SESSION FIVE
SEPTEMBER 26, 2023

NETWORKED & RESPONSIVE WEB

ENDLESS APPRECIATION
TO ERIC LI +
MICHAEL FEHRENBACH
FOR THE MATERIAL

DAVIS SCHERER
DSCHERER@NEWSCHOOL.EDU

AGENDA

1. RESPONSIVE WEB
2. NETWORKED WEB
3. EXERCISES
4. MIDTERMS
5. HOMEWORK

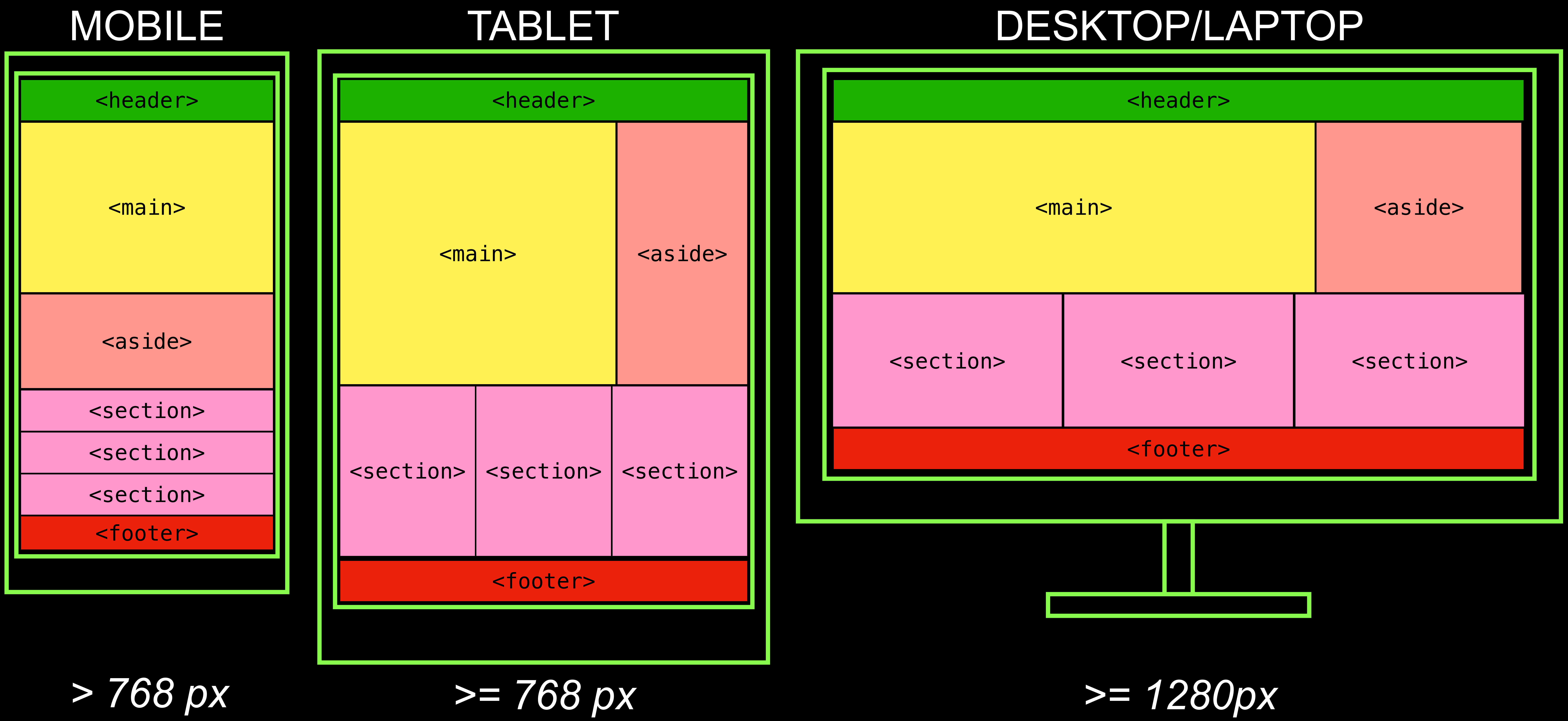
RESPONSIVE DESIGN

“RESPONSIVE DESIGN” as a term was coined in 2010 to reflect the mobile-first and progressive web design strategy that came about with the proliferation of web-surfing on smartphones.

RESPONSIVE DESIGN

It had previously been referred to as *liquid*, *flexible*, *fluid*, or *elastic* design. Essentially, it allowed for a single website to be adapted to different screens rather than forcing the development of a separate mobile site.

RESPONSIVE DESIGN



SESSION FIVE
SEPTEMBER 26, 2023

VIEWPORT

You’ll see this line in the head of many websites:

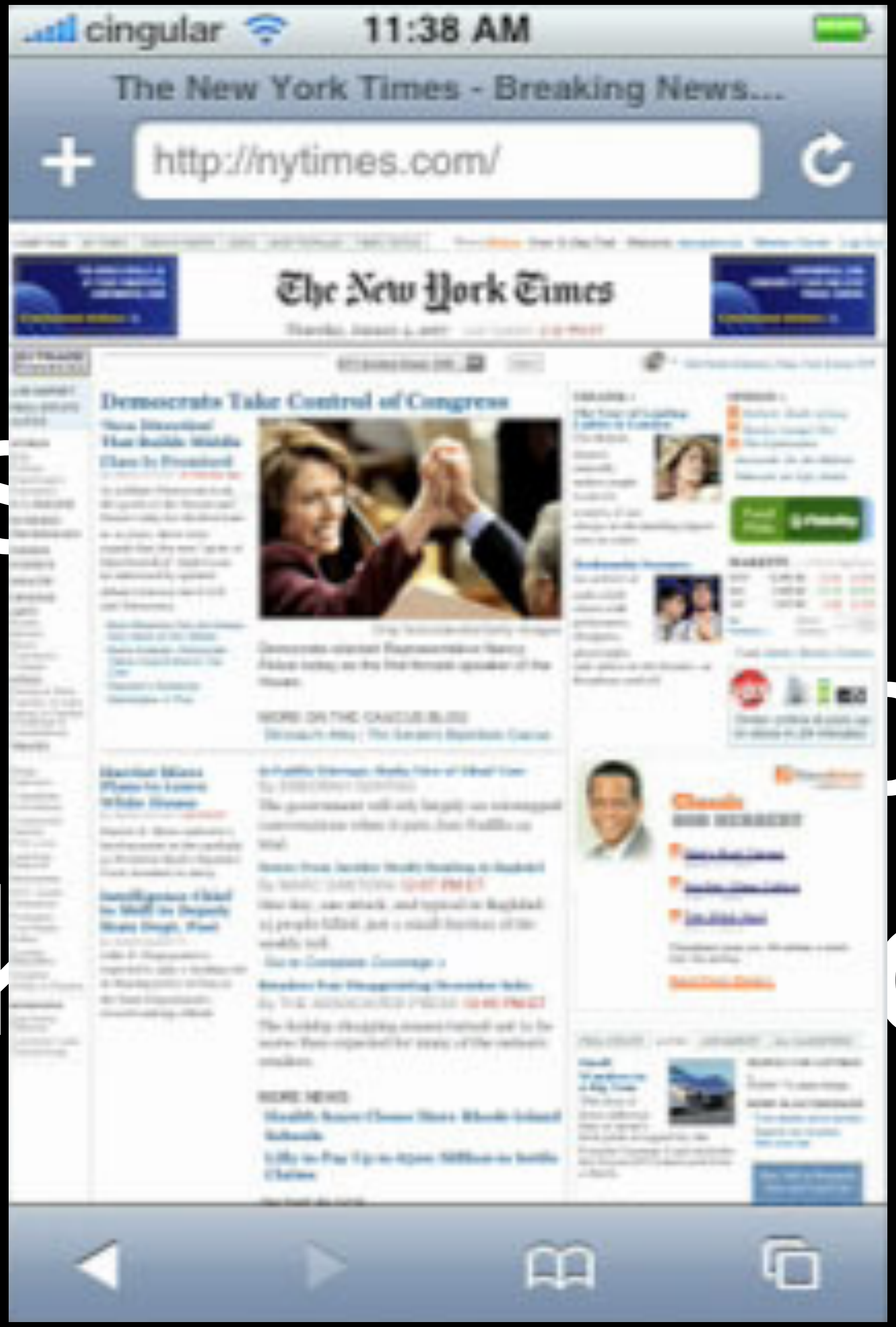
```
<meta name=“viewport”
content=“width=device-width, initial-scale=1”>
```

VIEWPORT

Before responsive design, most mobile browsers would just zoom out/scale a desktop site to fit.

VIEWPORT

Before respons n, most mobile
browsers wou om out/scale a
desk o fit.



cingular 11:38 AM

The New York Times - Breaking News...

http://nytimes.com/

The New York Times

Democrats Take Control of Congress

NYT on Mobile, 2007

VIEWPORT

This line of metadata tells the browser not to simply zoom out—that there is a responsive design in the code and to use that instead.

VIEWPORT

Tells the browser to refer the actual pixel dimension of the screen

```
<meta name=“viewport”
content=“width=device-width,
initial-scale=1”>
```

Sets the starting zoom to 100%

@MEDIA

`@media` is an ‘at-rule’—a CSS statement that tells the code how to behave. `@media` creates ‘media queries,’ which allow the code to check the dimension of the viewport, and to apply select CSS in only that scenario.

@MEDIA

What this ends up looking like is declarations in your code that essentially are conditional selectors by screen-size.

WIDTH BREAKPOINTS

Width is the most common variable used to assign breakpoints to your code—it's the core of responsive design.

WIDTH BREAKPOINTS

You will not be able to design a website
that will fit perfectly in every screen,
it’s just not possible.

WIDTH BREAKPOINTS

You wil website
that w reen,

WIDTH BREAKPOINTS

Width varies the most of screen dimensions—in general, phones screens are between ~375-428px wide, laptops are between ~1440-1680px wide, and desktops are ~2560-3440px wide…

WIDTH BREAKPOINTS

To design for all of these types of screens, we create steps, or **breakpoints,** in our code. This is where our content layouts starts to… break. Maybe images start to bleed off the screen, or lines of text are too short/ long… we need to add in adjustments.

WIDTH BREAKPOINTS

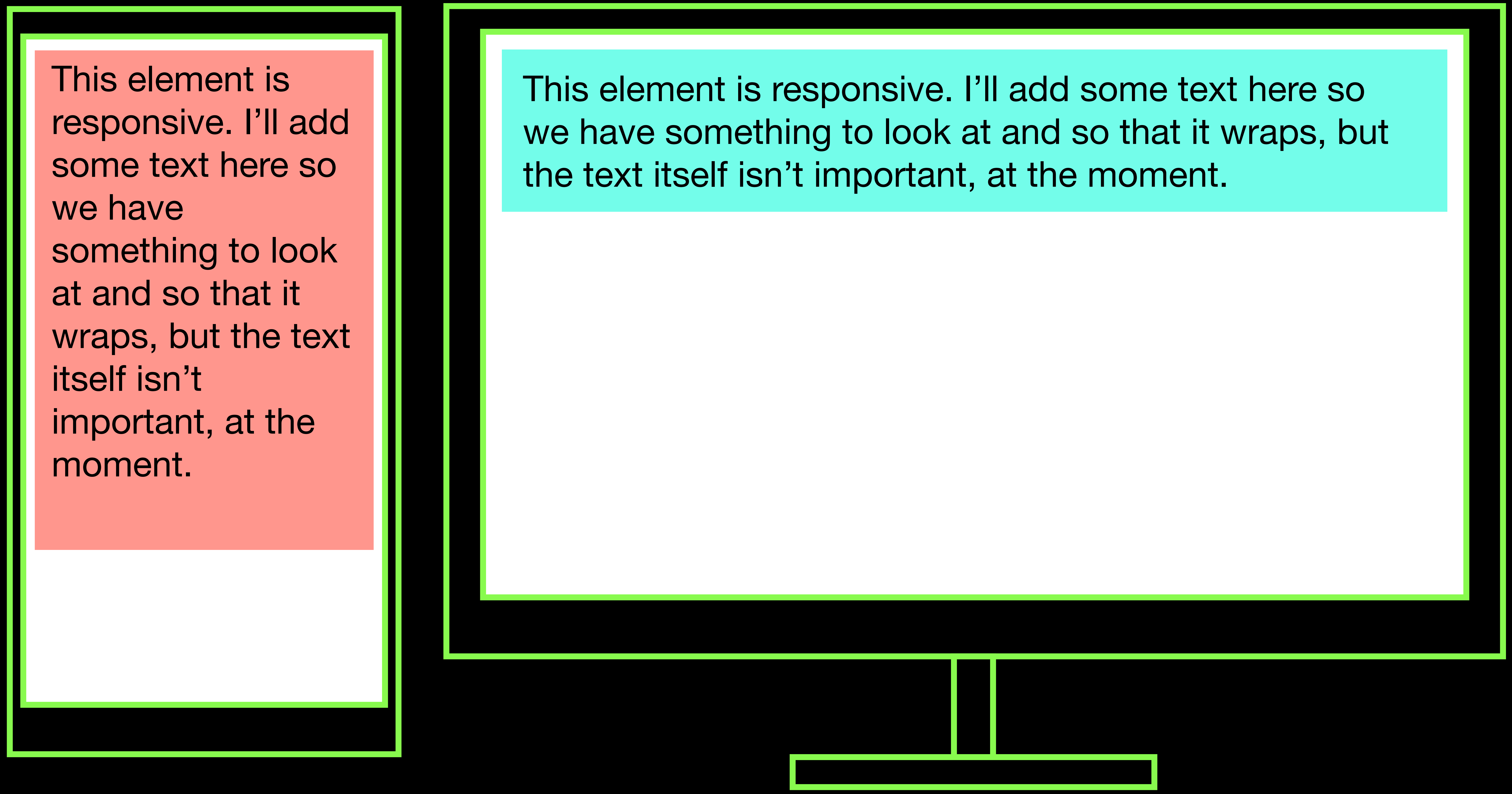
There is no limit to the number of breakpoints you can add into code, but you shouldn’t be code according to devices—rather, code for the design you’re making.

