen inches in area</li>
  <li>and two inches thick; the thin metal covers opening at the top.</li>
</ul>
```

CSS (SCSS) Compiled

```
.flex-container {
  display: flex;
  flex-direction: row-reverse;
}

```

JS



`flex-direction: column`

Flex items are stacked *in a column from top-to-bottom*

HTML

```
<ul class="flex-container">
  <li>The odd mechanism of the hooked fastener</li>
  <li>was perfectly well known to me</li>
  <li>and I snapped up the still rustless and workable lid and drew out the book within.</li>
  <li>The latter, as expected, was some twenty by fifteen inches in area</li>
  <li>and two inches thick; the thin metal covers opening at the top.</li>
</ul>
```

CSS (SCSS) Compiled

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

```

JS

1. The odd mechanism of the hooked fastener
2. was perfectly well known to me
3. and I snapped up the still rustless and workable lid and drew out the book within.
4. The latter, as expected, was some twenty by fifteen inches in area
5. and two inches thick; the thin metal covers opening at the top.

`flex-direction: column-reverse`

Flex items are stacked *in a column from bottom-to-top*

HTML

```
<ul class="flex-container">
  <li>The odd mechanism of the hooked fastener</li>
  <li>was perfectly well known to me</li>
  <li>and I snapped up the still rustless and workable lid and drew out the book within.</li>
  <li>The latter, as expected, was some twenty by fifteen inches in area</li>
  <li>and two inches thick; the thin metal covers opening at the top.</li>
</ul>
```

CSS (SCSS) Compiled

```
.flex-container {
  display: flex;
  flex-direction: column-reverse;
}

```

JS

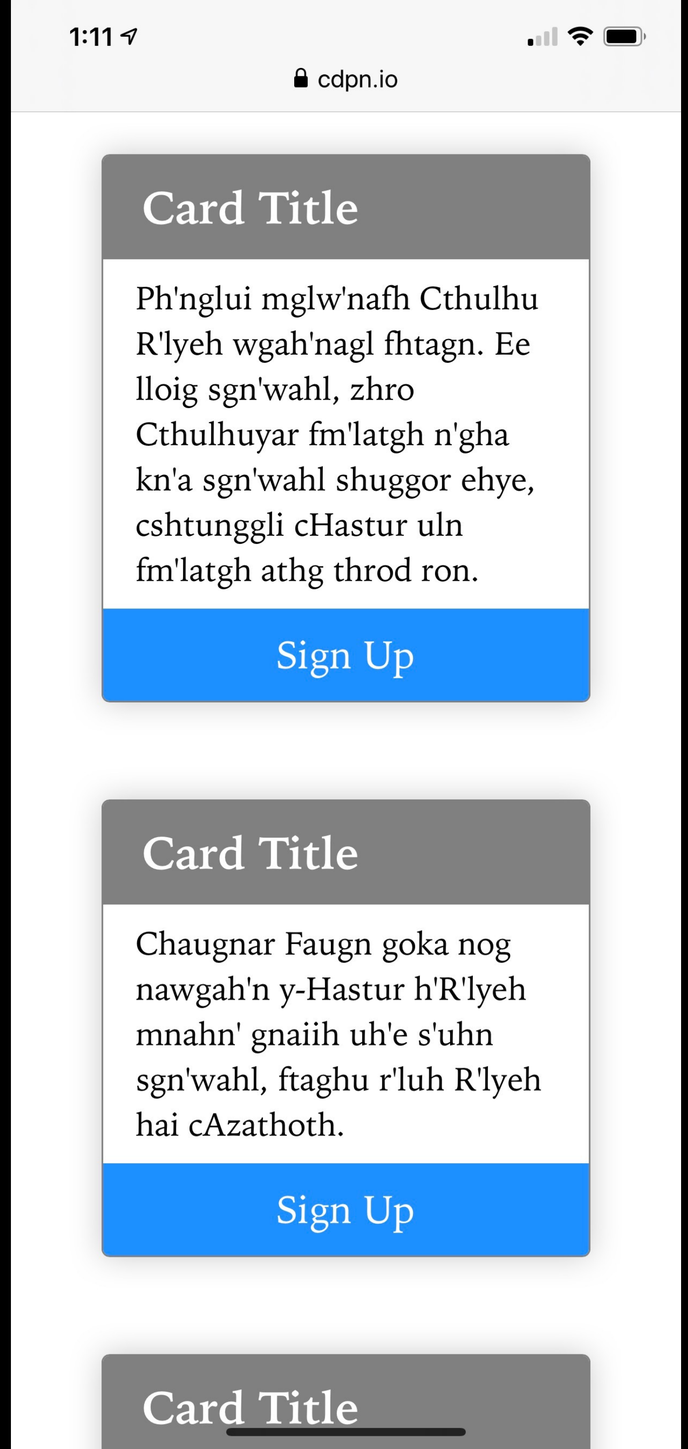
5. and two inches thick; the thin metal covers opening at the top.
4. The latter, as expected, was some twenty by fifteen inches in area
3. and I snapped up the still rustless and workable lid and drew out the book within.
2. was perfectly well known to me
1. The odd mechanism of the hooked fastener

You can apply `display: flex` on flex items

Add `flex-direction` to the mix & you can really have something

Each card is both a flex item & a flex container

`flex-direction: row` for the outer container;
`flex-direction: column` for each card



On a mobile device, responsive layout aligns them vertically & their height is now sized to their content

`flex-wrap`

Specifies if flex container *lays out flex items in single or multiple lines*, & the direction new lines are stacked in

Only applies if the flex container is too small to contain the flex items

`flex-wrap`

Values:

» `nowrap` (default)
» `wrap`
» `wrap-reverse`

`flex-wrap: nowrap`

Flex items are displayed in one row & they are *shrunk to fit the width of the flex container*

HTML

```
<ul class="flex-container">
  <li>The odd mechanism of the hooked fastener</li>
  <li>was perfectly well known to me</li>
  <li>and I snapped up the still rustless and workable lid and drew out the book within.</li>
  <li>The latter, as expected, was some twenty by fifteen inches in area</li>
  <li>and two inches thick; the thin metal covers opening at the top.</li>
</ul>
```

CSS (SCSS) Compiled

```
.flex-container {
  display: flex;
  flex-wrap: nowrap;
}
.flex-container > * {
  width: 200px;
}
```

JS

1. The odd mechanism of the hooked fastener
2. was perfectly well known to me
3. and I snapped up the still rustless and workable lid and drew out the book within.
4. The latter, as expected, was some twenty by fifteen inches in area
5. and two inches thick; the thin metal covers opening at the top.

HTML

```
<ul class="flex-container">
  <li>The odd mechanism of the hooked fastener</li>
  <li>was perfectly well known to me</li>
  <li>and I snapped up the still rustless and workable lid and drew out the book within.</li>
  <li>The latter, as expected, was some twenty by fifteen inches in area</li>
  <li>and two inches thick; the thin metal covers opening at the top.</li>
</ul>
```

CSS (SCSS) Compiled

```
.flex-container {
  display: flex;
  flex-wrap: nowrap;
}
.flex-container > * {
  width: 200px;
}
```

JS

1. The odd mechanism of the hooked fastener
2. was perfectly well known to me
3. and I snapped up the still rustless and workable lid and drew out the book within.
4. The latter, as expected, was some twenty by fifteen inches in area
5. and two inches thick; the thin metal covers opening at the top.

li 145.8 × 197
Color #00
Font 16px "Iowan Old Style", Georgia, serif
Background #D1E8FA
Padding 16px
Contrast Aa 16.63 ✓

These are *not* 200px wide

`flex-wrap: wrap`

Flex items are *displayed in multiple rows*, from left-to-right and top-to-bottom

HTML

```
<ul class="flex-container">
  <li>The odd mechanism of the hooked fastener</li>
  <li>was perfectly well known to me</li>
  <li>and I snapped up the still rustless and workable lid and drew out the book within.</li>
  <li>The latter, as expected, was some twenty by fifteen inches in area</li>
  <li>and two inches thick; the thin metal covers opening at the top.</li>
</ul>
```

CSS (SCSS) Compiled

```
.flex-container {
  display: flex;
  flex-wrap: wrap;
}
.flex-container > * {
  width: 200px;
}
```

JS

1. The odd mechanism of the hooked fastener
2. was perfectly well known to me
3. and I snapped up the still rustless and workable lid and drew out the book within.
4. The latter, as expected, was some twenty by fifteen inches in area
5. and two inches thick; the thin metal covers opening at the top.

HTML

```
<ul class="flex-container">
  <li>The odd mechanism of the hooked fastener</li>
  <li>was perfectly well known to me</li>
  <li>and I snapped up the still rustless and workable lid and drew out the book within.</li>
  <li>The latter, as expected, was some twenty by fifteen inches in area</li>
  <li>and two inches thick; the thin metal covers opening at the top.</li>
</ul>
```

CSS (SCSS) Compiled

```
.flex-container {
  display: flex;
  flex-wrap: wrap;
}
.flex-container > * {
  width: 200px;
}
```

JS

These *are* 200px wide

`flex-wrap: wrap-reverse`

Flex items are *displayed in multiple rows*, from left-to-right *but from bottom-to-top*

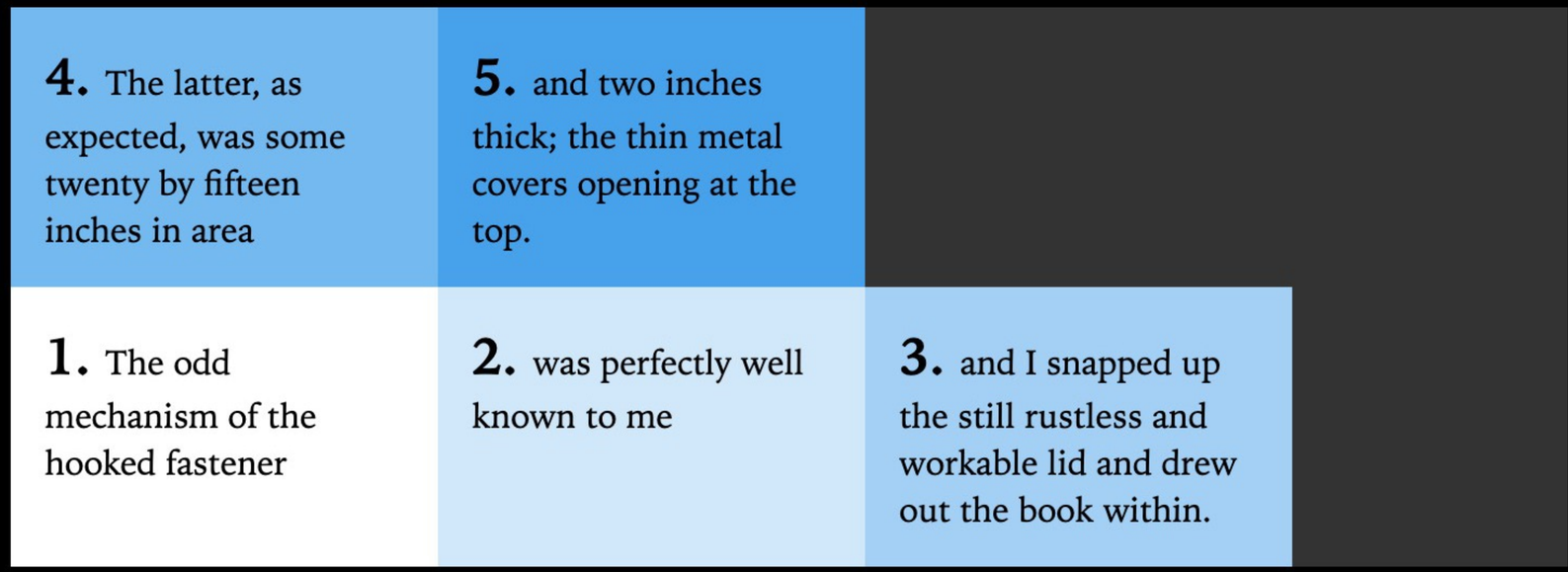
HTML

```
<ul class="flex-container">
  <li>The odd mechanism of the hooked fastener</li>
  <li>was perfectly well known to me</li>
  <li>and I snapped up the still rustless and workable lid and drew out the book within.</li>
  <li>The latter, as expected, was some twenty by fifteen inches in area</li>
  <li>and two inches thick; the thin metal covers opening at the top.</li>
</ul>
```

CSS (SCSS) Compiled

```
.flex-container {
  display: flex;
  flex-wrap: wrap-reverse;
}
.flex-container > * {
  width: 200px;
}
```

JS



`flex-flow`

Shorthand for setting `flex-direction` & `flex-wrap`

`flex-flow: <flex-direction> <flex-wrap>`

Default is `row nowrap`

`order`

Specifies *order in which flex items appear* inside the flex container

Default is `0`, which orders flex items according to order in HTML

Value: `<integer>`

HTML

```
<ul class="flex-container">
  <li>The odd mechanism of the hooked fastener</li>
  <li>was perfectly well known to me</li>
  <li>and I snapped up the still rustless and workable lid and drew
```

CSS (SCSS) Compiled

```
.flex-container {
  display: flex;
}
.flex-container > *:nth-child(1) {
  order: 2;
}
.flex-container > *:nth-child(2) {
  order: 1;
}
.flex-container > *:nth-child(3) {
  order: 0;
}
.flex-container > *:nth-child(4) {
  order: 0;
}
.flex-container > *:nth-child(5) {
  order: -1;
}
```

JS

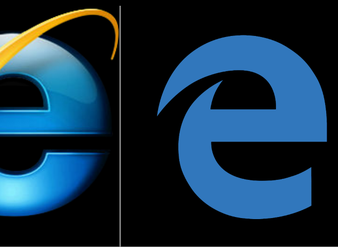
5. and two inches thick; the thin metal covers opening at the top.

3. and I snapped up the still rustless and workable lid and drew out the book within.

