SESSION FOUR
SEPTEMBER 19, 2023

LAYOUTS I: THE BOX MODEL

ENDLESS APPRECIATION
TO ERIC LI +
MICHAEL FEHRENBACH
FOR THE MATERIAL

DAVIS SCHERER
DSCHERER@NEWSCHOOL.EDU

HC UPDATES

PLEASE submit your entries on time, whatever you have.

Late entries will get a 0 and no feedback, starting next week.

HC UPDATES

I left comments on several of your all's work that it's ok to expand and adapt your concepts if you've boxed yourself into a corner. **IT IS NOT TOO LATE, BUT IT WILL BE SOON.**

HC UPDATES

Remember that each entry should be **unique**, and explore **different aspects** of your concept.

THESE SHOULD NOT BE TEMPLATED ENTRIES.

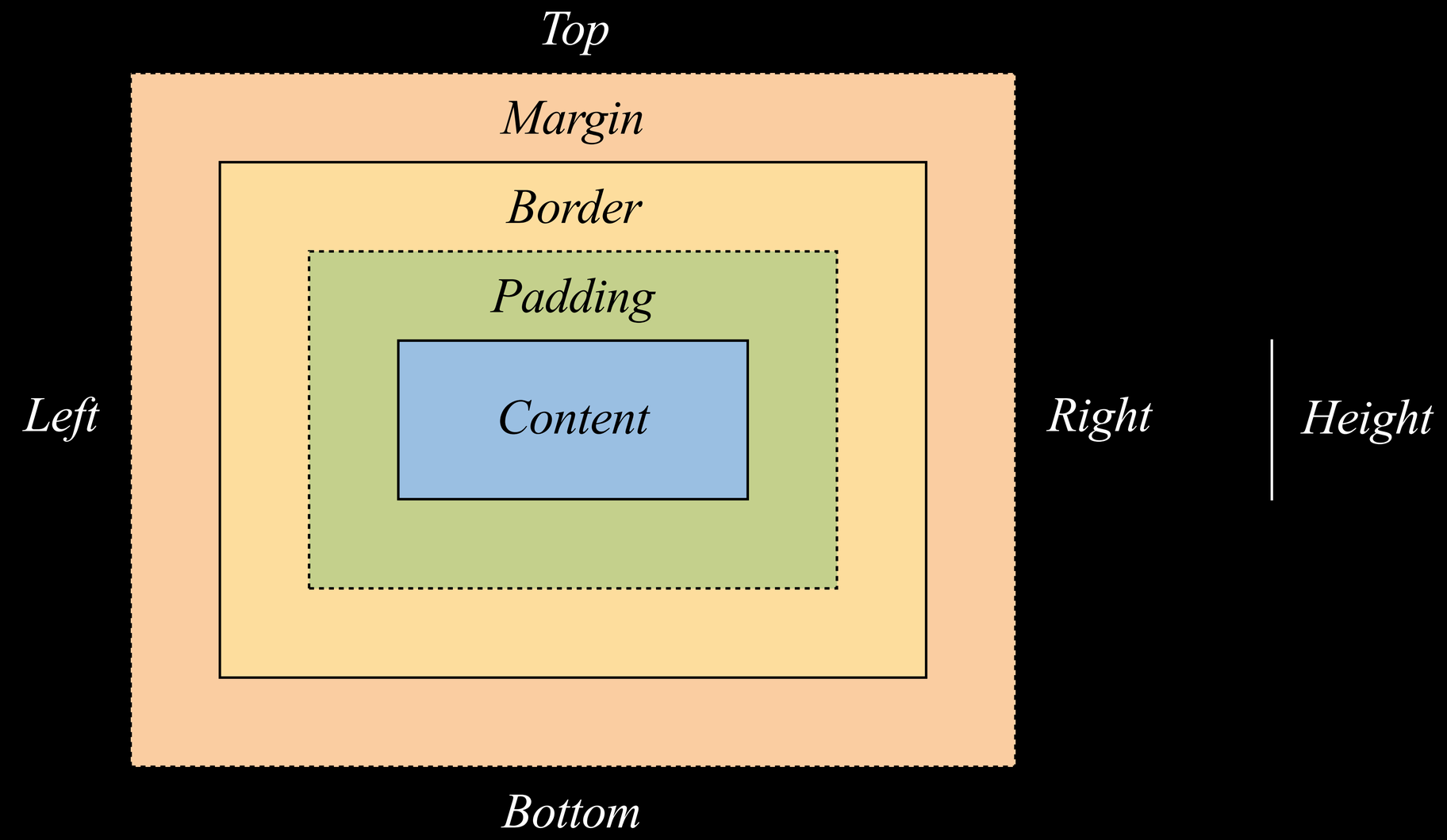
BOX MODEL

CSS sizes elements based on what is called the ‘box model.’

BOX MODEL

We talked a bit about block elements—basically, everything on the web begins as a rectangle.

BOX MODEL


Top
Margin
Border
Padding
Content
Left
Right
Height
Bottom
Width

BOX MODEL

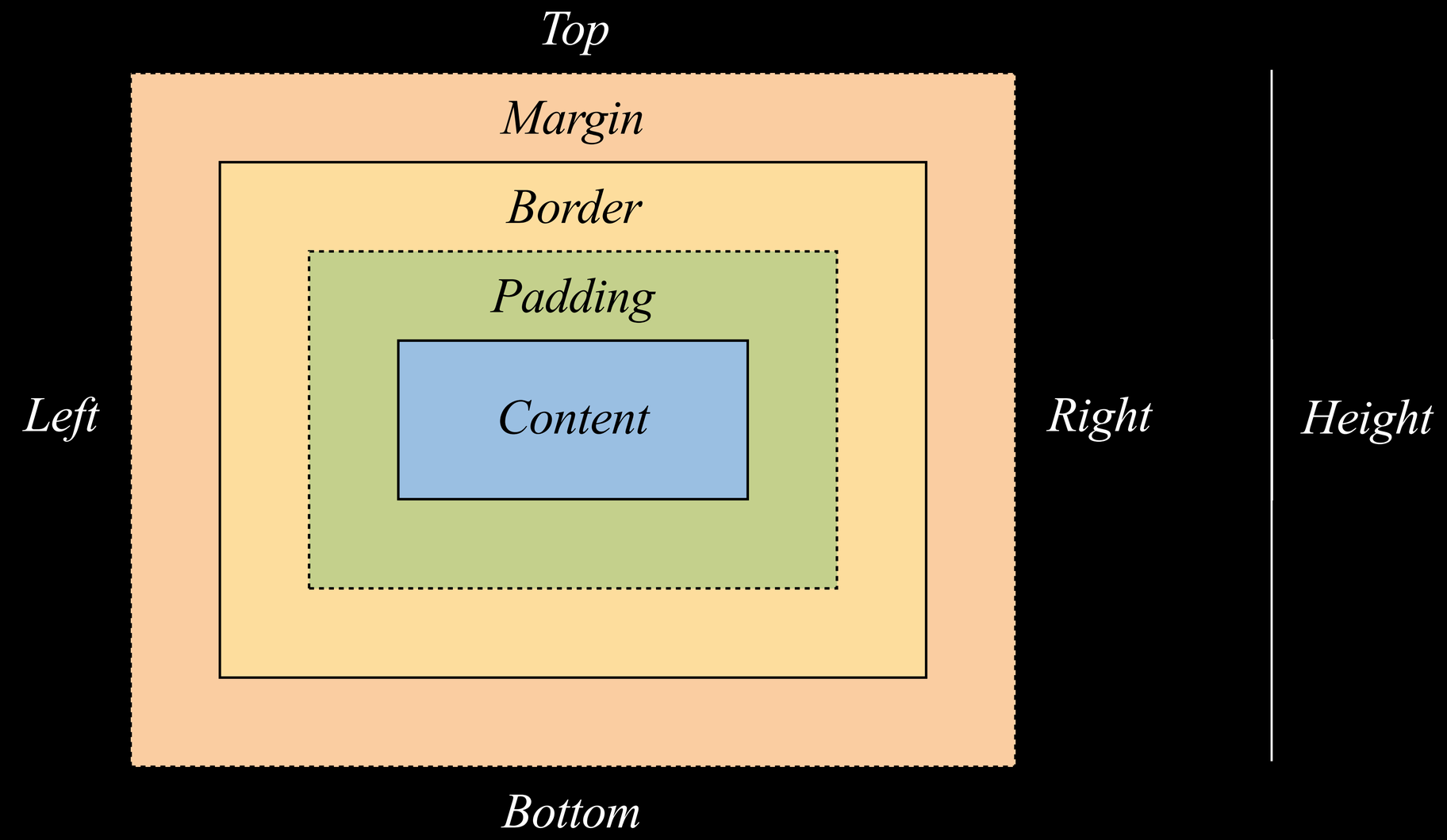
By default, browsers are set to `box-sizing: content-box;` which means that padding and the border exist outside of and in addition to the content's width & height

BOX MODEL

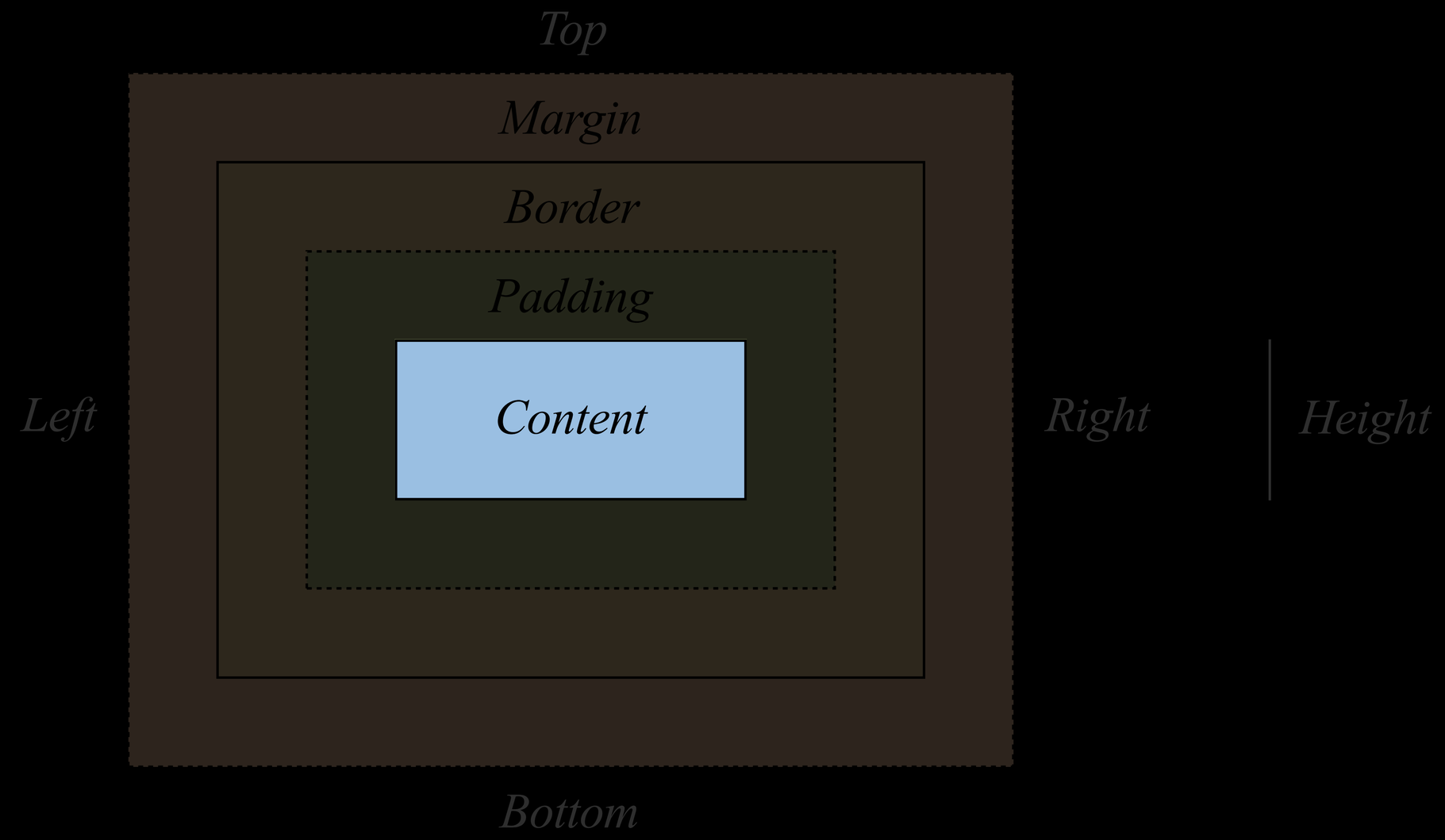
This isn’t intuitive! To change that—
so padding and borders inset into
your content dimensions—

```
box-sizing: border-box;
```

BOX MODEL


Top
Margin
Border
Padding
Content
Left
Right
Height
Bottom
Width

1. CONTENT


Top
Margin
Border
Padding
Content
Left
Right
Height
Bottom
Width

1. CONTENT

The **content** area is what you're filling the element with—generally text or an image. Its dimensions are generally dictated by its fill, but can be specified with the `height` and `width` properties.