WIDTH BREAKPOINTS

```html
<!DOCTYPE html>
<html>
  <head>
    <meta name="viewport" content="width=device-width, initial-
    scale=1">
    <link href="style.css" rel="stylesheet">
  </head>
  <body>
    <div>
      <p>This element is responsive. I’ll add some text here so we
      have something to look at and so that it wraps, but the text
      itself isn’t important, at the moment.</p>
    </div>
  </body>
</html>
```

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

div {
  padding: 10px;
  background-color: teal;
}

@media (min-width: 400px)
{
  div
    {
      background-color:
      coral;
    }
}
```

WIDTH BREAKPOINTS



This element is responsive. I’ll add some text here so we have something to look at and so that it wraps, but the text itself isn’t important, at the moment.

This element is responsive. I’ll add some text here so we have something to look at and so that it wraps, but the text itself isn’t important, at the moment.

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

div {
  padding: 10px;
  background-color: teal;
}

@media (min-width: 400px)
{
  div
    {
      background-color:
      coral;
    }
}
```

WIDTH BREAKPOINTS

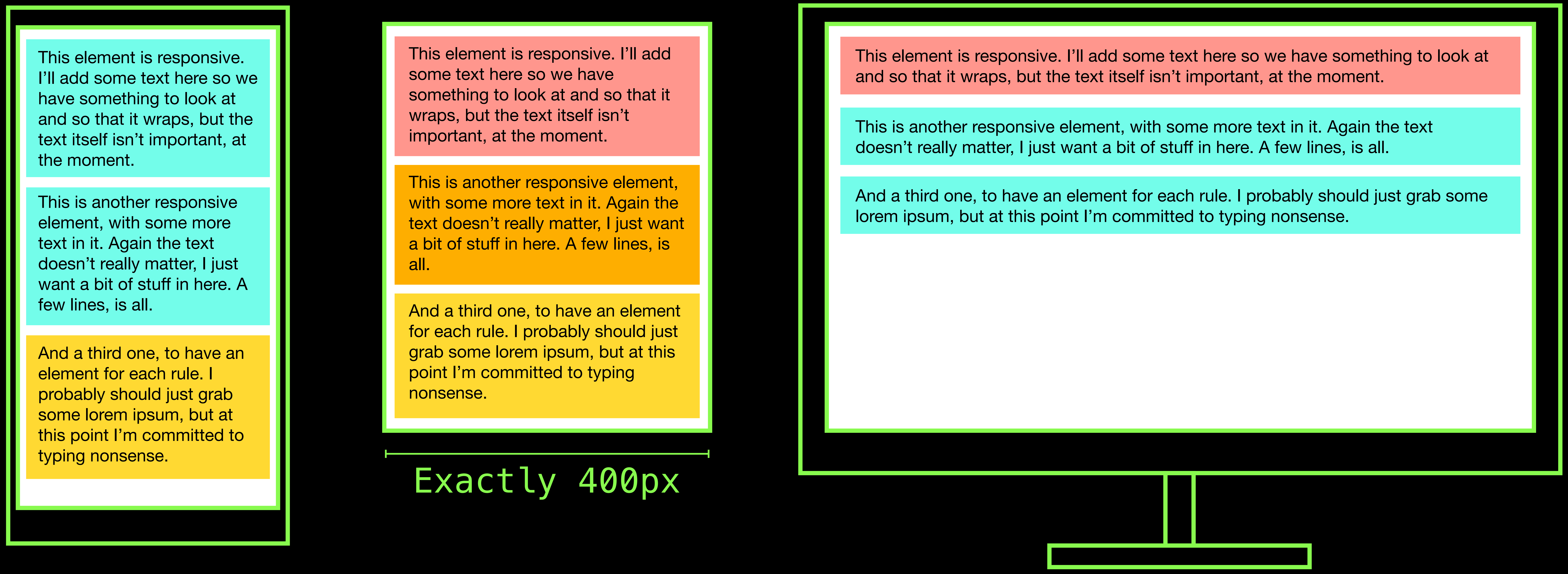
You can define these breakpoints using min-width, width, and max-properties.

WIDTH BREAKPOINTS

```html
<!DOCTYPE html>
<html>
  <head>
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link href="style.css" rel="stylesheet">
  </head>
  <body>
    <div>
      <p>This element is responsive. I’ll add some text here so we have
      something to look at and so that it wraps, but the text itself
      isn’t important, at the moment.</p>
    </div>
    <div>
      <p>This is another responsive element, with some more text in it.
      Again the text doesn’t really matter, I just want a bit of stuff
      in here. A few lines, is all.</p>
    </div>
    <div>
      <p>And a third one, to have an element for each rule. I probably
      should just grab some lorem ipsum, but at this point I’m committed
      to typing nonsense.</p>
    </div>
  </body>
</html>
```

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

div {
  padding: 10px;
  background-color: teal;
}

@media (min-width: 400px)
{ div:first-child {
  background-color: coral;
} }

@media (width: 400px) {
div:nth-child(2) {
  background-color: orange;
} }

@media (max-width: 400px)
{ div:last-child {
  background-color: gold;
} }
```

WIDTH BREAKPOINTS



This element is responsive. I’ll add some text here so we have something to look at and so that it wraps, but the text itself isn’t important, at the moment.

This is another responsive element, with some more text in it. Again the text doesn’t really matter, I just want a bit of stuff in here. A few lines, is all.

And a third one, to have an element for each rule. I probably should just grab some lorem ipsum, but at this point I’m committed to typing nonsense.

This element is responsive. I’ll add some text here so we have something to look at and so that it wraps, but the text itself isn’t important, at the moment.

This is another responsive element, with some more text in it. Again the text doesn’t really matter, I just want a bit of stuff in here. A few lines, is all.

And a third one, to have an element for each rule. I probably should just grab some lorem ipsum, but at this point I’m committed to typing nonsense.

Exactly 400px

This element is responsive. I’ll add some text here so we have something to look at and so that it wraps, but the text itself isn’t important, at the moment.

This is another responsive element, with some more text in it. Again the text doesn’t really matter, I just want a bit of stuff in here. A few lines, is all.

And a third one, to have an element for each rule. I probably should just grab some lorem ipsum, but at this point I’m committed to typing nonsense.

HEIGHT BREAKPOINTS

You can use height as a breakpoint as well… but given that scrolling is so ubiquitous it is rarely used.

WIDTH BREAKPOINTS

```html
<!DOCTYPE html>
<html>
  <head>
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link href="style.css" rel="stylesheet">
  </head>
  <body>
    <div>
      <p>This element is responsive. I’ll add some text here so we have
      something to look at and so that it wraps, but the text itself
      isn’t important, at the moment.</p>
    </div>
    <div>
      <p>This is another responsive element, with some more text in it.
      Again the text doesn’t really matter, I just want a bit of stuff
      in here. A few lines, is all.</p>
    </div>
    <div>
      <p>And a third one, to have an element for each rule. I probably
      should just grab some lorem ipsum, but at this point I’m committed
      to typing nonsense.</p>
    </div>
  </body>
</html>
```

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

div {
  padding: 10px;
  background-color: teal;
}

@media (min-height: 400px)
{ div:first-child {
  background-color: coral;
} }


@media (max-height: 400px)
{ div:last-child {
  background-color: gold;
} }
```

HEIGHT BREAKPOINTS

This element is responsive. I’ll add some text here so we have something to look at and so that it wraps, but the text itself isn’t important, at the moment.

This is another responsive element, with some more text in it. Again the text doesn’t really matter, I just want a bit of stuff in here. A few lines, is all.

And a third one, to have an element for each rule. I probably should just grab some lorem ipsum, but at this point I’m committed to typing nonsense.

This element is responsive. I’ll add some text here so we have something to look at and so that it wraps, but the text itself isn’t important, at the moment.

This is another responsive element, with some more text in it. Again the text doesn’t really matter, I just want a bit of stuff in here. A few lines, is all.

And a third one, to have an element for each rule. I probably should just grab some lorem ipsum, but at this point I’m committed to typing nonsense.

BREAKPOINTS

Basically… @media breakpoints are conditional statements.

IF this, **THEN** that.

BREAKPOINTS

You can specify by orientation as well—`orientation: landscape` or `portrait`

BREAKPOINTS

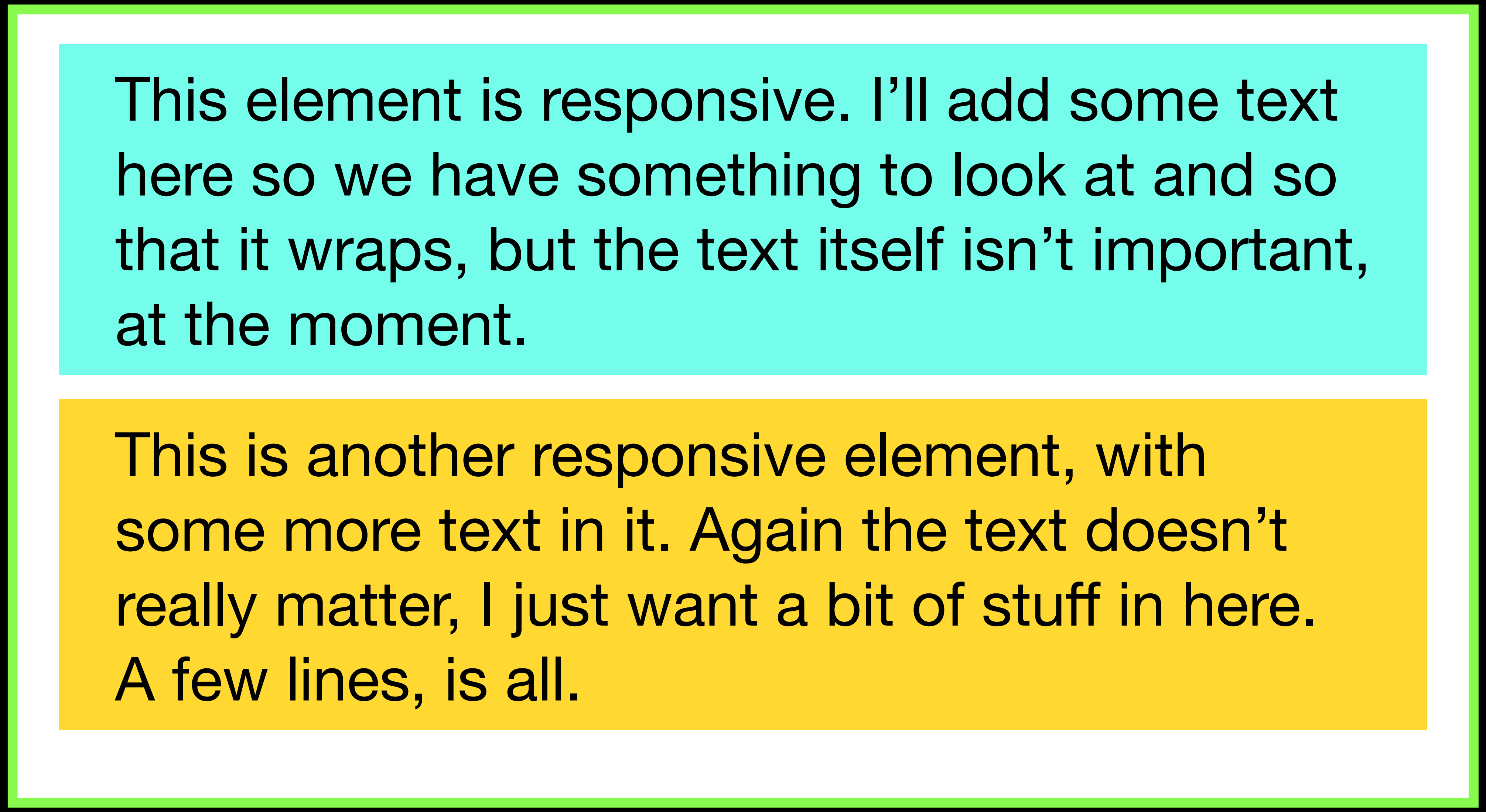
You can specify by orientation as well—`orientation: landscape` or `portrait`

BREAKPOINTS

```
<!DOCTYPE html>
<html>
  <head>
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link href="style.css" rel="stylesheet">
  </head>
  <body>
    <div>
      <p>This element is responsive. I’ll add some text here so we have
      something to look at and so that it wraps, but the text itself
      isn’t important, at the moment.</p>
    </div>
    <div>
      <p>This is another responsive element, with some more text in it.
      Again the text doesn’t really matter, I just want a bit of stuff
      in here. A few lines, is all.</p>
    </div>
  </body>
</html>
```

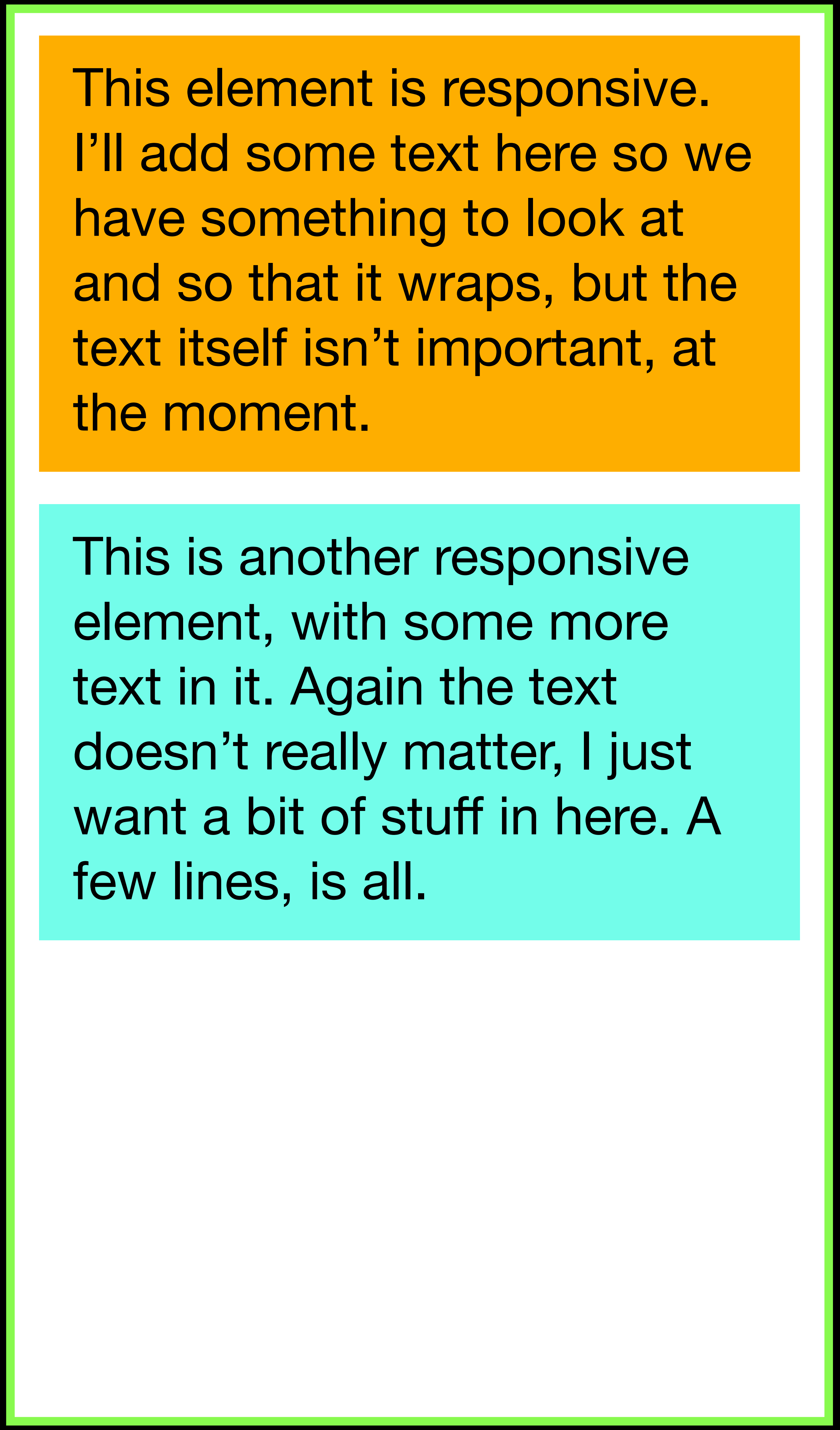
```
body {
  font-family: sans-serif;
  padding: 20px;
}

div {
  padding: 10px;
  background-color: teal;
}

@media (orientation: portrait)
{
  div:first-child {
    background-color: orange;
  }
}

@media (orientation:
landscape) {
  div:last-child
  {
      background-color: gold;
  }
}
```

HEIGHT BREAKPOINTS



This element is responsive. I’ll add some text here so we have something to look at and so that it wraps, but the text itself isn’t important, at the moment.