4. The latter, as expected, was some twenty by fifteen inches in area

2. was perfectly well known to me

1. The odd mechanism of the hooked fastener

					iOS		
`flex-direction`	11	12	28	9	9	29	4.4
`flex-wrap`	11*	12	28	9	9	29	4.4
`flex-flow`	11	12	28	9	9	29	4.4
`order`	11	12	28	9	9.2	29	4.4

* Very buggy; see chnsa.ws/1ik for more

Aligning Items & Lines

justify-content
align-content

place-content

safe

start
end

align-items
align-self

column-gap
row-gap

gap

Remember that *distributed alignment* focuses on distributing space among aligned boxes, e.g., `stretch`, `space-around`, `space-between`, & `space-evenly`

	Distributed	~~Distributed~~
Main axis	`justify-content`	
Cross axis	`align-content`	`align-items` `align-self`

This table helps simplify the relationship between box alignment keywords

💡 PRO TIP

Never use `width` & `height` with flexbox layout! Instead of `width` & `height`, size items using these for the cross axis:

- `align-items: stretch`
- `align-content: stretch`
- `align-self: stretch`

And this for the main axis:

- `flex-basis`

We're going to cover `align-*` in this section

Aligning Flex Items on the Main Axis

`justify-content`

Defines *how space is distributed between & around flex items* along the main axis of their container

Values for `justify-content`:

- `flex-start` (Default)
- `flex-end`
- `center`
- `space-between`
- `space-around`
- `space-evenly`
- `start`
- `end`

`justify-content: flex-start`

Aligns flex items to the *start of the main axis of the flex container*

Default value for `justify-content`

HTML

```
<ul class="flex-container">
  <li>The odd mechanism of the hooked fastener</li>
  <li>was perfectly well known to me</li>
  <li>and I snapped up the still rustless and workable lid and drew out the book within.</li>
  <li>The latter, as expected, was some twenty by fifteen inches in area</li>
  <li>and two inches thick; the thin metal covers opening at the top.</li>
</ul>
```

CSS (SCSS) Compiled

```
.flex-container {
  display: flex;
  justify-content: flex-start;
}
.flex-container > * {
  width: 80px;
}
```

JS

1. The odd mechanism of the hooked fastener
2. was perfectly well known to me
3. and I snapped up the still rustless and workable lid and drew out the book within.
4. The latter, as expected, was some twenty by fifteen inches in area
5. and two inches thick; the thin metal covers opening at the top.

```
justify-content: flex-end
```

Aligns flex items to the *end of the main axis of the flex container*

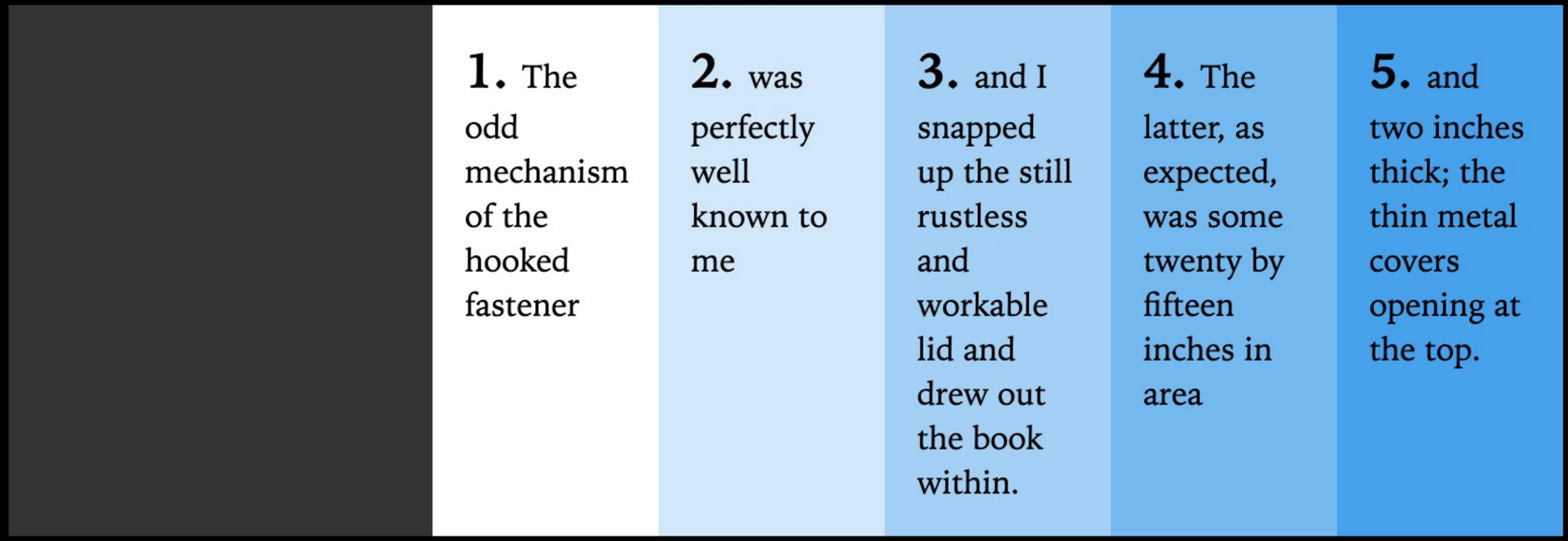
HTML

```
<ul class="flex-container">
  <li>The odd mechanism of the hooked fastener</li>
  <li>was perfectly well known to me</li>
  <li>and I snapped up the still rustless and workable lid and drew out the book within.</li>
  <li>The latter, as expected, was some twenty by fifteen inches in area</li>
  <li>and two inches thick; the thin metal covers opening at the top.</li>
</ul>
```

CSS (SCSS) Compiled

```
.flex-container {
  display: flex;
  justify-content: flex-end;
}
.flex-container > * {
  width: 80px;
}
```

JS



```
justify-content: center
```

Aligns flex items at the *center of the main axis of the flex container*

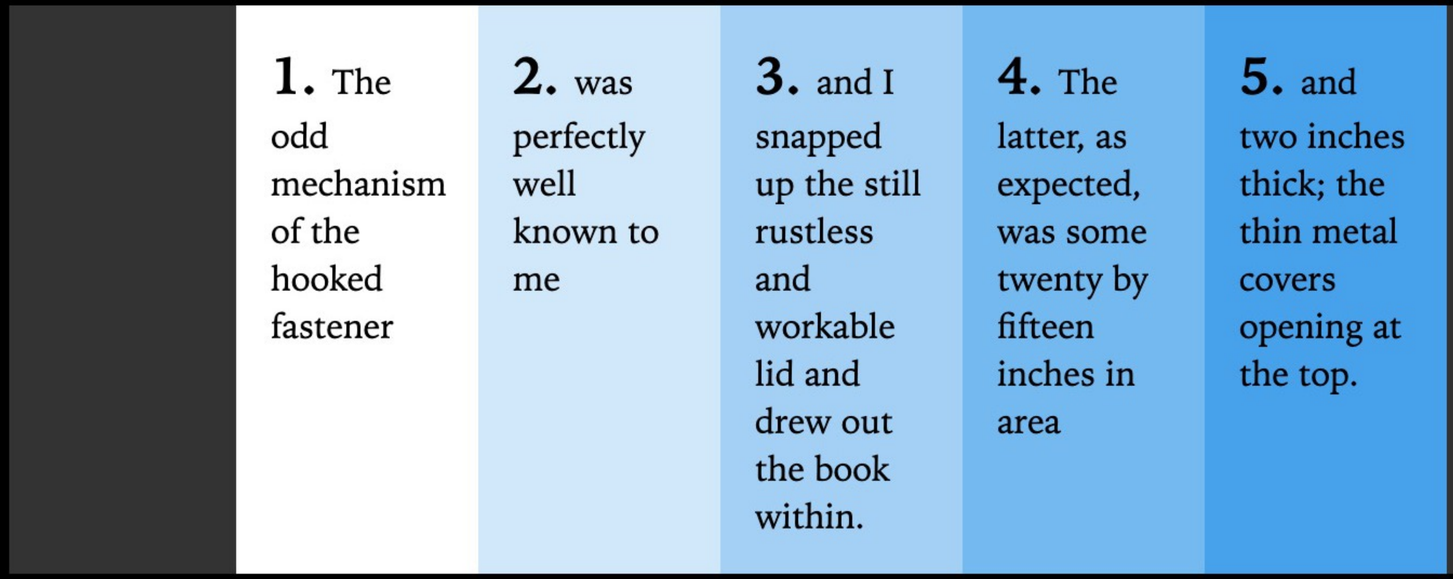
HTML

```
<ul class="flex-container">
  <li>The odd mechanism of the hooked fastener</li>
  <li>was perfectly well known to me</li>
  <li>and I snapped up the still rustless and workable lid and drew out the book within.</li>
  <li>The latter, as expected, was some twenty by fifteen inches in area</li>
  <li>and two inches thick; the thin metal covers opening at the top.</li>
</ul>
```

CSS (SCSS) Compiled

```
.flex-container {
  display: flex;
  justify-content: center;
}
.flex-container > * {
  width: 80px;
}
```

JS



`justify-content: space-between`

Flex items have *equal spacing between them*, with first & last flex items aligned to edges of the main axis of the flex container

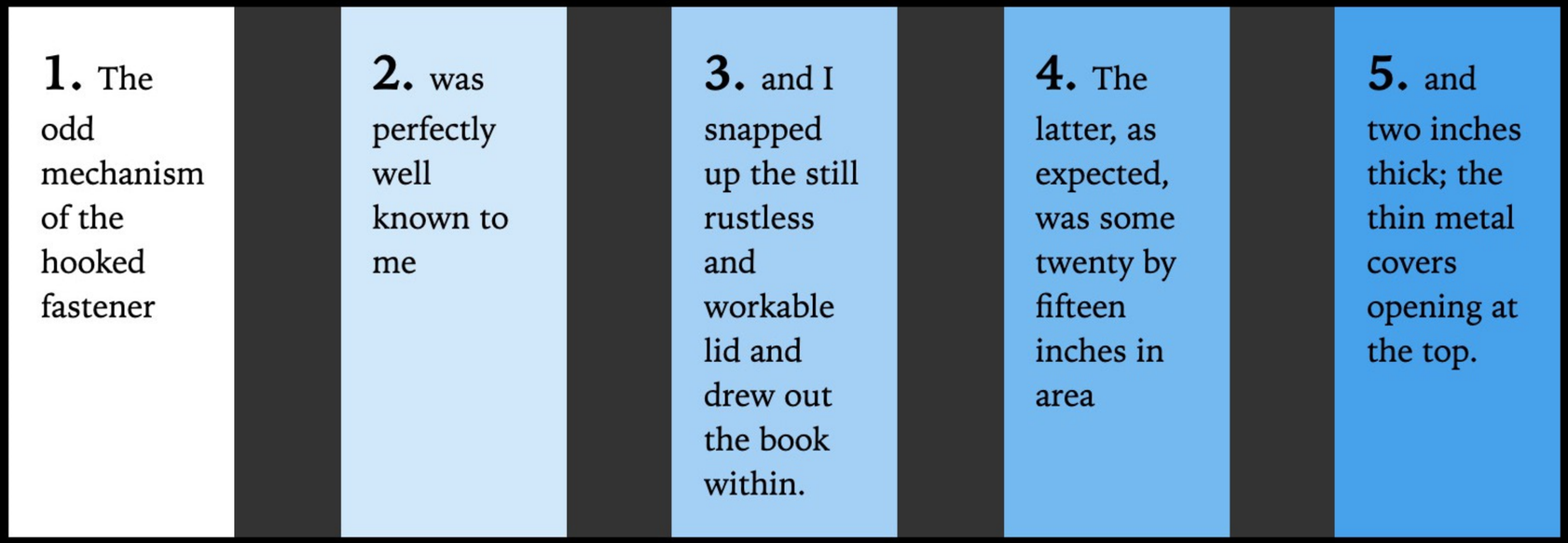
HTML

```
<ul class="flex-container">
  <li>The odd mechanism of the hooked fastener</li>
  <li>was perfectly well known to me</li>
  <li>and I snapped up the still rustless and workable lid and drew out the book within.</li>
  <li>The latter, as expected, was some twenty by fifteen inches in area</li>
  <li>and two inches thick; the thin metal covers opening at the top.</li>
</ul>
```

CSS (SCSS) Compiled

```
.flex-container {
  display: flex;
  justify-content: space-between;
}
.flex-container > * {
  width: 80px;
}
```

JS



No space before first flex item (or after last flex item), but equal space between flex items

HTML

Tidy

```html
<div class="flex-container">
  <p>← Prev</p>
  <p>Next →</p>
</div>
```

CSS

Tidy

View Compiled

```css
.flex-container {
  display: flex;
  justify-content: space-between;
  align-items: center;
  border-top: 1px solid gray;
  border-bottom: 1px solid gray;
}
```

← Prev

Next →

`justify-content: space-around`

Flex items have *equal spacing around them*, with first & last flex items getting half-sized spaces on the ends of the main axis of the flex container

Empty space before the first, and after the last, flex items equals half of the space between two adjacent items

HTML

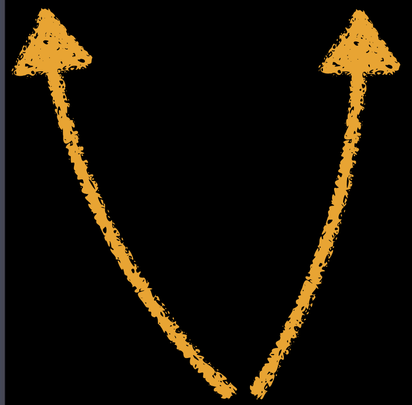
```
<ul class="flex-container">
  <li>The odd mechanism of the hooked fastener</li>
  <li>was perfectly well known to me</li>
  <li>and I snapped up the still rustless and workable lid and drew out the book within.</li>
  <li>The latter, as expected, was some twenty by fifteen inches in area</li>
  <li>and two inches thick; the thin metal covers opening at the top.</li>
</ul>
```

CSS (SCSS) Compiled

```
.flex-container {
  display: flex;
  justify-content: space-around;
}
.flex-container > * {
  width: 80px;
}
```

JS

1. The odd mechanism of the hooked fastener
2. was perfectly well known to me
3. and I snapped up the still rustless and workable lid and drew out the book within.
4. The latter, as expected, was some twenty by fifteen inches in area
5. and two inches thick; the thin metal covers opening at the top.