1. CONTENT

```
…
  <body>
    <section>
      <div>
        <p>This is some text in the
        first element.</p>
      </div>
      <div>
        <p>And some more in the
        second element.</p>
      </div>
      <div>
        <p>Then a third one, too.</
        p>
      </div>
    </section>
  </body>
…
```

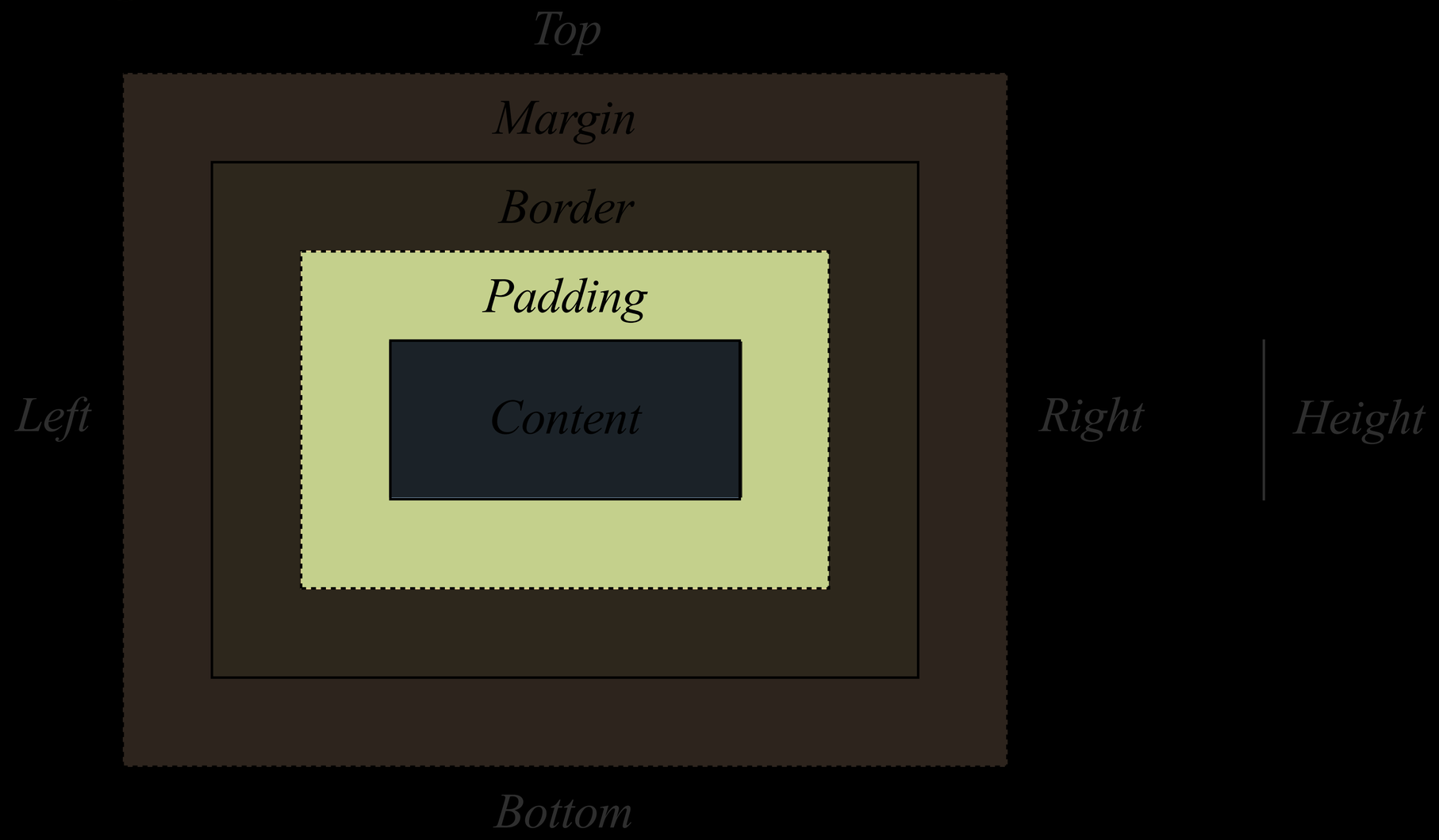
```
body {
  font-family: sans-serif;
  padding: 20px;
}

div {
  background-color: deepskyblue;
}
```

1. CONTENT

This is some text in the first element.
And some more in the second element.
Then a third one, too.

```
body {
  font-family: sans-serif;
  padding: 20px;
}

div {
  background-color: deepskyblue;
}
```

2. PADDING


Top
Margin
Border
Padding
Content
Left
Right
Height
Bottom
Width

2. PADDING

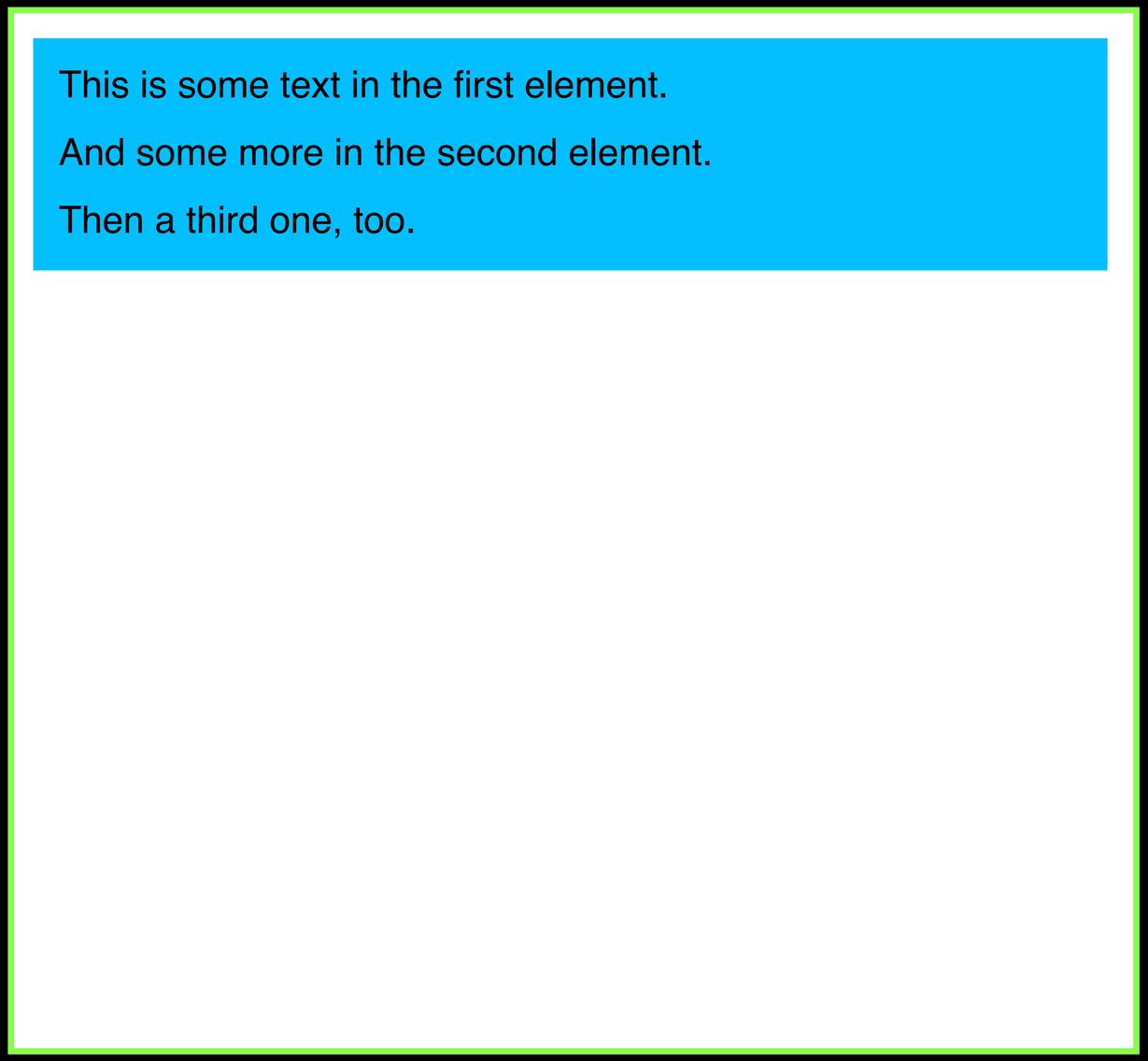
The **padding** extends around the element's content. If we set `box-sizing` to `border-box`, it's essentially the inset.

2. PADDING

```
…
  <body>
    <section>
      <div>
        <p>This is some text in the
        first element.</p>
      </div>
      <div>
        <p>And some more in the
        second element.</p>
      </div>
      <div>
        <p>Then a third one, too.</
        p>
      </div>
    </section>
  </body>
…
```

```
body {
  font-family: sans-serif;
  padding: 20px;
}

div {
  background-color: deepskyblue;
  padding: 20px;
}
```

2. PADDING




```
body {
  font-family: sans-serif;
  padding: 20px;
}

div {
  background-color: deepskyblue;
  padding: 20px;
}
```

(SHORTHAND)

Values for padding (or margin or border) do not have to be applied evenly. You can specify them one by one (e.g. `margin-top: 20px;`), OR!

(SHORTHAND)

You can use a shorthand property to name multiple at once.

(SHORTHAND)

1 value: all directions/sides

```
p { padding: 20px; }
```

2 values: top/bottom, left/right

```
p { padding: 20px 40px; }
```

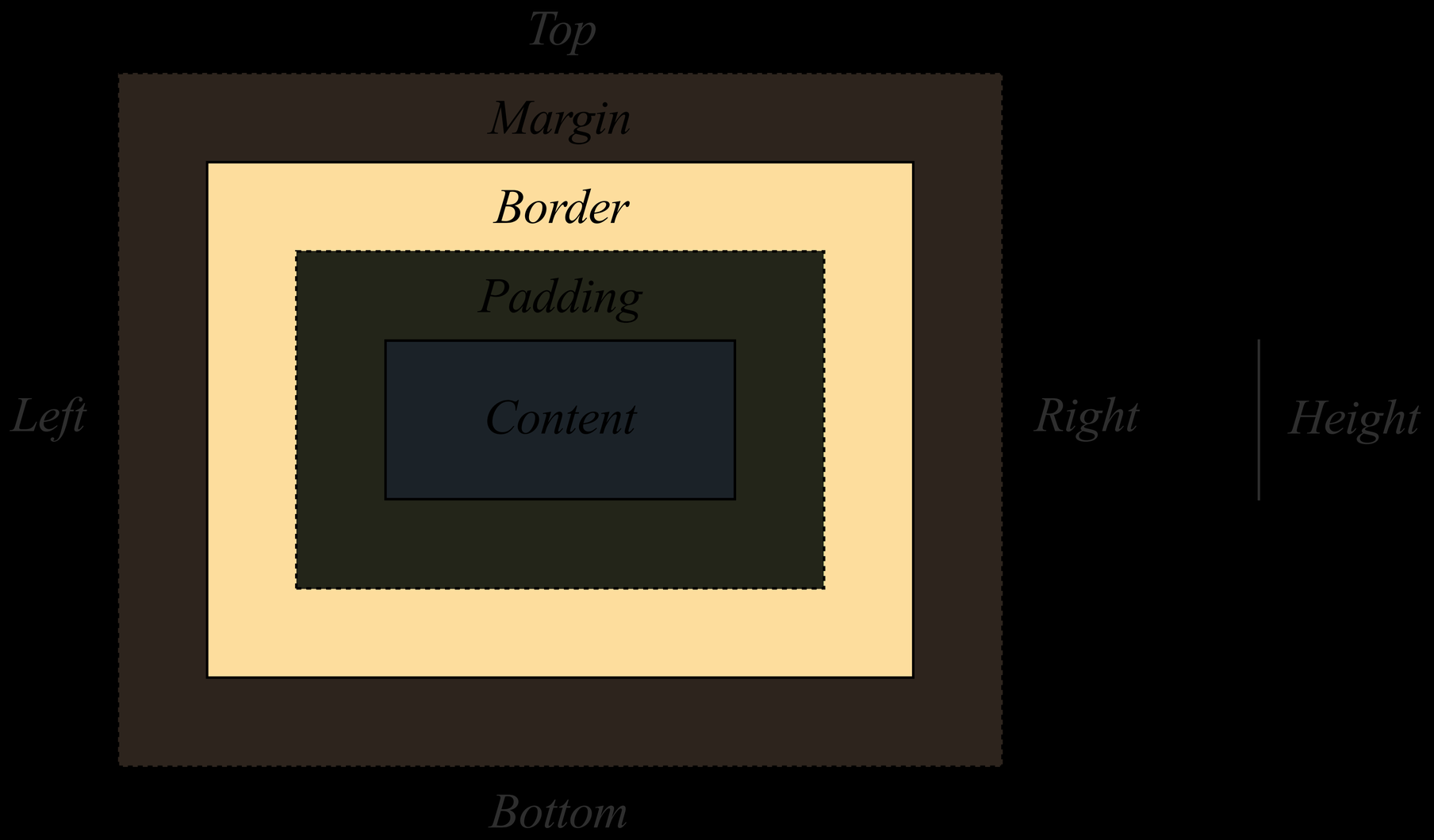
3 values: top, left/right, bottom

```
p { padding: 20px 40px 80px; }
```

4 values: top, left, bottom, left

```
p { padding: 20px 20px 40px 80px; }
```

3. BORDER


Top
Margin
Border
Padding
Content
Left
Right
Height
Bottom
Width

3. BORDER

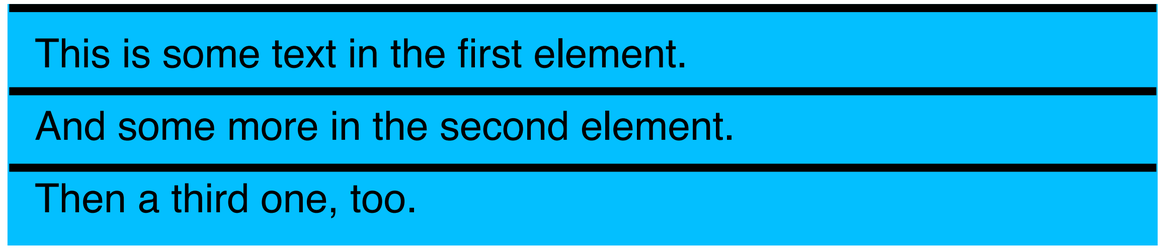
Border is exactly what it sounds like, and typically is used for decoration or delineation. It has its own specific CSS properties such as `border-width`, `border-color`, `border-height`.

3. BORDER

```
…
  <body>
    <section>
      <div>
        <p>This is some text in the
        first element.</p>
      </div>
      <div>
        <p>And some more in the
        second element.</p>
      </div>
      <div>
        <p>Then a third one, too.</
        p>
      </div>
    </section>
  </body>
…
```

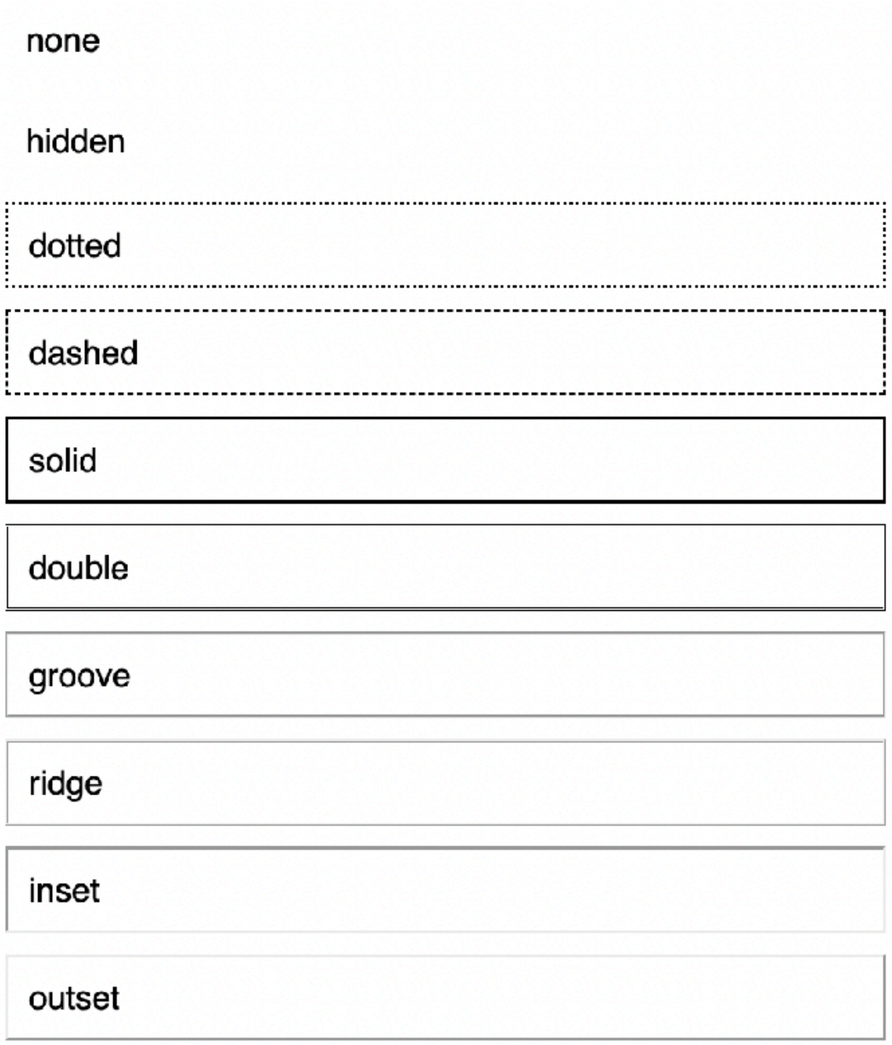
```
body {
  font-family: sans-serif;
  padding: 20px;
}

div {
  background-color: deepskyblue;
  border-top: solid black 4px;
  padding: 10px;
}

/* Non-shorthand. */
  /* div {
       border-top-color: black;
       border-top-style: solid;
       border-top-width: 4px;
     } */
```