This is another responsive element, with some more text in it. Again the text doesn’t really matter, I just want a bit of stuff in here. A few lines, is all.



This element is responsive. I’ll add some text here so we have something to look at and so that it wraps, but the text itself isn’t important, at the moment.

This is another responsive element, with some more text in it. Again the text doesn’t really matter, I just want a bit of stuff in here. A few lines, is all.

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

div {
  padding: 10px;
  background-color: teal;
}

@media (orientation: portrait)
{
  div:first-child {
    background-color: orange;
  }
}

@media (orientation:
landscape) {
  div:last-child
  {
        background-color: gold;
  }
}
```

BREAKPOINTS

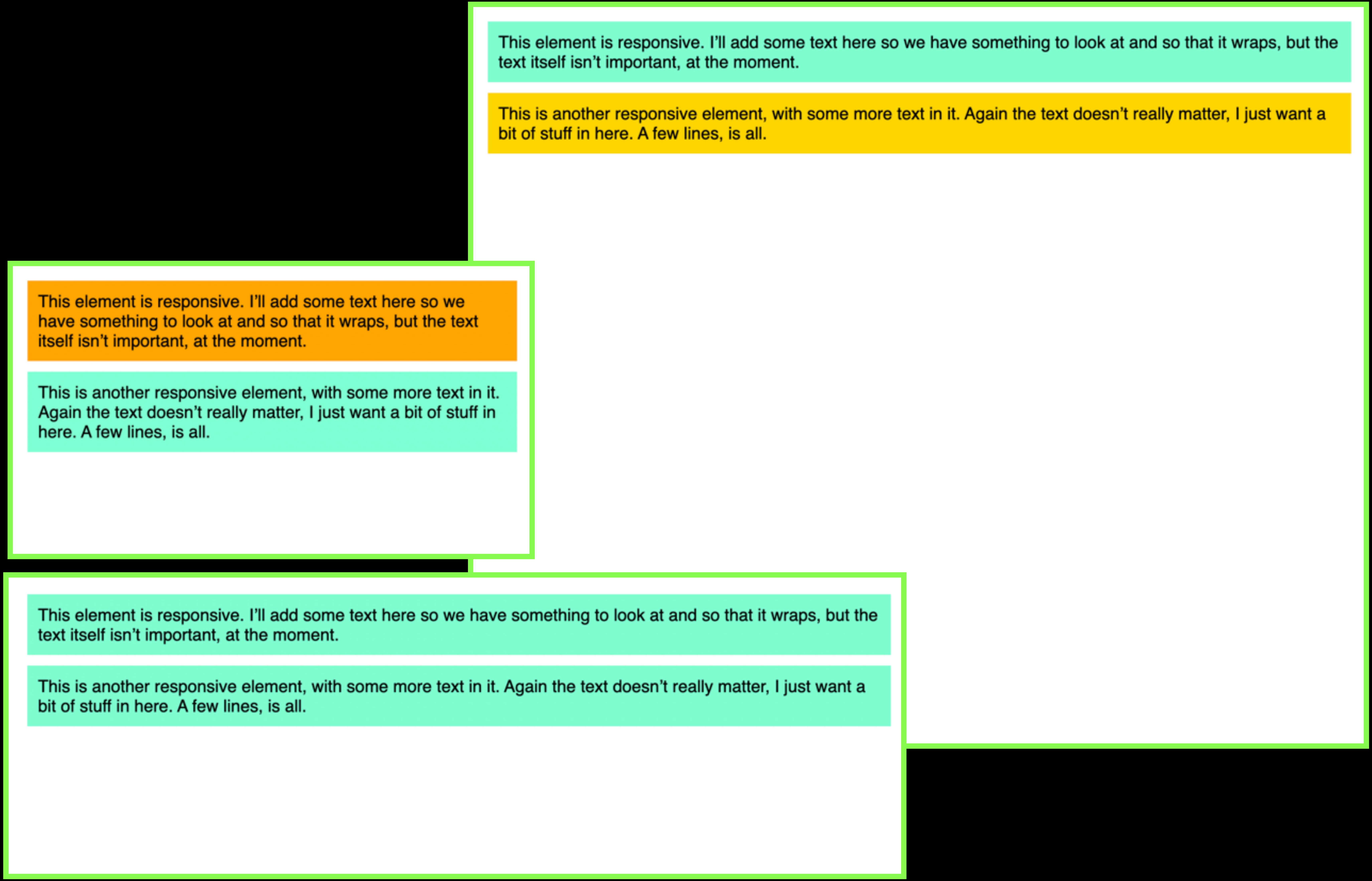
You can also create boolean statements with and/or to combine multiple media queries. Try using this for a range of sizes or to combine width and height.

BREAKPOINTS

```
<!DOCTYPE html>
<html>
  <head>
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link href="style.css" rel="stylesheet">
  </head>
  <body>
    <div>
      <p>This element is responsive. I’ll add some text here so we have
      something to look at and so that it wraps, but the text itself
      isn’t important, at the moment.</p>
    </div>
    <div>
      <p>This is another responsive element, with some more text in it.
      Again the text doesn’t really matter, I just want a bit of stuff
      in here. A few lines, is all.</p>
    </div>
  </body>
</html>
```

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

div {
  padding: 10px;
  background-color: teal;
}

@media (min-width: 300px) and
(max-width: 500px)
{
  div:first-child {
    background-color: orange;
  }
}

@media (min-width: 300px) and
(min-height: 300px)
{
  div:last-child {
    background-color: gold;
  }
}
```

BREAKPOINTS



This element is responsive. I'll add some text here so we have something to look at and so that it wraps, but the text itself isn't important, at the moment.

This is another responsive element, with some more text in it. Again the text doesn't really matter, I just want a bit of stuff in here. A few lines, is all.

This element is responsive. I'll add some text here so we have something to look at and so that it wraps, but the text itself isn't important, at the moment.

This is another responsive element, with some more text in it. Again the text doesn't really matter, I just want a bit of stuff in here. A few lines, is all.

This element is responsive. I'll add some text here so we have something to look at and so that it wraps, but the text itself isn't important, at the moment.

This is another responsive element, with some more text in it. Again the text doesn't really matter, I just want a bit of stuff in here. A few lines, is all.

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

div {
  padding: 10px;
  background-color: teal;
}

@media (min-width: 300px) and
(max-width: 500px)
{
  div:first-child {
    background-color: orange;
  }
}

@media (min-width: 300px) and
(min-height: 300px)
{
  div:last-child {
    background-color: gold;
  }
}
```

BREAKPOINTS

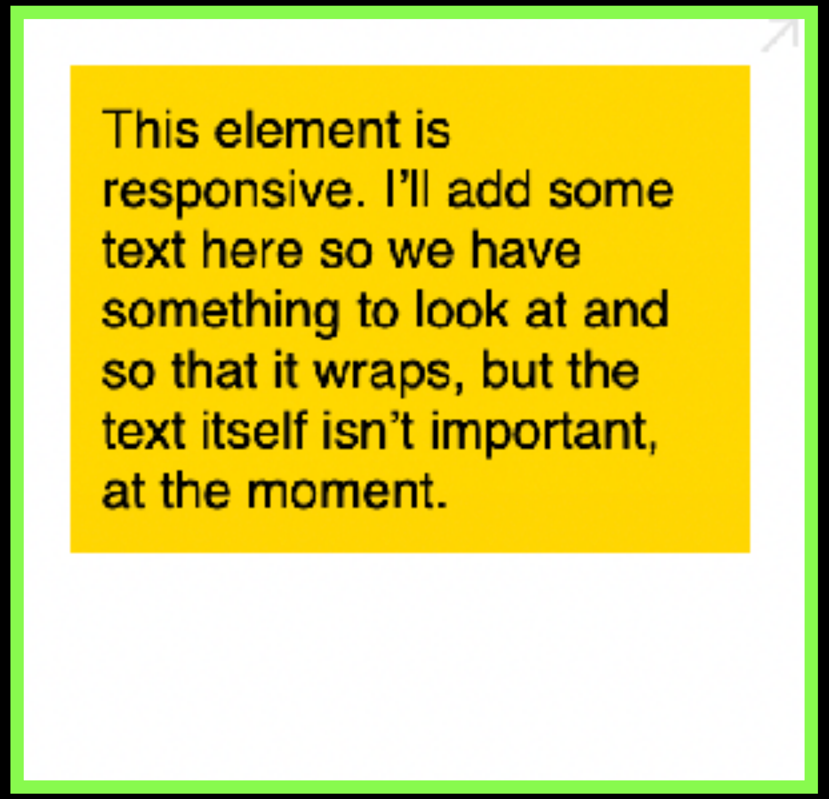
You can also use commas to apply *or* logic—one style, several scenarios.

BREAKPOINTS

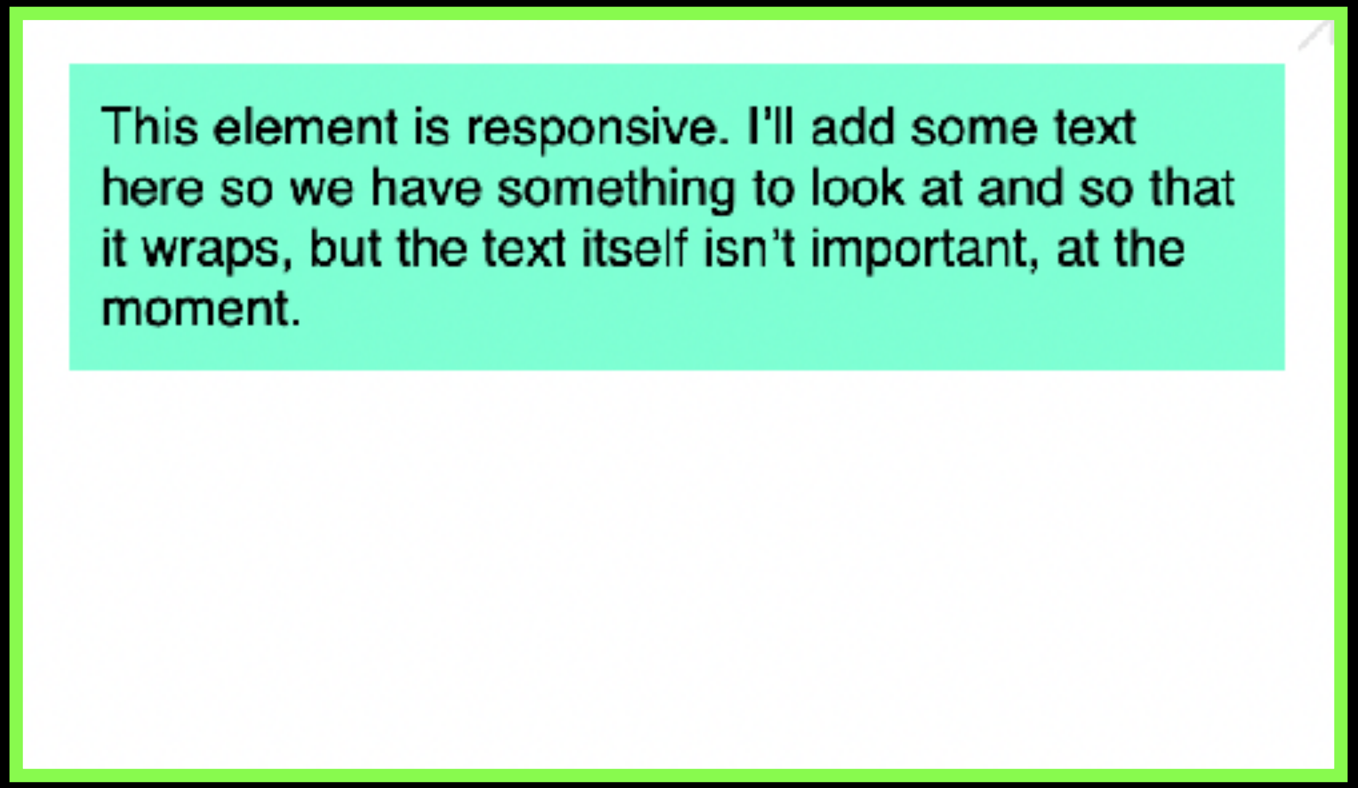
```
<!DOCTYPE html>
<html>
  <head>
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link href="style.css" rel="stylesheet">
  </head>
  <body>
    <div>
      <p>This element is responsive. I’ll add some text here so we have
      something to look at and so that it wraps, but the text itself
      isn’t important, at the moment.</p>
    </div>
  </body>
</html>
```

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

div {
  padding: 10px;
  background-color: teal;
}

@media (max-width: 300px),
(min-width: 500px)
{
  div
    {background-color: gold;}
}
```

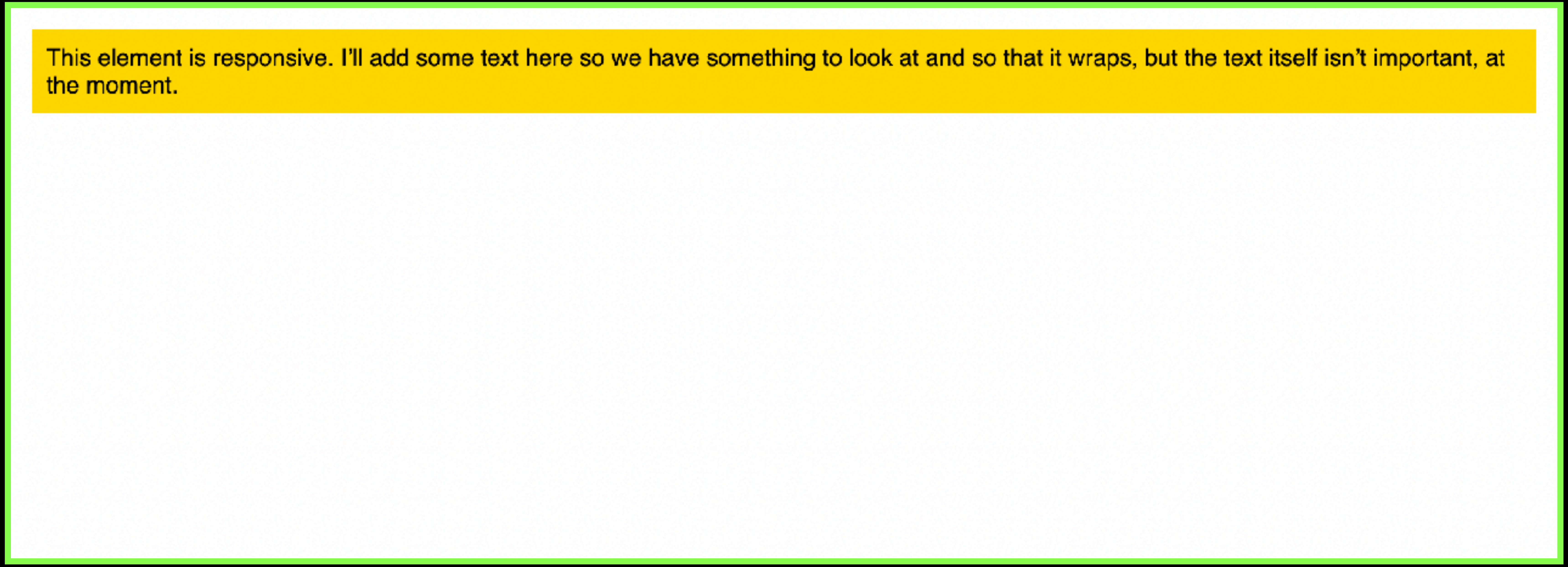
BREAKPOINTS



This element is responsive. I'll add some text here so we have something to look at and so that it wraps, but the text itself isn't important, at the moment.