Space before 1st flex item (& after last flex item) is 1/2 of space between flex items

`justify-content: space-evenly`

Flex items have *equal spacing around them*, with first & last flex items getting *equal spaces* on the ends of the main axis of the flex container

Empty space before the first, and after the last, flex items equals the space between two adjacent items

HTML

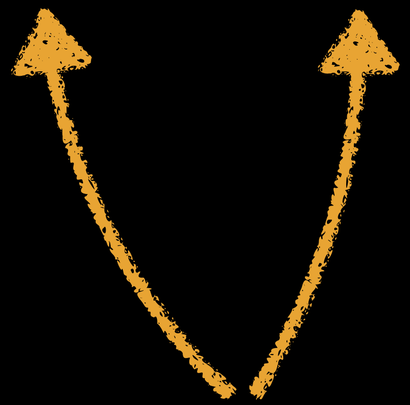
```
<ul class="flex-container">
  <li>The odd mechanism of the hooked fastener</li>
  <li>was perfectly well known to me</li>
  <li>and I snapped up the still rustless and workable lid and drew out the book within.</li>
  <li>The latter, as expected, was some twenty by fifteen inches in area</li>
  <li>and two inches thick; the thin metal covers opening at the top.</li>
</ul>
```

CSS (SCSS) Compiled

```
.flex-container {
  display: flex;
  justify-content: space-evenly;
}
.flex-container > * {
  width: 80px;
}
```

JS

1. The odd mechanism of the hooked fastener
2. was perfectly well known to me
3. and I snapped up the still rustless and workable lid and drew out the book within.
4. The latter, as expected, was some twenty by fifteen inches in area
5. and two inches thick; the thin metal covers opening at the top.



Space before first flex item & after last flex item equal to space between flex items

						
`justify-content*`	11	12	29	9	9.2	29	4.4
`: space-evenly`	–	79	52	11	11	60	Y

* Includes support for `flex-start`, `flex-end`, `center`, `space-between`, & `space-around`

Aligning Flex Items on the Cross Axis

`align-items`

Aligns flex items in the cross axis of the current flex line

Values for `align-items`:

- `stretch` (Default)
- `flex-start`
- `flex-end`
- `center`
- `baseline`
- `start`
- `end`

`align-items: stretch`

Flex items *fill the whole height (or width) from cross start to cross end* of the flex container

Obviously the flex item cannot have `height` set

Default value for `align-items`

HTML

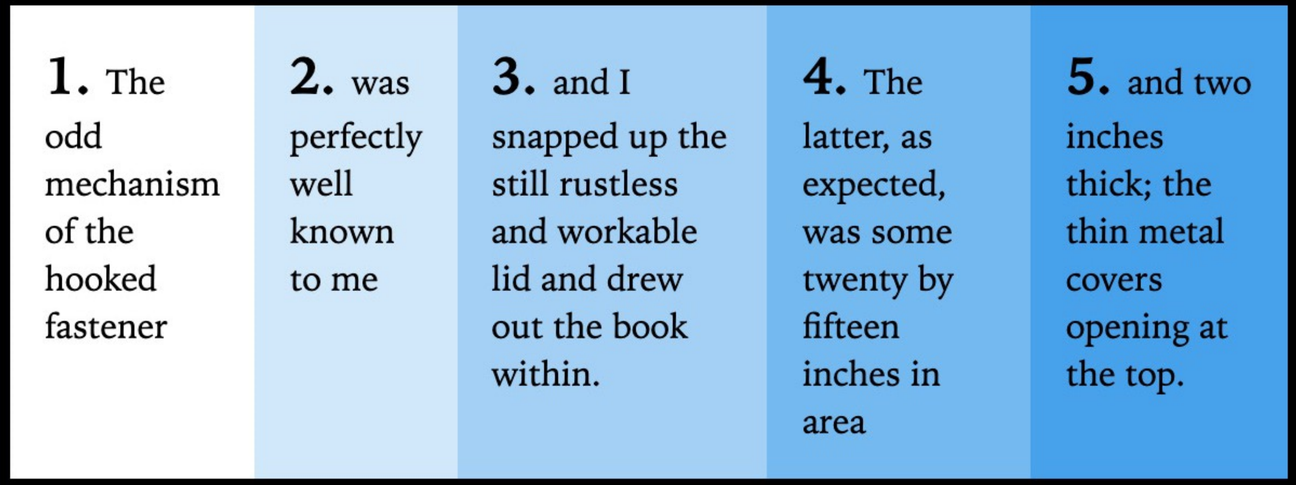
```
<ul class="flex-container">
  <li>The odd mechanism of the hooked fastener</li>
  <li>was perfectly well known to me</li>
  <li>and I snapped up the still rustless and workable lid and drew out the book within.</li>
  <li>The latter, as expected, was some twenty by fifteen inches in area</li>
  <li>and two inches thick; the thin metal covers opening at the top.</li>
</ul>
```

CSS (SCSS) Compiled

```
.flex-container {
  display: flex;
  align-items: stretch;
}

```

JS



`align-items: flex-start`

Flex items *stack from the cross start* of the flex container

HTML

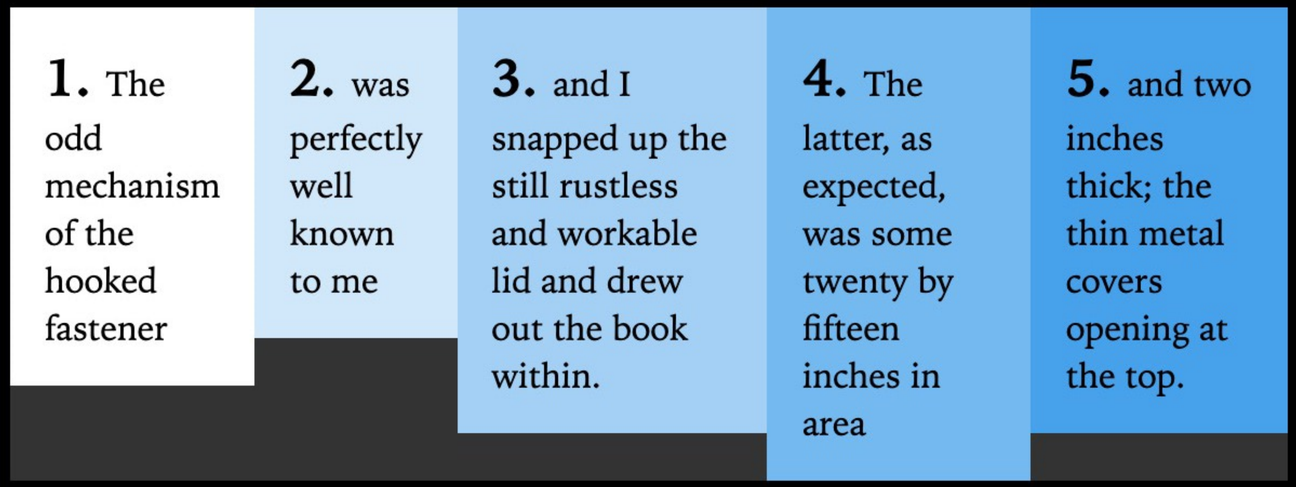
```html
<ul class="flex-container">
  <li>The odd mechanism of the hooked fastener</li>
  <li>was perfectly well known to me</li>
  <li>and I snapped up the still rustless and workable lid and drew out the book within.</li>
  <li>The latter, as expected, was some twenty by fifteen inches in area</li>
  <li>and two inches thick; the thin metal covers opening at the top.</li>
</ul>
```

CSS (SCSS) Compiled

```scss
.flex-container {
  display: flex;
  align-items: flex-start;
}

```

JS



```
align-items: flex-end
```

Flex items *stack from the cross end* of the flex container

HTML

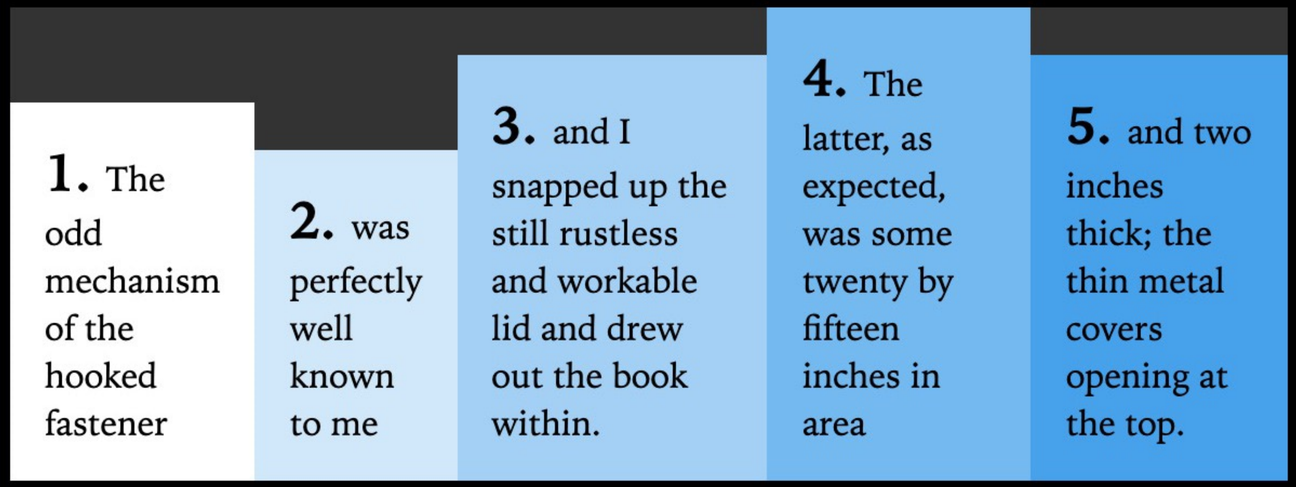
```
<ul class="flex-container">
  <li>The odd mechanism of the hooked fastener</li>
  <li>was perfectly well known to me</li>
  <li>and I snapped up the still rustless and workable lid and drew out the book within.</li>
  <li>The latter, as expected, was some twenty by fifteen inches in area</li>
  <li>and two inches thick; the thin metal covers opening at the top.</li>
</ul>
```

CSS (SCSS) Compiled

```
.flex-container {
  display: flex;
  align-items: flex-end;
}

```

JS



```
align-items: center
```

Flex items *stack from the center of the cross axis* of the flex container

HTML

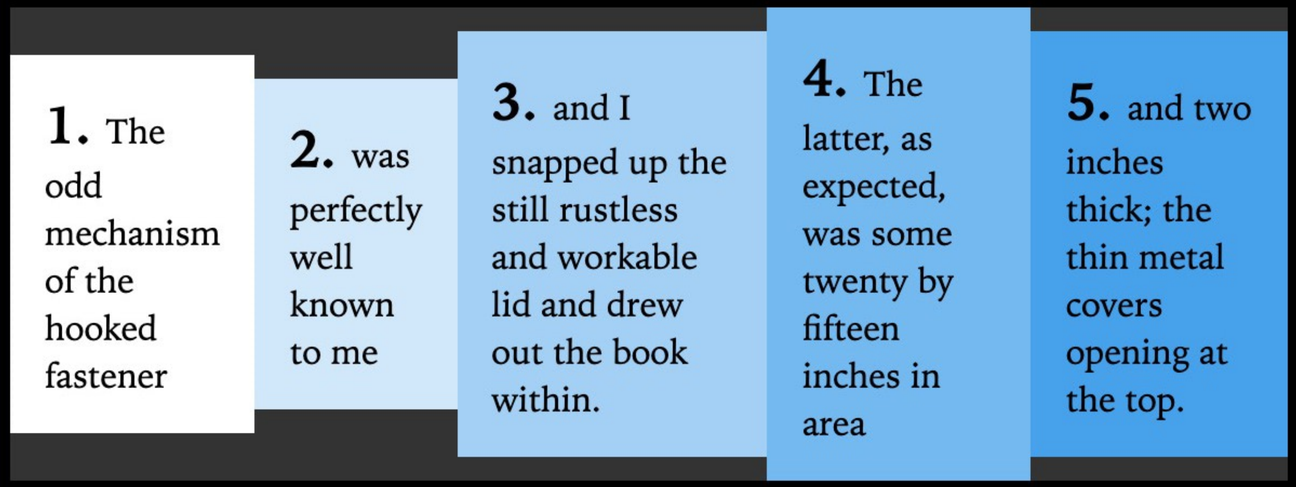
```
<ul class="flex-container">
  <li>The odd mechanism of the hooked fastener</li>
  <li>was perfectly well known to me</li>
  <li>and I snapped up the still rustless and workable lid and drew out the book within.</li>
  <li>The latter, as expected, was some twenty by fifteen inches in area</li>
  <li>and two inches thick; the thin metal covers opening at the top.</li>
</ul>
```

CSS (SCSS) Compiled

```
.flex-container {
  display: flex;
  align-items: center;
}

```

JS



HTML

```html
<ul class="flex-container">
  <li>The odd mechanism of the hooked fastener</li>
</ul>
```

CSS (SCSS) Compiled

```scss
.flex-container {
  display: flex;
  height: 300px;
  justify-content: center;
  align-items: center;
```

JS

1. The odd mechanism of the hooked fastener

Horizontally *and* vertically centered!

`align-items: baseline`

Flex items *stack so that the baselines are aligned* inside the flex container

HTML

```html
<ul class="flex-container">
  <li>The odd mechanism of the hooked fastener</li>
  <li>was perfectly well known to me</li>
  <li>and I snapped up the still rustless and workable lid and drew out the book within.</li>
  <li>The latter, as expected, was some twenty by fifteen inches in area</li>
  <li>and two inches thick; the thin metal covers opening at the top.</li>
</ul>
```

CSS (SCSS) Compiled

```scss
.flex-container {
  display: flex;
  align-items: baseline;
}
.flex-container > *:nth-child(2) {
  font-size: 1.5em;
}
.flex-container > *:nth-child(3) {
  font-size: .8em;
}
.flex-container > *:nth-child(4) {
  line-height: 1.6;
}
```

JS

1. The odd mechanism of the hooked fastener
2. was perfectly well known to me
3. and I snapped up the still rustless and workable lid and drew out the book within.
4. The latter, as expected, was some twenty by fifteen inches in area
5. and two inches thick; the thin metal covers opening at the top.