3. BORDER



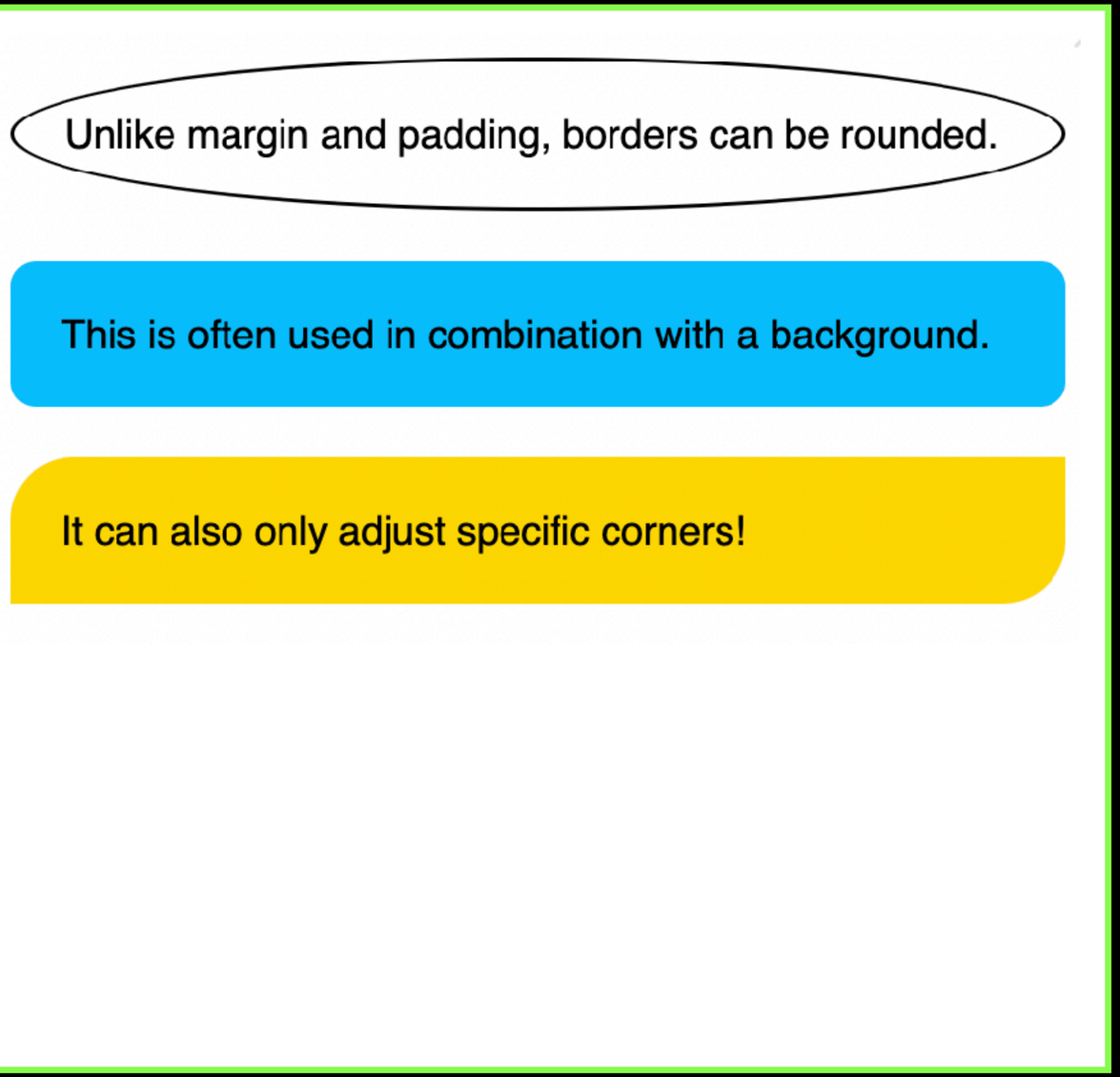

```
body {
  font-family: sans-serif;
  padding: 20px;
}

div {
  background-color: deepskyblue;
  border-top: solid black 4px;
  padding: 10px;
}

/* Non-shorthand. */
  /* div {
       border-top-color: black;
       border-top-style: solid;
       border-top-width: 4px;
     } */
```

3. BORDER STYLES



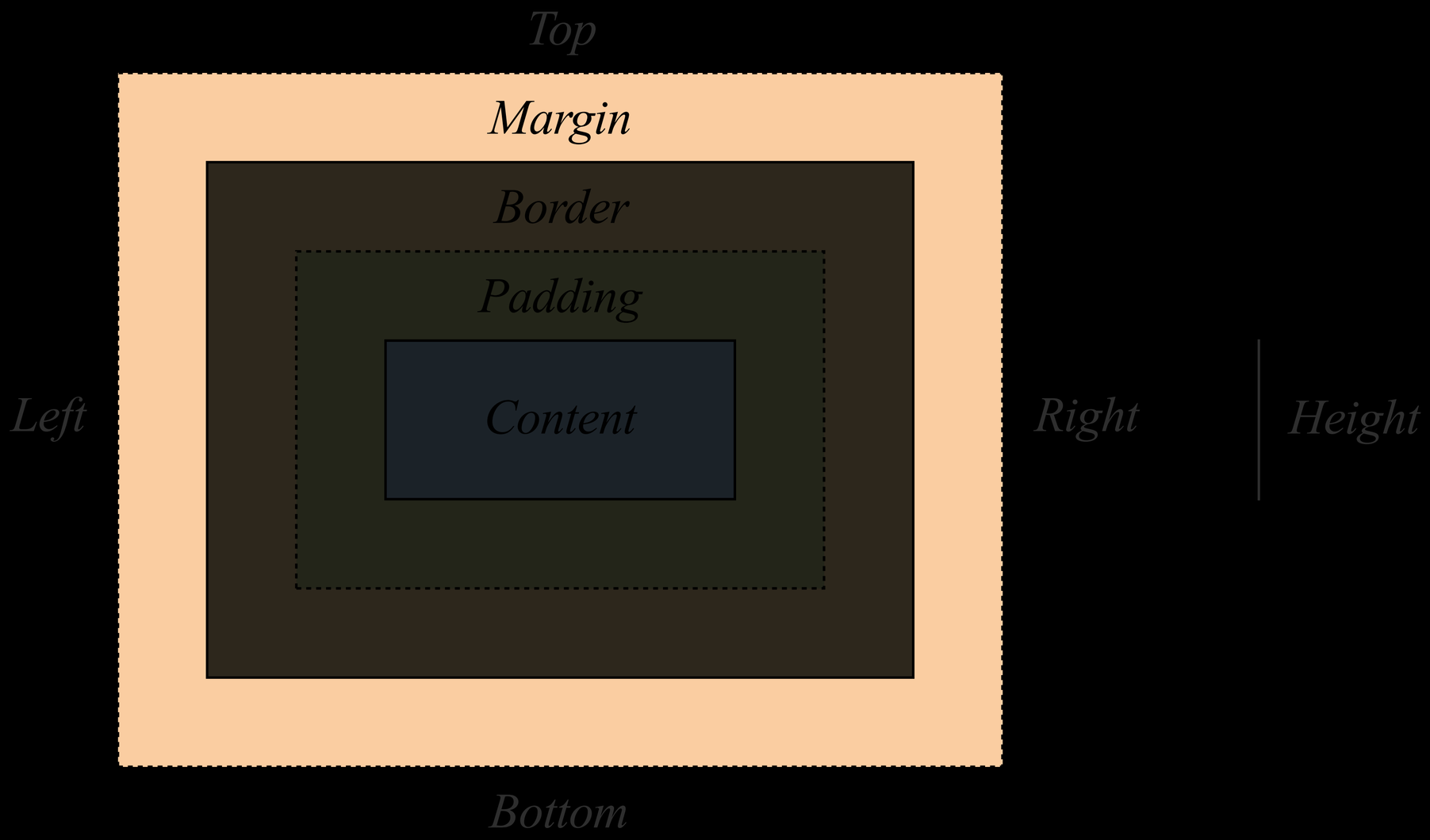

```
border-style: none;

border-style: hidden;

border-style: dotted;

border-style: dashed;

border-style: solid;

border-style: double;

border-style: groove;

border-style: ridge;

border-style: inset;

border-style: outset;
```

3. BORDER STYLES




```
div:nth-child(1) {
    border-radius: 50%;
    /* Relative to size.*/
    border-style: solid;
}

div:nth-child(2) {
    background-color: deepskyblue;
    border-radius: 10px;
    /* Absolute amount. */
}

div:nth-child(3) {
    background-color: gold;
    border-bottom-right-radius: 25px;
    border-top-left-radius: 25px;
}
```

4. MARGIN


Top
Margin
Border
Padding
Content
Left
Right
Height
Bottom
Width

4. MARGIN

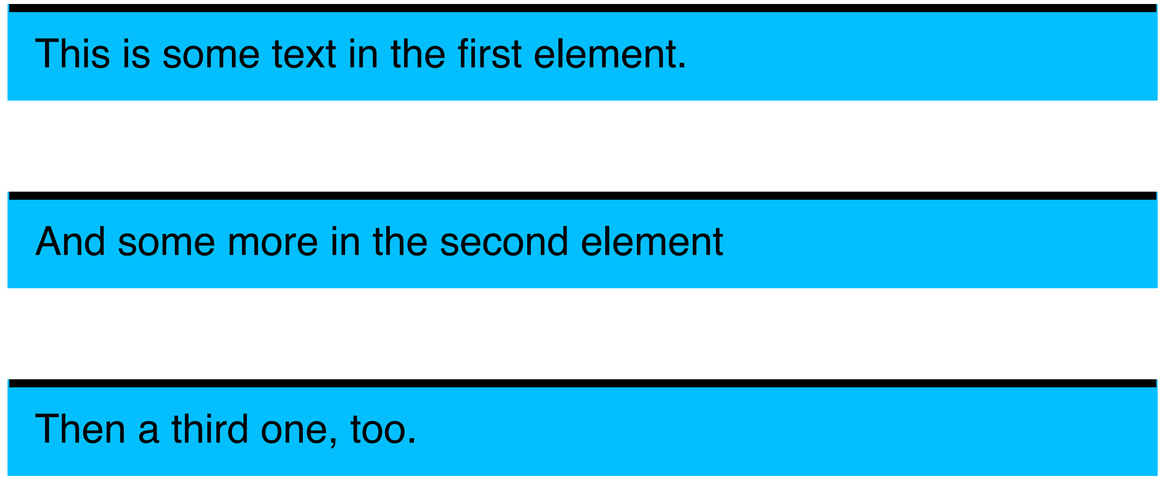
The **margin** is the last part of the box, and is the empty/negative/white space that separates one element from another. It can use the same shorthand and uneven application as padding.

4. MARGIN

```
…
  <body>
    <section>
      <div>
        <p>This is some text in the
        first element.</p>
      </div>
      <div>
        <p>And some more in the
        second element.</p>
      </div>
      <div>
        <p>Then a third one, too.</
        p>
      </div>
    </section>
  </body>
…
```

```
body {
  font-family: sans-serif;
  padding: 20px;
}

div {
  background-color: deepskyblue;
  border-top: solid black 4px;
  padding: 10px;
}

/* Every one but the first one. */
div:not(:first-child) {
  margin-top: 40px;
}
```

4. MARGIN

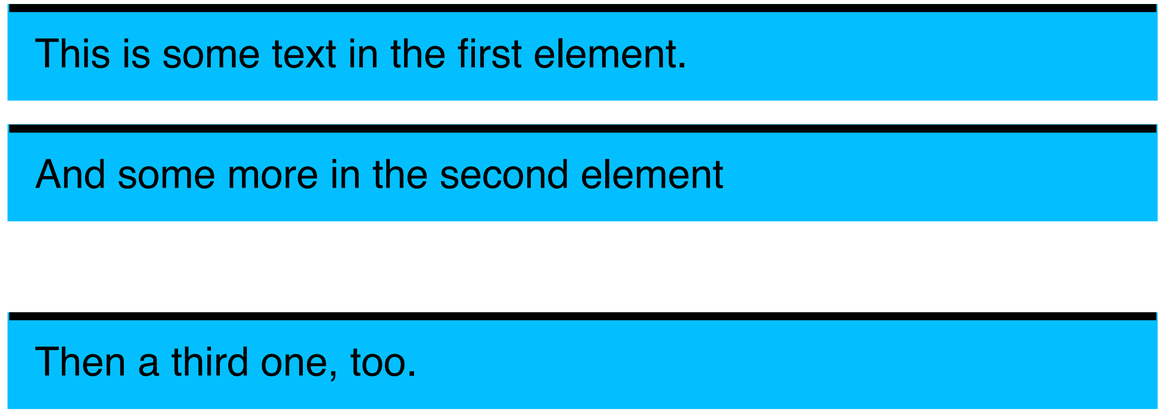



```
body {
  font-family: sans-serif;
  padding: 20px;
}

div {
  background-color: deepskyblue;
  border-top: solid black 4px;
  padding: 10px;
}

/* Every one but the first one. */
div:not(:first-child) {
  margin-top: 40px;
}
```

4. MARGIN

Margins can be positive or negative—which is unique among the parts of the box. A negative margin will bring elements closer together (but will not overlap them—more on this later)

