

Be the browser's mentor, not its micromanager

All Day Hey - May 2022

Andy Bell - @hankchizljaw
<https://set.studio>

I'm going to change
how you write CSS

Fluid Type

Fluid Space

Flexible Layouts

Progressive Enhancement

We build for everyone
Not just for ourselves, or our peer groups

**Everyone should
get an excellent
user experience**

**No one will ever complain
about getting a good
baseline experience**

**Let's take a look at
what we are building**

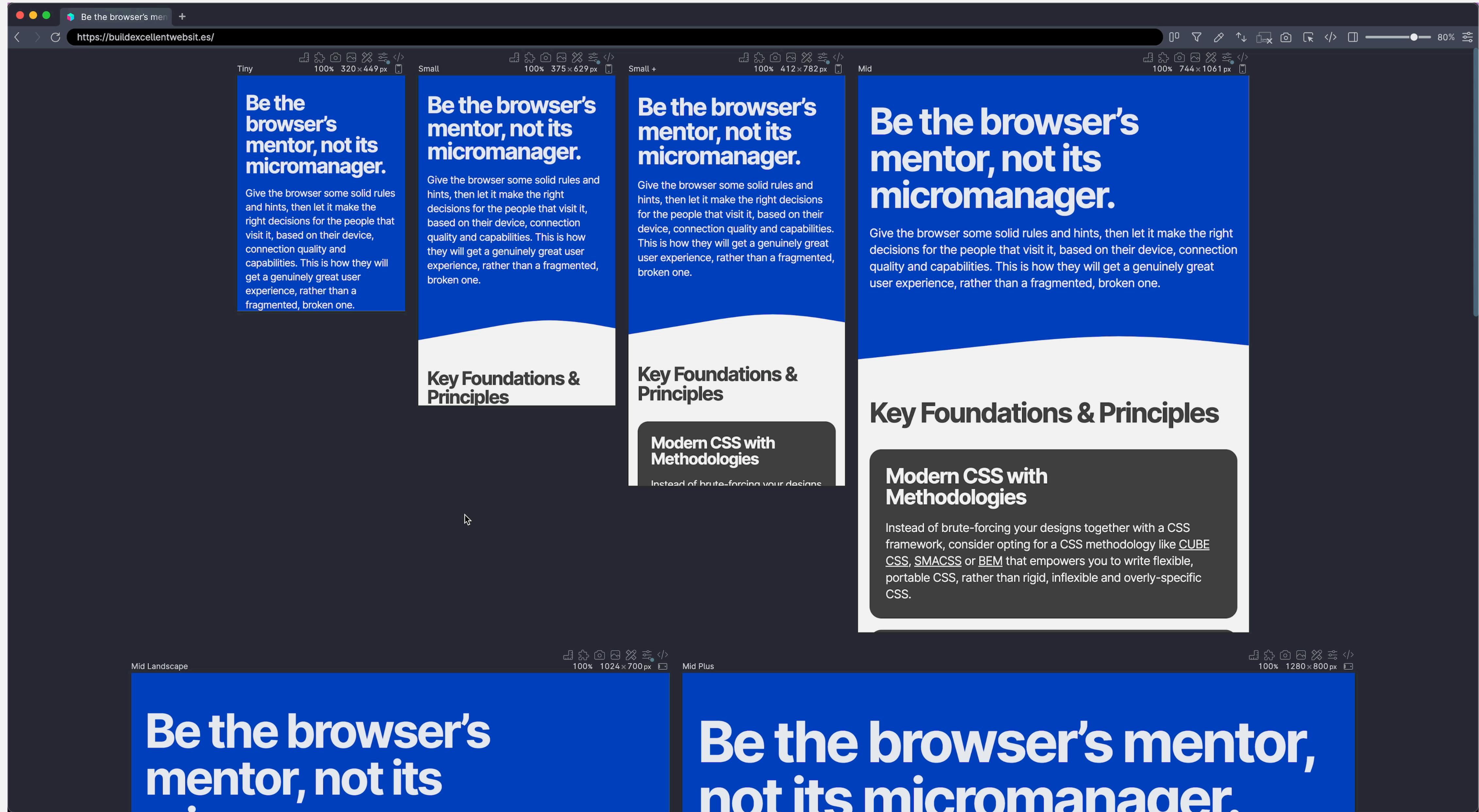


buildexcellentwebsit.es

#AIDayHey

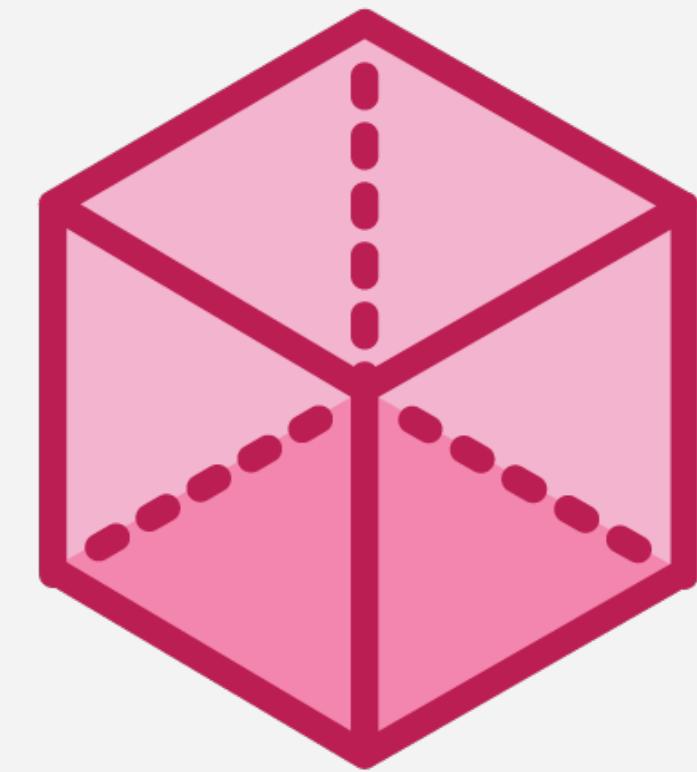


buildexcellentwebsites.es



**The “C” in CSS
stands for “Come
on, Andy, get on
with the good stuff”**

CUBE



CSS

**CUBE CSS is a CSS methodology
that's orientated towards simplicity,
pragmatism and consistency. It's
designed to work with the medium
that you're working in—often the
browser—rather than against it.**



cube.fyi



cube.fyi

Global CSS
Composition
Utilities
Blocks
Exceptions

Global CSS
Composition
Utilities
Blocks
Exceptions

**Global CSS
Composition
Utilities
Blocks
Exceptions**

**Global CSS
Composition
Utilities
Blocks
Exceptions**

**Global CSS
Composition
Utilities
Blocks
Exceptions**

**Home is where the
HTML is**

**It gives tools that help
others consume the
information on websites a
head start.**

**If the CSS doesn't load,
the website still makes
sense!**

**Old browsers will still
get a good baseline
experience**

**Get the HTML right and
you've built a solid
foundation. If not, you're
building on sand.**



```
1 <!DOCTYPE html>
2 <html lang="en">
3   <head> </head>
4   <body>
5     <main class="flow">
6       <header class="section spot-color-primary">...</header>
7       <article class="region flow">...</article>
8       <article class="region">...</article>
9       <article class="section spot-color-primary">...</article>
10      <article class="signoff region">...</article>
11    </main>
12  </body>
13 </html>
14
```

Why You Should Choose HTM x +

smashingmagazine.com

SMASHING MAGAZINE Articles Guides Books Events Membership Job Board More  Search articles...

Accessibility UX CSS JavaScript Performance Design Figma Wallpapers React Vue Round-Ups Web Design Guides Business Career Privacy

Bruce Lawson / JAN 7, 2020 / 27 comments

Why You Should Choose HTML5 `article` Over `section`

8 min read • CSS, HTML, Browsers Share on Twitter, LinkedIn

QUICK SUMMARY • Browsers' visual display of headings nested inside `<section>` elements makes it look as if they are assigning a logical hierarchy to those headings. However, this is purely visual and is not communicated to assistive technologies. What use is `<section>`, and how should authors mark up headings that are hugely important to AT users?

A few days ago, I was having a chat with some friends, one of whom asked me the difference between `<article>` and `<section>` in HTML. This is one of the eternal mysteries of web development, up there with “why is it white-space: nowrap, not white-space: no-wrap?” and “why is CSS ‘gray’ a darker color than ‘darkgray?’”.



ABOUT THE AUTHOR

Bruce has been working on accessibility, web standards, and browsers since 2001. That's why he looks that bad. You can follow him at [@brucel](#), or read his ... [More about Bruce](#) ↗

Email Newsletter 

<https://www.smashingmagazine.com/2020/01/html5-article-section/>

The screenshot shows a web browser window with the title bar "Be the browser's mentor, not its micromanager." and the URL "buildexcellentwebsites.es". The main content area contains the following text:

Be the browser's mentor, not its micromanager.

Give the browser some solid rules and hints, then let it make the right decisions for the people that visit it, based on their device, connection quality and capabilities. This is how they will get a genuinely great user experience, rather than a fragmented, broken one.

Key Foundations & Principles

- **Modern CSS with Methodologies**

Instead of brute-forcing your designs together with a CSS framework, consider opting for a CSS methodology like [CUBE CSS](#), [SMACSS](#) or [BEM](#) that empowers you to write flexible, portable CSS, rather than rigid, inflexible and overly-specific CSS.
- **Fluid type & Space**

Creating type scales that respond to the viewport, rather than setting explicit values for typography and space allows you to set rules once and forget about them, knowing that whatever device, regardless of its available size will be presented with appropriate sizes.
- **Flexible Layouts**

Using flexible, flexbox-based layouts, like the ones we provide in [Every Layout](#), ensures that regardless of conditions—be it content or available screen size: your front-end will be able to respond in the most appropriate way. Giving browsers hints and space to do what they do best, helps your front-end handle tricky scenarios where breakpoint-based layouts consistently fail.
- **Progressive Enhancement**

Building up with the lowest possible technological solution and enhancing it where device capability, connection speeds and context conditions allow, helps you build for everyone, not just the minority of people that have fast connections and powerful devices that work well, all the time.

Doing the opposite: building the best experience, then hacking it down for a handful of selected edge-cases means you're almost certainly going to build an experience that's excludes a lot of people.

Stick to those principles and making excellent websites that work for everyone suddenly becomes much, much easier.

Why though?