This element is responsive. I'll add some text here so we have something to look at and so that it wraps, but the text itself isn't important, at the moment.



This element is responsive. I'll add some text here so we have something to look at and so that it wraps, but the text itself isn't important, at the moment.

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

div {
  padding: 10px;
  background-color: teal;
}

@media (max-width: 300px),
(min-width: 500px)
{
  div
    {background-color: gold;}
}
```

BREAKPOINTS

You can also use *NOT*, but… it tends to be confusing with min and max; avoid double-negatives when you can and just say what you mean.

MOBILE-FIRST DESIGN

With complex rules and ranges of devices, it can be easier to start small, with your mobile design, and work up (rather than the other way around)

MOBILE-FIRST DESIGN

You code for mobile, and then add limits and rules to make it work more complexly on desktop.

MOBILE-FIRST DESIGN

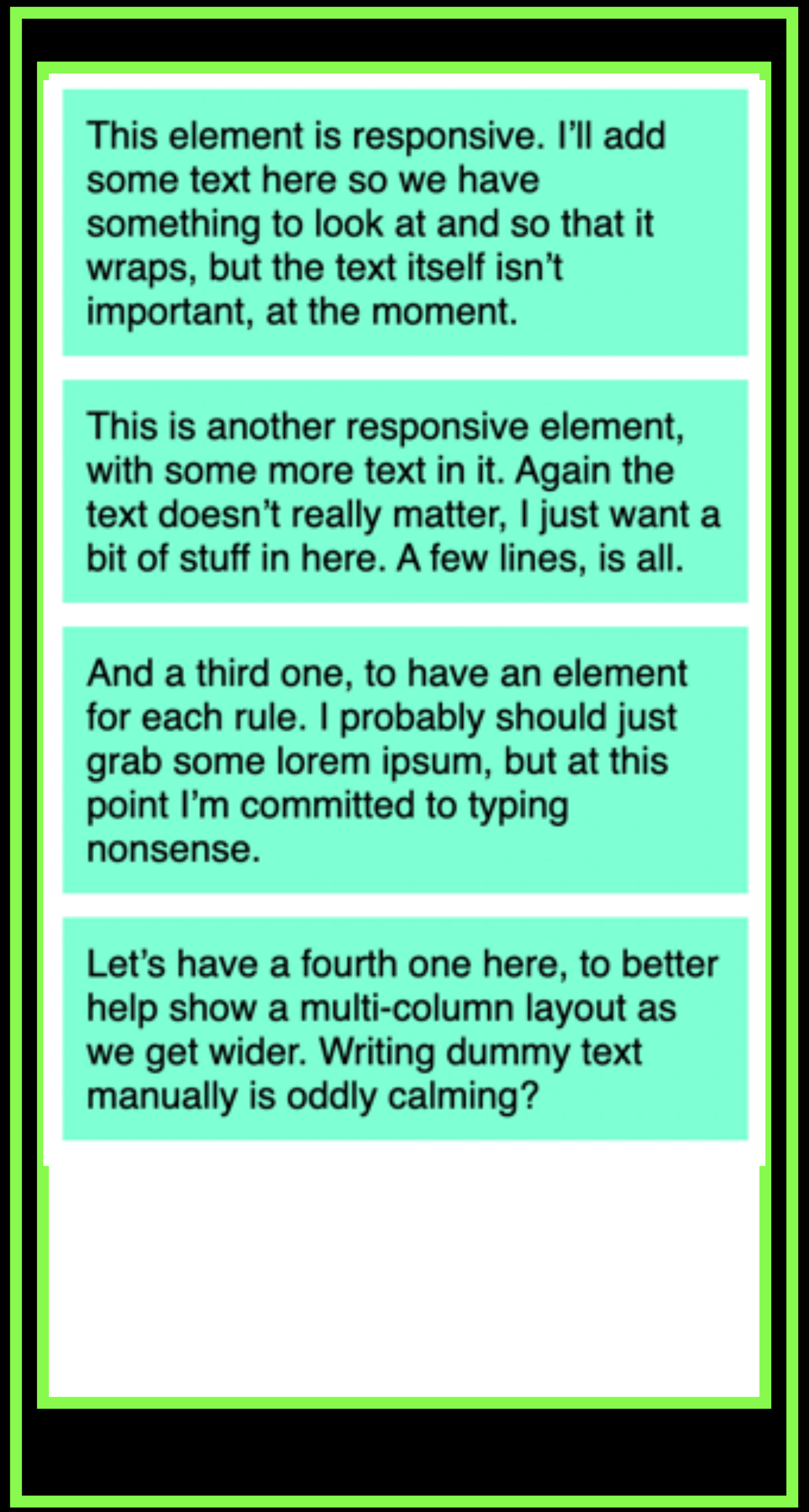
Practically, this looks like coding for mobile and then adding media queries that say min-width rather than max-width

MOBILE-FIRST DESIGN

```html
<!DOCTYPE html>
<html>
  <head>
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link href="style.css" rel="stylesheet">
  </head>
  <body>
    <div>
      <p>This element is responsive. I’ll add some text here so we have something
      to look at and so that it wraps, but the text itself isn’t important, at the
      moment.</p>
    </div>
    <div>
      <p>This is another responsive element, with some more text in it. Again the
      text doesn’t really matter, I just want a bit of stuff in here. A few lines,
      is all.</p>
    </div>
    <div>
      <p>And a third one, to have an element for each rule. I probably should just
      grab some lorem ipsum, but at this point I’m committed to typing nonsense.</
      p>
    </div>
    <div>
     <p>Let’s have a fourth one here, to better help s
      how a multi-column layout as we get wider. Writing dummy text manually is
      oddly calming?</p>
    </div>
  </body>
</html>
```

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

div {
  padding: 10px;
}

section {
  display: flex;
  flex-wrap: wrap;
  gap: 10px;
}

div {
  background-color: aquamarine;
}

@media (min-width: 400px) {
  div {
    background-color: gold;
    width: calc((100% - 10px) / 2);
  }
}

@media (min-width: 500px) {
  div {
    background-color: orange;
    width: calc((100% - 30px) / 4);
  }
}
```

WIDTH BREAKPOINTS

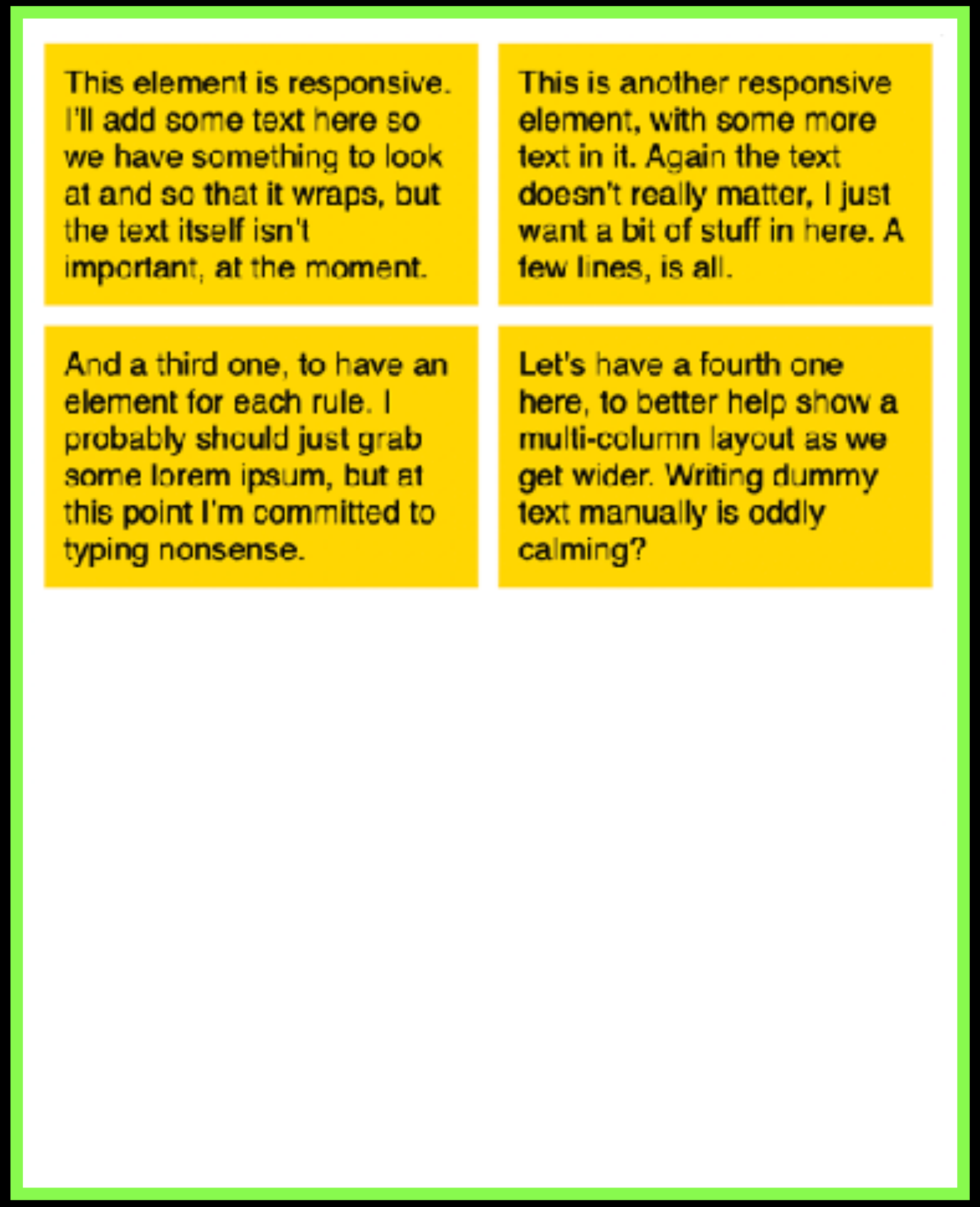


This element is responsive. I’ll add some text here so we have something to look at and so that it wraps, but the text itself isn’t important, at the moment.

This is another responsive element, with some more text in it. Again the text doesn’t really matter, I just want a bit of stuff in here. A few lines, is all.

And a third one, to have an element for each rule. I probably should just grab some lorem ipsum, but at this point I’m committed to typing nonsense.

Let’s have a fourth one here, to better help show a multi-column layout as we get wider. Writing dummy text manually is oddly calming?

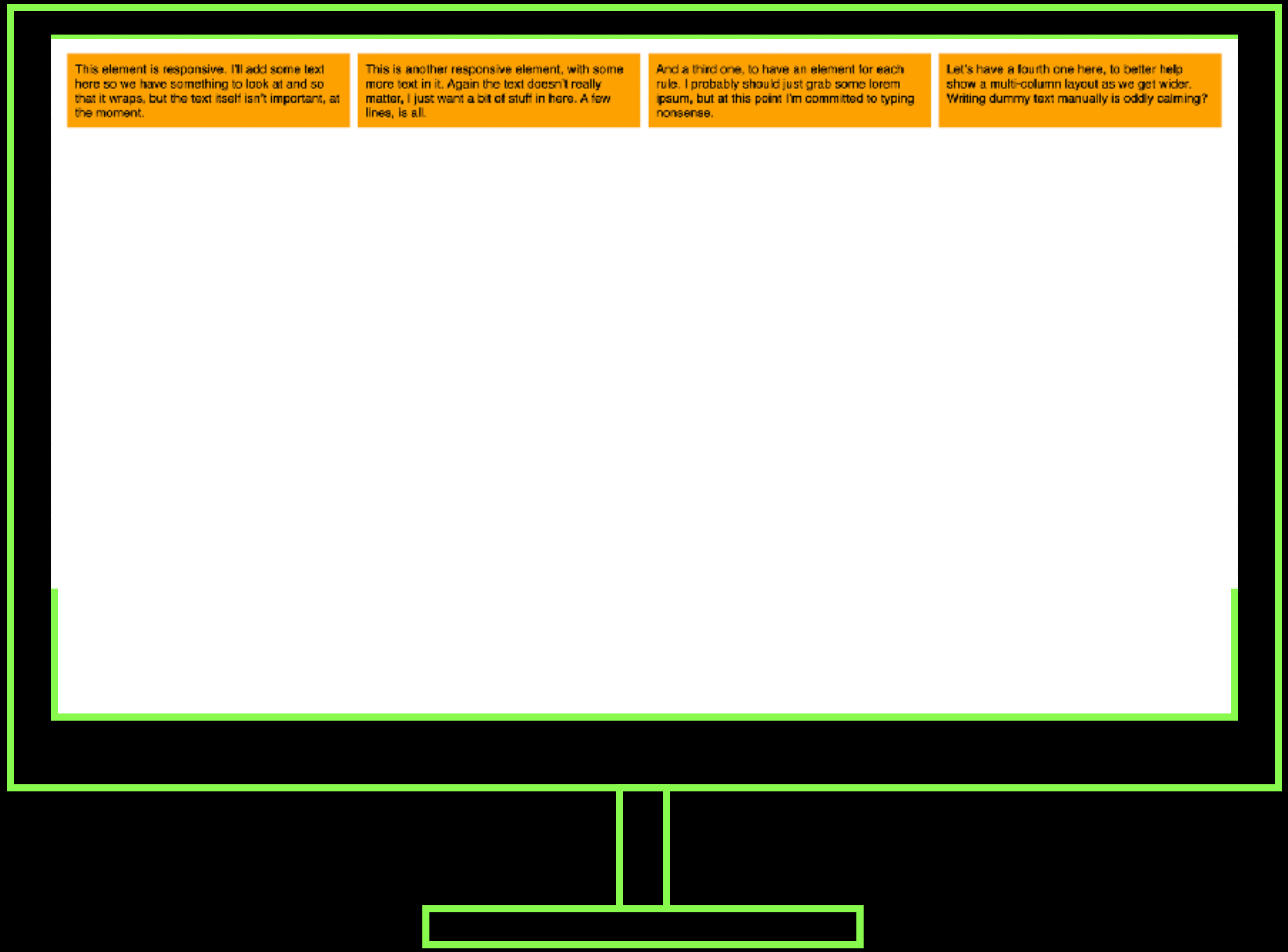


This element is responsive. I’ll add some text here so we have something to look at and so that it wraps, but the text itself isn’t important, at the moment.