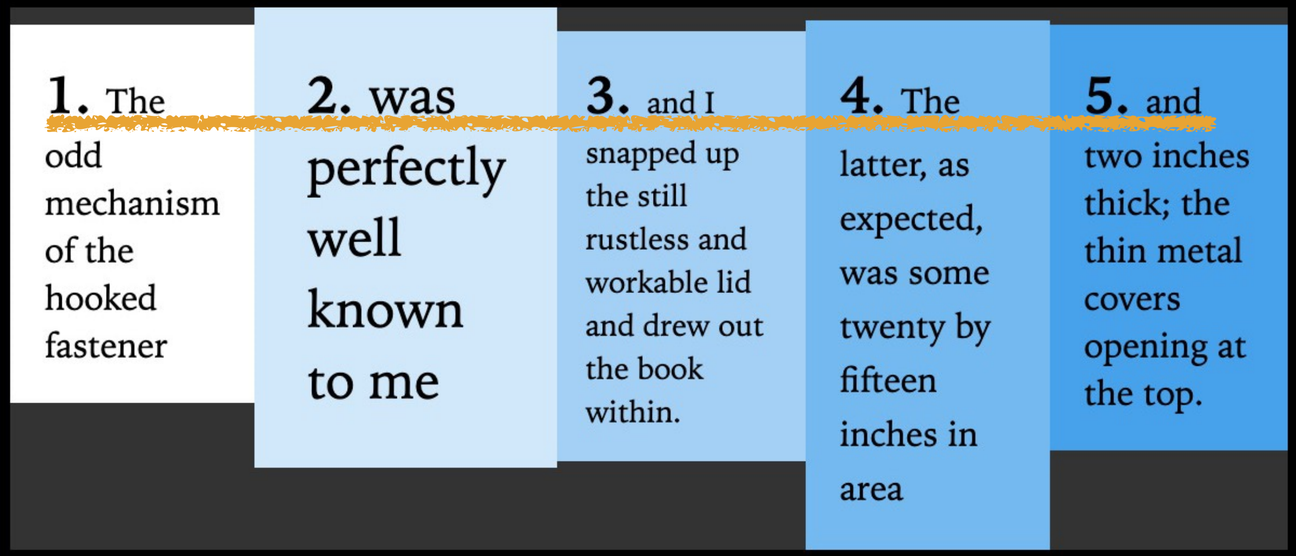
HTML

```
<ul class="flex-container">
  <li>The odd mechanism of the hooked fastener</li>
  <li>was perfectly well known to me</li>
  <li>and I snapped up the still rustless and workable lid and drew out the book within.</li>
  <li>The latter, as expected, was some twenty by fifteen inches in area</li>
  <li>and two inches thick; the thin metal covers opening at the top.</li>
</ul>
```

CSS (SCSS) Compiled

```
.flex-container {
  display: flex;
  align-items: baseline;
}
.flex-container > *:nth-child(2) {
  font-size: 1.5em;
}
.flex-container > *:nth-child(3) {
  font-size: .8em;
}
.flex-container > *:nth-child(4) {
  line-height: 1.6;
}
```

JS



					iOS		
`align-items`	11	12	27	9	9	52	4.4

`align-self`

Allows the default alignment, or the one specified for the flex container by `align-items`, to be overridden for individual flex items

Values for `align-self`:

- `auto` (Default)
- `flex-start`
- `flex-end`
- `center`
- `baseline`
- `stretch`
- `start`
- `end`

`align-self: auto`

Uses the *value of* `align-items` *on the flex container*

If `align-items` isn't set on the flex container,
remember that it defaults to `stretch`

HTML

```
<ul class="flex-container">
  <li>The odd mechanism of the hooked fastener</li>
  <li>was perfectly well known to me</li>
  <li>and I snapped up the still rustless and workable lid and drew out the book within.</li>
  <li>The latter, as expected, was some twenty by fifteen inches in area</li>
  <li>and two inches thick; the thin metal covers opening at the
```

CSS (SCSS) Compiled

```
.flex-container {
  display: flex;
  height: 300px;
  align-items: center;
}
.flex-container > *:nth-child(3) {
  align-self: auto;
}

```

JS

1. The odd mechanism of the hooked fastener
2. was perfectly well known to me
3. and I snapped up the still rustless and workable lid and drew out the book within.
4. The latter, as expected, was some twenty by fifteen inches in area
5. and two inches thick; the thin metal covers opening at the top.

`align-self: flex-start`/`start`

Flex item *stacks from the cross start* of the flex container

`flex-start` is well-supported; the future is `start`

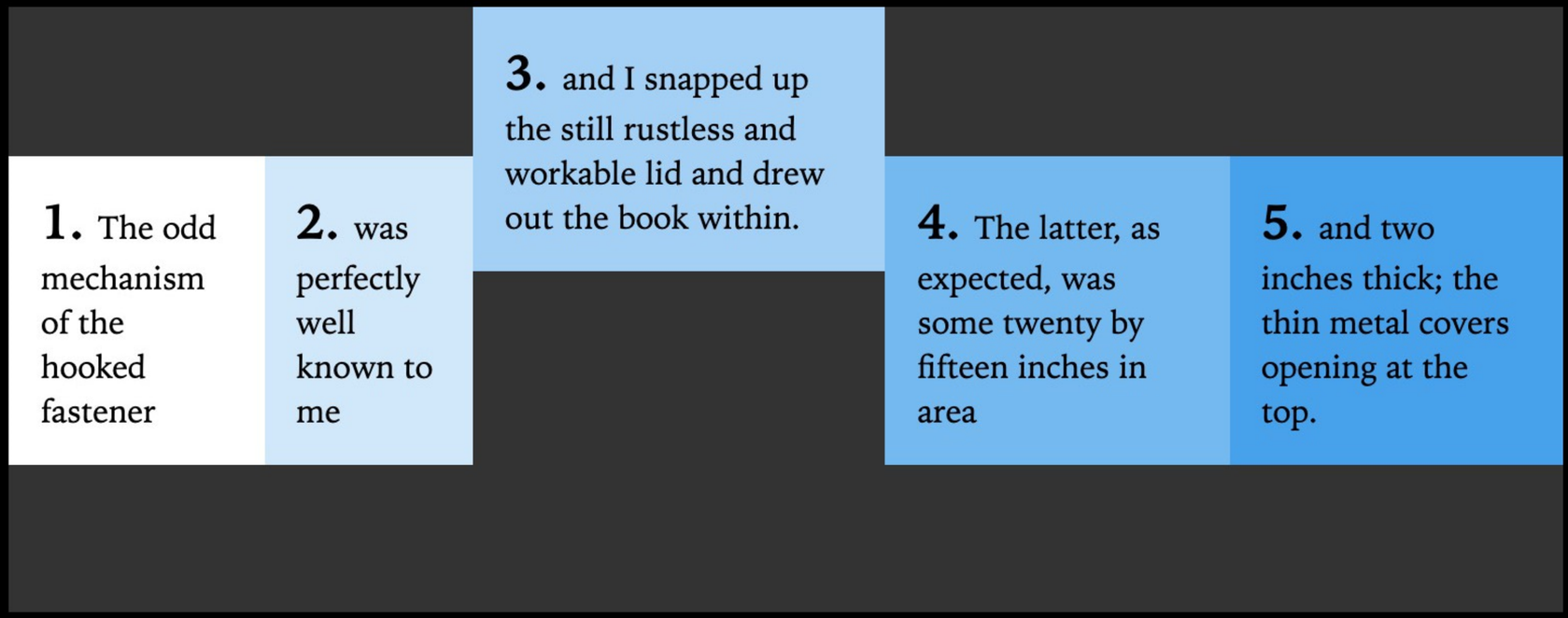
HTML

```
<ul class="flex-container">
  <li>The odd mechanism of the hooked fastener</li>
  <li>was perfectly well known to me</li>
  <li>and I snapped up the still rustless and workable lid and drew out the book within.</li>
  <li>The latter, as expected, was some twenty by fifteen inches in area</li>
  <li>and two inches thick; the thin metal covers opening at the
```

CSS (SCSS) Compiled

```
.flex-container {
  display: flex;
  height: 300px;
  align-items: center;
}
.flex-container > *:nth-child(3) {
  align-self: flex-start;
}
```

JS



`align-self: flex-end`/`end`

Flex item *stacks from the cross end* of the flex container

`flex-end` is well-supported; the future is `end`

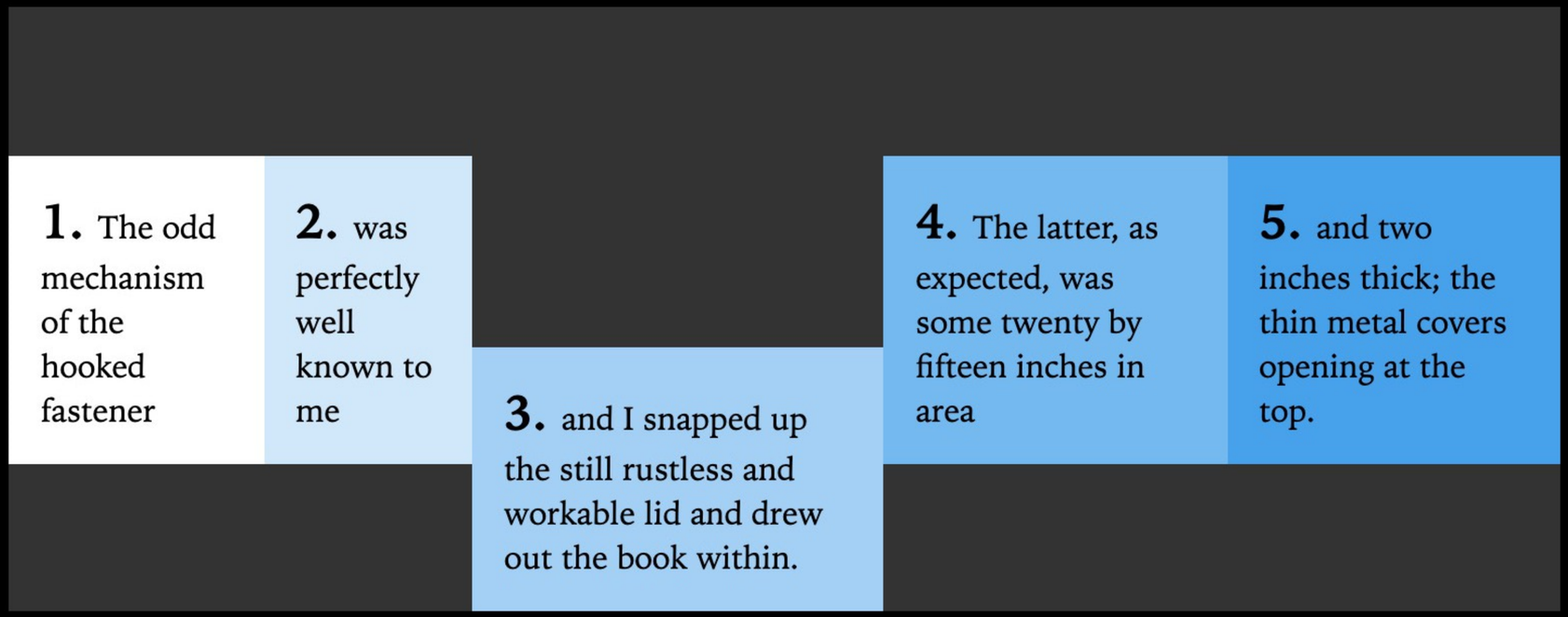
HTML

```
<ul class="flex-container">
  <li>The odd mechanism of the hooked fastener</li>
  <li>was perfectly well known to me</li>
  <li>and I snapped up the still rustless and workable lid and drew out the book within.</li>
  <li>The latter, as expected, was some twenty by fifteen inches in area</li>
  <li>and two inches thick; the thin metal covers opening at the
```

CSS (SCSS) Compiled

```
.flex-container {
  display: flex;
  height: 300px;
  align-items: center;
}
.flex-container > *:nth-child(3) {
  align-self: flex-end;
}
```

JS



`align-self: center`

Flex item *stacks from the center of the cross axis* of the flex container

HTML

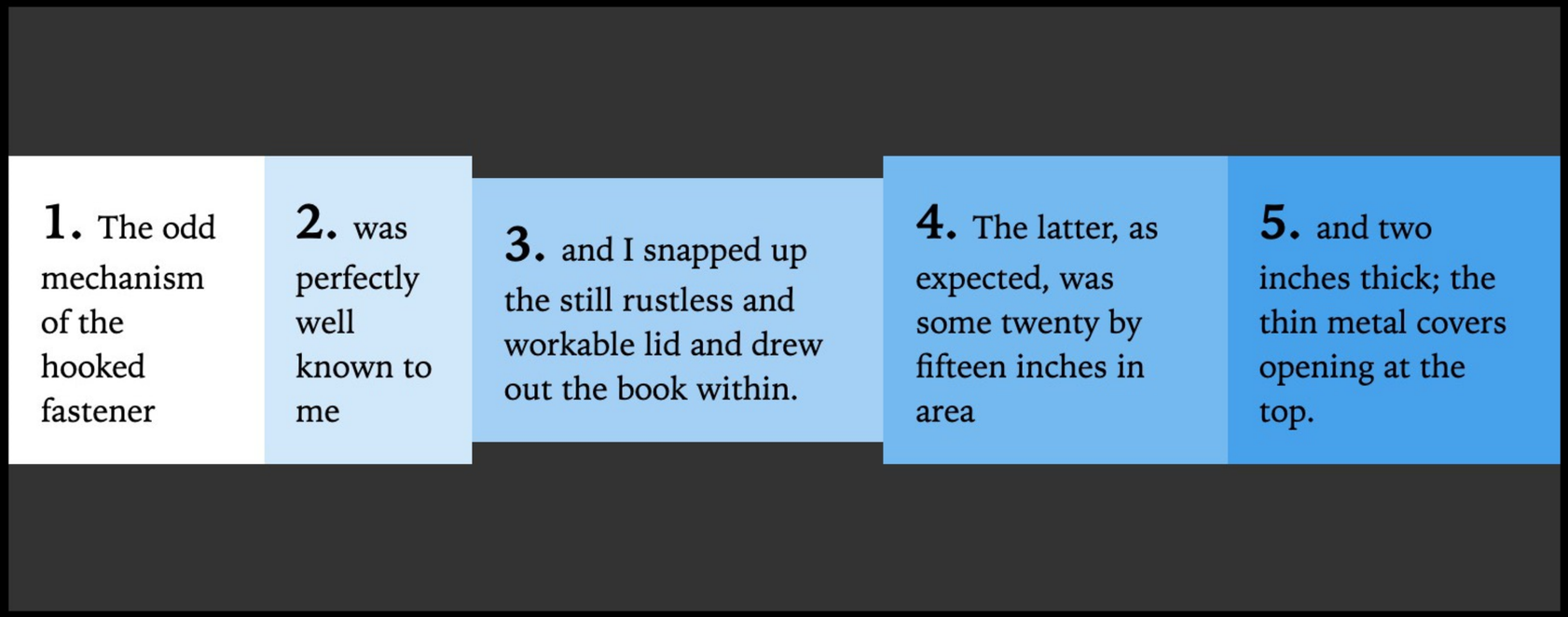
```html
<ul class="flex-container">
  <li>The odd mechanism of the hooked fastener</li>
  <li>was perfectly well known to me</li>
  <li>and I snapped up the still rustless and workable lid and drew out the book within.</li>
  <li>The latter, as expected, was some twenty by fifteen inches in area</li>
  <li>and two inches thick; the thin metal covers opening at the
```