It was in 2010 when [Ethan Marcotte](#) published the legendary [Responsive Web Design](#) article. It completely changed how we built websites for an ever-growing variety of device types and sizes.

The article has aged really well, but the practice of web design has not. Oftentimes, designers and developers get stuck into pixel-pushing a design into shape with rigid methods to ensure it looks exactly like that Figma, Sketch or even Photoshop design. This attitude has stuck around for a long time though, even as far back as the very early days of the web, which [Jeremy Keith](#) touched on in [Resilient Web Design](#):

It was as though the web design community were participating in a shared consensual hallucination. Rather than acknowledge the flexible nature of the browser window, they chose to settle on one set width as the ideal ...even if that meant changing the ideal every few years.

Jeremy Keith - Resilient Web Design

We absolutely **don't know** what our audience device sizes are going to be or whether or not they have a decent connection speed, fully working browser or even a bright enough screen to present our designs how **we want them to be presented**. We are all guilty of micromanaging the browser in some aspects, and in turn, are creating an inflexible and fragile user experience.

A better way to approach this is to **be the browser's mentor** by setting some base rules and hints, then getting out of its way to let it make decisions based on the challenges it will **undoubtedly face**. Even looking at [this 2015 report on Android device sizes](#) tells us just how fragmented devices are. This was also conducted **7 years ago** and at the time of writing, it's **2022**. That's a long time for even more fragmentation to occur. Now factor in all of the other brands and types of device

Let's get stuck into some CSS Programming



<https://piccalil.li/blog/a-modern-css-reset/>

The screenshot shows a web browser window with the title bar "Be the browser's mentor, not its micromanager." and the URL "buildexcellentwebsites.es". The main content area contains the following text:

Be the browser's mentor, not its micromanager.

Give the browser some solid rules and hints, then let it make the right decisions for the people that visit it, based on their device, connection quality and capabilities. This is how they will get a genuinely great user experience, rather than a fragmented, broken one.

Key Foundations & Principles

- **Modern CSS with Methodologies**

Instead of brute-forcing your designs together with a CSS framework, consider opting for a CSS methodology like [CUBE CSS](#), [SMACSS](#) or [BEM](#) that empowers you to write flexible, portable CSS, rather than rigid, inflexible and overly-specific CSS.
- **Fluid type & Space**

Creating type scales that respond to the viewport, rather than setting explicit values for typography and space allows you to set rules once and forget about them, knowing that whatever device, regardless of its available size will be presented with appropriate sizes.
- **Flexible Layouts**

Using flexible, flexbox-based layouts, like the ones we provide in [Every Layout](#), ensures that regardless of conditions—be it content or available screen size: your front-end will be able to respond in the most appropriate way. Giving browsers hints and space to do what they do best, helps your front-end handle tricky scenarios where breakpoint-based layouts consistently fail.
- **Progressive Enhancement**

Building up with the lowest possible technological solution and enhancing it where device capability, connection speeds and context conditions allow, helps you build for everyone, not just the minority of people that have fast connections and powerful devices that work well, all the time.

Doing the opposite: building the best experience, then hacking it down for a handful of selected edge-cases means you're almost certainly going to build an experience that's excludes a lot of people.

Stick to those principles and making excellent websites that work for everyone suddenly becomes much, much easier.

Why though?

It was in 2010 when [Ethan Marcotte](#) published the legendary [Responsive Web Design](#) article. It completely changed how we built websites for an ever-growing variety of device types and sizes.

The article has aged really well, but the practice of web design has not. Oftentimes, designers and developers get stuck into pixel-pushing a design into shape with rigid methods to ensure it looks exactly like that Figma, Sketch or even Photoshop design. This attitude has stuck around for a long time though, even as far back as the very early days of the web, which [Jeremy Keith](#) touched on in [Resilient Web Design](#):

It was as though the web design community were participating in a shared consensual hallucination. Rather than acknowledge the flexible nature of the browser window, they chose to settle on one set width as the ideal ...even if that meant changing the ideal every few years.

Jeremy Keith - Resilient Web Design

We absolutely **don't know** what our audience device sizes are going to be or whether or not they have a decent connection speed, fully working browser or even a bright enough screen to present our designs how **we want them to be presented**. We are all guilty of micromanaging the browser in some aspects, and in turn, are creating an inflexible and fragile user experience.

A better way to approach this is to **be the browser's mentor** by setting some base rules and hints, then getting out of its way to let it make decisions based on the challenges it will **undoubtedly face**. Even looking at [this 2015 report on Android device sizes](#) tells us just how fragmented devices are. This was also conducted **7 years ago** and at the time of writing, it's **2022**. That's a long time for even more fragmentation to occur. Now factor in all of the other brands and types of device



```
1 *::before,  
2 *::after {  
3   box-sizing: border-box;  
4 }  
5  
6 body, h1, h2, h3, h4,  
7 p, figure, blockquote, dl, dd {  
8   margin: 0;  
9 }  
10  
11 ul[role='list'],  
12 ol[role='list'] {  
13   list-style: none;  
14 }  
15  
16 html {  
17   text-size-adjust: none;  
18   -webkit-text-size-adjust: none;  
19 }  
20  
21 html:focus-within {  
22   scroll-behavior: smooth;  
23 }  
24  
25 body {  
26   min-height: 100vh;  
27   text-rendering: optimizeSpeed;  
28   line-height: 1.5;  
29 }  
30  
31 a:not([class]) {  
32   text-decoration-skip-ink: auto;  
33 }  
34  
35 img, picture {  
36   max-width: 100%;  
37   display: block;  
38 }  
39  
40 input, button, textarea, select {  
41   font: inherit;  
42 }  
43
```

The screenshot shows a web browser window with the title bar "Be the browser's mentor, not its micromanager." The address bar says "localhost". The main content area displays the following text:

Be the browser's mentor, not its micromanager.

Give the browser some solid rules and hints, then let it make the right decisions for the people that visit it, based on their device, connection quality and capabilities. This is how they will get a genuinely great user experience, rather than a fragmented, broken one.

Key Foundations & Principles

- Modern CSS with Methodologies**
Instead of brute-forcing your designs together with a CSS framework, consider opting for a CSS methodology like [CUBE CSS](#), [SMACSS](#) or [BEM](#) that empowers you to write flexible, portable CSS, rather than rigid, inflexible and overly-specific CSS.
- Fluid type & Space**
Creating type scales that respond to the viewport, rather than setting explicit values for typography and space allows you to set rules once and forget about them, knowing that whatever device, regardless of its available size will be presented with appropriate sizes.
- Flexible Layouts**
Using flexible, flexbox-based layouts, like the ones we provide in [Every Layout](#), ensures that regardless of conditions—be it content or available screen size: your front-end will be able to respond in the most appropriate way. Giving browsers hints and space to do what they do best, helps your front-end handle tricky scenarios where breakpoint-based layouts consistently fail.
- Progressive Enhancement**
Building up with the lowest possible technological solution and enhancing it where device capability, connection speeds and context conditions allow, helps you build for everyone, not just the minority of people that have fast connections and powerful devices that work well, all the time.
Doing the opposite: building the best experience, then hacking it down for a handful of selected edge-cases means you're almost certainly going to build an experience that's excludes a lot of people.

Stick to those principles and making excellent websites that work for everyone suddenly becomes much, much easier.

Why though?

It was in 2010 when [Ethan Marcotte](#) published the legendary [Responsive Web Design](#) article. It completely changed how we built websites for an ever-growing variety of device types and sizes. The article has aged really well, but the practice of web design has not. Oftentimes, designers and developers get stuck into pixel-pushing a design into shape with rigid methods to ensure it looks exactly like that Figma, Sketch or even Photoshop design. This attitude has stuck around for a long time though, even as far back as the very early days of the web, which [Jeremy Keith](#) touched on in [Resilient Web Design](#):
It was as though the web design community were participating in a shared consensual hallucination. Rather than acknowledge the flexible nature of the browser window, they chose to settle on one set width as the ideal ...even if that meant changing the ideal every few years.

Jeremy Keith - Resilient Web Design
We absolutely **don't know** what our audience device sizes are going to be or whether or not they have a decent connection speed, fully working browser or even a bright enough screen to present our designs how **we want them to be presented**. We are all guilty of micromanaging the browser in some aspects, and in turn, are creating an inflexible and fragile user experience.

A better way to approach this is to **be the browser's mentor** by setting some base rules and hints, then getting out of its way to let it make decisions based on the challenges it will **undoubtedly face**. Even looking at [this 2015 report on Android device sizes](#) tells us just how fragmented devices are. This was also conducted **7 years ago** and at the time of writing, it's **2022**. That's a long time for even more fragmentation to occur. Now factor in all of the other brands and types of device and you can really see how problematic it is, building websites in a rigid, specific manner.

It makes sense to lose a bit of **perceived control** and instead, get even **greater control** by being the browser's mentor and not its micromanger, right?

Go ahead and open up this website on multiple devices or just resize the browser window. You'll see it deals with anything that is thrown at it. It also uses progressive enhancement to leverage modern CSS, while providing a solid, base experience for browsers that don't yet support those features, thanks to the usage of semantic HTML. All in, it's around 2kb of CSS **in total**.