This is another responsive element, with some more text in it. Again the text doesn’t really matter, I just want a bit of stuff in here. A few lines, is all.

And a third one, to have an element for each rule. I probably should just grab some lorem ipsum, but at this point I’m committed to typing nonsense.

Let’s have a fourth one here, to better help show a multi-column layout as we get wider. Writing dummy text manually is oddly calming?



This element is responsive. I’ll add some text here so we have something to look at and so that it wraps, but the text itself isn’t important, at the moment.

This is another responsive element, with some more text in it. Again the text doesn’t really matter, I just want a bit of stuff in here. A few lines, is all.

And a third one, to have an element for each rule. I probably should just grab some lorem ipsum, but at this point I’m committed to typing nonsense.

Let’s have a fourth one here, to better help show a multi-column layout as we get wider. Writing dummy text manually is oddly calming?

CSS VARIABLES

In code, **variables** are entities you declare and give a persistent value. It’s the programming equivalent of *x* in algebra.

CSS VARIABLES

CSS can support **custom properties** which act just like variables, allowing you to reuse values throughout a CSS document (and adjust them en masse)

CSS VARIABLES

You declare (or **set**) the variable with a -- and you reference it with var(). So!

```
:root (
 --brand-color:#0000ff
)

div.brand-color{
 color:var(--brand-color);
}
```

CSS VARIABLES

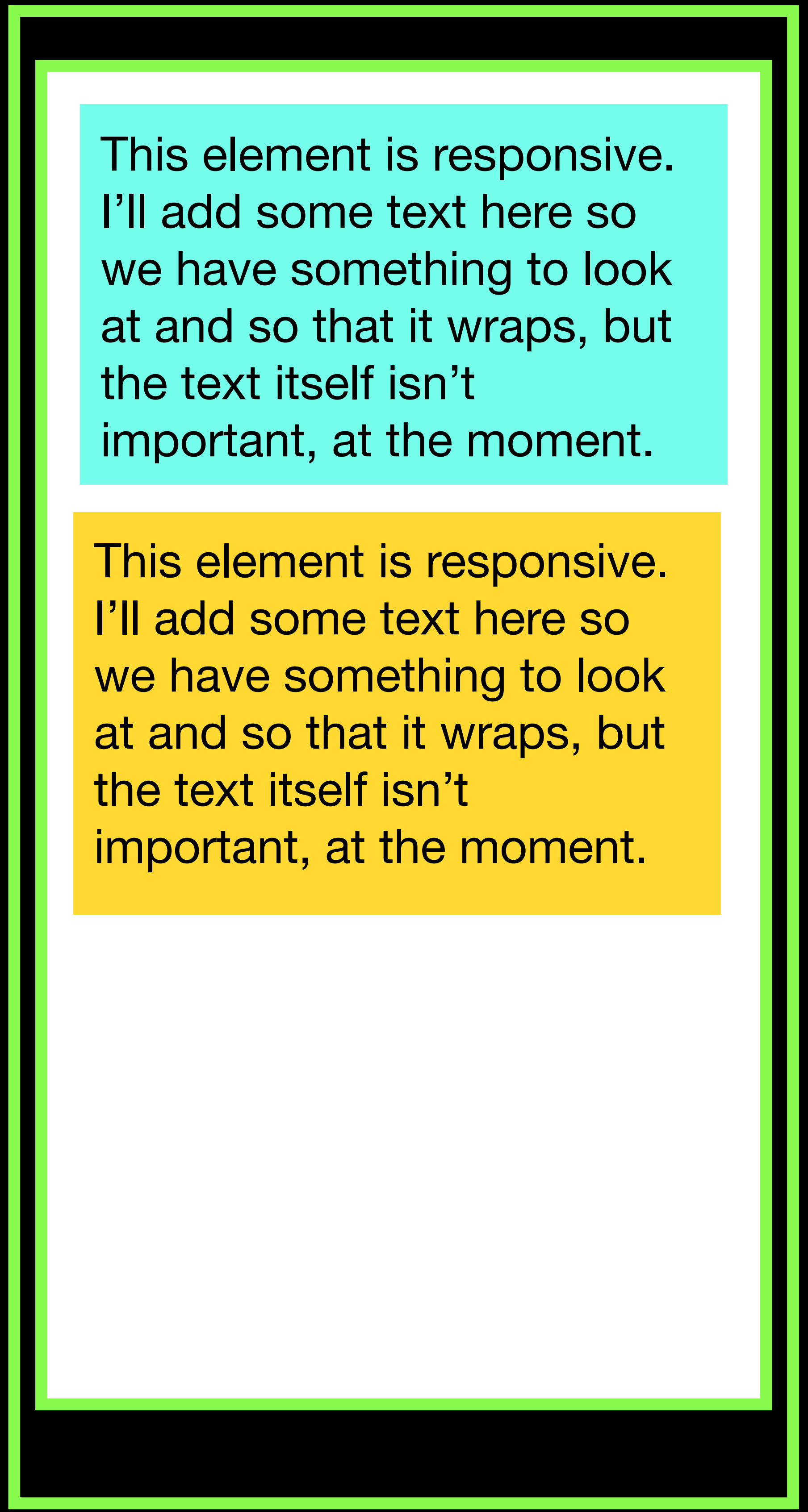
You can use this for any property that you want to use repeatedly, to be consistent, or to easily adjust.

CSS VARIABLES

```html
<!DOCTYPE html>
<html>
  <head>
    <meta name="viewport" content="width=device-width,
    initial-scale=1">
    <link href="style.css" rel="stylesheet">
  </head>
  <body>
    <div>
      <p>This element is responsive. I’ll add some text here
      so we have something to look at and so that it wraps,
      but the text itself isn’t important, at the moment.</p>
    </div>
    <div>
      <p>This is another responsive element, with some more
      text in it. Again the text doesn’t really matter, I just
      want a bit of stuff in here. A few lines, is all.</p>
    </div>
  </body>
</html>
```

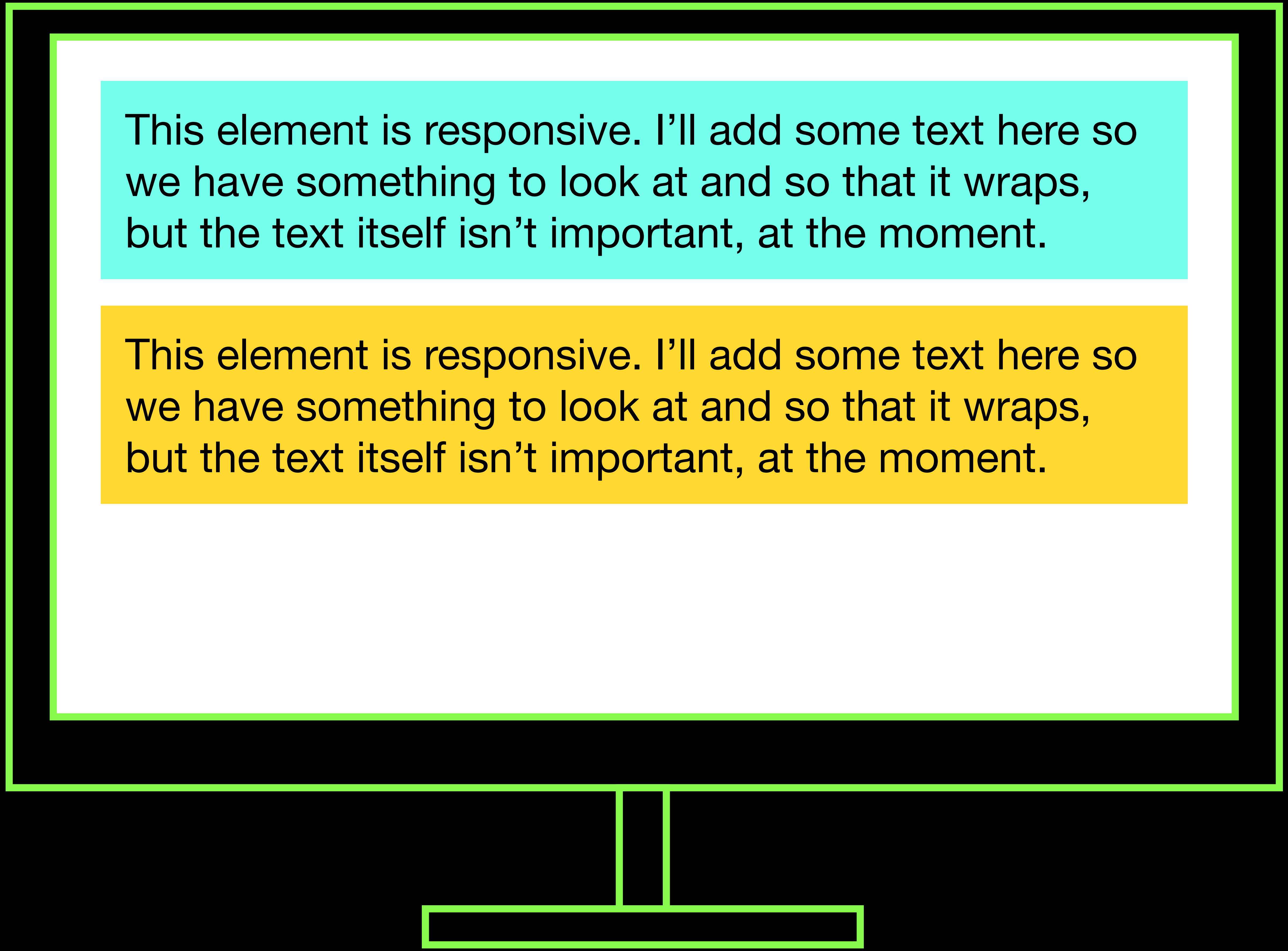
```css
/* Special selector that means
“default”. */
:root {
  --font-size: 16px;
  --spacing: 20px;
}

@media (min-width: 400px) {
  :root {
    --font-size: 24px;
    --spacing: 40px;
  }
}

body {
  font-family: sans-serif;
  font-size: var(--font-size);
  padding: var(--spacing);
}

div {
  background-color: var(--
  background, aquamarine);
  padding: calc(var(--spacing) / 2);
  /
}

div:not(:first-child) {
  --background: gold;
  margin-top: var(--spacing);
}
```

CSS VARIABLES



This element is responsive. I’ll add some text here so we have something to look at and so that it wraps, but the text itself isn’t important, at the moment.

This element is responsive. I’ll add some text here so we have something to look at and so that it wraps, but the text itself isn’t important, at the moment.



This element is responsive. I’ll add some text here so we have something to look at and so that it wraps, but the text itself isn’t important, at the moment.

This element is responsive. I’ll add some text here so we have something to look at and so that it wraps, but the text itself isn’t important, at the moment.

```
/* Special selector that means
“default”. */
:root {
  --font-size: 16px;
  --spacing: 20px;
}

@media (min-width: 400px) {
  :root {
    --font-size: 24px;
    --spacing: 40px;
  }
}

body {
  font-family: sans-serif;
  font-size: var(--font-size);
  padding: var(--spacing);
}

div {
  background-color: var(--
  background, aquamarine);
  padding: calc(var(--spacing) / 2);
  /
}

div:not(:first-child) {
  --background: gold;
  margin-top: var(--spacing);
}
```

CSS VARIABLES

In a responsive design context, think about using variables for type sizes, spacing, etc. This allows you to change values just once for each breakpoint.

OTHER MEDIA FEATURES

`screen` vs `print`

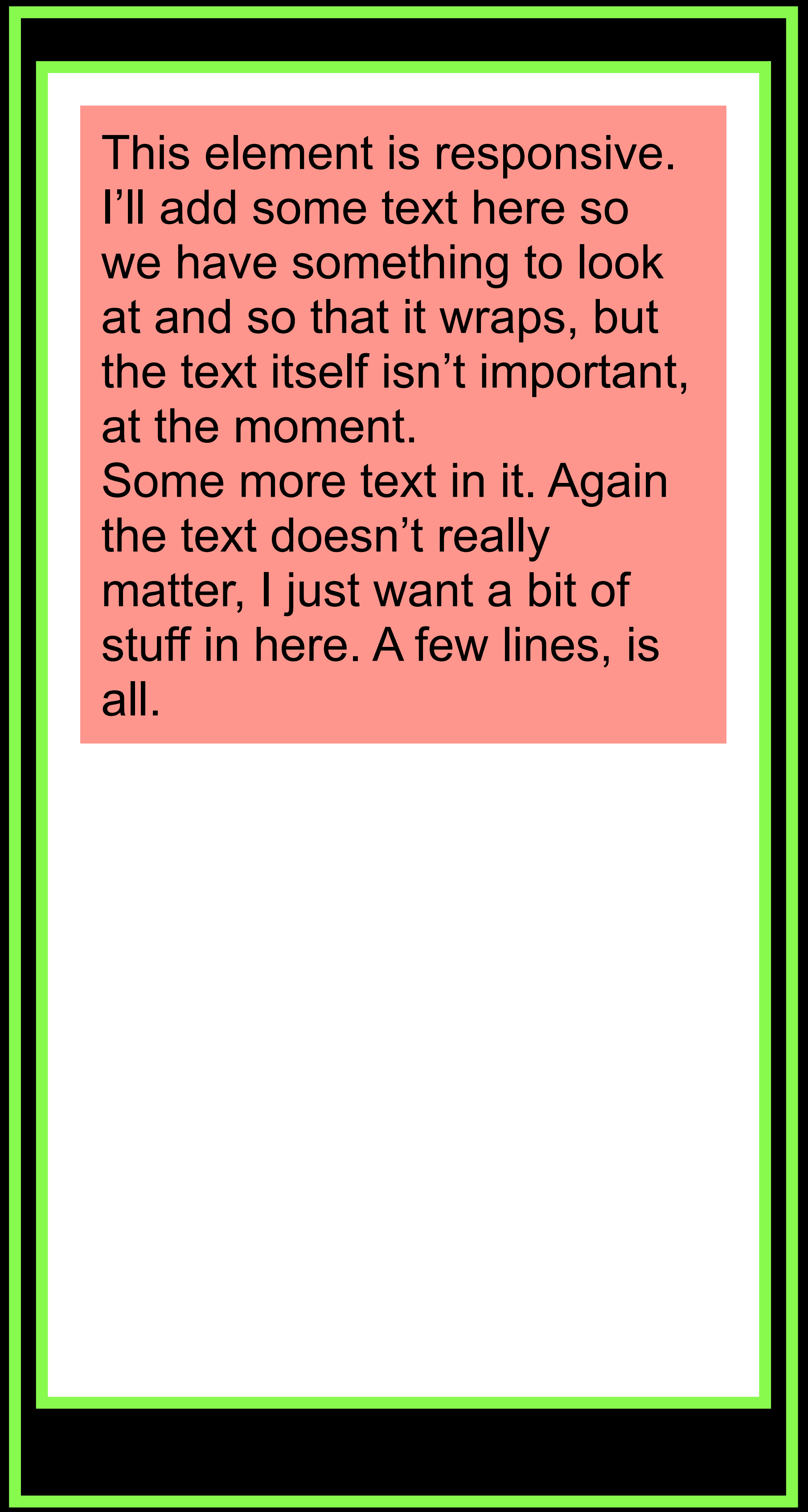
You can code things to print out with specific styles—you may have seen this when you print out a news article or recipe.