CSS (SCSS) Compiled

```scss
.flex-container {
  display: flex;
  height: 300px;
  align-items: center;
}
.flex-container > *:nth-child(3) {
  align-self: center;
}

```

JS



`align-self: baseline`

Flex item *stacks so that the baselines are aligned* inside the flex container

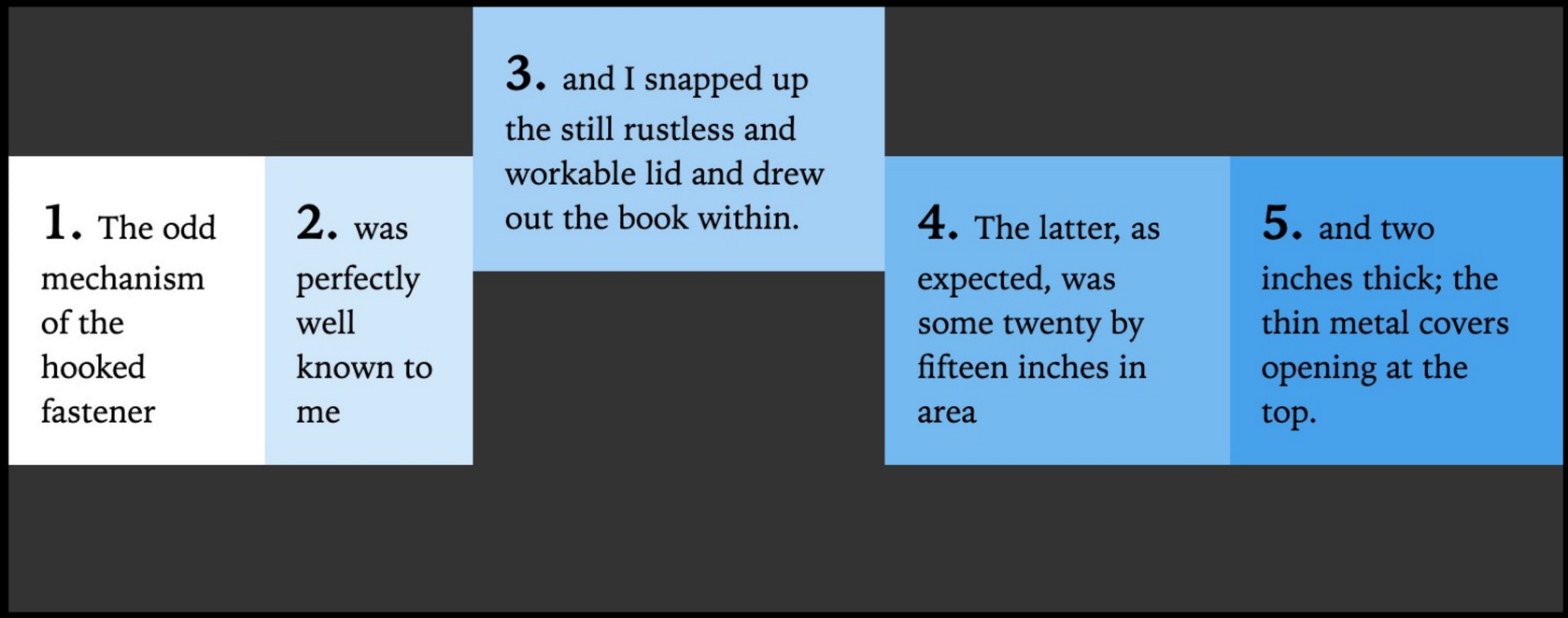
HTML

```
<ul class="flex-container">
  <li>The odd mechanism of the hooked fastener</li>
  <li>was perfectly well known to me</li>
  <li>and I snapped up the still rustless and workable lid and drew out the book within.</li>
  <li>The latter, as expected, was some twenty by fifteen inches in area</li>
  <li>and two inches thick; the thin metal covers opening at the
```

CSS (SCSS) Compiled

```
.flex-container {
  display: flex;
  height: 300px;
  align-items: center;
}
.flex-container > *:nth-child(3) {
  align-self: baseline;
}
```

JS



`align-self: stretch`

Flex item *fills the whole height (or width) from cross start to cross end* of the flex container

Default value for `align-items` on the flex container

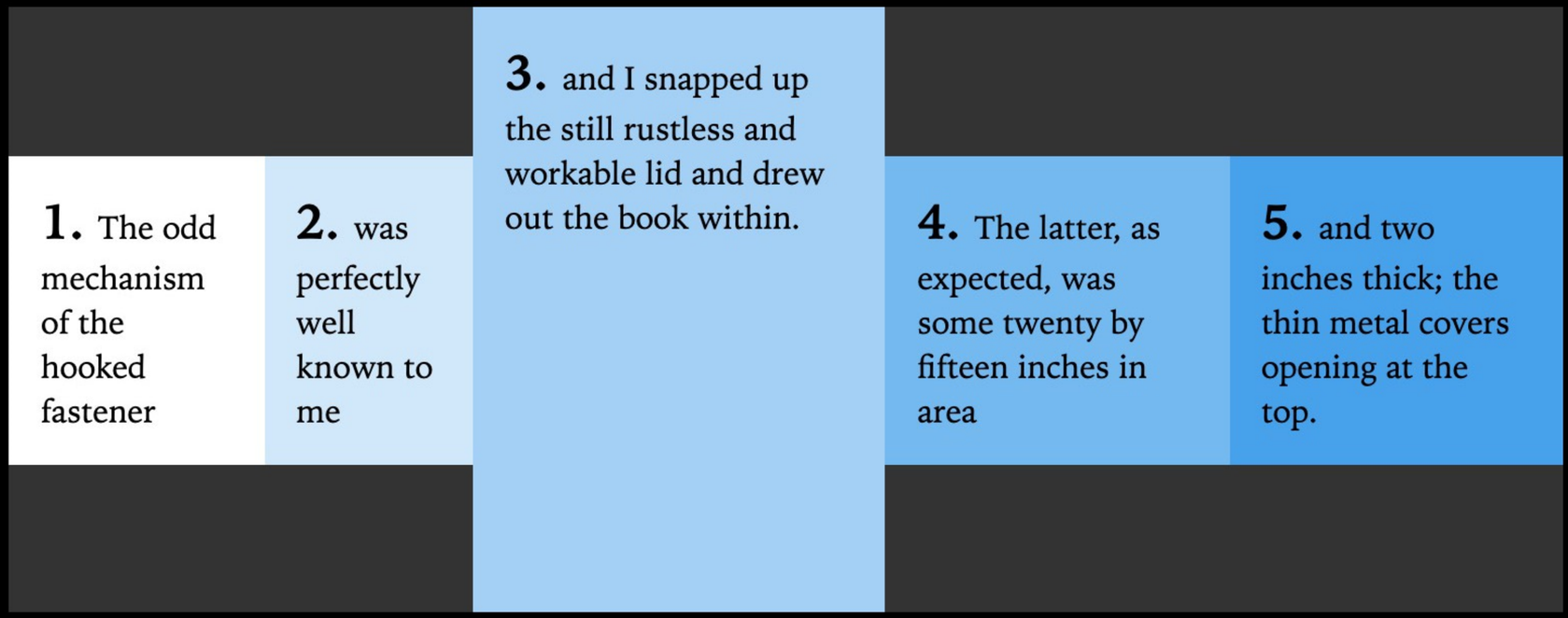
HTML

```
<ul class="flex-container">
  <li>The odd mechanism of the hooked fastener</li>
  <li>was perfectly well known to me</li>
  <li>and I snapped up the still rustless and workable lid and drew out the book within.</li>
  <li>The latter, as expected, was some twenty by fifteen inches in area</li>
  <li>and two inches thick; the thin metal covers opening at the
```

CSS (SCSS) Compiled

```
.flex-container {
  display: flex;
  height: 300px;
  align-items: center;
}
.flex-container > *:nth-child(3) {
  align-self: stretch;
}
```

JS



					iOS		
align-self	11	12	27	9	9	36	4.4
: baseline	–	79	45	–	–	57	Y
: stretch	11	79	52	–	–	57	Y

Aligning Flex Lines on the Cross Axis

`align-content`

Aligns multiple lines of flex items within the flex container when there is extra space in the cross axis

Similar to how `justify-content` aligns individual items within the main axis

HTML

```
<ul class="flex-container">
  <li>Abhoth</li>
  <li>Azathoth</li>
  <li>Cthulhu</li>
  <li>Dagon</li>
  <li>Deep One</li>
  <li>Dunwich Horror</li>
  <li>Formless Spawn</li>
  <li>Hound of Tindalos</li>
  <li>Hunting Horror</li>
  <li>Mi-go</li>
  <li>Nightgaunt</li>
  <li>Nyarlathotep</li>
  <li>Shoggoth</li>
  <li>Shub-Niggurath</li>
  <li>Spider of Leng</li>
```

CSS (SCSS) Compiled

```
.flex-container {
  height: 80vh;
  flex-wrap: wrap;
  display: flex;
  align-content: flex-start;
}
.flex-container > * {
  width: 100px;
}
```

JS

1. Abhoth	2. Azathoth	3. Cthulhu	4. Dagon	5. Deep One	6. Dunwich Horror	7. Formless Spawn
8. Hound of Tindalos	9. Hunting Horror	10. Mi-go	11. Nightgaunt	12. Nyarlathote	13. Shoggoth	14. Shub-Niggurath
15. Spider of Leng	16. Tsathoggua	17. Yig	18. Yog-Sothoth	19. Yuggoth		

What about all this extra space?

Values for `align-content`:

- `stretch` (Default)
- `flex-start`
- `flex-end`
- `center`
- `space-between`
- `space-around`
- `space-evenly`
- `start`
- `end`

`align-content` only effects layout when there are *multiple lines* of flex items inside the flex container & `flex-wrap` is set to `wrap` or `wrap-reverse`

If there is only a single line of flex items, `align-content` has no effect on the layout

`align-content: stretch`

Flex items display with *distributed space after every line* of flex items

Default for `align-content`

HTML

```
<ul class="flex-container">
  <li>Abhoth</li>
  <li>Azathoth</li>
  <li>Cthulhu</li>
  <li>Dagon</li>
  <li>Deep One</li>
  <li>Dunwich Horror</li>
  <li>Formless Spawn</li>
  <li>Hound of Tindalos</li>
  <li>Hunting Horror</li>
  <li>Mi-go</li>
  <li>Nightgaunt</li>
  <li>Nyarlathotep</li>
  <li>Shoggoth</li>
  <li>Shub-Niggurath</li>
  <li>Spider of Leng</li>
```

CSS (SCSS) Compiled

```
.flex-container {
  height: 80vh;
  flex-wrap: wrap;
  display: flex;
  align-content: stretch;
}
.flex-container > * {
  width: 100px;
}
```

JS

1. Abhoth
2. Azathoth
3. Cthulhu
4. Dagon
5. Deep One
6. Dunwich Horror
7. Formless Spawn
8. Hound of Tindalos
9. Hunting Horror
10. Mi-go
11. Nightgaunt
12. Nyarlathote
13. Shoggoth
14. Shub-Niggurath
15. Spider of Leng
16. Tsathoggua
17. Yig
18. Yog-Sothoth
19. Yuggoth

`align-content: flex-start`

Flex items *begin at the cross start* of the flex container based on flex-direction

HTML

```
<ul class="flex-container">
  <li>Abhoth</li>
  <li>Azathoth</li>
  <li>Cthulhu</li>
  <li>Dagon</li>
  <li>Deep One</li>
  <li>Dunwich Horror</li>
  <li>Formless Spawn</li>
  <li>Hound of Tindalos</li>
  <li>Hunting Horror</li>
  <li>Mi-go</li>
  <li>Nightgaunt</li>
  <li>Nyarlathotep</li>
  <li>Shoggoth</li>
  <li>Shub-Niggurath</li>
  <li>Spider of Leng</li>
```

CSS (SCSS) Compiled

```
.flex-container {
  height: 80vh;
  flex-wrap: wrap;
  display: flex;
  align-content: flex-start;
}
.flex-container > * {
  width: 100px;
}
```

JS

1. Abhoth
2. Azathoth
3. Cthulhu
4. Dagon
5. Deep One
6. Dunwich Horror
7. Formless Spawn
8. Hound of Tindalos
9. Hunting Horror
10. Mi-go
11. Nightgaunt
12. Nyarlathote
13. Shoggoth
14. Shub-Niggurath
15. Spider of Leng
16. Tsathoggua
17. Yig
18. Yog-Sothoth
19. Yuggoth

`align-content: flex-end`

Flex items are *stacked at the cross end* of the flex container, but not starting there

HTML

```html
<ul class="flex-container">
  <li>Abhoth</li>
  <li>Azathoth</li>
  <li>Cthulhu</li>
  <li>Dagon</li>
  <li>Deep One</li>
  <li>Dunwich Horror</li>
  <li>Formless Spawn</li>
  <li>Hound of Tindalos</li>
  <li>Hunting Horror</li>
  <li>Mi-go</li>
  <li>Nightgaunt</li>
  <li>Nyarlathotep</li>
  <li>Shoggoth</li>
  <li>Shub-Niggurath</li>
  <li>Spider of Leng</li>
```

CSS (SCSS) Compiled

```scss
.flex-container {
  height: 80vh;
  flex-wrap: wrap;
  display: flex;
  align-content: flex-end;
}
.flex-container > * {
  width: 100px;
}
```

JS

1. Abhoth
2. Azathoth
3. Cthulhu
4. Dagon
5. Deep One
6. Dunwich Horror
7. Formless Spawn
8. Hound of Tindalos
9. Hunting Horror
10. Mi-go
11. Nightgaunt
12. Nyarlathote
13. Shoggoth
14. Shub-Niggurath
15. Spider of Leng
16. Tsathoggua
17. Yig
18. Yog-Sothoth
19. Yuggoth

```
align-content: center
```

Flex items are *stacked in the center of the cross axis* of the flex container

HTML

```
<ul class="flex-container">
  <li>Abhoth</li>
  <li>Azathoth</li>
  <li>Cthulhu</li>
  <li>Dagon</li>
  <li>Deep One</li>
  <li>Dunwich Horror</li>
  <li>Formless Spawn</li>
  <li>Hound of Tindalos</li>
  <li>Hunting Horror</li>
  <li>Mi-go</li>
  <li>Nightgaunt</li>
  <li>Nyarlathotep</li>
  <li>Shoggoth</li>
  <li>Shub-Niggurath</li>
  <li>Spider of Leng</li>
```