Tools of the trade

Tools are just tools. They don't really matter—especially to the people trying to use the websites you build. The same goes for frameworks too. The most important thing is that you stick to the key principles. Even so, here are some useful tools I

Global CSS
Composition
Utilities
Blocks
Exceptions



```
1 :root {  
2   --color-primary: #0042bf;  
3   --color-primary-glare: #d8e2f4;  
4   --color-secondary: #ee5141;  
5   --color-secondary-glare: #ffe3e5;  
6   --space-s: clamp(1rem,0.92rem + 0.39vw,1.25rem);  
7   --space-m: clamp(1.5rem,1.38rem + 0.58vw,1.875rem);  
8   --space-l: clamp(2rem,1.84rem + 0.78vw,2.5rem);  
9   --size-step-1: clamp(1.1875rem,1.01rem + 0.87vw,1.75rem);  
10  --size-step-2: clamp(1.4375rem,1.11rem + 1.65vw,2.5rem);  
11  --size-step-3: clamp(1.75rem,1.19rem + 2.82vw,3.5625rem);  
12 }
```



```
1 :root {  
2   --color-primary: #0042bf;  
3   --color-primary-glare: #d8e2f4;  
4   --color-secondary: #ee5141;  
5   --color-secondary-glare: #ffe3e5;  
6   --space-s: clamp(1rem,0.92rem + 0.39vw,1.25rem);  
7   --space-m: clamp(1.5rem,1.38rem + 0.58vw,1.875rem);  
8   --space-l: clamp(2rem,1.84rem + 0.78vw,2.5rem);  
9   --size-step-1: clamp(1.1875rem,1.01rem + 0.87vw,1.75rem);  
10  --size-step-2: clamp(1.4375rem,1.11rem + 1.65vw,2.5rem);  
11  --size-step-3: clamp(1.75rem,1.19rem + 2.82vw,3.5625rem);  
12 }
```



```
1 :root {  
2   --gutter: var(--space-s-m);  
3   --border-radius: var(--size-step-1);  
4   --transition-base: 250ms ease;  
5   --transition-movement: 200ms linear;  
6   --transition-fade: 200ms ease;  
7   --transition-bounce: 500ms cubic-bezier(0.5, 0.05, 0.2, 1.5);  
8   --tracking: -0.05ch;  
9   --tracking-s: -0.075ch;  
10 }
```



```
1 body {  
2   color: var(--color-dark);  
3   background: var(--color-light);  
4   font-size: var(--size-step-1);  
5   font-family: var(--font-base);  
6   line-height: 1.4;  
7   letter-spacing: var(--tracking);  
8 }
```

The screenshot shows a web browser window with the title bar "Be the browser's mentor, not its micromanager." The address bar says "localhost". The main content area features a large, bold heading "Be the browser's mentor, not its micromanager." followed by a paragraph of text: "Give the browser some solid rules and hints, then let it make the right decisions for the people that visit it, based on their device, connection quality and capabilities. This is how they will get a genuinely great user experience, rather than a fragmented, broken one." Below this, there is a section titled "Key Foundations & Principles" with four subsections: "Modern CSS with Methodologies", "Fluid type & Space", "Flexible Layouts", and "Progressive Enhancement", each with a brief explanatory paragraph.

Be the browser's mentor, not its micromanager.

Give the browser some solid rules and hints, then let it make the right decisions for the people that visit it, based on their device, connection quality and capabilities. This is how they will get a genuinely great user experience, rather than a fragmented, broken one.

Key Foundations & Principles

Modern CSS with Methodologies

Instead of brute-forcing your designs together with a CSS framework, consider opting for a CSS methodology like [CUBE CSS](#), [SMACSS](#) or [BEM](#) that empowers you to write flexible, portable CSS, rather than rigid, inflexible and overly-specific CSS.

Fluid type & Space

Creating type scales that respond to the viewport, rather than setting explicit values for typography and space allows you to set rules once and forget about them, knowing that whatever device, regardless of its available size will be presented with appropriate sizes.

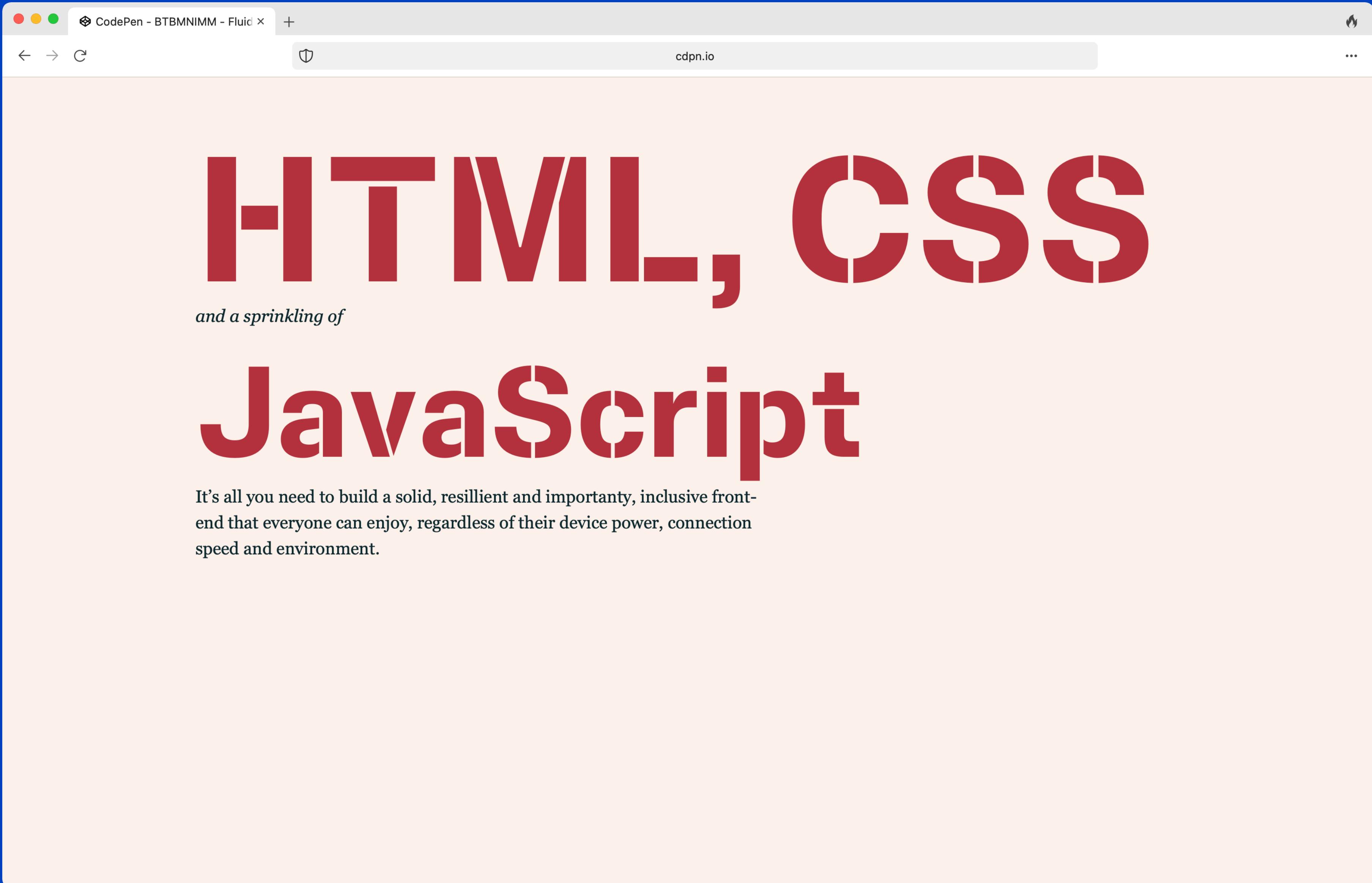
Flexible Layouts

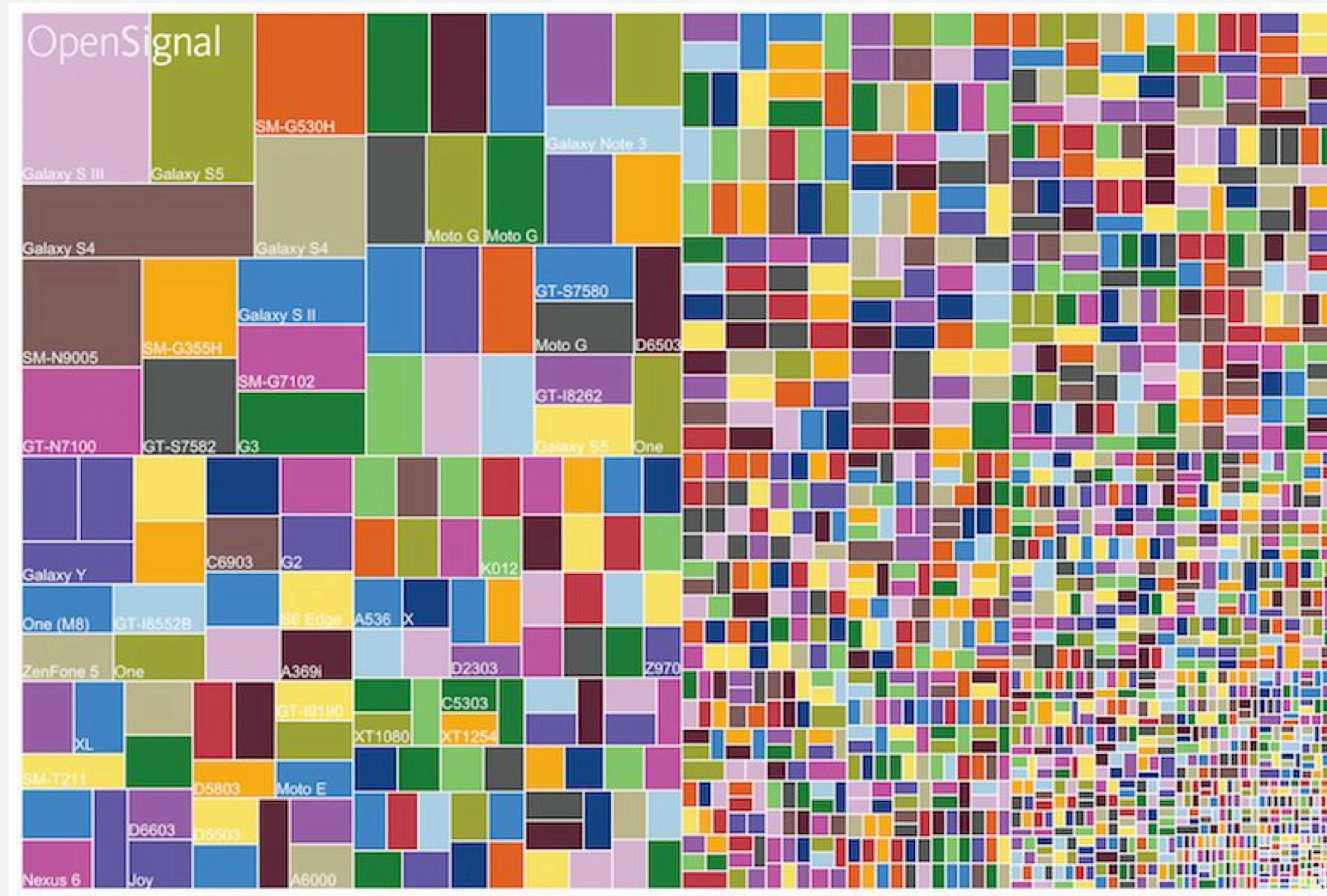
Using flexible, flexbox-based layouts, like the ones we provide in [Every Layout](#), ensures that regardless of conditions—be it content or available screen size: your front-end will be able to respond in the most appropriate way. Giving browsers hints and space to do what they do best, helps your front-end handle tricky scenarios where breakpoint-based layouts consistently fail.