4. MARGIN

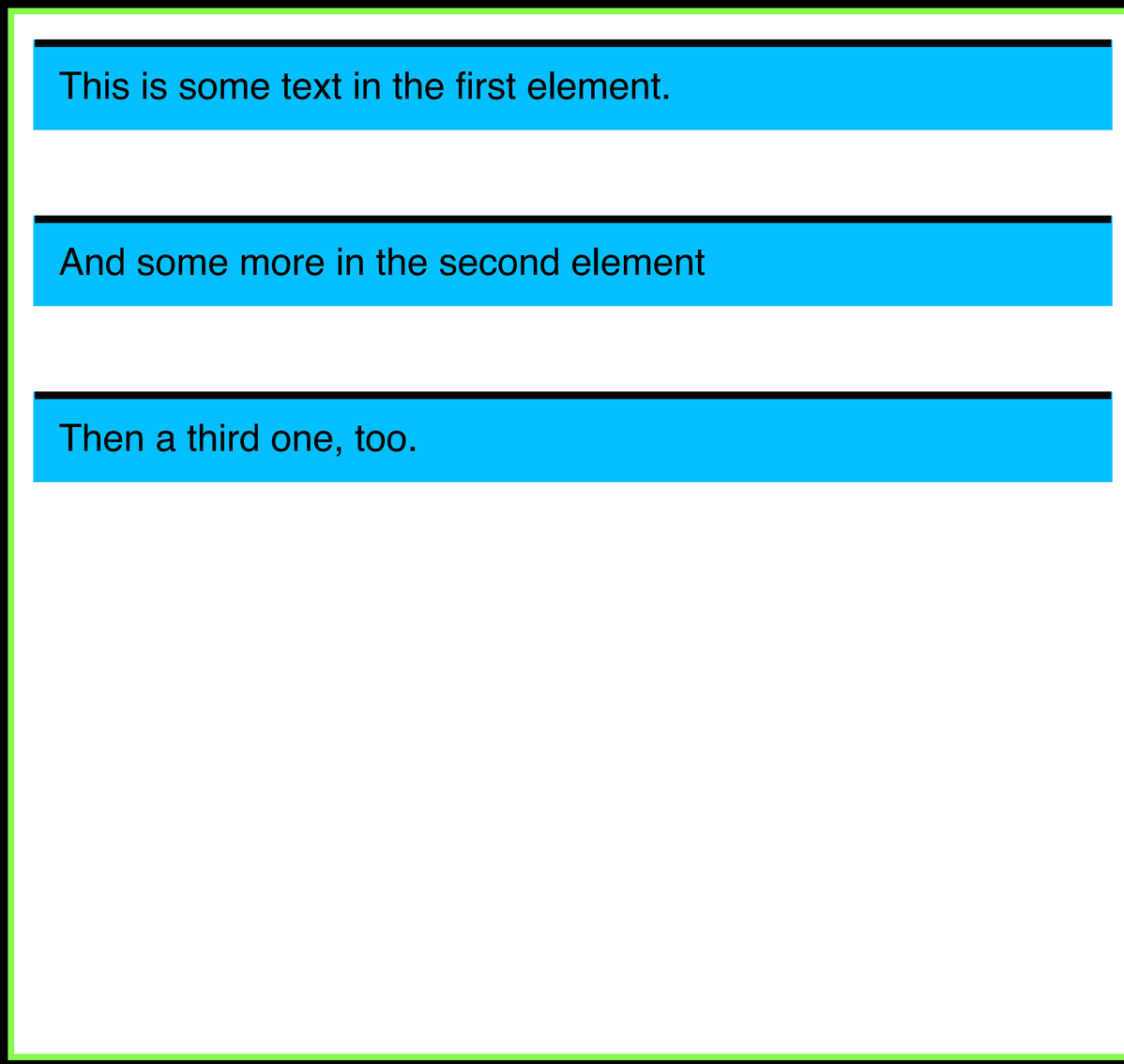



```
body {
  font-family: sans-serif;
  padding: 20px;
}

div {
  background-color: deepskyblue;
  border-top: solid black 4px;
  padding: 10px;
}

div:not(:first-child) {
  margin-top: 40px;
}

/* But this negates some of it. */
div:first-child
{ margin-bottom: -30px; }
```

4. MARGIN

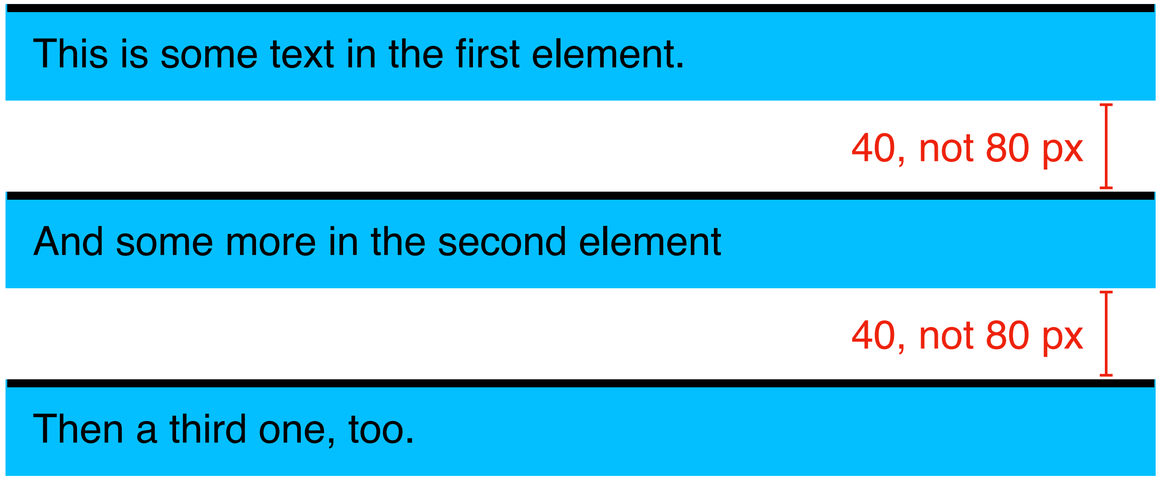
Margins defining the same space will **collapse** rather than compound.

4. MARGIN




```
body {
  font-family: sans-serif;
  padding: 20px;
}

div {
  background-color: deepskyblue;
  border-top: solid black 4px;
  padding: 10px;
}

div:not(:first-child) {
  margin-top: 40px;
}

div:first-child
{ margin-bottom: 40px; }
```

4. MARGIN




```
body {
  font-family: sans-serif;
  padding: 20px;
}

div {
  background-color: deepskyblue;
  border-top: solid black 4px;
  padding: 10px;
}

div:not(:first-child) {
  margin-top: 40px;
}

div:first-child
{ margin-bottom: 40px; }
```

UNITS

We’ve used a lot of units in CSS so far, to specify dimensions, parts of the box, font sizes and typography… Let’s go through some of them.

ABSOLUTE LENGTH

These units are fixed to actual, printable sizes—screen dimensions will vary them a bit.

ABSOLUTE LENGTH

These units are fixed to actual, printable sizes—screen dimensions will vary them a bit.

```
.pixels {
  height: 360px;
  width: 720px;
}

.inches {
  height: 5in;
  width: 10in;
}

.mm {
  height: 84mm;
  width: 400mm;
}
```

RELATIVE LENGTH

These units depend on context—they are explicitly and innately web-based measurements.

```
/* Relative to nearest sized ancestor. */
.percentage {
  height: 90%;
  width: 85%;
}

/* Relative to viewport height/width. */
.viewport {
  height: 75vh;
  width: 80vw;
}

/* Relative to element font-size. */
.em {
  height: 14em; /* 1em is one line. */
  width: 4.8em;
}

/* Also relative to font size */
.ch {
  width: 1ch; /* 1ch is one letter. */
}

/* Relative to :root font-size. */
.rem {
  height: 12rem;
  width: 2.4rem;
}
```

CALC

You could do some math in CSS if you're so inclined…

```
.absolute-and-relative {
   width: calc(50% - 20px);
}

.computer-do-the-math {
   width: calc(100% / 12);
}
```

LIMITS

You can also set **limits** or constraints on dimension—which will become key as we start to talk about responsive design. Your elements flex based on screen size, but maybe you want them to only do so within a range.

```
.constrained-width {
  min-width: 200px;
  width: 50%;
  max-width: 400px;
}

.constrained-height {
  min-height: 100px;
  height: 100%;
  max-height: 200px;
}

/* Handy to watch your line lengths! */
p {
  max-width: 65ch;
/* 65ish letters. */
}
```

POSITION

We know how to **fill** and **size** elements, now… so let’s talk about how they can flow together. The CSS property `position` sets this relationship.

POSITION: STATIC

Static is the default position—it's not something you set, it's something you change a position from.

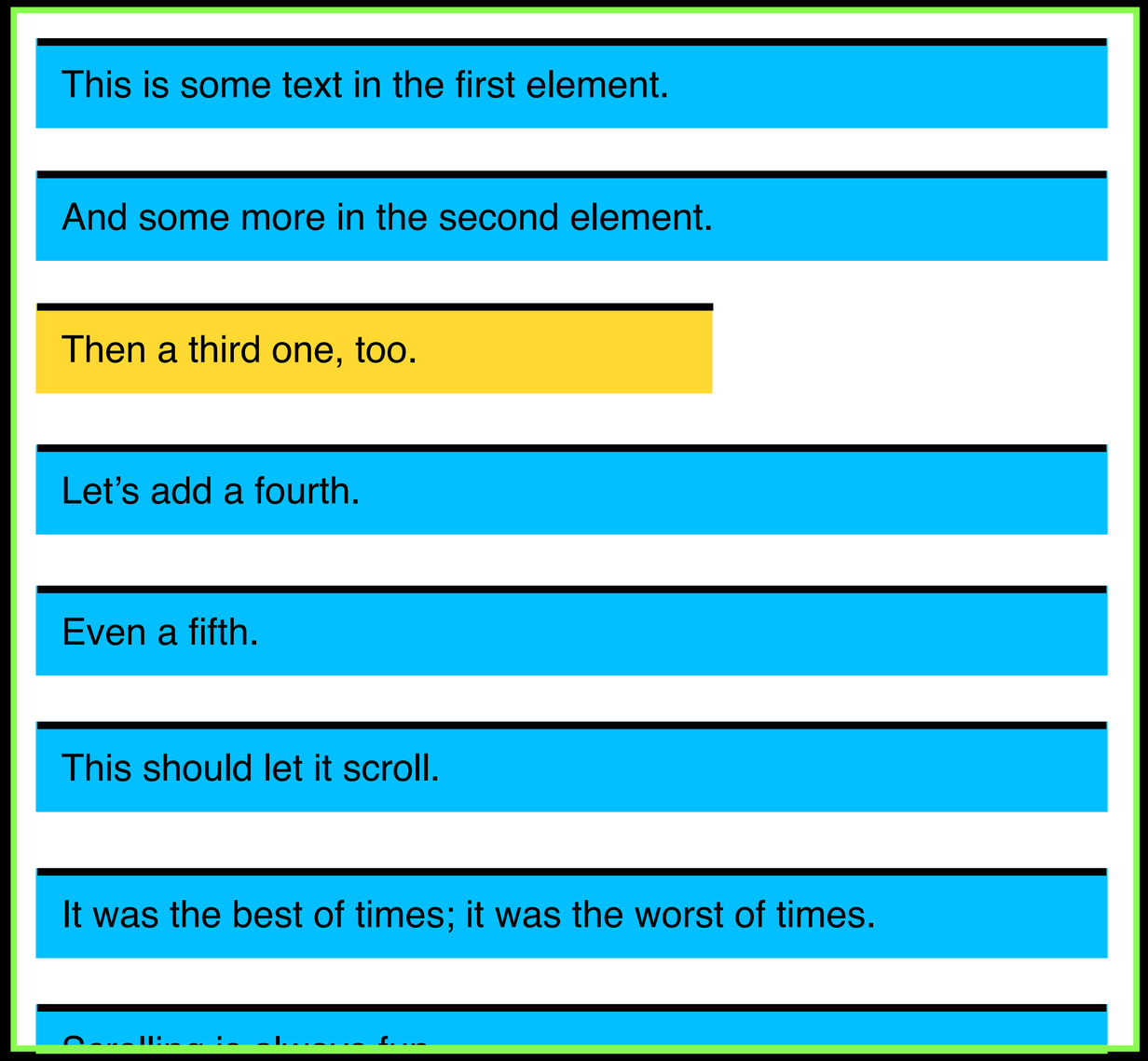
POSITION: STATIC

```
…
    <body>
        <section>
            <div>
                <p>This is some text in the first element.</p>
            </div>
            <div>
                <p>And some more in the second element.</p>
            </div>
            <div>
                <p>Then a third one, too.</p>
            </div>
            <div>
                <p>Let’s add a fourth.</p>
            </div>
            <div>
                <p>Even a fifth.</p>
            </div>
            <div>
                <p>This should let it scroll.</p>
            </div>
            <div>
                <p>It was the best of times; it was the worst of times.</p>
            </div>
            <div>
                <p>Scrolling is always fun.</p>
            </div>
            <div>
                <p>I guess that isn’t true.</p>
            </div>
            <div>
                <p>Sometimes it goes on and on, you know.</p>
            </div>
            <div>
                <p>Okay, this is enough.</p>
            </div>
            <div>
                <p>One more, for good measure.</p>
            </div>
        </section>
    </body>
…
```

```
body {
  font-family: sans-serif;
  padding: 20px;
}

div {
  background-color: deepskyblue;
  border-top: solid black 4px;
  padding: 10px;
}

div:not(:first-child) {
  margin-top: 20px;
}
```


this is going to be constant for all position code… won’t repeat it every time

POSITION: STATIC

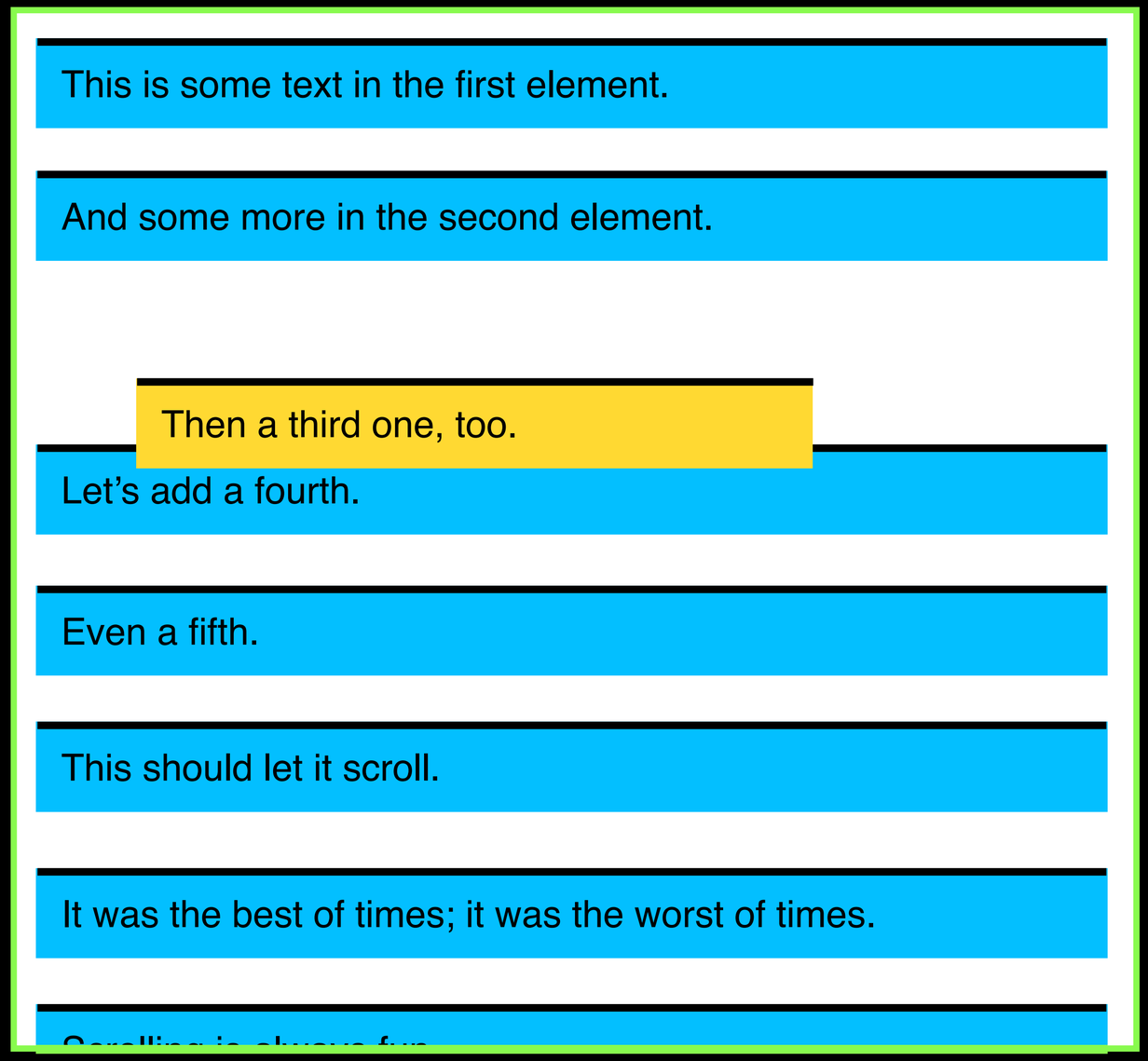



```
…
div:nth-child(3) {
  background-color: gold;
  position: static;
  width: 66%;
}
```

POSITION: RELATIVE

Once you set `position: relative`, you can use the `top`, `left`, `right`, and `bottom` values (with any unit) to move elements away from their default positioning.