CSS (SCSS) Compiled

```
.flex-container {
  height: 80vh;
  flex-wrap: wrap;
  display: flex;
  align-content: center;
}
.flex-container > * {
  width: 100px;
}
```

JS

1. Abhoth
2. Azathoth
3. Cthulhu
4. Dagon
5. Deep One
6. Dunwich Horror
7. Formless Spawn
8. Hound of Tindalos
9. Hunting Horror
10. Mi-go
11. Nightgaunt
12. Nyarlathote
13. Shoggoth
14. Shub-Niggurath
15. Spider of Leng
16. Tsathoggua
17. Yig
18. Yog-Sothoth
19. Yuggoth

`align-content: space-between`

Rows of flex items have *equal spacing between them*, with *first & last rows aligned to the top & bottom edges* of the flex container

HTML

```html
<ul class="flex-container">
  <li>Abhoth</li>
  <li>Azathoth</li>
  <li>Cthulhu</li>
  <li>Dagon</li>
  <li>Deep One</li>
  <li>Dunwich Horror</li>
  <li>Formless Spawn</li>
  <li>Hound of Tindalos</li>
  <li>Hunting Horror</li>
  <li>Mi-go</li>
  <li>Nightgaunt</li>
  <li>Nyarlathotep</li>
  <li>Shoggoth</li>
  <li>Shub-Niggurath</li>
  <li>Spider of Leng</li>
```

CSS (SCSS) Compiled

```scss
.flex-container {
  height: 80vh;
  flex-wrap: wrap;
  display: flex;
  align-content: space-between;
}
.flex-container > * {
  width: 100px;
}
```

JS

1. Abhoth
2. Azathoth
3. Cthulhu
4. Dagon
5. Deep One
6. Dunwich Horror
7. Formless Spawn
8. Hound of Tindalos
9. Hunting Horror
10. Mi-go
11. Nightgaunt
12. Nyarlathote
13. Shoggoth
14. Shub-Niggurath
15. Spider of Leng
16. Tsathoggua
17. Yig
18. Yog-Sothoth
19. Yuggoth

No space after last row (or before 1st row), but equal space between rows

`align-content: space-around`

Rows of flex items have *equal spacing between them*, but space before the first row & after the last row *equals half of the space* between adjacent rows

Similar to `justify-content: space-around`, but focuses on rows instead of flex items

HTML

```
<ul class="flex-container">
  <li>Abhoth</li>
  <li>Azathoth</li>
  <li>Cthulhu</li>
  <li>Dagon</li>
  <li>Deep One</li>
  <li>Dunwich Horror</li>
  <li>Formless Spawn</li>
  <li>Hound of Tindalos</li>
  <li>Hunting Horror</li>
  <li>Mi-go</li>
  <li>Nightgaunt</li>
  <li>Nyarlathotep</li>
  <li>Shoggoth</li>
  <li>Shub-Niggurath</li>
  <li>Spider of Leng</li>
```

CSS (SCSS) Compiled

```
.flex-container {
  height: 80vh;
  flex-wrap: wrap;
  display: flex;
  align-content: space-around;
}
.flex-container > * {
  width: 100px;
}
```

JS

1. Abhoth
2. Azathoth
3. Cthulhu
4. Dagon
5. Deep One
6. Dunwich Horror
7. Formless Spawn
8. Hound of Tindalos
9. Hunting Horror
10. Mi-go
11. Nightgaunt
12. Nyarlathote
13. Shoggoth
14. Shub-Niggurath
15. Spider of Leng
16. Tsathoggua
17. Yig
18. Yog-Sothoth
19. Yuggoth

Space after last row (& before 1st row) is 1/2 of space between rows

`align-content: space-evenly`

Rows of flex items have *equal spacing around them, even the first & last rows*

HTML

```html
<ul class="flex-container">
  <li>Abhoth</li>
  <li>Azathoth</li>
  <li>Cthulhu</li>
  <li>Dagon</li>
  <li>Deep One</li>
  <li>Dunwich Horror</li>
  <li>Formless Spawn</li>
  <li>Hound of Tindalos</li>
  <li>Hunting Horror</li>
  <li>Mi-go</li>
  <li>Nightgaunt</li>
  <li>Nyarlathotep</li>
  <li>Shoggoth</li>
  <li>Shub-Niggurath</li>
  <li>Spider of Leng</li>
```

CSS (SCSS) Compiled

```scss
.flex-container {
  height: 80vh;
  flex-wrap: wrap;
  display: flex;
  align-content: space-evenly;
}
.flex-container > * {
  width: 100px;
}
```

JS

1. Abhoth
2. Azathoth
3. Cthulhu
4. Dagon
5. Deep One
6. Dunwich Horror
7. Formless Spawn
8. Hound of Tindalos
9. Hunting Horror
10. Mi-go
11. Nightgaunt
12. Nyarlathote
13. Shoggoth
14. Shub-Niggurath
15. Spider of Leng
16. Tsathoggua
17. Yig
18. Yog-Sothoth
19. Yuggoth

Space after last row (& before 1st row) is equal to space between rows

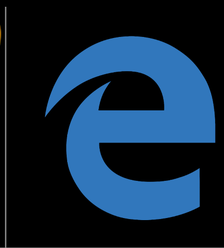
✏️ SIDE NOTE

`place-content`

Shorthand for `align-content` (which aligns *lines* on the *cross* axis) & `justify-content` (which aligns *items* on the *main* axis)

Values: <`align-content`>-values <`justify-content`>-values

If only 1 value is provided, it applies to both

					iOS		
align-content	11	12	28	9	9.2	29	4.4
: stretch	–	79	52	9	9	57	&
: space-evenly	–	79	52	11	11	60	Y
place-content	–	79	45	9	9	59	Y

SCOOBY-DOO & THE DANGERS OF DATA LOSS

Every property previously covered in this section — `justify-content`, `align-content`, `place-content`, `align-items`, & `align-self` — also supports 2 values that work in conjunction with the other values: `safe` & `unsafe`, e.g.:

```
div {
  display: flex;
  align-items: safe center;
  justify-content: safe center;
}
```

HTML

```html
<div>
  <img
src="https://assets.codepen.io
/122116/tsathaggua.jpg" alt="Cthugha
the Sleeper of N'kai">
</div>
```

The viewport is easily big enough for our centered image

CSS (SCSS) Compiled

```scss
div {
  display: flex;
  align-items: center;
  justify-content: center;
}
```

JS

Now make the viewport smaller than the centered image

The image is still centered, therefore all 4 edges go outside the viewport

Let’s see what happens…

HTML

```html
<div>
  <img src="https://assets.codepen.io/122116/tsathaggua.jpg"
alt="Cthugha  the Sleeper of N'kai">
</div>
```

CSS (SCSS) Compiled

```scss
div {
  display: flex;
  align-items: center;
  justify-content: center;
}
```

JS

Uh oh — the top & left of the image is completely out of reach, so that we can’t even scroll to it!

HTML

```
<div>
  <img src="https://assets.codepen.io/122116/tsathaggua.jpg"
alt="Cthugha  the Sleeper of N'kai">
</div>
```

CSS (SCSS) Compiled

```
div {
  display: flex;
  align-items: center;
  justify-content: center;
}
```

JS

but we can scroll to the bottom & right without a problem

Why?

Since we were unable to scroll past the top or left of the viewport to see the top or left of the image, you experience what the W3C calls “data loss”

You can still scroll right & down, however, thanks to the left-to-right *direction* & top-to-bottom *writing mode* of English

Let's enable `safe center` for `align-items` & `justify-content`

Now when scrolling is triggered, the image is no longer centered & is now aligned *as though it was*:

```
align-items: start;
justify-content: start;
```

HTML

```
<div>
  <img src="https://assets.codepen.io/122116/tsathaggua.jpg"
alt="Cthugha  the Sleeper of N'kai">
</div>

```

CSS (SCSS) Compiled

```
div {
  display: flex;
  align-items: center;
  justify-content: center;
  align-items: safe center;
  justify-content: safe center;
}
```

JS

Now we can see the top & left of the image (before, it was out of reach)…

HTML

```
<div>
  <img src="https://assets.codepen.io/122116/tsathaggua.jpg"
alt="Cthugha  the Sleeper of N'kai">
</div>

```

CSS (SCSS) Compiled

```
div {
  display: flex;
  align-items: center;
  justify-content: center;
  align-items: safe center;
  justify-content: safe center;
}
```

JS

& the bottom & right, like we always could

					iOS		
`align-items: safe` & `unsafe`	–	–	63	–	–	–	–
`justify-content: safe` & `unsafe`	–	–	63	–	–	–	–

Since only Firefox supports `safe` & `unsafe` for now, make sure your design handles the problem via other responsive behaviors such as image resizing & word wrapping

Marathon Man (1976)

Aligning via Writing Mode

Every property previously covered in this section — `justify-content`, `align-content`, `place-content`, `align-items`, & `align-self` — also supports 2 values: `start` & `end`, e.g.:

```
div {
  display: flex;
  justify-content: start;
}
```

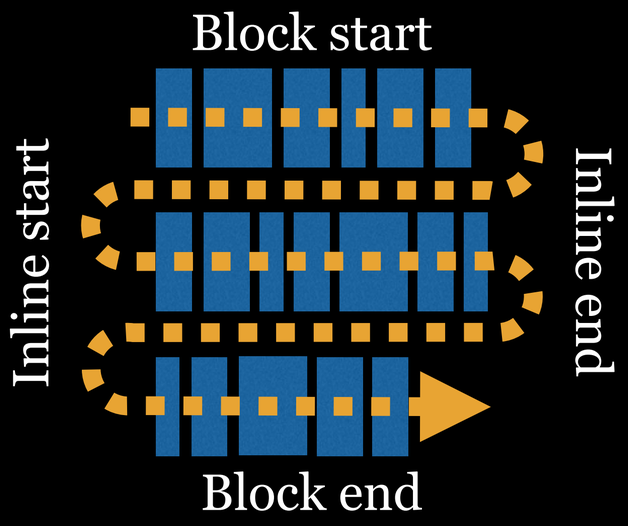
`start`

Flex items are *aligned at the logical start edge* of the flex container relative to reading direction or writing mode

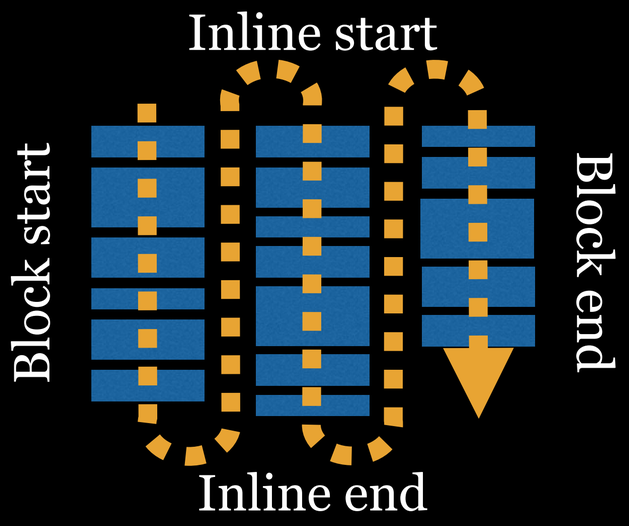
`end`

Flex items are *aligned at the logical end edge* of the flex container relative reading direction or writing-mode

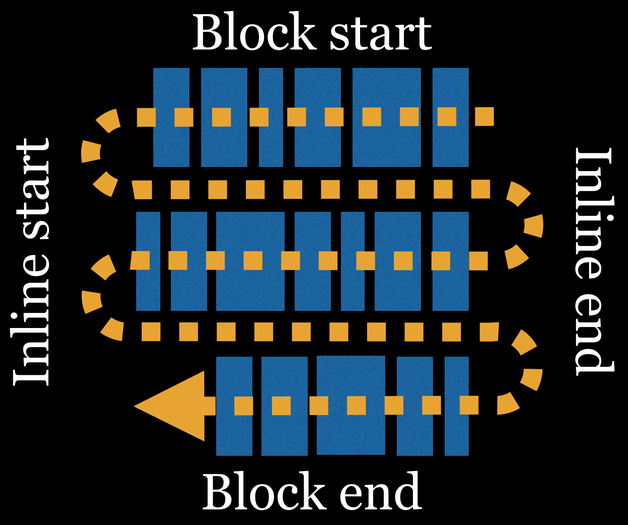
Latin- & Han-based



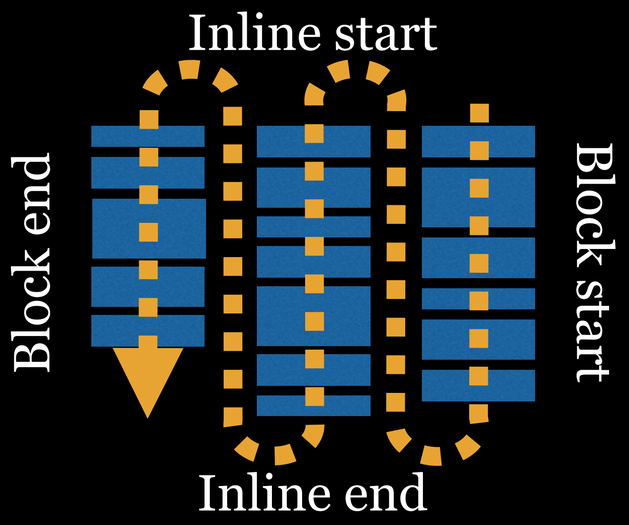
Mongolian-based



Arabic-based



Han-based



					iOS		
`justify-content: start & end`	–	–	45	9	9	–	–
`align-items: start & end`	–	–	45	–	–	–	–
`align-self: start & end`	–	79	45	–	–	57	Y
`align-content: start & end`	–	–	45	–	–	–	–
`place-content: start & end`	?	?	?	?	?	?	?