Progressive Enhancement

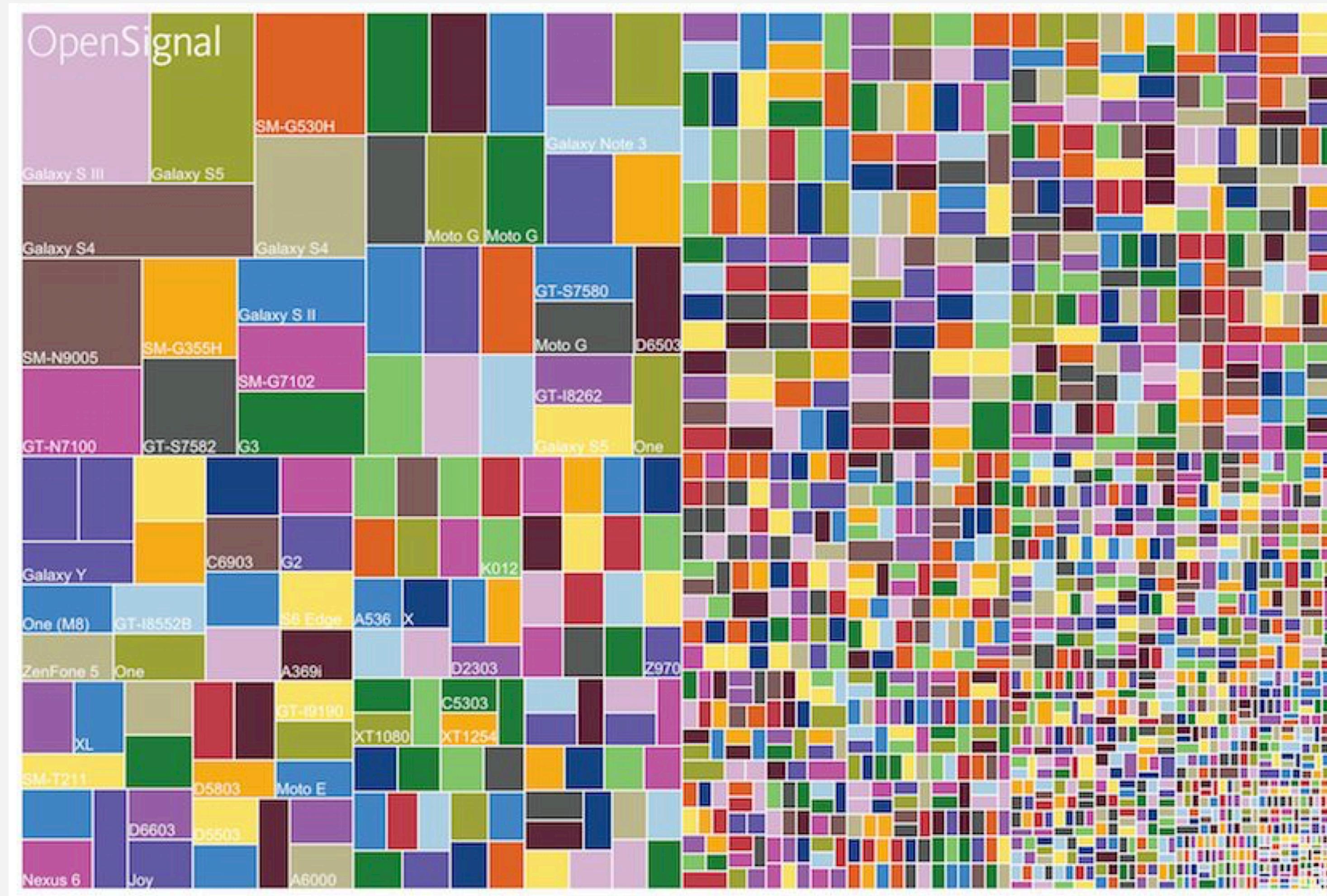
Building up with the lowest possible technological solution and enhancing it where device capability, connection speeds and context conditions allow, helps you build for everyone, not just the minority of people that have fast connections and powerful devices that work well, all the time.

Fluid type and fluid space





https://www.opensignal.com/sites/opensignal-com/files/data/reports/global/data-2015-08/2015_08_fragmentation_report.pdf



https://www.opensignal.com/sites/opensignal-com/files/data/reports/global/data-2015-08/2015_08_fragmentation_report.pdf

A screenshot of a web browser window displaying a static type example on codepen.io. The page title is "BTBMNIMM - Static type" by Piccalilli (PRO). The main content features large red text: "HTML, CSS" on top and "and a sprinkling of" followed by "JavaScript" below it. A descriptive paragraph states: "It's all you need to build a solid, resilient and importantly, inclusive front-end that everyone can enjoy, regardless of their device power, connection speed and environment." On the left side of the browser, there are three code editors: "HTML" showing a snippet of HTML with some parts visually hidden; "CSS" showing a font-face rule for "slussen_stencilbold"; and "JS" which is currently empty. At the bottom of the browser window, there are tabs for "Console", "Assets", "Comments", "⌘ Keys", and a Squarespace template selection bar. The browser has a dark theme with a light gray header.

BTBMNIMM - Static type

Piccalilli PRO + Follow

codepen.io

HTML

```
1 <main class="wrapper flow">
2   <h1 class="visually-hidden">HTML,
3     CSS and a sprinkling of JavaScript...
4   </h1>
5   <div aria-hidden="true"
6     class="header flow">
7     <div class="header__display-
8       text">HTML, CSS</div>
9     <div>and a sprinkling of</div>
10    <div class="header__display-
11      text">JavaScript</div>
12  </div>
```

CSS

```
1 @font-face {
2   font-family: "slussen_stencilbold";
3   src:
4     url("https://assets.codepen.io/174183
5 /slussen-stencil-bold-webfont.woff2")
6     format("woff2");
7   font-weight: normal;
8   font-style: normal;
9 }
```

JS

```
1
```

Squarespace: Choose from hundreds of Squarespace templates

Console Assets Comments ⌘ Keys

Fork Embed Export Share

```
1 .header__display-text {  
2   font-size: 2rem;  
3 }  
4  
5 .header__display-text:first-of-type {  
6   font-size: 2.3rem;  
7 }  
8  
9 @media (min-width: 768px) {  
10   .header__display-text {  
11     font-size: 6rem;  
12   }  
13  
14   .header__display-text:first-of-type {  
15     font-size: 8rem;  
16   }  
17 }  
18  
19 @media (min-width: 1100px) {  
20   .header__display-text {  
21     font-size: 8rem;  
22   }  
23  
24   .header__display-text:first-of-type {  
25     font-size: 10rem;  
26   }  
27 }
```



```
1 @media (min-width: 768px) {
2   .header__display-text {
3     font-size: 6rem;
4   }
5
6   .header__display-text:first-of-type {
7     font-size: 8rem;
8   }
9 }
10
11 @media (min-width: 856px) {
12   .header__display-text {
13     font-size: 7rem;
14   }
15
16   .header__display-text:first-of-type {
17     font-size: 9rem;
18   }
19 }
20
21 @media (min-width: 1100px) {
22   .header__display-text {
23     font-size: 8rem;
24   }
25
26   .header__display-text:first-of-type {
27     font-size: 10rem;
28   }
29 }
```

BTBMNIMM - Fluid type

codepen.io

BTBMNIMM - Fluid type

Piccalilli PRO + Follow

HTML

```
1 <main class="wrapper flow">
2   <h1 class="visually-hidden">HTML,
  CSS and a sprinkling of JavaScript...
</h1>
3   <div aria-hidden="true"
  class="header flow">
4     <div class="header__display-
  text">HTML, CSS</div>
5     <div>and a sprinkling of</div>
6     <div class="header__display-
  text">JavaScript</div>
```

CSS

```
1 @font-face {
2   font-family: "slussen_stencilbold";
3   src:
  url("https://assets.codepen.io/174183
  /slussen-stencil-bold-webfont.woff2")
    format("woff2");
5   font-weight: normal;
6   font-style: normal;
7 }
```

JS

```
1
```

Console Assets Comments ⌘ Keys Squarespace: Choose from hundreds of Squarespace templates Fork Embed Export Share

HTML, CSS and a sprinkling of JavaScript

It's all you need to build a solid, resilient and importantly, inclusive front-end that everyone can enjoy, regardless of their device power, connection speed and environment.

```
.my-element {  
  font-size: clamp(2rem, calc(1rem + 5vw), 10rem);  
}
```

```
.my-element {  
  font-size: clamp(2rem, calc(1rem + 5vw), 10rem);  
}
```

BTBMNIMM - Fluid type

codepen.io

BTBMNIMM - Fluid type

Piccalilli PRO + Follow

HTML

```
1 <main class="wrapper flow">
2   <h1 class="visually-hidden">HTML,
  CSS and a sprinkling of JavaScript...
</h1>
3   <div aria-hidden="true"
  class="header flow">
4     <div class="header__display-
  text">HTML, CSS</div>
5     <div>and a sprinkling of</div>
6     <div class="header__display-
  text">JavaScript</div>
```

CSS

```
1 @font-face {
2   font-family: "slussen_stencilbold";
3   src:
  url("https://assets.codepen.io/174183
  /slussen-stencil-bold-webfont.woff2")
    format("woff2");
5   font-weight: normal;
6   font-style: normal;
7 }
```

JS

```
1
```

Console Assets Comments ⌘ Keys Squarespace: Choose from hundreds of Squarespace templates Fork Embed Export Share

HTML, CSS
and a sprinkling of

JavaScript

It's all you need to build a solid, resilient and importantly, inclusive front-end that everyone can enjoy, regardless of their device power, connection speed and environment.

Size and space scales

Step 0: 1rem

Step 1: 1.25rem ($1 * 1.25$)

Step 2: 1.56rem ($1.25 * 1.25$)

Step 3: 1.95rem ($1.56 * 1.25$)

Step 4: 2.43rem ($1.95 * 1.25$)

The screenshot shows a web browser window titled "Fluid type scale calculator | Ut". The address bar displays "utopia.fyi". The main content area features the UTOPIA logo at the top left. Below it is a section titled "FLUID TYPE SCALE CALCULATOR". The interface is divided into two main sections: "MIN VIEWPORT" and "MAX VIEWPORT".

MIN VIEWPORT

Width	Font size	Type scale
320 <input type="text"/>	21 <input type="text"/>	1.2

Minor Third

MAX VIEWPORT

Width	Font size	Type scale
1140 <input type="text"/>	24 <input type="text"/>	1.25

Major Third

CALCULATED FONT SIZES

This table lists font size values in px for your type scales at the min and max viewport widths entered above.