SCREEN/PRINT

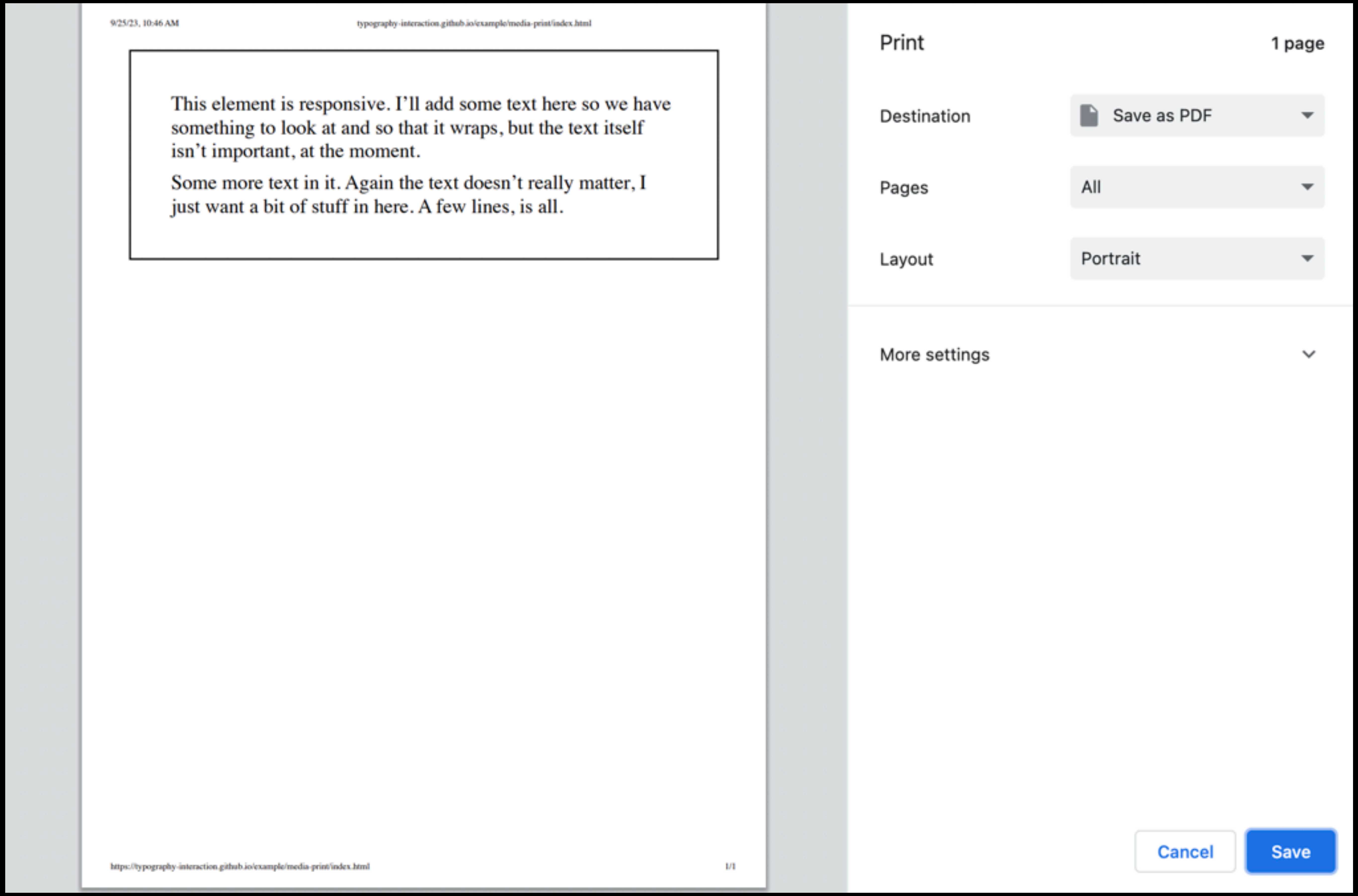
```
<!DOCTYPE html>
<html>
  <head>
    <meta name="viewport" content="width=device-width,
    initial-scale=1">
    <link href="style.css" rel="stylesheet">
  </head>
  <body>
    <div>
      <p>This element is responsive. I’ll add some text here
      so we have something to look at and so that it wraps,
      but the text itself isn’t important, at the moment.</p>
      <p>Some more text in it. Again the text doesn’t really
      matter, I just want a bit of stuff in here. A few lines,
      is all.</p>
    </div>
  </body>
</html>
```

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

div { padding: 10px; }

p:not(:first-child) {
   margin-top: 10px;
}

div { background-color: aquamarine; }

@media screen and (min-width: 400px)
{
   div {
      background-color: coral;
   }
}

@media print {
   div {
      background-color: white;
      border: 2px solid black;
      font-family: serif;
      font-size: 18pt; padding:
      0.5in;
   }
}
```

SCREEN/PRINT



This element is responsive. I’ll add some text here so we have something to look at and so that it wraps, but the text itself isn’t important, at the moment.
Some more text in it. Again the text doesn’t really matter, I just want a bit of stuff in here. A few lines, is all.



9/25/23, 10:46 AM typography-interaction.github.io/example/media-print/index.html

This element is responsive. I’ll add some text here so we have something to look at and so that it wraps, but the text itself isn’t important, at the moment.

Some more text in it. Again the text doesn’t really matter, I just want a bit of stuff in here. A few lines, is all.

https://typography-interaction.github.io/example/media-print/index.html 1/1

Print 1 page

Destination: Save as PDF

Pages: All

Layout: Portrait

OTHER MEDIA FEATURES

It’s good to add media queries relating to hover states as well—mobile browsing/touch-screens don’t support hover, so build in solves for these states.

OTHER MEDIA FEATURES

For example, this would make it so a div was always visible on mobile, instead of only on hover:

```
@media (hover: hover) {
 aside { visibility: hidden; }

 div:hover + aside { visibility: visible; }
}
```

OTHER MEDIA FEATURES

`prefers-color-scheme` switches the styles if a user is in light or dark mode.

```
@media (prefers-color-scheme: dark) {
  body { background: black; color: white; }
}
```

OTHER MEDIA FEATURES

`prefers-reduced-motion` / `prefers-contrast` are accessibility features.

```
@media (prefers-reduced-motion) {
 button { animation: none; }
}

@media(prefers-contrast: more){
 p { background-color: white; color: black; }
}
```

OTHER MEDIA FEATURES

`prefers-color-scheme` switches the styles if a user is in light or dark mode.

DEV TOOLS

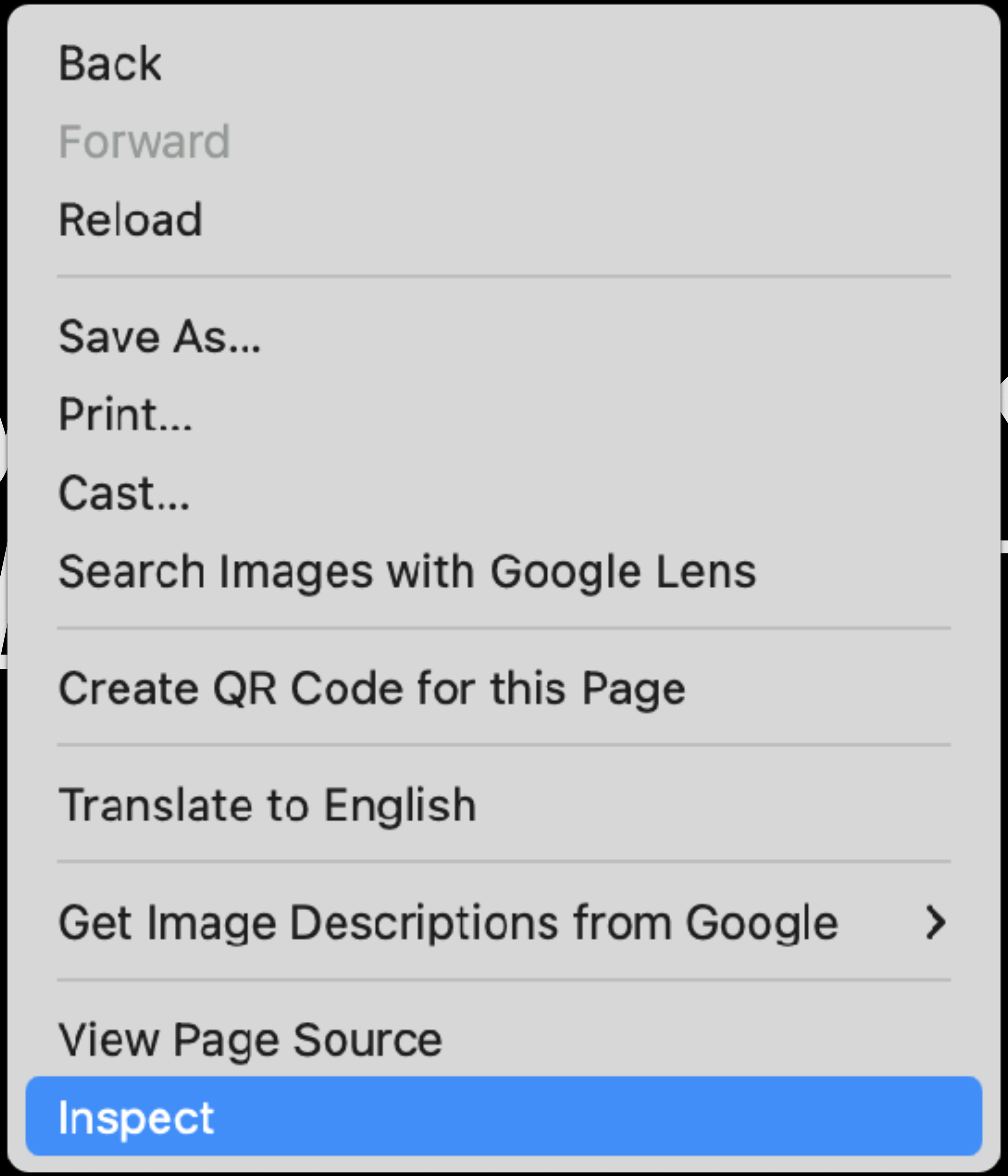
A brief aside on devtools—which are easiest to use on Chrome.

INSPECT

In Chrome, if you right-click any element, you can *INSPECT* it (or ⌘⌥I) to see the source code

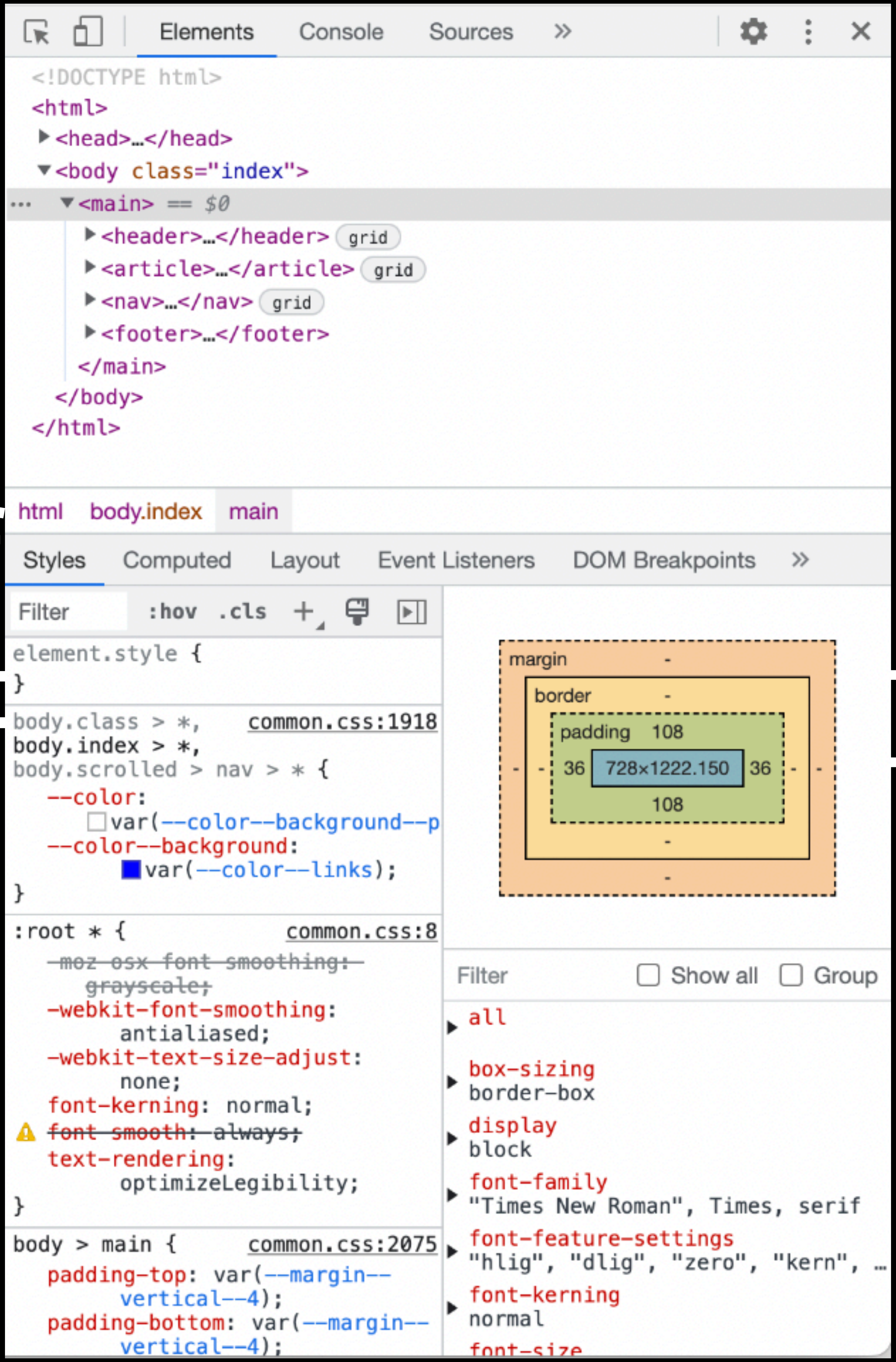
INSPECT

In Chrome, if y k any element,
you can *INSP* ⌥ I) to see the


Back
Forward
Reload
Save As...
Print...
Cast...
Search Images with Google Lens
Create QR Code for this Page
Translate to English
Get Image Descriptions from Google
View Page Source
Inspect

INSPECT

In Chrome, if any element, you can *INSP* I) to see the



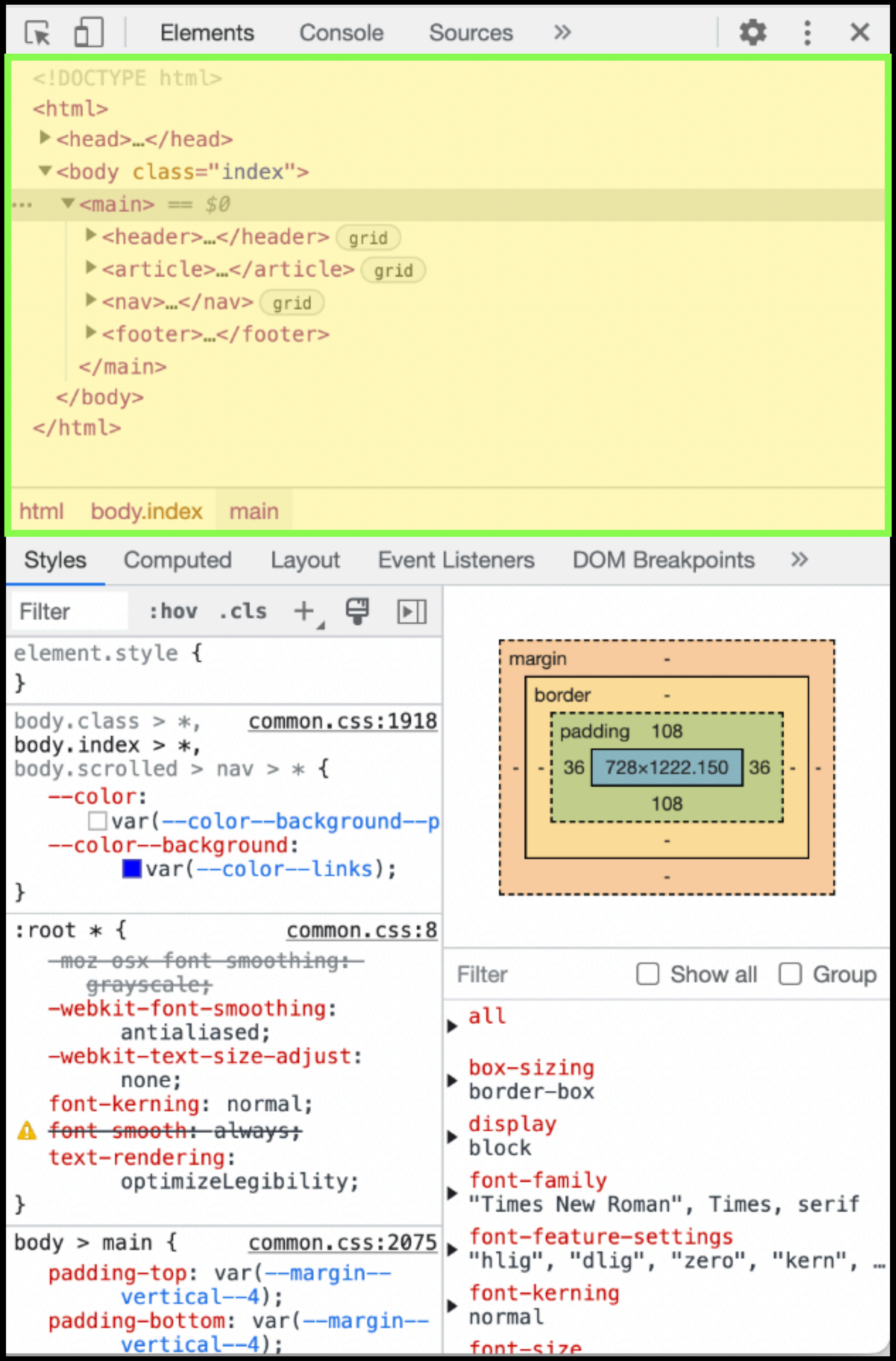
```
<!DOCTYPE html>
<html>
  ▶<head>…</head>
  ▼<body class="index">
…   ▼<main> == $0
      ▶<header>…</header> grid
      ▶<article>…</article> grid
      ▶<nav>…</nav> grid
      ▶<footer>…</footer>
     </main>
   </body>
</html>
```