POSITION: RELATIVE




```
…
div:nth-child(3) {
  background-color: gold;
  position: relative;
  left: 30px;
  top: 30px;
  width: 66%;
}
```

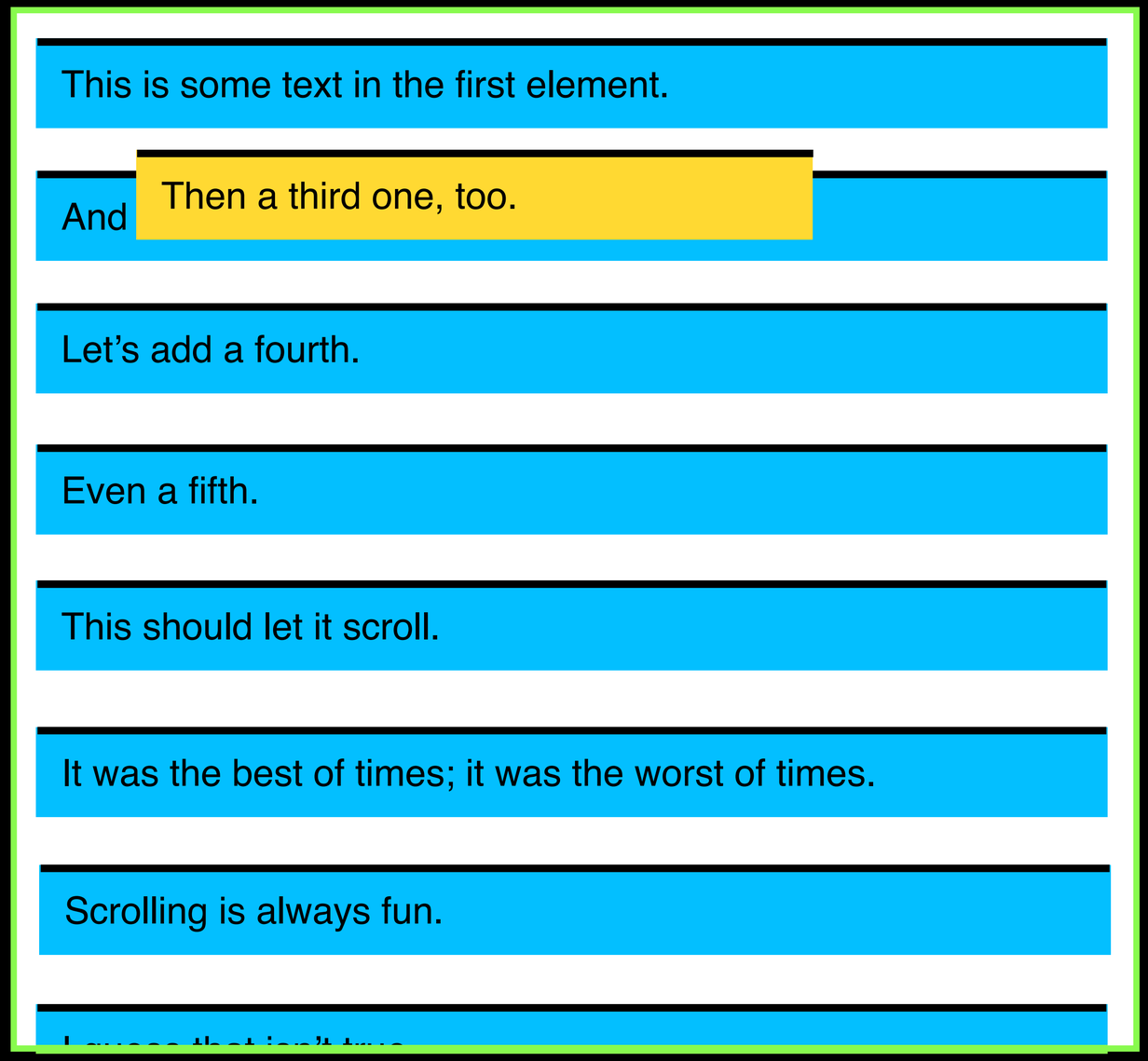
POSITION: ABSOLUTE

`position: absolute` acts a bit like position: relative, except that rather than referring to its own default position, it is relative to the **next relatively-positioned ancestor** up the cascade.

POSITION: ABSOLUTE

Note that absolute **removes** the element from the normal document flow—it doesn’t take up space in the page layout. Use this for specific design elements that need to be placed very careful.

POSITION: ABSOLUTE



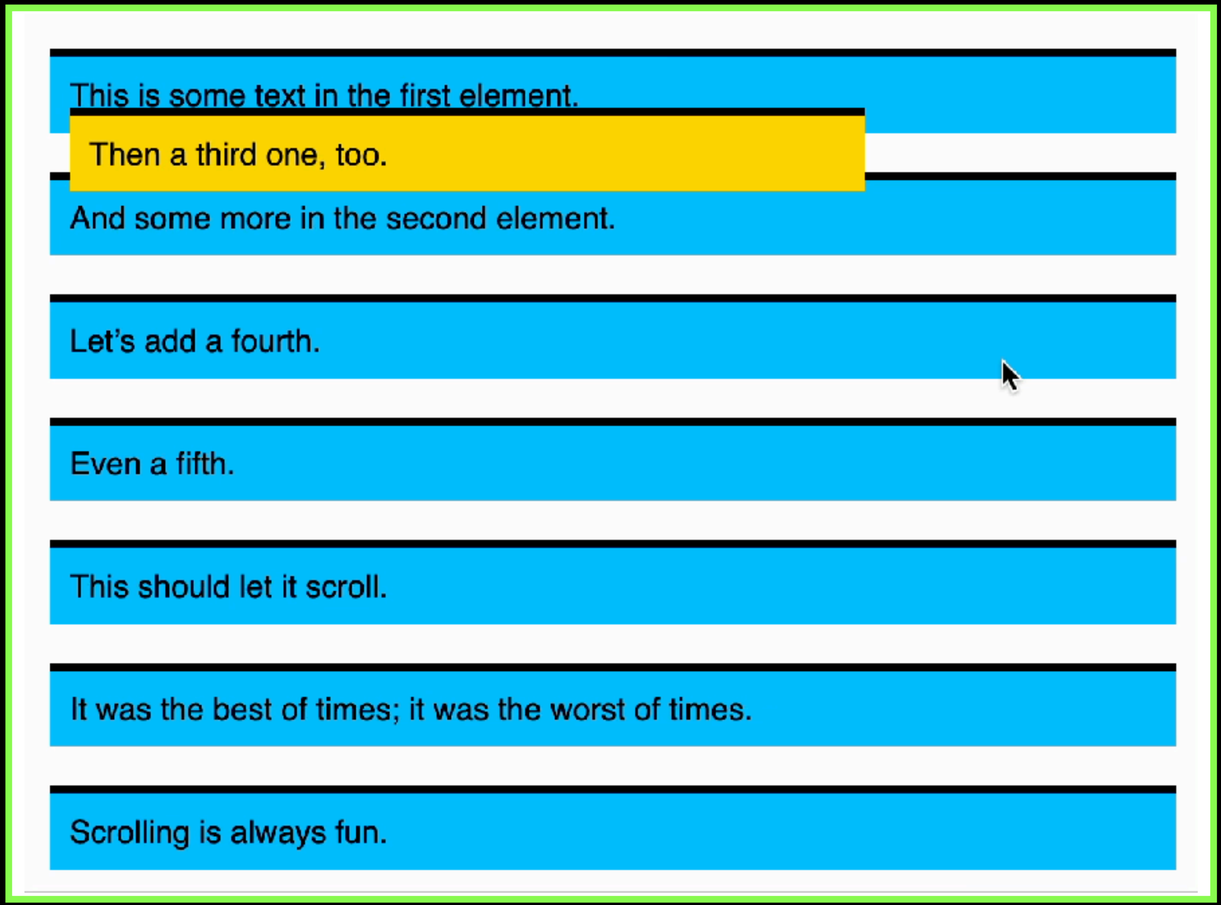

```
…
section {
  position: relative;
}

div:nth-child(3) {
  background-color: gold;
  position: absolute;
  left: 30px;
  top: 30px;
  width: 66%;
}
```

POSITION: FIXED

Fixed positioning also removes an element from the flow, but positions the element **relative to the browser ‘viewport’**, not the rest of the code.

It’s like putting it on another layer. This is often used for things like navigations.

POSITION: FIXED

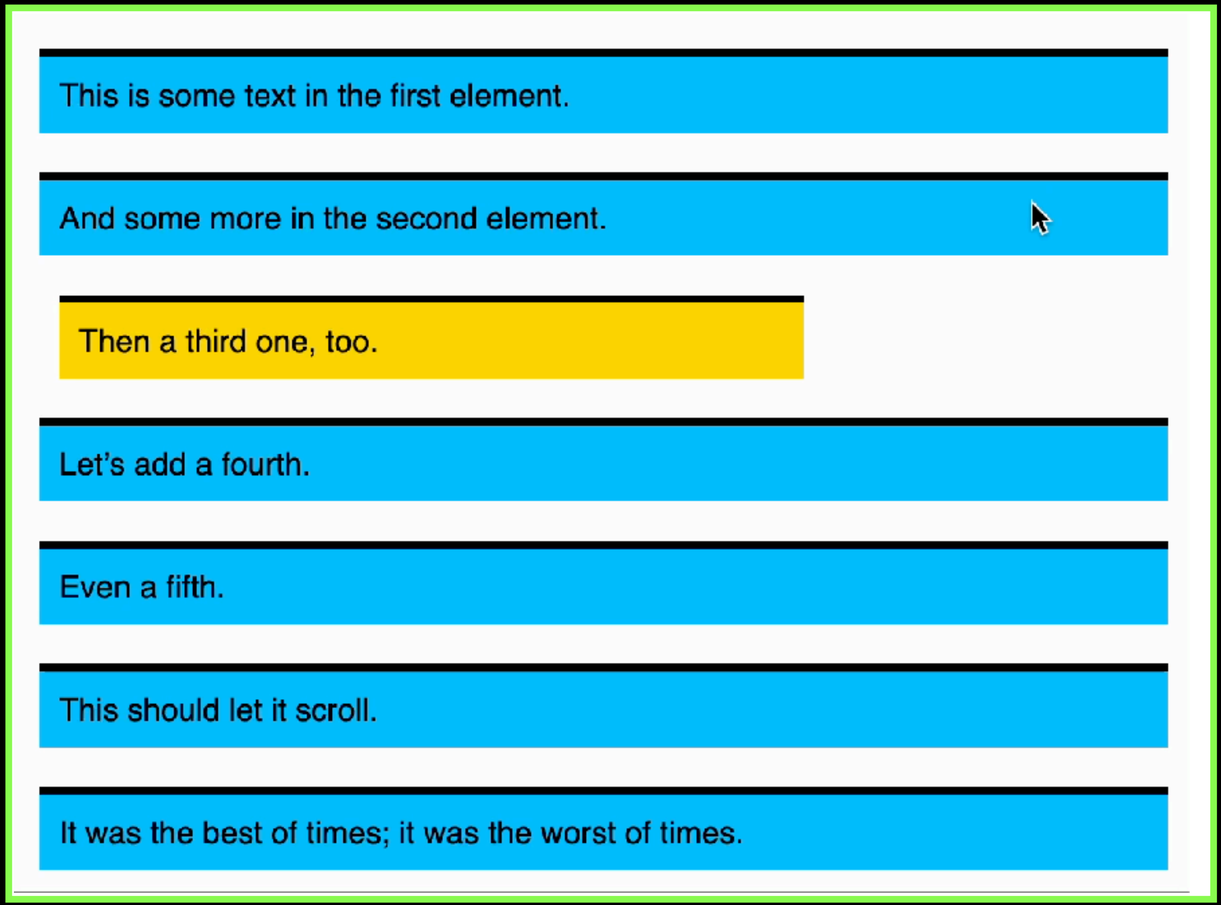



```
…
div:nth-child(3) {
  background-color: gold;
  position: fixed;
  left: 30px;
  top: 30px;
  width: 66%;
}
```

POSITION: STICKY

Sticky is new to position, and will start an element in the flow of their document (like static0 until their nearest ancestor moves past them (usually the viewport). The element ‘sticks’ in relation to that element. Try it for headers on tables and lists.