Add a **viewport width** to show its corresponding font size values.

Add a **scale step** to extend your scale up or down.

Scale step	Viewport width
<input type="button" value="+"/>	320
	1140
	<input type="button" value="+"/>

The screenshot shows a web browser window titled "Fluid type scale calculator | Ut". The URL "utopia.fyi" is visible in the address bar. The page content is a fluid type scale calculator for UTOPIA. The header features the "UTOPIA." logo with "FLUID RESPONSIVE DESIGN" below it. A navigation menu icon is on the right.

FLUID TYPE SCALE CALCULATOR

MIN VIEWPORT

Width 320 px	Font size 21 px	Type scale 1.2
-----------------	--------------------	-------------------

Minor Third

MAX VIEWPORT

Width 1140 px	Font size 24 px	Type scale 1.25
------------------	--------------------	--------------------

Major Third

CALCULATED FONT SIZES

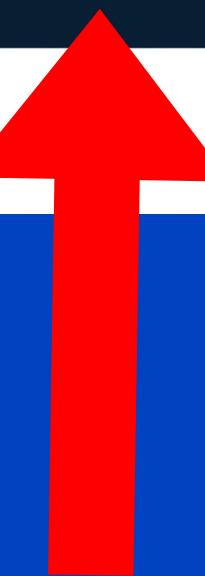
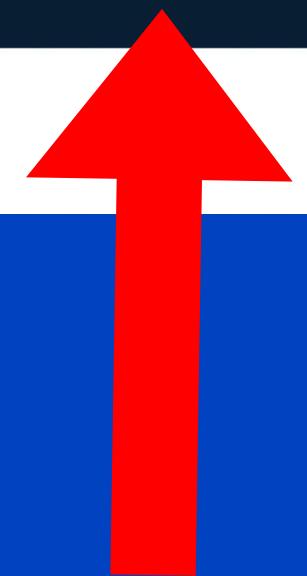
This table lists font size values in px for your type scales at the min and max viewport widths entered above.

Add a **viewport width** to show its corresponding font size values.

Add a **scale step** to extend your scale up or down.

Scale step	Viewport width
+	320
	1140
+	

MIN VIEWPORT			MAX VIEWPORT		
Width	Font size	Type scale	Width	Font size	Type scale
320 px	16 px	1.2	1350 px	20 px	1.414
Minor Third			Augmented Fourth		





```
1 :root {  
2   --step--1: clamp(0.83rem, calc(0.82rem + 0.08vw), 0.88rem);  
3   --step-0: clamp(1.00rem, calc(0.92rem + 0.39vw), 1.25rem);  
4   --step-1: clamp(1.20rem, calc(1.02rem + 0.88vw), 1.77rem);  
5   --step-2: clamp(1.44rem, calc(1.11rem + 1.65vw), 2.50rem);  
6   --step-3: clamp(1.73rem, calc(1.17rem + 2.80vw), 3.53rem);  
7   --step-4: clamp(2.07rem, calc(1.17rem + 4.54vw), 5.00rem);  
8   --step-5: clamp(2.49rem, calc(1.07rem + 7.11vw), 7.07rem);  
9   --step-6: clamp(2.99rem, calc(0.81rem + 10.88vw), 9.99rem);  
10 }
```

The screenshot shows a web browser window with the URL <https://utopia.fyi/type/calculator/?c=320,16,1.2,1350,20,1.414,8,1&s=0.75%7C0.5%7C0.25,1.5%7C2%7C3%7C4%7C6,s-l>. The main content area contains a vertical list of text elements:

- Step 8 (font size: 177.38px)
- Step 7 (font size: 133.61px)
- Step 6 (font size: 100.65px)
- Step 5 (font size: 75.81px)
- Step 4 (font size: 57.11px)
- Step 3 (font size: 43.02px)
- Step 2 (font size: 32.40px)
- Step 1 (font size: 24.41px)

The DevTools sidebar on the right shows the DOM structure and the CSS styles applied to the 'body' element. The 'body' element has a font size of 24.41px and a color of #var(--navy). It also includes vendor-specific properties for text-size-adjust and font-smoothing.

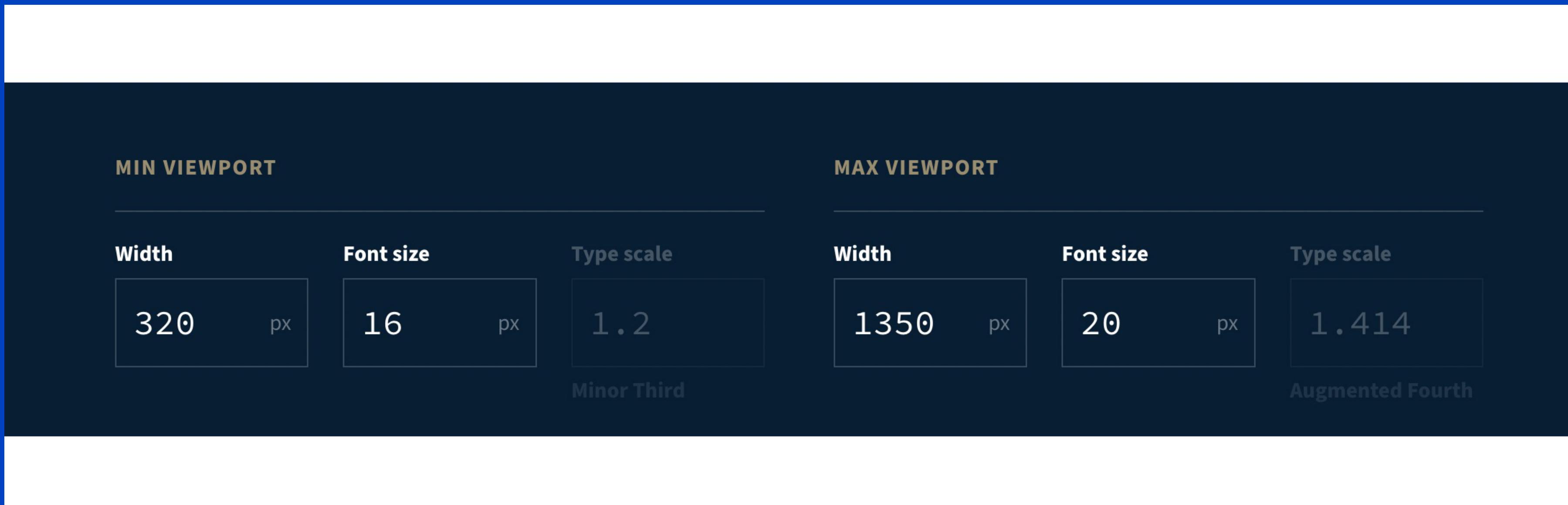
```
<!DOCTYPE html>
<html>
  <head>...</head>
  <body>
    <div id="__nuxt">
      <!-->
      <div id="__layout">
        <div class="layout">
          <header role="banner" class="top">...</header>
          <div class="row-spacing--major">
            <header class="wrap">...</header>
            <form class="calculator">...</form>
          </div>
        </div>
      </div>
    </div>
    ... html body div#__nuxt div#__layout div.layout div.row-spacing--major div.wrap.row-spacing--major ...

```

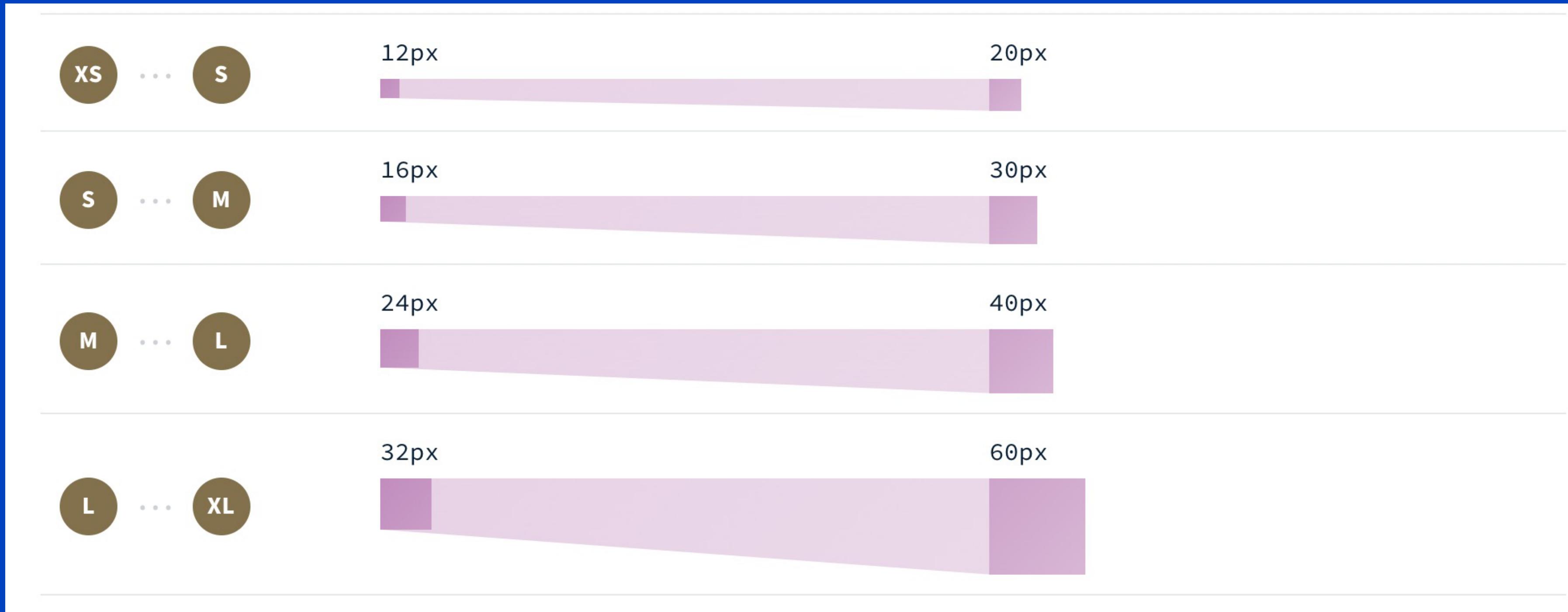
Styles Computed Layout Event Listeners DOM Breakpoints Properties Accessibility