```
element.style {
}
body.class > *,                    common.css:1918
body.index > *,
body.scrolled > nav > * {
    --color:
        □var(--color--background--p
    --color--background:
          ■var(--color--links);
}
:root * {                             common.css:8
    ~~-moz-osx-font-smoothing:~~
      ~~grayscale;~~
    -webkit-font-smoothing:
          antialiased;
    -webkit-text-size-adjust:
          none;
    font-kerning: normal;
⚠   ~~font-smooth: always;~~
    text-rendering:
          optimizeLegibility;
}
body > main {                      common.css:2075
    padding-top: var(--margin--
          vertical--4);
    padding-bottom: var(--margin--
          vertical--4);
```

```
▶ all
▶ box-sizing
  border-box
▶ display
  block
▶ font-family
  "Times New Roman", Times, serif
▶ font-feature-settings
  "hlig", "dlig", "zero", "kern", …
▶ font-kerning
  normal
  font-size
```

INSPECT

DOM



```
<!DOCTYPE html>
<html>
 ▶<head>…</head>
 ▼<body class="index">
…  ▼<main> == $0
     ▶<header>…</header> grid
     ▶<article>…</article> grid
     ▶<nav>…</nav> grid
     ▶<footer>…</footer>
    </main>
  </body>
</html>
```

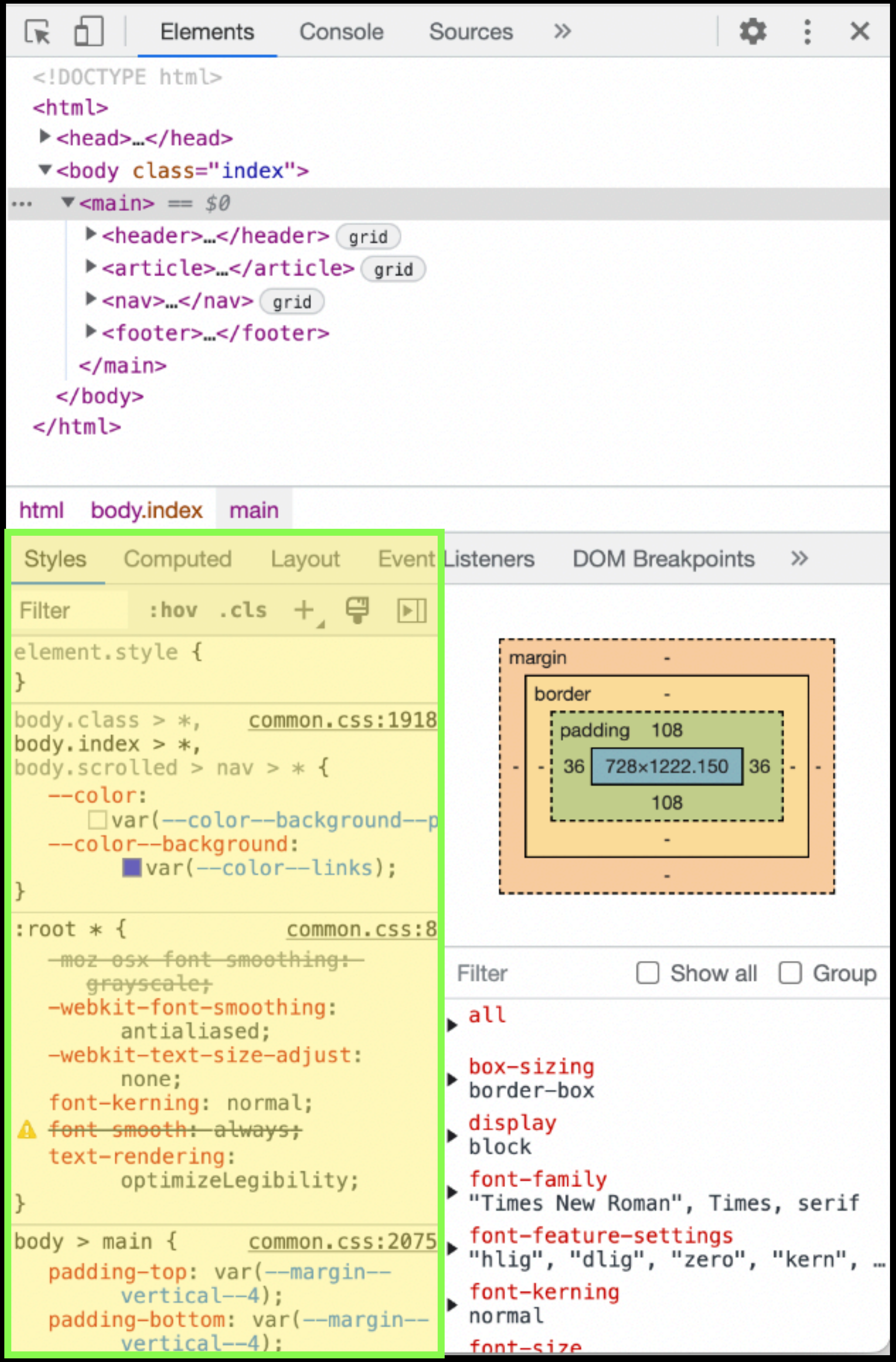
```
element.style {
}
body.class > *,                    common.css:1918
body.index > *,
body.scrolled > nav > * {
  --color:
     □var(--color--background--p
  --color--background:
     ■var(--color--links);
}
:root * {                          common.css:8
  ~~-moz-osx-font-smoothing:~~
     ~~grayscale;~~
  -webkit-font-smoothing:
     antialiased;
  -webkit-text-size-adjust:
     none;
  font-kerning: normal;
⚠ ~~font-smooth: always;~~
  text-rendering:
     optimizeLegibility;
}
body > main {                      common.css:2075
  padding-top: var(--margin--
     vertical--4);
  padding-bottom: var(--margin--
     vertical--4);
```

```
▶ all
▶ box-sizing
  border-box
▶ display
  block
▶ font-family
  "Times New Roman", Times, serif
▶ font-feature-settings
  "hlig", "dlig", "zero", "kern", …
▶ font-kerning
  normal
  font-size
```

html elements, nested hierarchically. You can select an element from here or the webpage to find it in the code

INSPECT

styles pane



```
<!DOCTYPE html>
<html>
 ▶<head>…</head>
 ▼<body class="index">
…  ▼<main> == $0
    ▶<header>…</header> grid
    ▶<article>…</article> grid
    ▶<nav>…</nav> grid
    ▶<footer>…</footer>
   </main>
  </body>
</html>
```

```
element.style {
}
body.class > *,                 common.css:1918
body.index > *,
body.scrolled > nav > * {
   --color:
        □var(--color--background--p
   --color--background:
          ■var(--color--links);
}
:root * {                          common.css:8
   ~~-moz-osx-font-smoothing:~~
     ~~grayscale;~~
   -webkit-font-smoothing:
          antialiased;
   -webkit-text-size-adjust:
          none;
   font-kerning: normal;
⚠  ~~font-smooth: always;~~
   text-rendering:
          optimizeLegibility;
}
body > main {                   common.css:2075
   padding-top: var(--margin--
          vertical--4);
   padding-bottom: var(--margin--
          vertical--4);
```

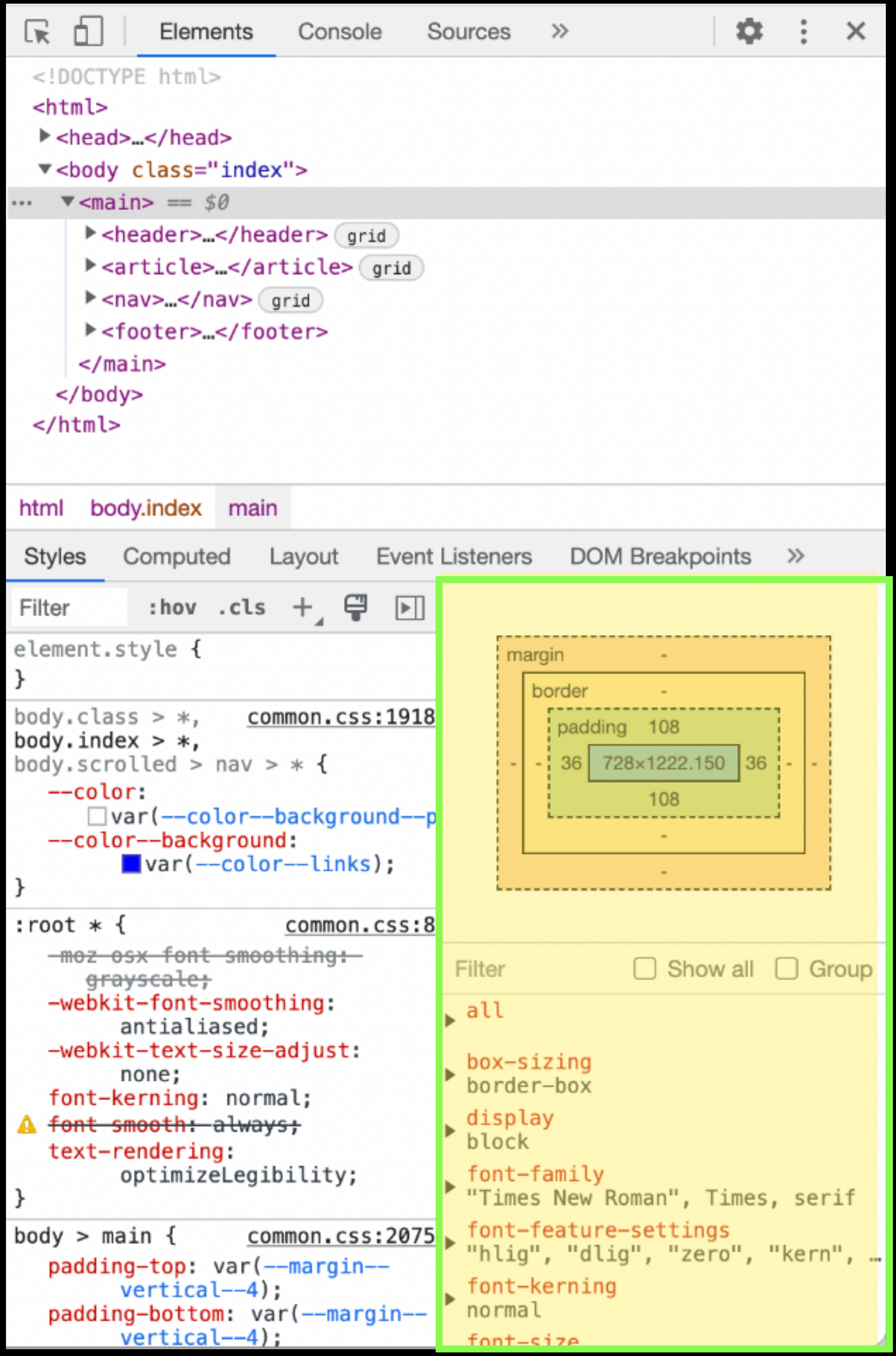
```
▶ all