POSITION: STICKY




```
…
div:nth-child(3) {
  background-color: gold;
  position: sticky;
  left: 30px;
  top: 30px;
  width: 66%;
}
```

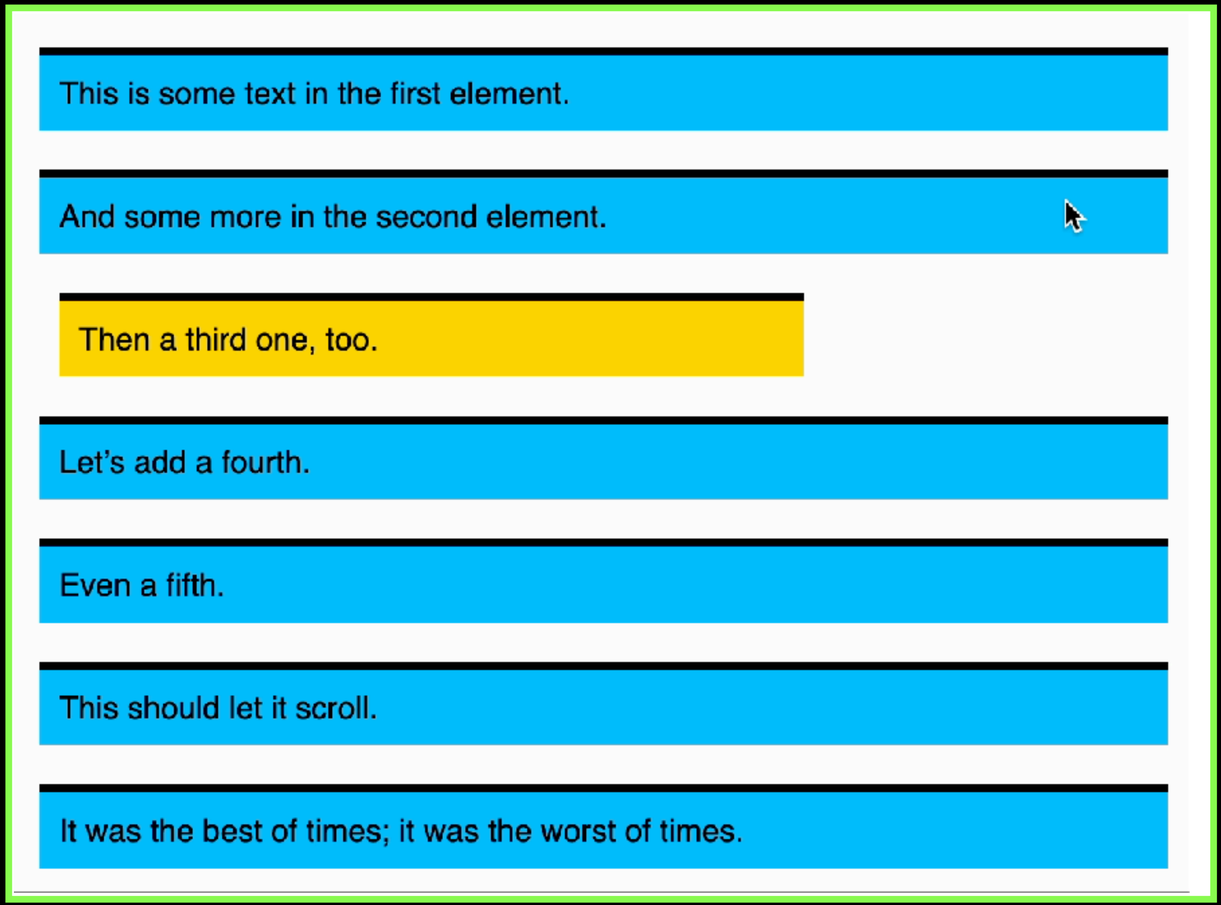
DEPTH

This isn’t exactly… position… but you can also define the `z-index` of any element as a separate property. This defines how things overlap, and in what order—you’re ordering the layers along the z-axis.

DEPTH

By default, later elements are in front of earlier elements.

DEPTH




```
…
div:nth-child(3) {
  background-color: gold;
  position: sticky;
  left: 30px;
  top: 30px;
  width: 66%;
}

div:nth-child(even) {
  position: relative;
  z-index: 1;
}
```

DEPTH

Stacking can be tricky, as all sorts of CSS changes and elements will automatically add context to the stack.

FLOATS

Floats allow you to adjust the flow of an element within another element.

FLOATS

You can assign the property `float` the value `left` or `right`, which are exactly what they sound like.

FLOATS

`float: left` or `right;` takes the element out of the document flow (to the left or to the right), and its siblings will basically wrap around it.

FLOATS

This is an aside element.

This is some text in the first element. Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Arcu risus quis varius quam. Egestas fringilla phasellus. Faucibus scelerisque eleifend donec

And some more in the second element. Ut etiam sit amet nisl. Sed libero enim sed faucibus turpis in eu. Congue nisi vitae suscipit tellus mauris. Nisi est sit amet facilisis magna etiam tempor orci. Leo urna molestie at elementum. Morbi tristique senectus et netus et malesuada.

This is an aside element, but with more content

And a third one too.

```
…
aside {
  background-color: gold;
  float: left;
  margin-right: 10px;
  padding: 10px;
  width: 100px;
}

div:nth-child(2) aside {
  float: right;
  margin-left: 10px;
  margin-right: initial;
}
```

FLOATS

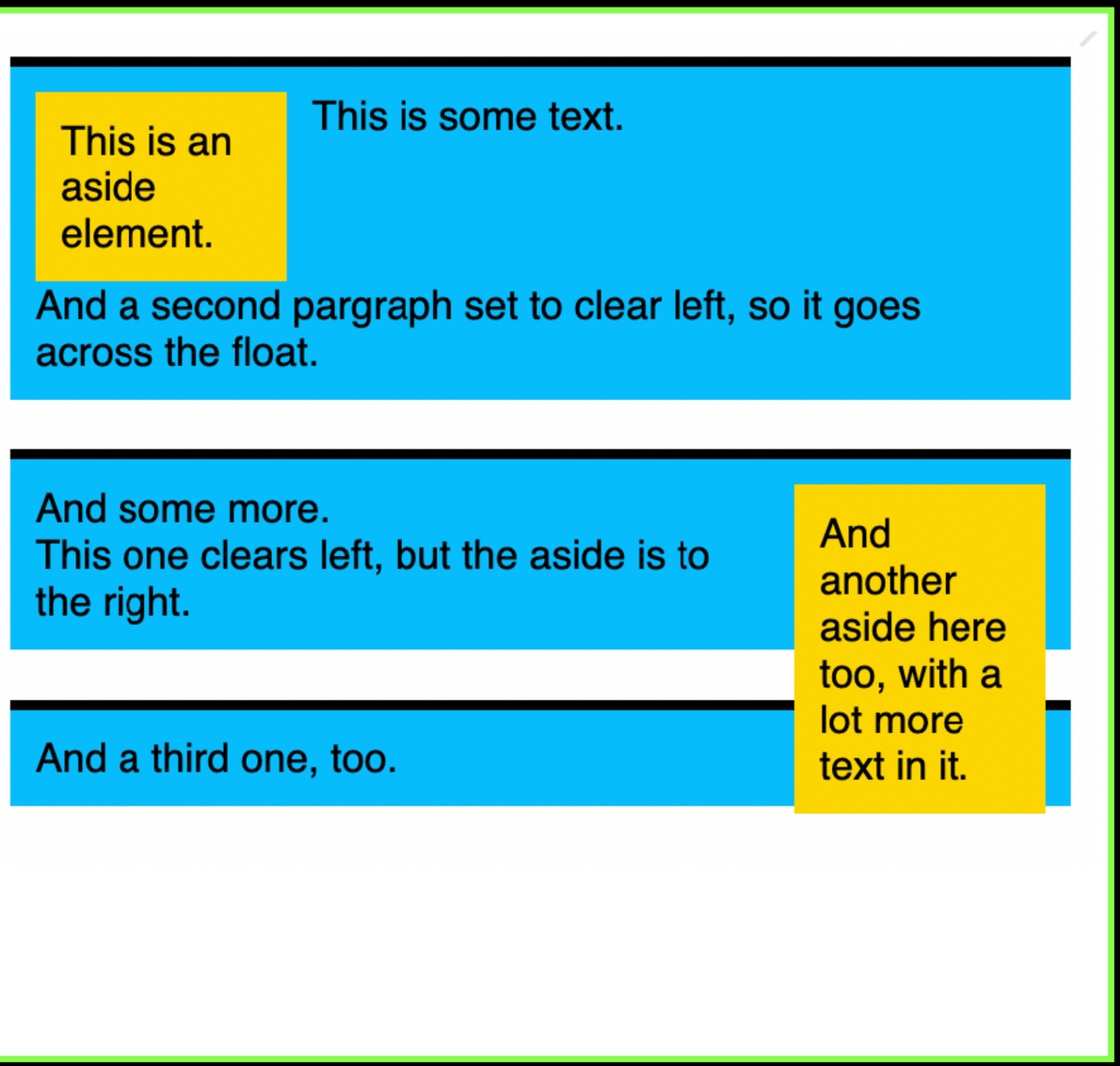
To keep a float from affecting its siblings' flows, you have to clear—

```
clear: left;
clear: right;
clear: both;
```

FLOATS

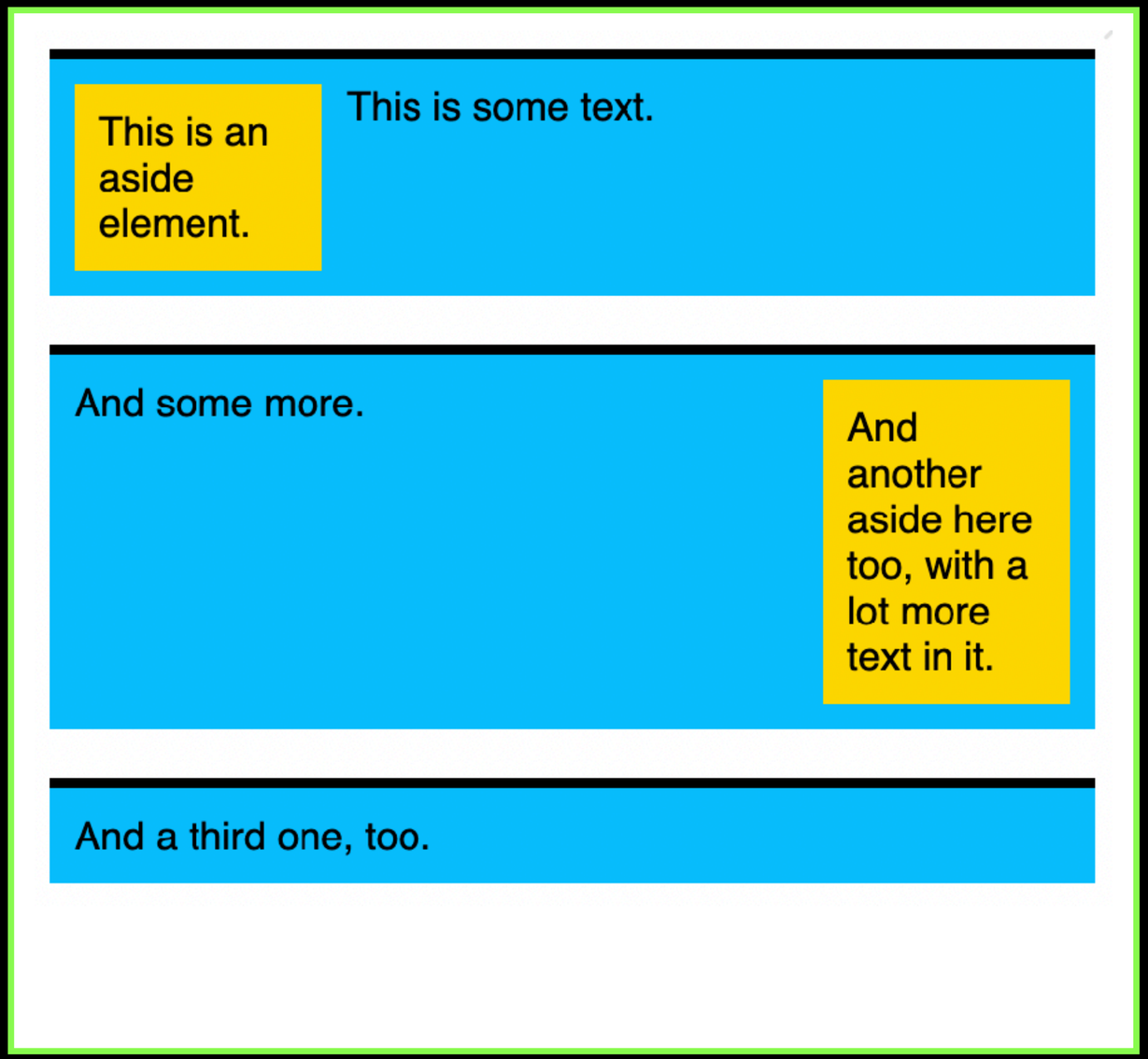
This will help the next element to sit entirely below a floated element.

FLOATS




```
…
aside {
  background-color: gold;
  float: left;
  margin-right: 10px;
  padding: 10px;
  width: 100px;
}

div:nth-child(2) aside {
  float: right;
  margin-left: 10px;
  margin-right: initial;
}

p:last-child {
  clear: left;
}
```

FLOATS: FIXES (CLEARFIX HACK)




```
…
aside {
  background-color: gold;
  float: left;
  margin-right: 10px;
  padding: 10px;
  width: 100px;
}

div:nth-child(2) aside {
  float: right;
  margin-left: 10px;
  margin-right: initial;
}

div: after{
  clear: both;
  content: ‘’;
  display: block;
}
```

FLOATS

Like so many elements in HTML, floats aren’t perfect, and there’s often (BUT NOT ALWAYS) a better way to do what you need to do.

DISPLAY

We've talked about **block** and **inline** elements—these are actually values of the CSS property `display`

DISPLAY

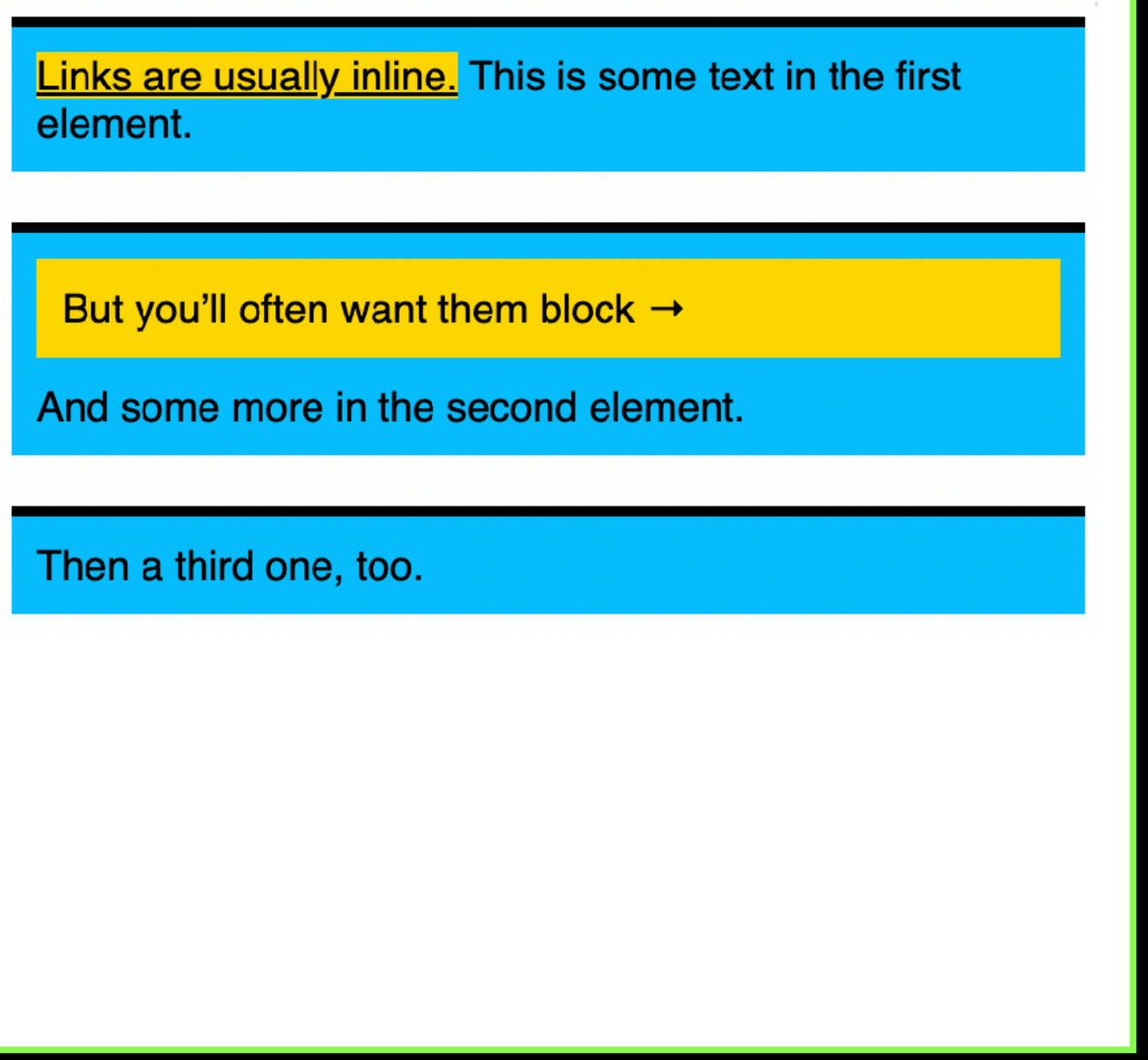
Most elements are block by default, but you can assign `display: block;` on an inline element as well (and then set height, width, etc).