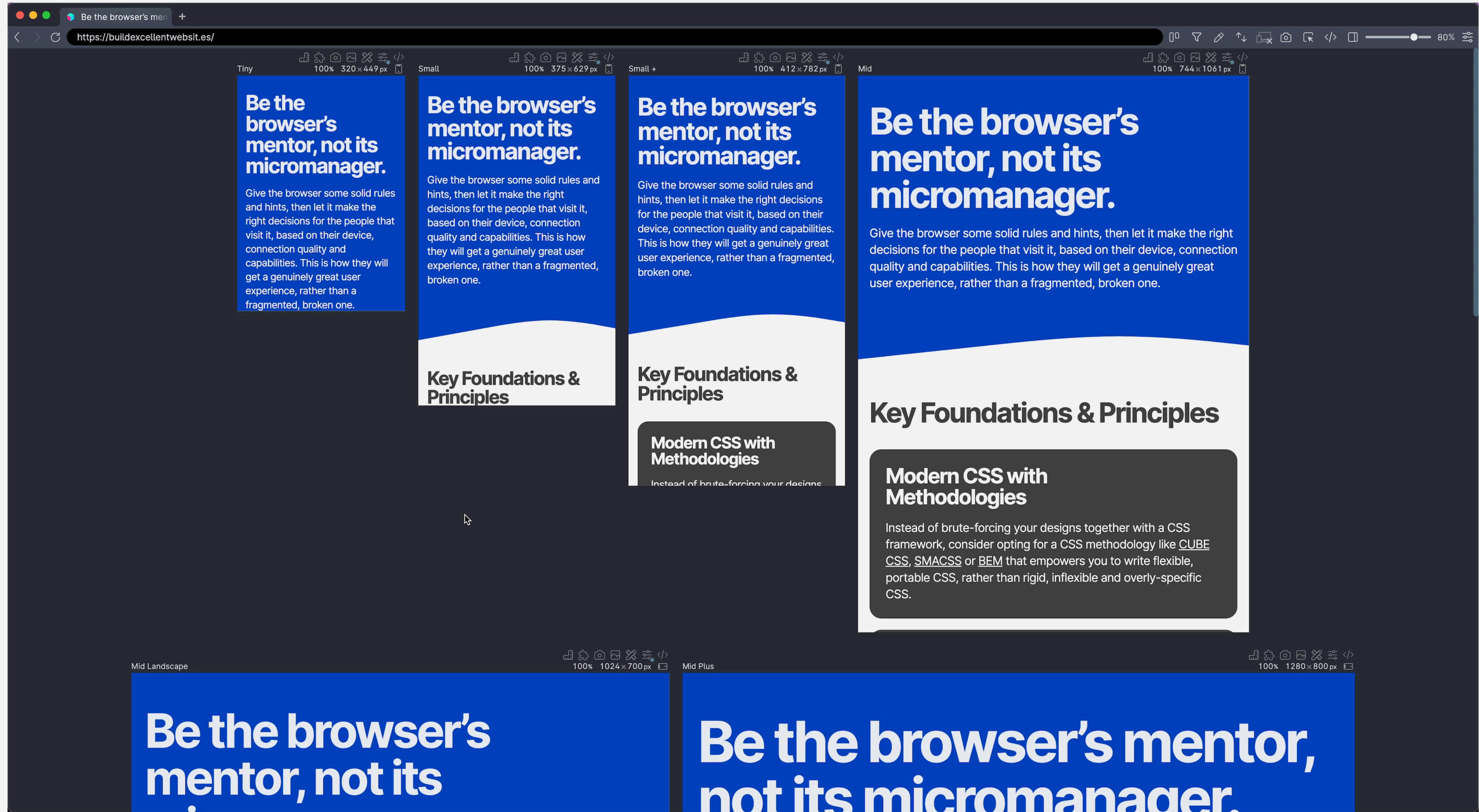
Filter :hov .cls +

```
element.style { }
.layout {
  display: flex;
  -webkit-box-orient: vertical;
  -webkit-box-direction: normal;
  flex-direction: column;
  -webkit-box-align: start;
  align-items: flex-start;
}
*, :after, :before {
  box-sizing: inherit;
}
div {
  display: block;
}
Inherited from div.layout
.layout {
  display: flex;
  -webkit-box-orient: vertical;
  -webkit-box-direction: normal;
  flex-direction: column;
  -webkit-transition: background-color .3s;
  transition: background-color .3s;
}
Inherited from body
body {
  font: var(--step-0)/1.5 var(--font-serif);
  -ms-text-size-adjust: 100%;
  -webkit-text-size-adjust: 100%;
  -moz-osx-font-smoothing: grayscale;
  -webkit-font-smoothing: antialiased;
  color: #var(--navy);
}
```



			Multiplier	@min	@max	
3XS			0.25	4px ■	5px ■	
2XS			0.5	8px ■	10px ■	
XS			0.75	12px ■	15px ■	
S			1	16px ■	20px ■	
M			1.5	24px ■	30px ■	
L			2	32px ■	40px ■	
XL			3	48px ■	60px ■	





Back to our CSS



```
1 body {  
2   color: var(--color-dark);  
3   background: var(--color-light);  
4   font-size: var(--size-step-1);  
5   font-family: var(--font-base);  
6   line-height: 1.4;  
7   letter-spacing: var(--tracking);  
8 }
```



```
1 h1, h2, h3 {  
2   line-height: 1;  
3   letter-spacing: var(--tracking-s);  
4 }  
5  
6 h1 {  
7   font-size: var(--size-step-5);  
8 }  
9  
10 h2 {  
11   font-size: var(--size-step-4);  
12 }  
13  
14 h3 {  
15   font-size: var(--size-step-3);  
16 }  
17
```



```
1 p, li,  
2 blockquote:not([class]) {  
3   max-width: 50ch;  
4 }  
5  
6 h1, h2, h3 {  
7   max-width: 20ch;  
8 }  
9
```

```
1 blockquote:not([class]) {  
2   font-family: var(--font-serif);  
3   font-size: var(--size-step-2);  
4 }  
5  
6 blockquote:not([class]) p:last-of-type {  
7   font-family: var(--font-base);  
8   font-size: var(--size-step-1);  
9   font-weight: normal;  
10 }  
11  
12 svg {  
13   height: 2ex;  
14   width: auto;  
15   flex: none;  
16 }  
17  
18 a {  
19   color: currentcolor;  
20 }  
21  
22 a:hover {  
23   text-decoration: none;  
24 }  
25  
26 :focus {  
27   outline: 2px solid;  
28   outline-offset: 0.3ch;  
29 }  
30  
31 :target {  
32   scroll-margin-top: 2ex;  
33 }
```

Be the browser's mentor, not its micromanager.

Give the browser some solid rules and hints, then let it make the right decisions for the people that visit it, based on their device, connection quality and capabilities. This is how they will get a genuinely great user experience, rather than a fragmented, broken one.

Key Foundations & Principles

Modern CSS with Methodologies

Instead of brute-forcing your designs together with a CSS framework, consider opting for a CSS methodology like [CUBE CSS](#), [SMACSS](#) or [BEM](#) that empowers you to write flexible, portable CSS, rather than rigid, inflexible and overly-specific CSS.

Fluid type & Space

Creating type scales that respond to the viewport, rather than setting explicit values for typography and space allows you to set rules once and forget about them, knowing that whatever device, regardless of

**Global CSS
Composition
Utilities
Blocks
Exceptions**



```
1 .flow > * + * {  
2   margin-top: var(--flow-space, 1em);  
3 }
```

Hello I am a heading

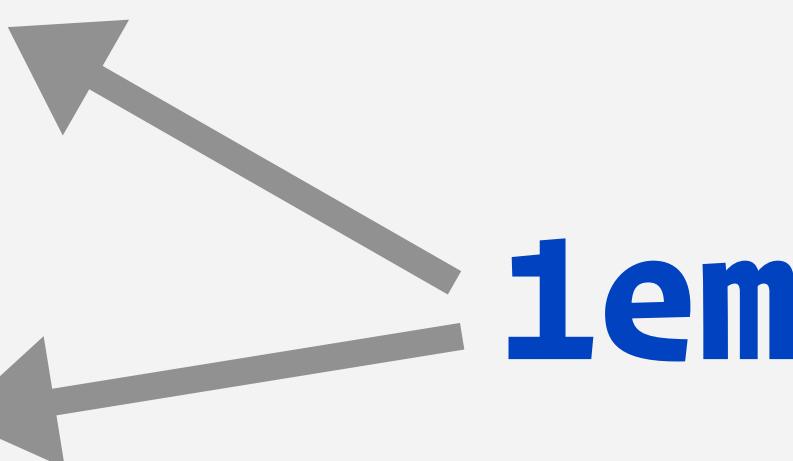
Sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Duis mollis, est non commodo luctus, nisi erat porttitor ligula, eget lacinia odio sem nec elit. Donec ullamcorper nulla non metus auctor fringilla.

Fusce dapibus, tellus ac cursus commodo, tortor mauris condimentum nibh, ut fermentum massa justo sit amet risus. Donec id elit non mi porta gravida at eget metus.

A subheading

Sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Duis mollis, est non commodo luctus, nisi erat porttitor ligula, eget lacinia odio sem nec elit. Donec ullamcorper nulla non metus auctor fringilla.