Gutters

`column-gap`

Defines minimum size of space (the *gutter*) between items

`row-gap`

Defines minimum size of gutter between wrapped lines

`gap`

Defines minimum size of gutter between rows & columns

Shorthand for setting `grid-column-gap` & `row-gap`

					iOS		
`column-gap`	–	–	63	3.1*	3.2*	–	–
`row-gap`	–	–	63	–	–	–	–
`gap`	–	–	63	–	–	84	–

* Requires `-webkit-` prefix

💡 PRO TIP

Due to lack of support, for now use fixed-length margins — e.g., `margin-left: 20px;` — but understand that this brings with them limitations & complications

Limitations: since you can't do a first or last line selector, you cannot exclude margins from the first or last line if wrapping occurs

Complications: if wrapping is not going to occur, you need to use `:nth-child`, `:first-child`, or `:last-child`, as needed

Or just use grid if the design allows

margin: auto

`margin: auto`

Takes all available space (horizontally & vertically) within a flex line

Helpful since `justify-self` property isn’t available in Flexbox

HTML

```
<ul class="flex-container">
  <li>Abhoth</li>
  <li>Azathoth</li>
  <li>Cthulhu</li>
  <li>Dagon</li>
  <li class="flex-end">Deep One</li>
</ul>

<ul class="flex-container">
  <li>Dunwich Horror</li>
  <li>Formless Spawn</li>
  <li class="space-around">Hound of Tindalos</li>
  <li>Mi-go</li>
</ul>
```

CSS (SCSS) Compiled

```
.flex-container {
  display: flex;
}
.flex-container .flex-end {
  margin-left: auto;
}
.flex-container .space-around {
  margin: auto;
}
```

JS

1. Abhoth
2. Azathoth
3. Cthulhu
4. Dagon
5. Deep One

1. Dunwich Horror
2. Formless Spawn
3. Hound of Tindalos
4. Mi-go

HTML

```html
<ul class="flex-container">
  <li>Abhoth</li>
  <li>Azathoth</li>
  <li>Cthulhu</li>
  <li>Dagon</li>
  <li class="flex-end">Deep One</li>
</ul>
```

CSS (SCSS) Compiled

```scss
.flex-container {
  min-height: 100vh;
  display: flex;
  flex-direction: column;
}
.flex-container .flex-end {
  margin-top: auto;
}
```

JS

1. Abhoth
2. Azathoth
3. Cthulhu
4. Dagon
5. Deep One

HTML

```
  <div class="listed-item">
    <i class="fa fa-heart-o"></i>
    <p class="listed-item_title">My Favorite Tee</p>
    <p class="listed-item_price">$100</p>
  </div>
```

CSS (SCSS) Compiled

```
}
.listed-item .fa {
  color: #EE96AE;
  font-size: 24px;
  flex-basis: 45px;
}
.listed-item > p {
  font-family: 'Bitter', serif;
  font-size: 21px;
}
.listed-item .listed-item_title {
  color: #5F5F5F;
}
.listed-item .listed-item_price {
  color: #54CACD;
  font-weight: bold;
  margin-left: auto;
}
```

JS

Sizing Flex Items

flex-grow

flex-shrink

flex-basis

flex

By default, flex items are sized to their content, similarly to table cells

💡 PRO TIP

Never use `width` & `height` with flexbox layout! Instead of `width` & `height`, size items using these for the cross axis:

- `align-items: stretch`
- `align-content: stretch`
- `align-self: stretch`

And this for the main axis:

- `flex-basis`

We're going to cover `flex-basis` in this section

`flex-grow`

Specifies *if a flex item can grow if necessary to take up available space* inside its flex container

Value: `<number>`

» `0`: do not grow (default)
» Positive `<number>`: flex item can grow

Any positive number for `flex-grow` allows the flex item to grow to fill all available space

All flex items that have the same number grow the same amount

If those numbers differ, flex items grow *proportionally* based upon those numbers, with larger numbers growing more

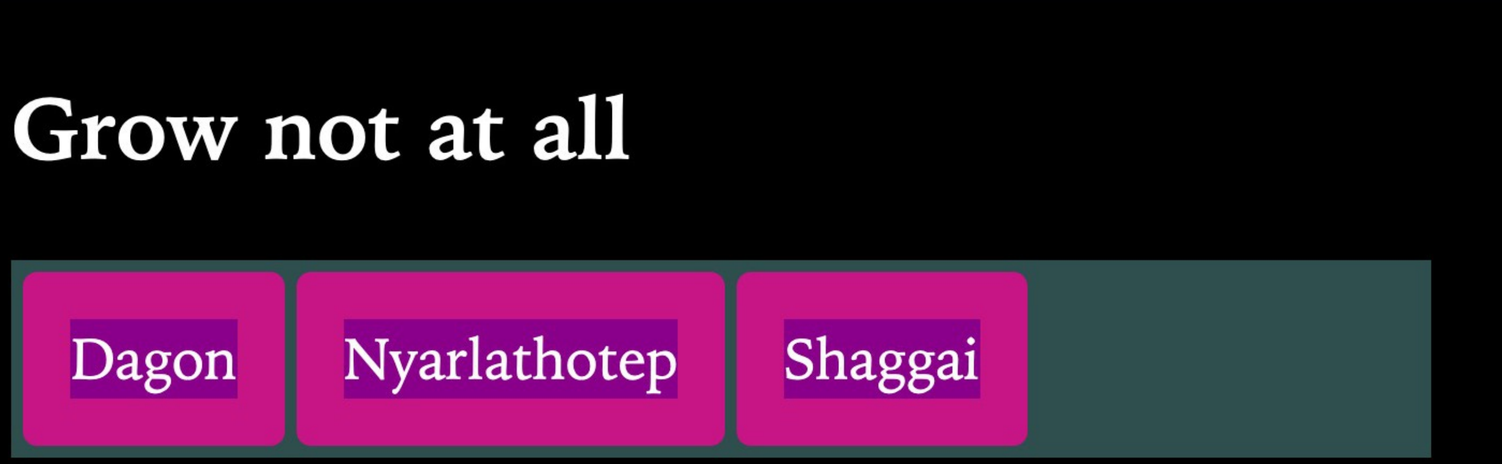
HTML

```html
<h2>Grow not at all</h2>
<ul>
  <li><span>Dagon</span></li>
  <li><span>Nyarlathotep</span></li>
  <li><span>Shaggai</span></li>
</ul>
```

CSS (SCSS) Compiled

```scss
ul {
  display: flex;
}
ul li {
  border: 20px solid mediumvioletred;
  background-color: deeppink;
}
ul li span {
  background-color: darkmagenta;
}
```

JS



HTML

```
<h2>Grow evenly</h2>
<ul class="grow-same">
  <li><span>Dagon</span></li>
  <li><span>Nyarlathotep</span></li>
  <li><span>Shaggai</span></li>
</ul>
```

CSS (SCSS) Compiled

```
ul.grow-same li {
  flex-grow: 1;
}
```

JS

Grow not at all

Dagon Nyarlathotep Shaggai

Grow evenly

Dagon Nyarlathotep Shaggai

HTML

```
<h2>Grow differently</h2>
<ul class="grow-different">
  <li><span>Dagon</span></li>
  <li><span>Nyarlathotep</span></li>
  <li><span>Shaggai</span></li>
</ul>
```

CSS (SCSS) Compiled

```
ul.grow-different *:nth-child(1) {
  flex-grow: 1;
}
ul.grow-different *:nth-child(2) {
  flex-grow: 2;
}
ul.grow-different *:nth-child(3) {
  flex-grow: 3;
}
```

JS

Grow not at all

Dagon Nyarlathotep Shaggai

Grow evenly

Dagon Nyarlathotep Shaggai

Grow differently

Dagon Nyarlathotep Shaggai

`flex-shrink`

Specifies *if a flex item can shrink if necessary to fit* inside its flex container

Value: `<number>`

- `1`: all flex items can shrink to fit (default)
- `0`: do not shrink below the original size
- Positive `<number>`: flex item can shrink

Any positive number for `flex-shrink` allows the flex item to shrink to fit within its flex container

All flex items that have the same number shrink the same amount

If those numbers differ, flex items shrink *proportionally* based upon those numbers, with larger numbers shrinking more

HTML

```html
<h2>Shrink evenly</h2>
<ul class="shrink-same">
  <li><span>The Blackness from the Stars</span></li>
  <li><span>The Blackness from the Stars</span></li>
  <li><span>The Blackness from the Stars</span></li>
</ul>
```

CSS (SCSS) Compiled

```scss
ul.shrink-same li {
  flex-shrink: 1;
}
```

JS

Shrink evenly

- The Blackness from the Stars
- The Blackness from the Stars
- The Blackness from the Stars

HTML

```
<h2>Shrink not at all</h2>
<ul class="shrink-not-at-all">
  <li><span>The Blackness from the Stars</span></li>
  <li><span>The Blackness from the Stars</span></li>
  <li><span>The Blackness from the Stars</span></li>
</ul>
```

CSS (SCSS) Compiled

```
ul.shrink-not-at-all li {
  flex-shrink: 0;
}
```

JS

Shrink evenly

The Blackness from the Stars

The Blackness from the Stars

The Blackness from the Stars

Shrink not at all

The Blackness from the Stars

The Blackness from the Stars

HTML

```
<h2>Shrink different</h2>
<ul class="shrink-different">
  <li><span>The Blackness from the Stars</span></li>
  <li><span>The Blackness from the Stars</span></li>
  <li><span>The Blackness from the Stars</span></li>
</ul>
```

CSS (SCSS) Compiled

```
ul.shrink-different *:nth-child(1) {
  flex-shrink: 1;
}
ul.shrink-different *:nth-child(2) {
  flex-shrink: 2;
}
ul.shrink-different *:nth-child(3) {
  flex-shrink: 3;
}
```

JS

Shrink evenly

The Blackness from the Stars

The Blackness from the Stars

The Blackness from the Stars

Shrink not at all

The Blackness from the Stars

The Blackness from the Stars

Shrink different

The Blackness from the Stars

The Blackness from the Stars

The Blackness from the Stars

Note that…

» the default for `flex-grow` is `0`: do not grow
» the default for `flex-shrink` is `1`: shrink as necessary

💡 PRO TIP

We have said not to use `width` or `height` — & we mean that

You will, however, need to use `max-width` & `min-width` on flex items to limit their maximum growth or shrinking

`flex-basis`

Defines *initial main size of a flex item* before changes caused by `flex-grow` or `flex-shrink` are applied

Values:

- `auto` (Default)
- `content`
- `max-content`
- `min-content`
- `<width>`

What about `width`/`height`? Why use `flex-basis` instead?

- `flex-basis` avoids content overflow problems
- `flex-basis` works along the main axis, so you don't need to worry about `flex-direction` vs `width`/`height`
- `flex-basis` is designed to work together with `flex-grow` & `flex-shrink`
- `flex-basis` is part of the `flex` shorthand, which the W3C recommends you use

`flex-basis: auto`

Sizes flex items *based on their content*

Warning: this assumes you have not used `width`/`height`, which we told you not to do!

If you do have `width`/`height` set, `auto` uses that value instead (see why we told you not to use it?)

`flex-basis: content`

Sizes flex items *according to content while ignoring* *`width`*/*`height`*

Only supported in a few browsers!

You won't need this because you won't use `width`/`height`, right? Right? Riiiiiight?

`flex-basis: max-content`

Sizes flex items to *longest the content can be* (if it were not wrapped)

`flex-basis: min-content`

Sizes flex items to *narrowest the content can be*, e.g., the longest word or image

`flex-basis: <length>|<percentage>`

Overrides the CSS `height` or `width` properties of a flex item (as if you were to disobey us & use them! Never!)

Can be either a `<length>` (e.g., `px`, `em`, `rem`, `vh`) or `<percentage>` of the flex container

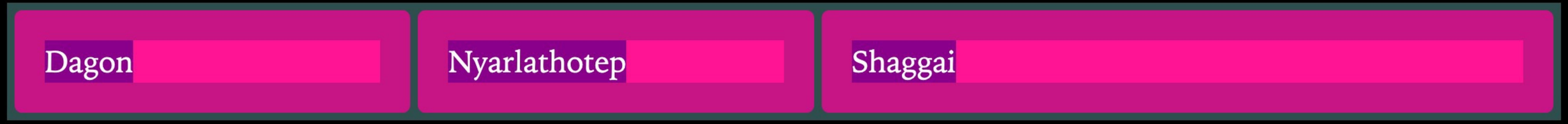
flex-basis: 200px

flex-basis: 125px

flex-basis: auto & flex-grow: 1, 1, 2

flex-basis: 0

flex-basis: 0 & flex-grow: 1, 1, 2



`flex-grow` & `flex-shrink`

`flex-basis` (not `auto`)

`width` or `height` (No!)

Content (aka `flex-basis: auto`, the default)

Order of sizing effect for flex items

`flex`

Instead of `flex-grow`, `flex-shrink`, & `flex-basis`, the W3C encourages the use of the shorthand `flex` instead

Why?

- `flex-grow`, `flex-shrink`, & `flex-basis` always work together when it comes to sizing flex items, & `flex` combines them intelligently
- `flex` sets values that make sense for common uses

`flex` values:

» `<flex-grow>` `<flex-shrink>` `<flex-basis>`
» `initial` (Default)
» `auto`
» `none`
» `<number>`
» `<length>`|`<percentage>`

`flex` takes 1, 2, or 3 values

1 is a keyword or `<number>`

2 (e.g., `1 25px`), but this is really confusing

3 for `<flex-grow> <flex-shrink> <flex-basis>`

Of these, we recommend 1 or 3

Single value	<flex-grow> <flex-shrink> <flex-basis>
initial	0 1 auto
auto	1 1 auto
none	0 0 auto
<number>	<number> 1 0
<length>	1 1 <length>
<percentage>	1 1 <percentage>

`flex: initial`

Equivalent to `0 1 auto`

» Do not grow
» Shrink if not enough space in container
» Size based on content

Default for `flex`

`flex: auto`

Equivalent to `1 1 auto`

» Grow to absorb free space
» Shrink if not enough space
» Size based on content

} Fully flexible!

`flex: none`

Equivalent to `0 0 auto`

» Do not grow
» Do not shrink
} Inflexible!
» Size based on content

`flex: <number>`

Equivalent to `<number> 1 0`

- Grow to absorb free space
- Shrink if not enough space in container
- Base size is *not* based on content, so growing & shrinking occur without content in mind

Typically this will results in all flex items having the same size

					iOS		
`flex-grow`	11	12	28	9	9.2	29	4.4
`flex-shrink`	11	12	28	9	9.2	29	4.4
`flex-basis`	11*	12	28	9	9.2	29	4.4
`flex`	11	12	28	9	9.2	29	4.4

* See note at chnsa.ws/1ii

flex-basis					iOS		
: auto	11	12	18	7	7	22	Y
: content	–	12*	61	–	–	–	–
: max-content	–	–	66	–	–	–	–
: min-content	–	–	66	–	–	–	–

* No longer supported as of 79!