DISPLAY

```
…
  <body>
    <section>
      <div>
        <p><a href="#">Links are
        usually inline.</a> This is
        some text in the first
        element.</p>
      </div>
      <div>
        <p><a href="#">But you'll often
        want them block →</a> And some
        more in the second element.</p>
      </div>
      <div>
        <p>Then a third one, too.</p>
      </div>
    </section>
  </body>
…
```

```
…
a {
  background-color: gold;
  text-decoration: underline;
}

div:nth-child(2) a {
  display: block;
  margin-bottom: 10px;
  padding: 10px;
  text-decoration: none;
}
```

DISPLAY




```
…
a {
  background-color: gold;
  text-decoration: underline;
}

div:nth-child(2) a {
  display: block;
  margin-bottom: 10px;
  padding: 10px;
  text-decoration: none;
}
```

DISPLAY

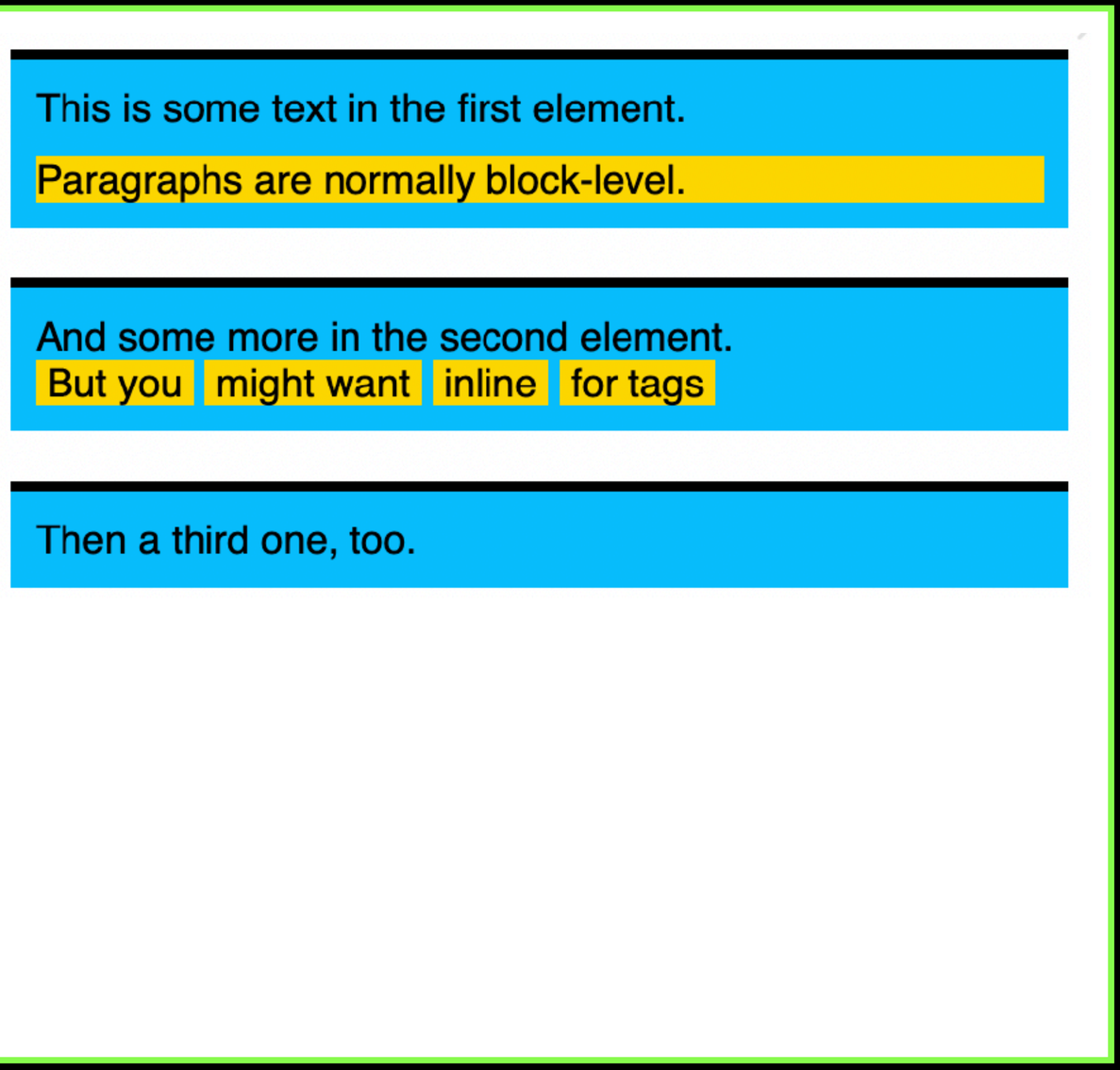
When a whole area (e.g. background and text) is a link, generally you'll want to make it's display block.

DISPLAY

```
…
  <body>
    <section>
      <div>
        <p>This is some text in the first
        element.</p>
        <p>Paragraphs are normally block-
        level.</p>
        </div>
        <div>
        <p>And some more in the second
        element.</p>
        <p> But you </p>
        <p> might want </p>
        <p> inline </p>
        <p> for tags </p>
      </div>
      <div>
        <p>Then a third one, too.</p>
      </div>
    </section>
  </body>
…
```

```
…
p:not(:first-child) {
  background-color: gold;
  margin-top: 10px;
}

div:nth-child(2)
p:not(:first-child) {
  display: inline;
  white-space: pre;
}
```

DISPLAY




```
…
p:not(:first-child) {
  background-color: gold;
  margin-top: 10px;
}

div:nth-child(2)
p:not(:first-child) {
  display: inline;
  white-space: pre;
}
```

DISPLAY

The `white-space` property with the value `pre` keeps white space from being collapsed inline

DISPLAY

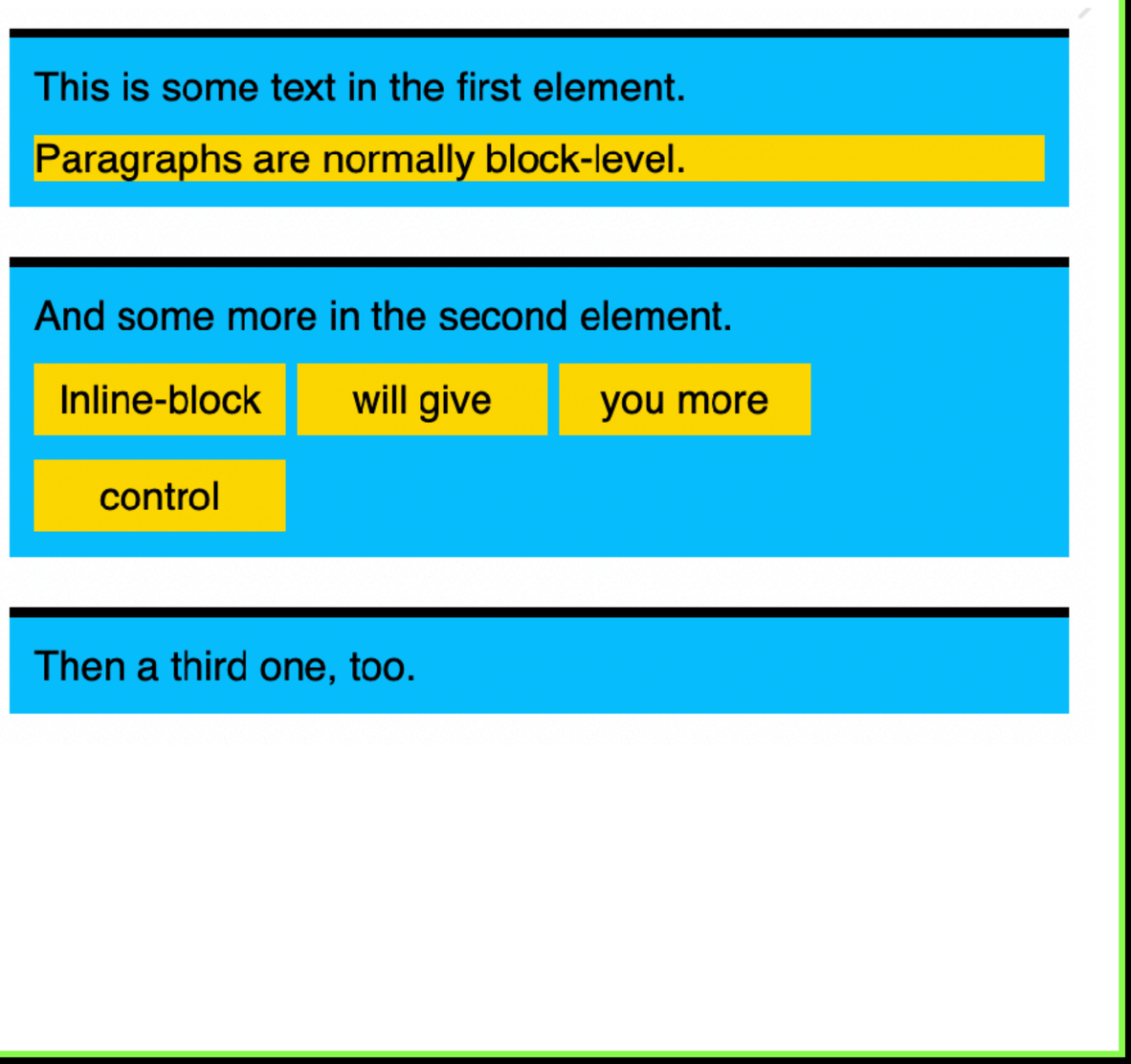
There's also `inline-block`. These elements get properties like `height` and `width`, but do not start on a new line.

DISPLAY

```
…
  <body>
    <section>
      <div>
        <p>This is some text in the first
        element.</p>
        <p>Paragraphs are normally block-
        level.</p>
        </div>
        <div>
        <p>And some more in the second
        element.</p>
        <p> Inline-block </p>
        <p> will give </p>
        <p> you more </p>
        <p> control </p>
      </div>
      <div>
        <p>Then a third one, too.</p>
      </div>
    </section>
  </body>
…
```

```
…
p:not(:first-child) {
  background-color: gold;
  margin-top: 10px;
}

div:nth-child(2)
p:not(:first-child) {
  display: inline-block;
  margin-top: 10px;
  padding: 5px;
  text-align: center;
  white-space: pre;
  width: 100px;
}
```

DISPLAY




```
…
p:not(:first-child) {
  background-color: gold;
  margin-top: 10px;
}

div:nth-child(2)
p:not(:first-child) {
  display: inline-block;
  margin-top: 10px;
  padding: 5px;
  text-align: center;
  white-space: pre;
  width: 100px;
}
```

DISPLAY

And then there’s none. `display: none` takes an element out of the flow entirely —and hides it from view.

DISPLAY

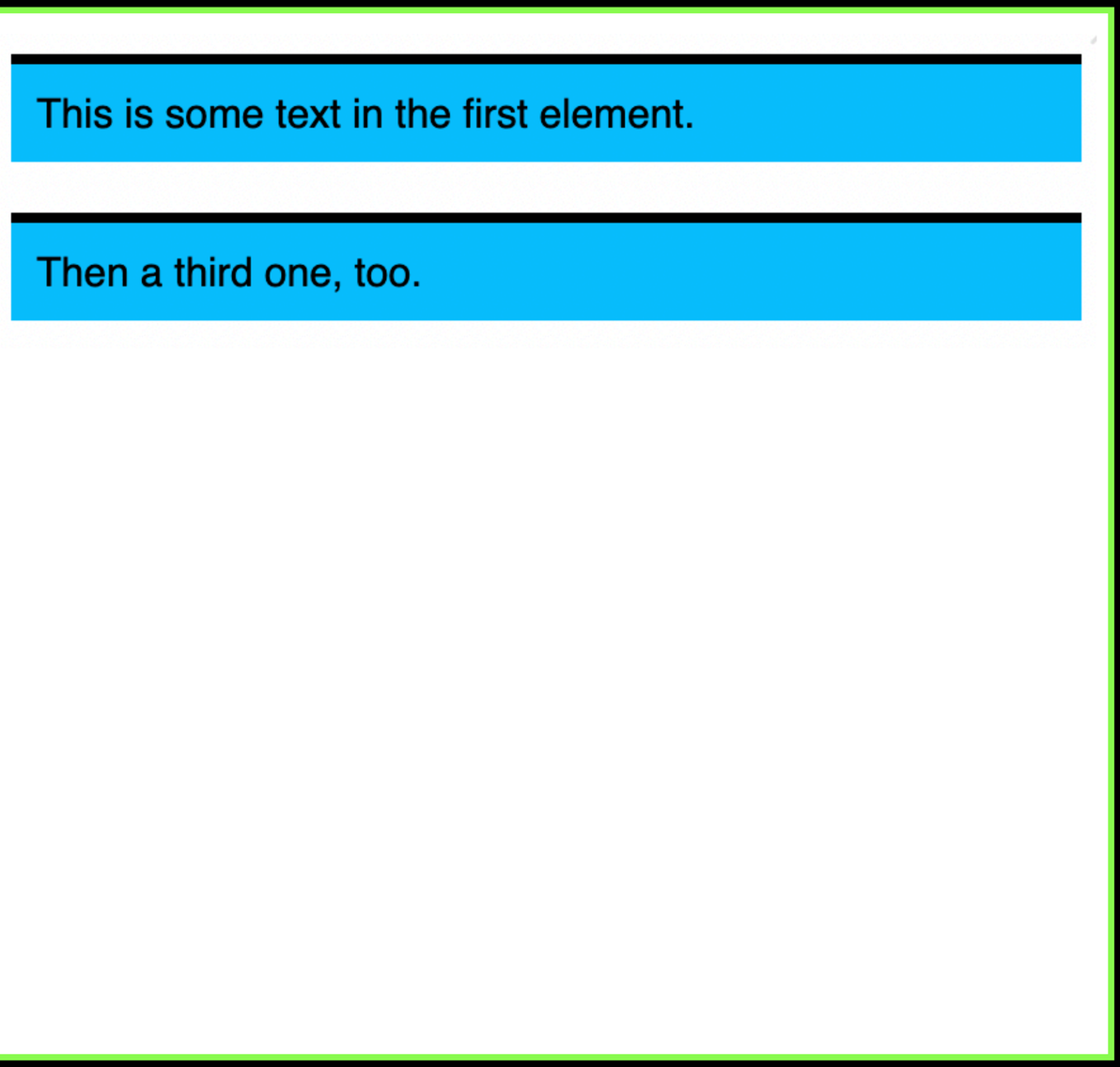
You can also do the inverse, and set a typically block element to be inline.