Fusce dapibus, tellus ac cursus commodo, tortor mauris condimentum nibh, ut fermentum massa justo sit amet risus. Donec id elit non mi porta gravida at eget metus.





```
1 .my-context {  
2   --flow-space: 10rem;  
3 }  
4  
5 .flow > * + * {  
6   margin-top: var(--flow-space, 1em);  
7 }
```

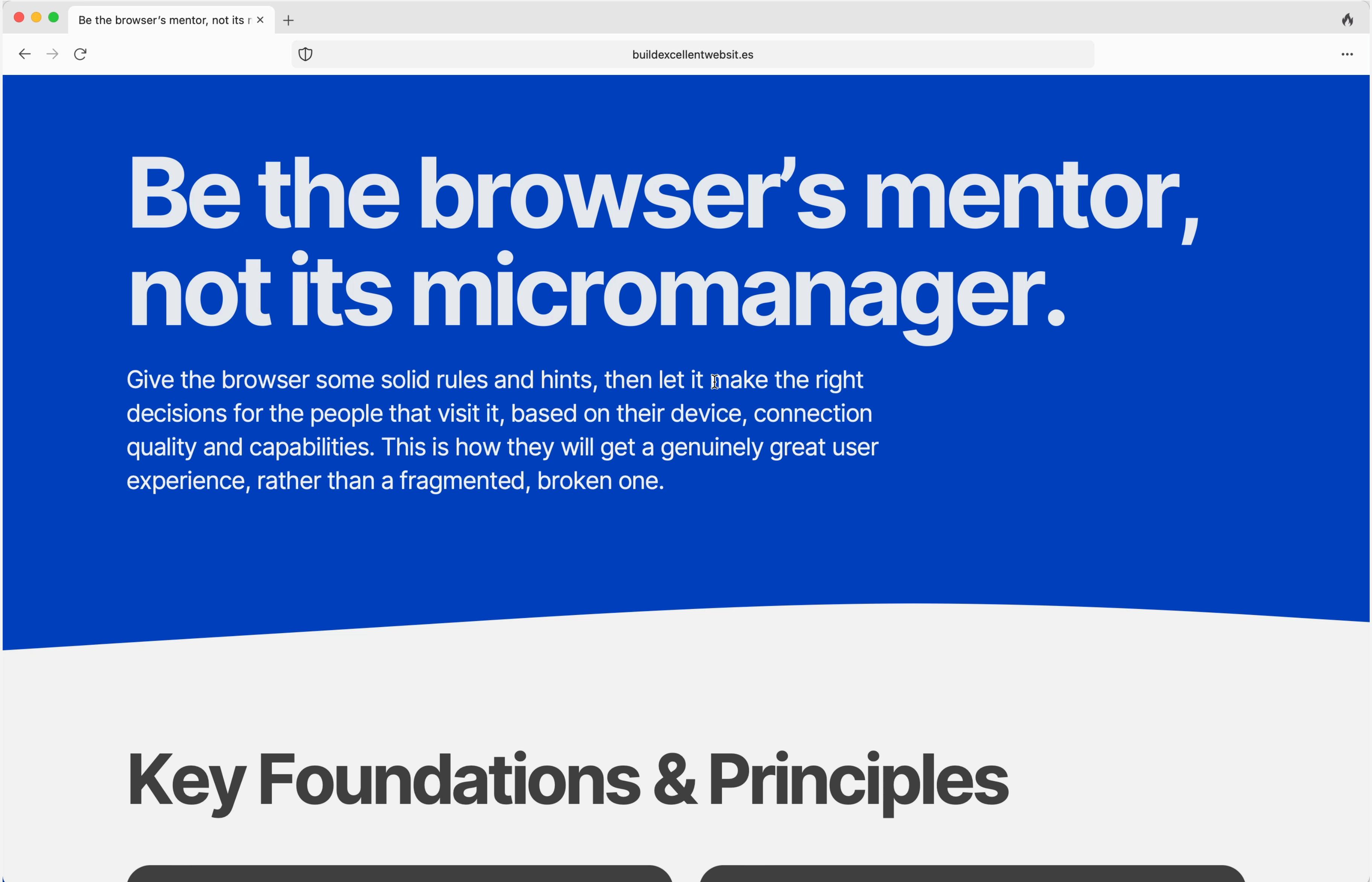
Hello I am a heading

Sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Duis mollis, est non commodo luctus, nisi erat porttitor ligula, eget lacinia odio sem nec elit. Donec ullamcorper nulla non metus auctor fringilla.

10rem

A subheading

Fusce dapibus, tellus ac cursus commodo, tortor mauris condimentum nibh, ut fermentum massa justo sit amet risus. Donec id elit non mi porta gravida at eget metus.



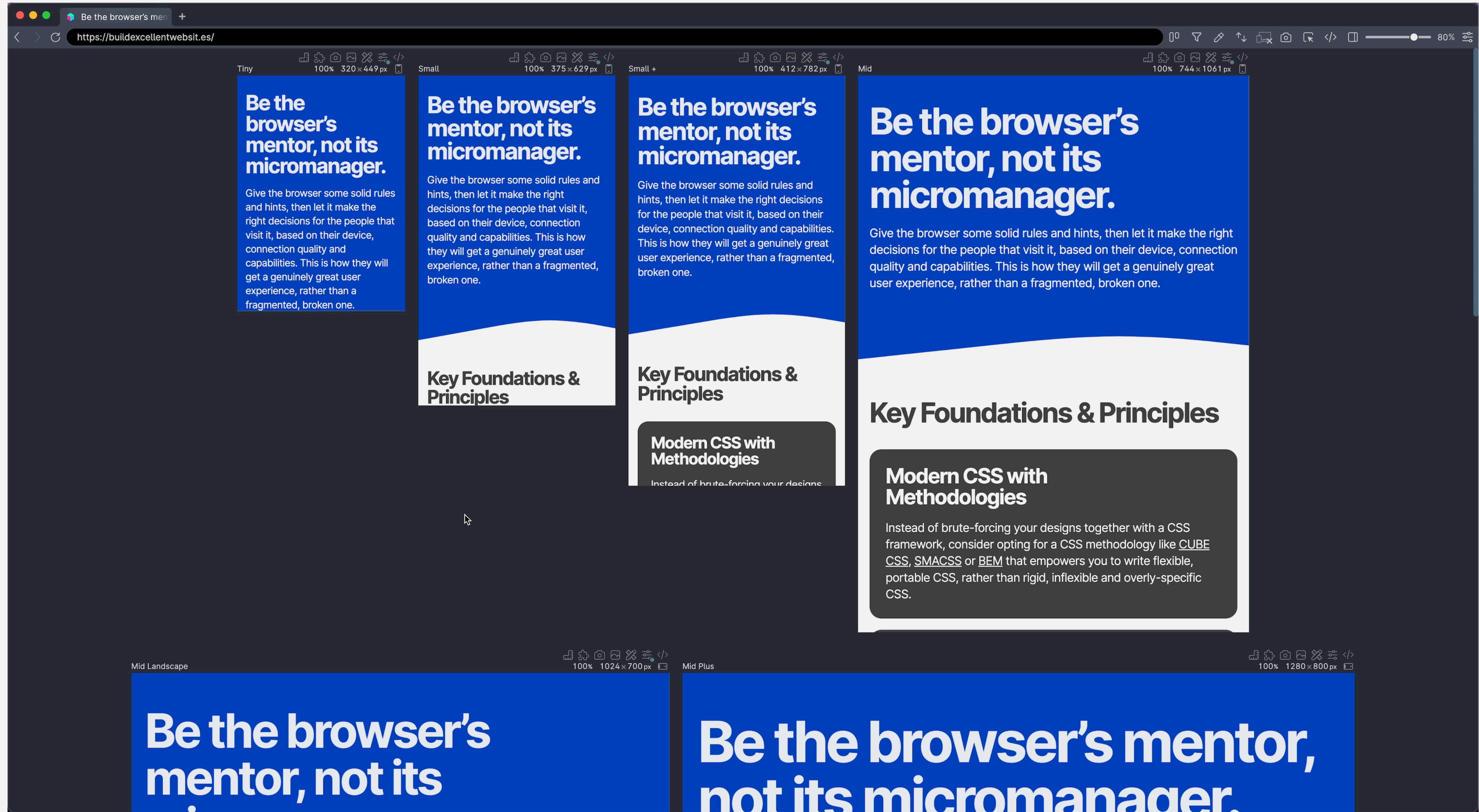
Be the browser's mentor, not its micromanager.

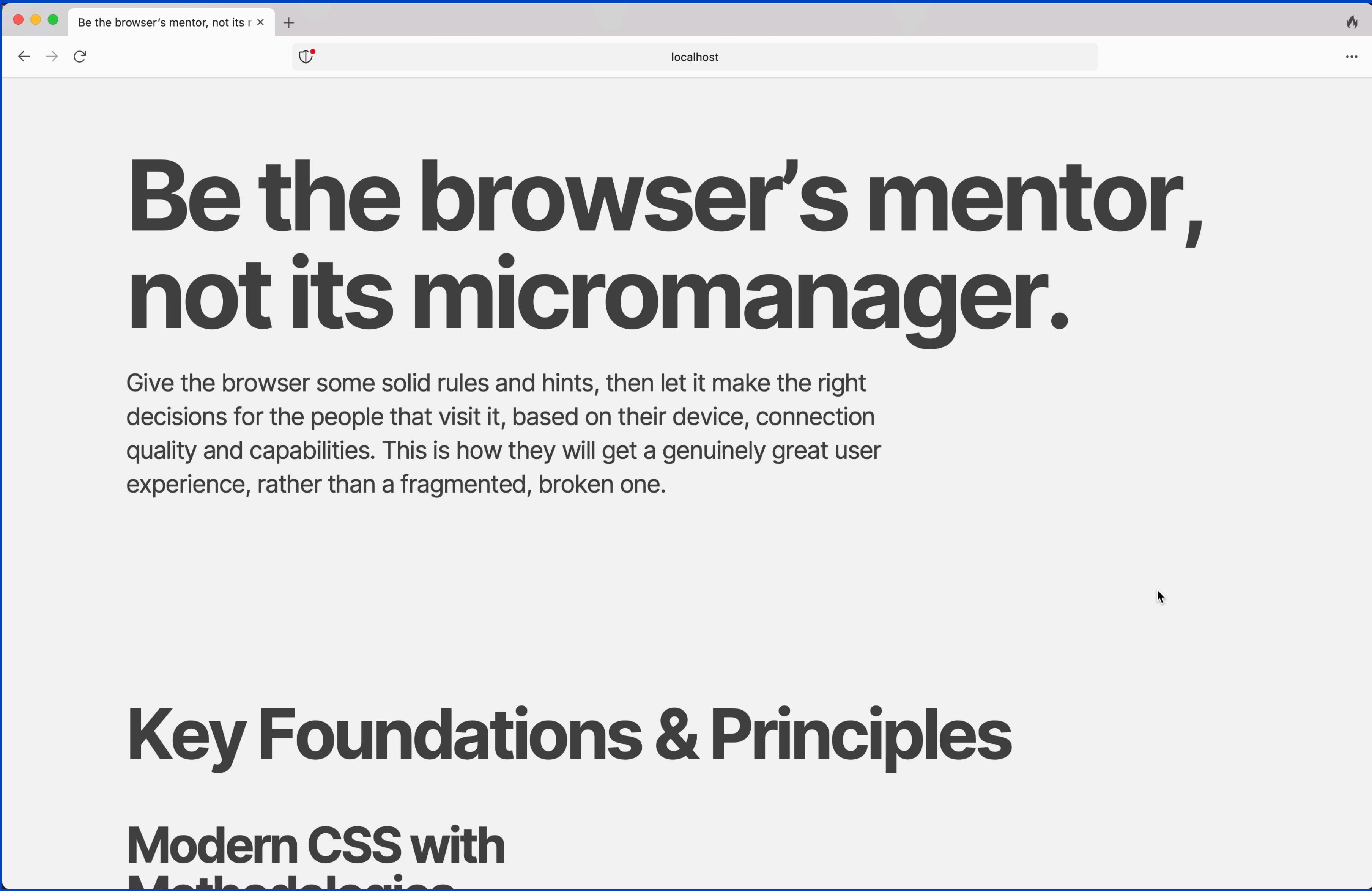
Give the browser some solid rules and hints, then let it make the right decisions for the people that visit it, based on their device, connection quality and capabilities. This is how they will get a genuinely great user experience, rather than a fragmented, broken one.