▶ box-sizing
  border-box
▶ display
  block
▶ font-family
  "Times New Roman", Times, serif
▶ font-feature-settings
  "hlig", "dlig", "zero", "kern", …
▶ font-kerning
  normal
  font-size
```

Tells you the styles attached to whatever is selected in the DOM (HTML) panel

Listed by selector specificity (most specific at the top)

INSPECT

styles pane



```
<!DOCTYPE html>
<html>
 ▶<head>…</head>
 ▼<body class="index">
 … ▼<main> == $0
     ▶<header>…</header> grid
     ▶<article>…</article> grid
     ▶<nav>…</nav> grid
     ▶<footer>…</footer>
    </main>
  </body>
</html>
```

```
element.style {
}
body.class > *,                common.css:1918
body.index > *,
body.scrolled > nav > * {
   --color:
       □var(--color--background--p
   --color--background:
       ■var(--color--links);
}
:root * {                       common.css:8
   ~~-moz-osx-font-smoothing:~~
       ~~grayscale;~~
   -webkit-font-smoothing:
       antialiased;
   -webkit-text-size-adjust:
       none;
   font-kerning: normal;
⚠  ~~font-smooth: always;~~
   text-rendering:
       optimizeLegibility;
}
body > main {                  common.css:2075
   padding-top: var(--margin--
       vertical--4);
   padding-bottom: var(--margin--
       vertical--4);
```

```
▶ all
▶ box-sizing
  border-box
▶ display
  block
▶ font-family
  "Times New Roman", Times, serif
▶ font-feature-settings
  "hlig", "dlig", "zero", "kern", …
▶ font-kerning
  normal
  font-size
```

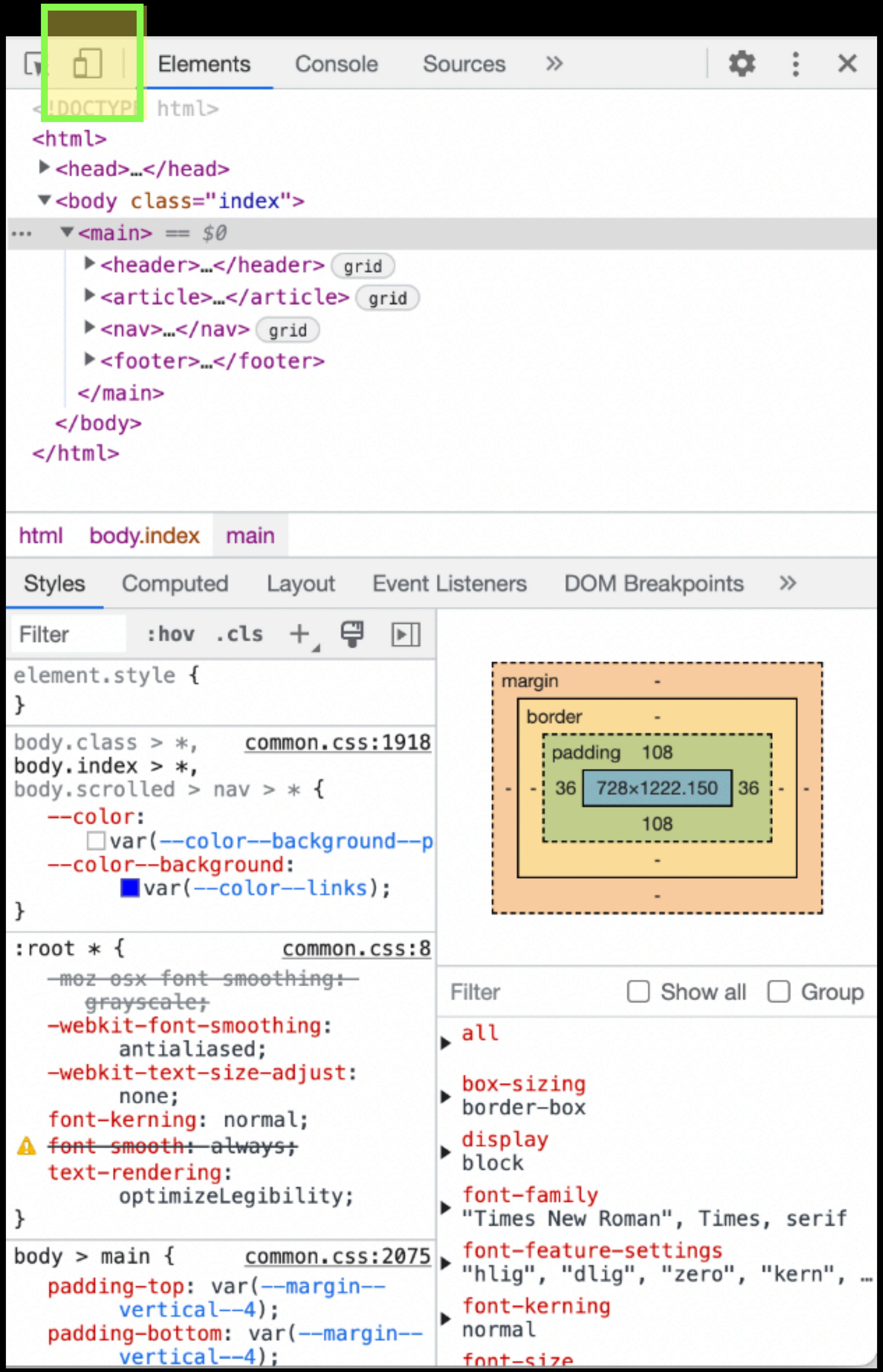
Lists the computed/
rendered values of styles

INSPECT

You can edit the code live in the inspector and see how things would change in the display.

INSPECT

device mode



```
<!DOCTYPE html>
<html>
 ▶<head>…</head>
 ▼<body class="index">
…  ▼<main> == $0
     ▶<header>…</header> grid
     ▶<article>…</article> grid
     ▶<nav>…</nav> grid
     ▶<footer>…</footer>
    </main>
  </body>
</html>
```

html body.index main

```
element.style {
}
body.class > *,               common.css:1918
body.index > *,
body.scrolled > nav > * {
   --color:
      □var(--color--background--p
   --color--background:
      ■var(--color--links);
}
:root * {                     common.css:8
   ~~-moz-osx-font-smoothing:
      grayscale;~~
   -webkit-font-smoothing:
         antialiased;
   -webkit-text-size-adjust:
         none;
   font-kerning: normal;
⚠  ~~font-smooth: always;~~
   text-rendering:
         optimizeLegibility;
}
body > main {                 common.css:2075
   padding-top: var(--margin--
         vertical--4);
   padding-bottom: var(--margin--
         vertical--4);
```

```
▶ all
▶ box-sizing
  border-box
▶ display
  block
▶ font-family
  "Times New Roman", Times, serif
▶ font-feature-settings
  "hlig", "dlig", "zero", "kern", …
▶ font-kerning
  normal
  font-size
```

Changes the aspect ratio of your display so you can see mobile (or device specific) renders of your code from web

GITHUB

Github is a platform for saving and sharing code, with a focus on **versioning**.

GITHUB

Essentially, it’s a way to track changes of code over time for yourself, while also sharing your work widely.

GITHUB

It works a bit like google drive, but instead of folders you have **repositories**, and each upload (or **commit**) does not overwrite the evidence of former files.

GITHUB

Let’s make GitHub accounts together.

GITHUB

Zach Scheinfeld’s Tutorials for 🔗 making and 🔗 publishing GitHub repositories

QUESTIONS?

BREAK (10 MIN)

EXERCISE 1

Take a website brought in by a classmate and make a quick wireframe. How are pages or elements linked—by what logic and mechanism?

EXERCISE 2

Take one of your previous three harmonic collection entries and build in some level of responsive design—what breakpoints or revisions in layout make sense for the work you've already done?

MIDTERMS

8/29

The Internet

9/05

HTML

9/12

CSS

9/19

Layouts

9/26

Web Hosting
+ Responsive
Design

10/3

Advanced
Layouts
+ CD LECTURE
@10h30

10/10

Even More
CSS

10/17

Midterms

MIDTERMS

1.

A **FULLY** **FUNCTIONAL** set of five harmonic collection entries, linked by a hub page, submitted either as a zip file or a GitHub link

MIDTERMS

2.

A **SHORT, 1-2 PAGE REFLECTION***
on your Harmonic Collection so far—
how has your theme evolved? How are you finding coding? How can you expand your entries, and what are some challenges you'd like to tackle next?

*** DOUBLE SPACED, 12PT… ~250-500 WDS**

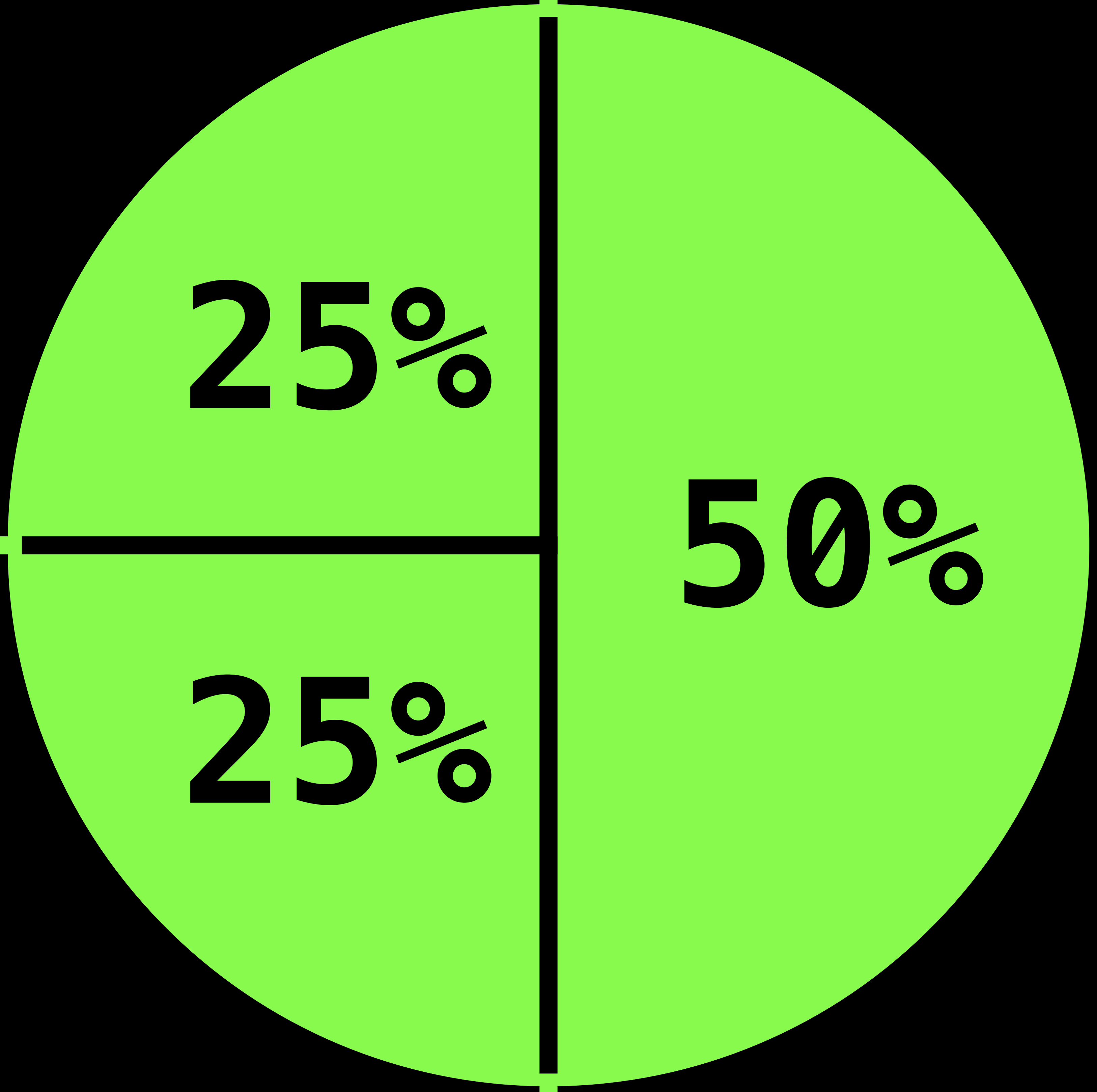
MIDTERMS

3.

A **BRIEF** presentation of your work to the class—for 2-5 minutes, walk the class through your progress live in your code or via a presentation; the remaining time (4-7 minutes) will be a short critique.

PARTICIPATION IS FOR A GRADE

MIDTERMS



WRITTEN RESPONSE

25%

IN-CLASS CRITIQUE

25%

50%

HARMONIC COLLECTION

MIDTERMS

A All required elements are **completely functional, exceptionally rendered. conceptually robust,** and **well-designed.**

B All required elements are **present, conceptually robust, considered, and graphically sound.**

C Work is mostly functional, and of average quality and consideration.

D Work is nonfunctional, unlinked, or otherwise flawed in a major way, but can be accessed.

F Work is not submitted, nonfunctional, unlinked, or otherwise flawed in a major way that prevents access and review.

MIDTERMS

ABSOLUTELY NO LATE WORK WILL BE ACCEPTED.

MIDTERMS

🔗 **SIGN UP FOR YOUR MIDTERM TIME SLOT AT YOUR EARLIEST CONVENIENCE**

HOMEWORK

FOR YOU

- X

FOR ME

- Harmonic Collection Entry 4
- Begin to think about how your entries will connect—bring in either some code or a sketch next week to discuss
- Upload your entries, in an appropriately structured format, to GitHub. You will continue to submit sketches through canvas, but it’s important to be able to access the entire network as well.

SESSION FIVE
SEPTEMBER 26, 2023

THANK YOU

DAVIS SCHERER
DSCHERER@NEWSCHOOL.EDU