DISPLAY

```
…
  <body>
    <section>
      <div>
        <p>This is some text in the first
        element.</p>
        <p>Paragraphs are normally block-
        level.</p>
        </div>
        <div>
        <p>And some more in the second
        element.</p>
      </div>
      <div>
        <p>Then a third one, too.</p>
      </div>
    </section>
  </body>
…
```

```
…
div:nth-child(2) {
  display: none;
}
```

DISPLAY

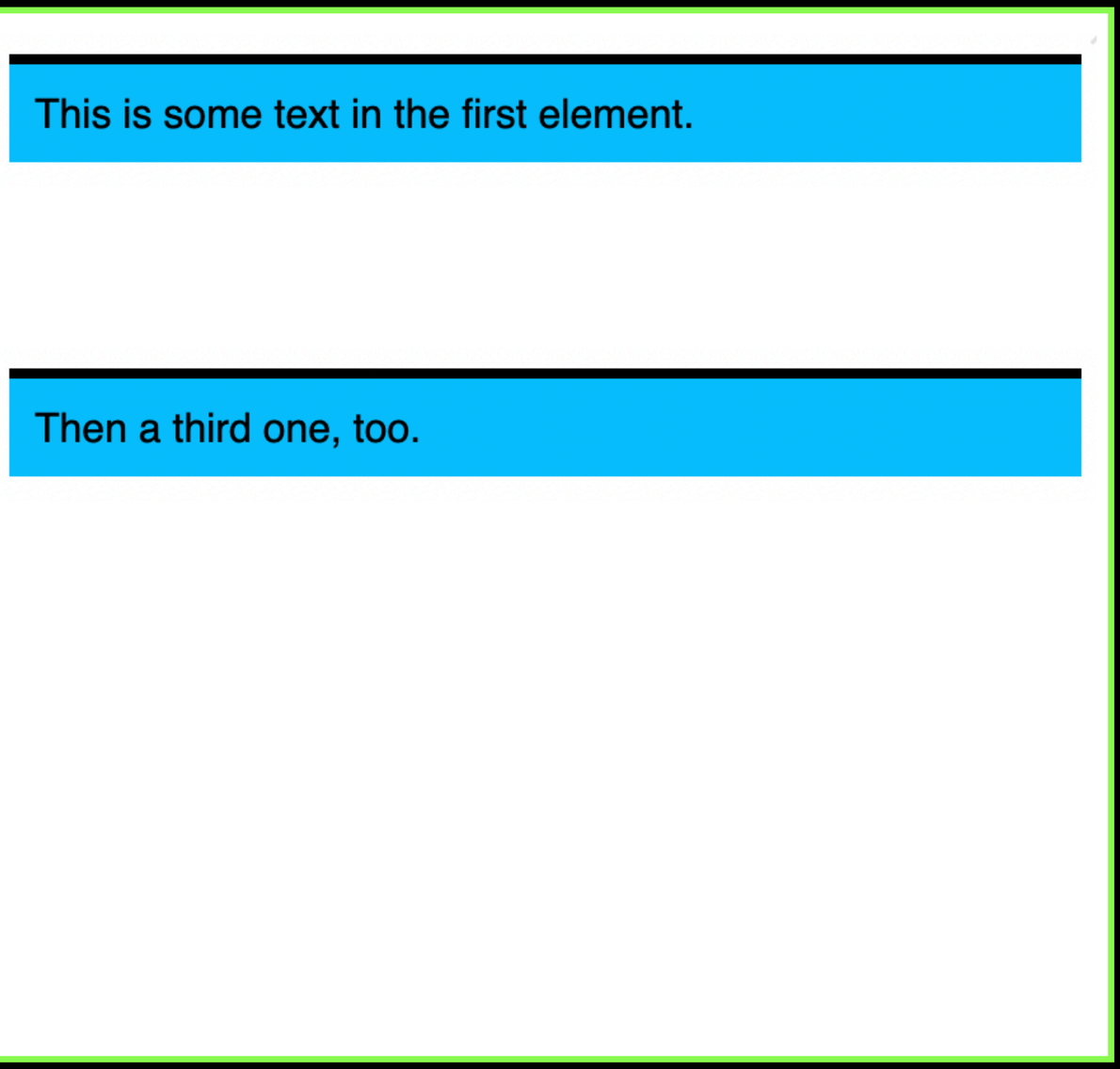



```
…
div:nth-child(2) {
  display: none;
}
```

VISIBILITY

Another way to hide an element, without taking it out of the document's flow, is to declare `visibility: hidden`. This keeps the space that something takes up, but just hides the content.

VISIBILITY

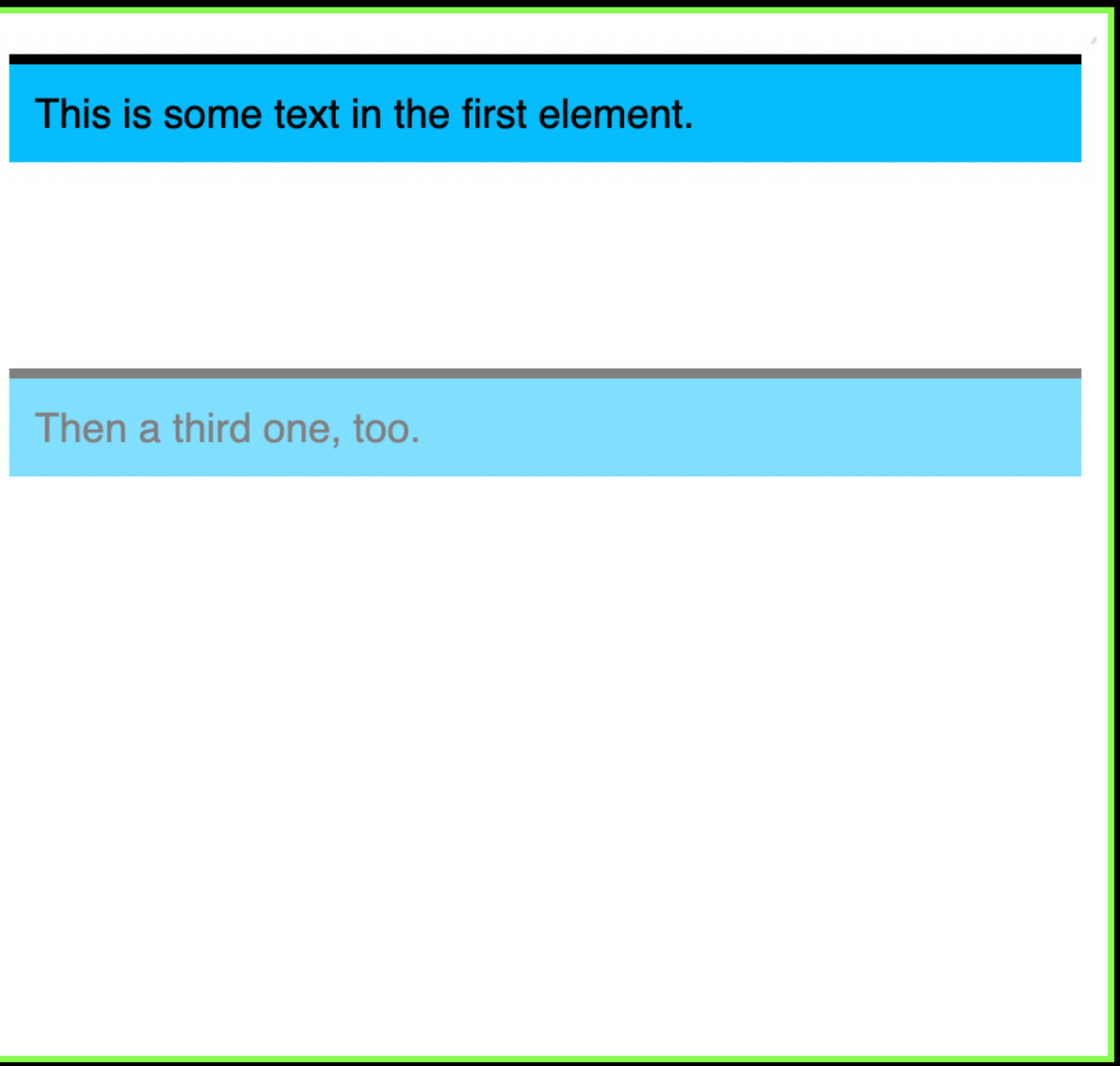



```
…
div:nth-child(2) {
  visibility: hidden;
}
```

OPACITY

You can also set the opacity of a block to hide it (like we did with color). Elements with opacity (even 0 opacity) can still be interacted with.

OPACITY




```
…
div:nth-child(2) {
  opacity: 0;
}

div:nth-child(3) {
  opacity: 0.5;
}
```

JUST WAIT… NEXT WEEK

We’re going to talk a bit about **responsive design** and **web hosting.** You’ll start to think about the structure of, and container for, your Harmonic Collection. We might also make GitHub accounts… stay tuned

QUESTIONS?

BREAK (10 MINS)

EXERCISE 1

Let’s set up some elements together.

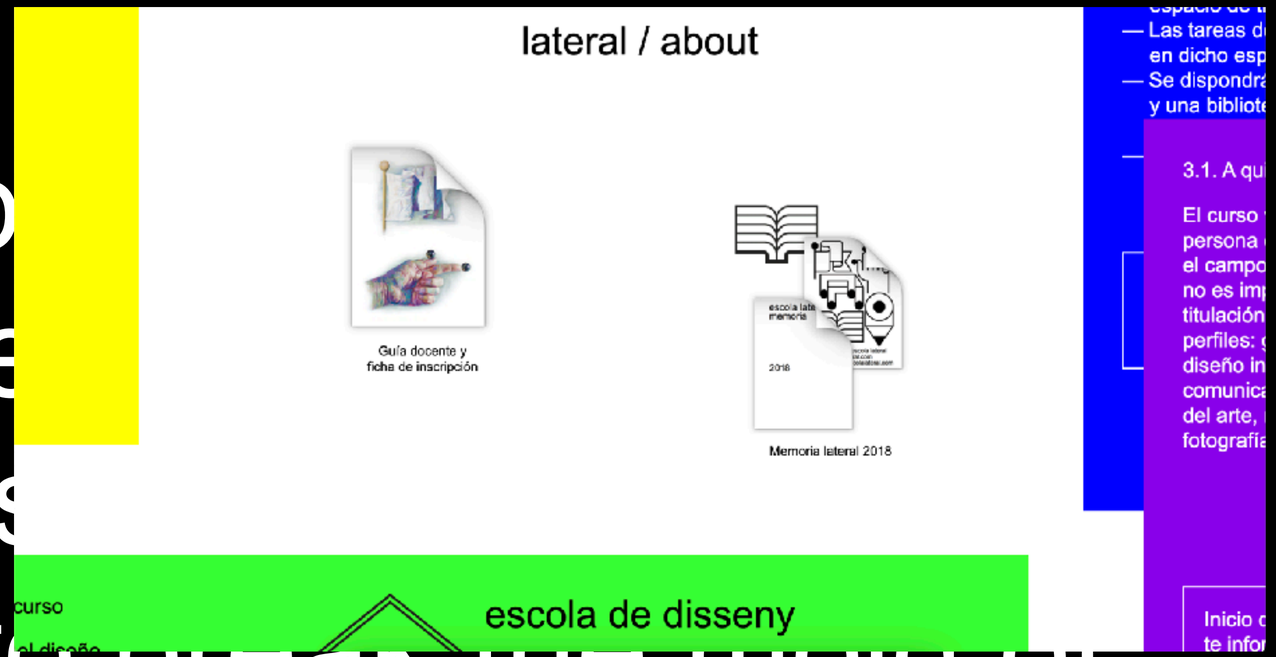
EXERCISE 2

Compose a webpage made of divs arranged creatively. Perhaps it’s a cityscape, a dreamscape, a mind map. I want you to start to break the mold of what you know the web to be.

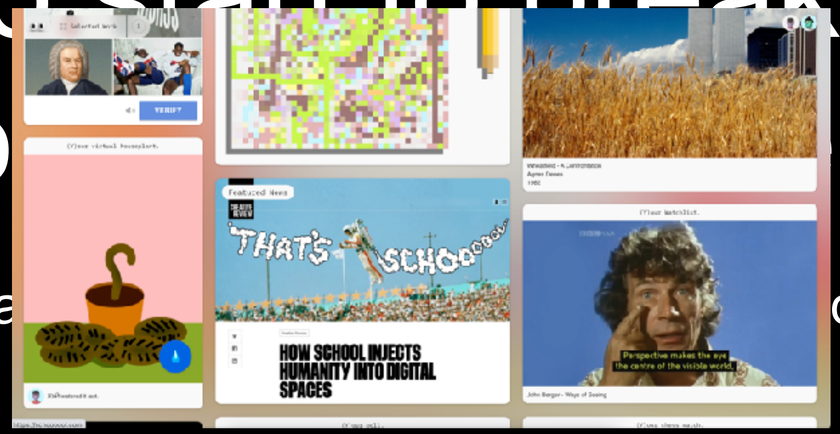
If you’re happy with what you’ve started, feel very free to adapt it into an HC entry.

EXERCISE 2


Gerrit Rietveld Academie
Take a Walk on the Wild Side
Fabulating Alternative Imaginaries in Art & Life
Intro
Calendar
Preliminary Programme
Conference
Rietveld Uncut
Attendance
Lexi/con
Fabula
Telling Stories
Against the day
Reading Groups
Colophon


lateral / about
Guía docente y
ficha de inscripción
Memoria lateral 2018
escola de disseny


Feel Good Factor (Re-Run)
3 Angels Radio (Re-Run)
SAMEDI 19 AOÛT
THANK YOU


THAT'S SCHOOL
HOW SCHOOL INJECTS HUMANITY INTO DIGITAL SPACES


CLOUD
AIRCRAFT

QUESTIONS?

HOMEWORK FOR YOU

> Practice layouts using https://learnlayout.com/

FOR ME

> Harmonic Collection Entry 3

> Email me a webpage you find enticing—design-wise, functionally, in terms of content. Next week we're going to start about the web writ-large, and I want us to have some examples to talk about. **BE PREPARED TO DISCUSS.**

SESSION FOUR
SEPTEMBER 19, 2023

THANK YOU

DAVIS SCHERER
DSCHERER@NEWSCHOOL.EDU