References

philipwalton / flexbugs

Watch 110 Star 1,637 Fork 40

A community-curated list of flexbox issues and cross-browser workarounds for them.

37 commits 2 branches 0 releases 3 contributors

branch: master flexbugs / +

Update flexbug 11's language and demos.

philipwalton authored 19 days ago latest commit cd3eb48e9a

LICENSE	Initial commit	5 months ago
README.md	Update flexbug 11's language and demos.	19 days ago

README.md

Flexbugs

This repository is a community-curated list of flexbox issues and cross-browser workarounds for them. The goal is that if you're building a website using flexbox and something isn't working as you'd expect, you can find the solution here.

As the spec continues to evolve and vendors nail down their implementations, this repo will be updated with newly discovered issues and remove old issues as they're fixed or become obsolete. If

Code
Issues
Pull requests
Wiki
Pulse
Graphs

HTTPS clone URL

https://github.com/

You can clone with HTTPS, SSH, or Subversion.

Clone in Desktop

Download ZIP

Guide

A Complete Guide to Flexbox

Last Updated
Jun 12, 2020

Our comprehensive guide to CSS flexbox layout. This complete guide explains everything about flexbox, focusing on all the different possible properties for the parent element (the flex container) and the child elements (the flex items). It also includes history, demos, patterns, and a browser support chart.

Get the poster!

Reference this guide a lot? Pin a copy up on the office wall.

Examples

Solved *by* Flexbox

Cleaner, hack-free CSS

View Project Source Spread the Word

Introduction

CSS has been lacking proper layout mechanisms for far too long. Transitions, animations, filters, all of these are great and useful additions to the language, but they don't address the major problems that Web developers have been complaining about for what seems like an eternity.

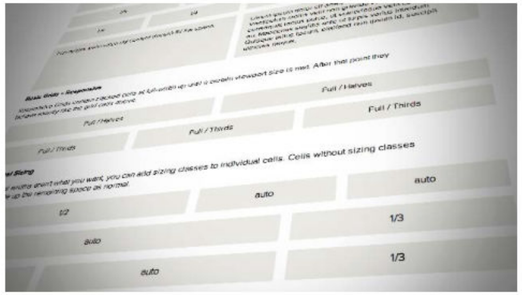
Finally, thanks to Flexbox, we have a solution.

This site is not another CSS framework. Instead, its purpose is to showcase problems once hard or impossible to solve with CSS alone, now made trivially easy with Flexbox. And with the recent release of Internet Explorer 11 and Safari 6.1, the latest Flexbox spec is now supported in every modern browser.

Check out the demos below. View the styles in the Web inspector or dive into the source to see just how easy CSS layout will become once Flexbox becomes mainstream.

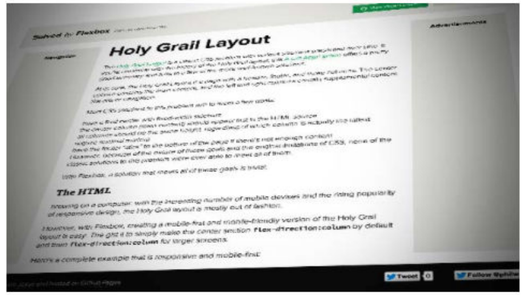
Showcase

Showcase



Better, Simpler Grid Systems

Flexbox gives us most of the features we want from a grid system out of the box. And sizing and alignment are just one or two properties away.

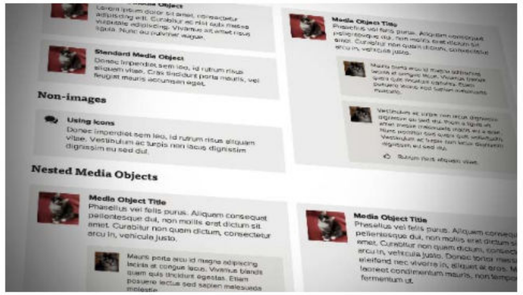


Holy Grail Layout

This classic problem has been challenging CSS hackers for years, yet none of the historical solutions have fully solved it. With Flexbox, it's finally possible.

Input Add-ons

Creating full-width, fluid input/button pairs has been impossible for most of the history of CSS. With Flexbox it couldn't be easier.

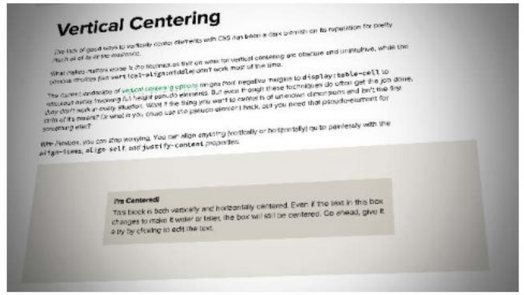


Media Object

Create media objects with fixed or varying figure sizes without worrying about overflow, clearfixing, or block formatting context hacks.

Sticky Footer

Getting your footer to stick to the bottom of sparsely contented pages has always been tricky. And if the footer's height is unknown, it's basically impossible. Not so anymore.



Vertical Centering

This classic problem has been challenging CSS hackers for years, yet none of the historical solutions have fully solved it. With Flexbox, it's finally possible.

Flexbox Playground

Children Width

width: 12%

Parent Flex Properties – flex container

flex-direction

- (•) row
- ( ) row-reverse
- ( ) column
- ( ) column-reverse

flex-wrap

- (•) nowrap
- ( ) wrap
- ( ) wrap-reverse

justify-content

- (•) flex-start
- ( ) flex-end
- ( ) center
- ( ) space-between
- ( ) space-around

align-items

- (•) stretch
- ( ) flex-start
- ( ) flex-end
- ( ) center
- ( ) baseline

align-content

- (•) stretch
- ( ) flex-start
- ( ) flex-end
- ( ) center
- ( ) space-between
- ( ) space-around

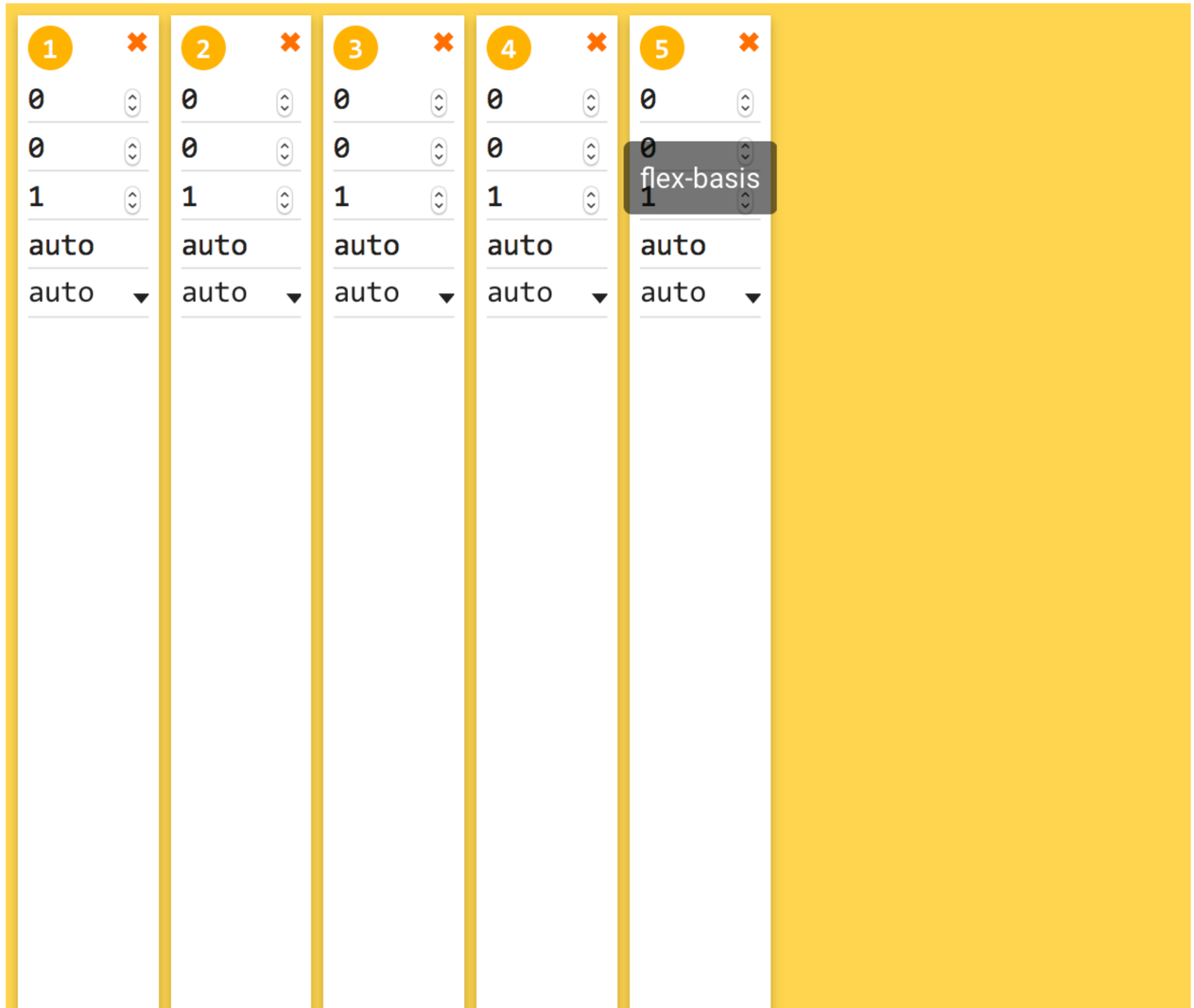
* The default properties are highlighed

Children Flex Properties – flex items

The children flex properties can be applied at child level, separate for each child. See the results below and change some of their properties. Hover with the mouse pointer or touch the fields to see the property name.

Result

ADD CHILD



MadnessBook

Cthulhu is currently sleeping under the sea at R'yleh. Do not disturb. Or else.

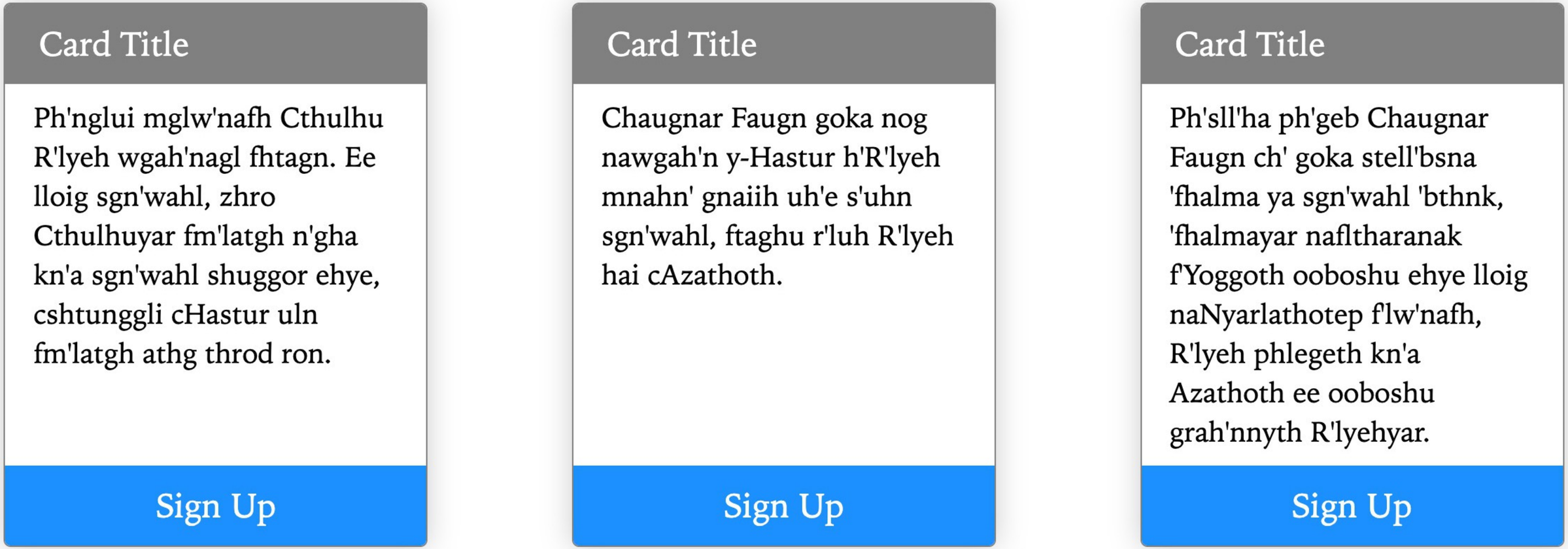
It is a monster of vaguely anthropoid outline, but with an octopus-like head whose face was a mass of feelers, a scaly, rubbery-looking body, prodigious claws on hind and fore feet, and long, narrow wings behind.

Azathoth, the boundless daemon sultan whose name no lips dare speak aloud, & who gnaws hungrily in inconceivable, unlighted chambers beyond time & space.

We're going to make this in class

Abdul Alhazred, the "Mad Arab" who wrote the *Kitab al-Azif* (the *Necronomicon*). Devoured by an invisible beast in broad daylight in a public marketplace.

Nyarlathotep frequently walks the Earth in the guise of a human being, usually a tall, dark, slim man. He has a thousand other forms, most reputed to be maddeningly horrific.



We’re going to make this in class

Thank you!

scott@granneman.com
www.granneman.com
ChainsawOnATireSwing.com
@scottgranneman

jans@websanity.com
websanity.com

Flexbox Layout

Flexible, Robust Line-Based Layout

R. Scott Granneman & Jans Carton

Changelog

2020-07-13 3.1: Changed subtitle

2020-07-10 3.0: (con't. from ↓) added new section *Scooby-Doo & The Dangers of Data Loss* covering `safe`; added new section *Aligning via Writing Mode* covering `start` & `end`

Changelog

2020-07-10 3.0: Moved Flexbox out of CSS Layout into its own slide deck; added diagram explaining basic Flexbox flow; completely re-did all screenshots; re-did almost all compatibility charts; added new section *Direction, Wrapping, & Order*; better examples for using multiple `flex-direction`; created new section *Aligning Lines & Items*; lots of warnings not to use `width` & `height`; renamed *Sources* to *References*; added CSS-Tricks' "A Complete Guide to Flexbox"; added 2 assignments to Examples; (con't. ↑)