Key Foundations & Principles



```
1 .region {  
2   padding-top: var(--region-space, var(--space-l-2xl));  
3   padding-bottom: var(--region-space, var(--space-l-2xl));  
4 }
```





Global CSS
Composition
Utilities
Blocks
Exceptions

The screenshot shows a web browser window with a presentation slide. The title bar says "Be the browser's mentor, not its奴" and the address bar says "buildexcellentwebsit.es". The main content is a large heading "Key Foundations & Principles" followed by four dark gray callout boxes.

- Modern CSS with Methodologies**

Instead of brute-forcing your designs together with a CSS framework, consider opting for a CSS methodology like CUBE CSS, SMACSS or BEM that empowers you to write flexible, portable CSS, rather than rigid, inflexible and overly-specific CSS.
- Fluid type & Space**

Creating type scales that respond to the viewport, rather than setting explicit values for typography and space allows you to set rules once and forget about them, knowing that whatever device, regardless of its available size will be presented with appropriate sizes.
- Flexible Layouts**

Using flexible, flexbox-based layouts, like the ones we provide in Every Layout, ensures that regardless of conditions—be it content or available screen size: your front-
- Progressive Enhancement**

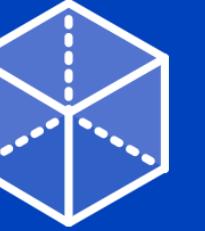
Building up with the lowest possible technological solution and enhancing it

Be the browser's mentor, not its奴 +

← → ⌂

buildexcellentwebsites.es

...



CUBE CSS

A CSS methodology that's orientated towards simplicity, pragmatism and consistency.



Utopia

A handy collection of generators and tools that let you build up various fluid type and space scales depending on viewport sizes to help with responsive design.



Every Layout

A series of simple, composable layouts that can be ported to any project. There's also heaps of learning material to help you *really* learn CSS layout.



Design Tokens

Centralised data—almost like a database of design values—that could be consumed by anything that understands a standard, like JSON to help with design consistency.



Polypane

A handy app that lets you spin up multiple viewports in various configurations to help you build truly responsive sites.



Tailwind

A utility-first CSS framework that is very useful for generating utility classes on demand for CUBE CSS.



```
1 .grid {  
2   display: grid;  
3   grid-template-columns: repeat(  
4     var(--grid-placement, auto-fill),  
5     minmax(var(--grid-min-item-size, 16rem), 1fr)  
6   );  
7   gap: var(--gutter, var(--space-s-l));  
8 }
```

The slide has a dark blue header bar with the title 'Be the browser's mentor, not its r' and a '+' icon. Below the header is a navigation bar with back, forward, and refresh buttons, followed by a URL bar containing 'localhost'. The main content area features a large, bold, dark gray title 'Key Foundations & Principles'. Below the title are three main sections, each with a bold title and a detailed description:

- . Modern CSS with Methodologies**

Instead of brute-forcing your designs together with a CSS framework, consider opting for a CSS methodology like CUBE CSS, SMACSS or BEM that empowers you to write flexible, portable CSS, rather than rigid, inflexible and overly-specific CSS.
- . Fluid type & Space**

Creating type scales that respond to the viewport, rather than setting explicit values for typography and space allows you to set rules once and forget about them, knowing that whatever device, regardless of its available size will be presented with appropriate sizes.
- . Flexible Layouts**

Using flexible, flexbox-based layouts, like the ones we provide in Every Layout, ensures that regardless of conditions—be it content or available screen size: your front-end will be able to respond in the most appropriate way. Giving browsers hints and space to do what they do best, helps your front-end handle tricky scenarios where
- . Progressive Enhancement**

Building up with the lowest possible technological solution and enhancing it where device capability, connection speeds and context conditions allow, helps you build for everyone, not just the minority of people that have fast connections and powerful devices that work well, all the time.

Doing the opposite:



```
1 <ul class="grid" role="list">
2   ...
3 </ul>
```



```
1 [role='list'] {  
2   padding: 0;  
3 }
```

Be the browser's mentor, not its奴隸

localhost

Key Foundations & Principles

Modern CSS with Methodologies

Instead of brute-forcing your designs together with a CSS framework, consider opting for a CSS methodology like [CUBE CSS](#), [SMACSS](#) or [BEM](#) that empowers you to write flexible, portable CSS, rather than rigid, inflexible and overly-specific CSS.

Fluid type & Space

Creating type scales that respond to the viewport, rather than setting explicit values for typography and space allows you to set rules once and forget about them, knowing that whatever device, regardless of its available size will be presented with appropriate sizes.

Flexible Layouts

Using flexible, flexbox-based layouts, like the ones we provide in [Every Layout](#), ensures that regardless of conditions—be it content or available screen size: your front-end will be able to respond in the most appropriate way. Giving browsers hints and space to do what they do best, helps your front-end handle tricky scenarios where

Progressive Enhancement

Building up with the lowest possible technological solution and enhancing it where device capability, connection speeds and context conditions allow, helps you build for everyone, not just the minority of people that have fast connections and powerful devices that work well, all the time.

Doing the opposite: [building the best](#)



```
1 .grid[data-layout='50-50'] {  
2   --grid-placement: auto-fit;  
3   --grid-min-item-size: clamp(16rem, 50vw, 26rem);  
4 }
```

Tiny 100% 320×449 px

Small 100% 375×629 px

Small + 100% 412×782 px

Mid 100% 744×1061 px

Mid Landscape 100% 1024×700 px

Mid Plus 100% 1280×800 px

Key Foundations & Principles

Modern CSS with Methodologies

Instead of brute-forcing your designs together with a CSS framework, consider opting for a CSS methodology like [CUBE CSS](#), [SMACSS](#) or [BEM](#) that empowers you to write flexible, portable CSS, rather than rigid, inflexible and overly-specific CSS.

Fluid type & Space

Creating type scales that respond to the viewport, rather than setting explicit values for typography and space allows you to set rules once and forget about them, knowing that whatever device, regardless of its

quality and capabilities. This is how they will get a genuinely great user experience, rather than a fragmented, broken one.

Key Foundations & Principles

Modern CSS with Methodologies

Instead of brute-forcing your designs together with a CSS framework, consider opting for a CSS methodology like [CUBE CSS](#), [SMACSS](#) or [BEM](#) that empowers you to write flexible, portable CSS, rather than rigid, inflexible and overly-specific CSS.

Fluid type & Space

Creating type scales that respond to the viewport, rather than setting explicit values for typography and space allows you to set rules once and forget about them, knowing that whatever device, regardless of its

Flexible Layouts

Using flexible, flexbox-based layouts, like the ones we provide in [Every Layout](#),

Key Foundations & Principles

Modern CSS with Methodologies

Instead of brute-forcing your designs together with a CSS framework, consider opting for a CSS methodology like [CUBE CSS](#), [SMACSS](#) or [BEM](#) that empowers you to write flexible, portable CSS, rather than rigid, inflexible and overly-specific CSS.

Fluid type & Space

Creating type scales that respond to the viewport, rather than setting explicit values for typography and space allows you to set rules once and forget about them, knowing that whatever device, regardless of its

Flexible Layouts

Using flexible, flexbox-based layouts, like the ones we provide in [Every Layout](#), ensures that regardless of conditions—be it content or available screen size: your front-end will be able to respond in the most appropriate way. Giving browsers hints and space to do what they do best, helps your front-end handle tricky scenarios where breakpoint-based layouts consistently fail.

Progressive Enhancement

quality and capabilities. This is how they will get a genuinely great user experience, rather than a fragmented, broken one.

Key Foundations & Principles

**Global CSS
Composition
Utilities
Blocks
Exceptions**

The screenshot shows a web browser window with a presentation slide. The title bar says "Be the browser's mentor, not its奴" and the address bar says "buildexcellentwebsit.es". The main content is a large heading "Key Foundations & Principles" followed by four dark gray callout boxes.

- Modern CSS with Methodologies**

Instead of brute-forcing your designs together with a CSS framework, consider opting for a CSS methodology like CUBE CSS, SMACSS or BEM that empowers you to write flexible, portable CSS, rather than rigid, inflexible and overly-specific CSS.
- Fluid type & Space**

Creating type scales that respond to the viewport, rather than setting explicit values for typography and space allows you to set rules once and forget about them, knowing that whatever device, regardless of its available size will be presented with appropriate sizes.
- Flexible Layouts**

Using flexible, flexbox-based layouts, like the ones we provide in Every Layout, ensures that regardless of conditions—be it content or available screen size: your front-
- Progressive Enhancement**

Building up with the lowest possible technological solution and enhancing it



```
1 .card {  
2   background: var(--color-dark);  
3   color: var(--color-light);  
4   padding: var(--space-m-l);  
5   border-radius: var(--border-radius);  
6   max-width: unset;  
7 }
```



```
1 p,  
2 li,  
3blockquote:not([class]) {  
4   max-width: 50ch;  
5 }
```



```
1 .card {  
2   background: var(--color-dark);  
3   color: var(--color-light);  
4   padding: var(--space-m-l);  
5   border-radius: var(--border-radius);  
6   max-width: unset;  
7 }
```

A screenshot of a web browser window titled "Be the browser's mentor, not its奴" (Be the browser's mentor, not its slave). The address bar shows "localhost". The page contains four dark gray rectangular cards with white text, arranged in a 2x2 grid.

- Modern CSS with Methodologies**

Instead of brute-forcing your designs together with a CSS framework, consider opting for a CSS methodology like [CUBE CSS](#), [SMACSS](#) or [BEM](#) that empowers you to write flexible, portable CSS, rather than rigid, inflexible and overly-specific CSS.
- Fluid type & Space**

Creating type scales that respond to the viewport, rather than setting explicit values for typography and space allows you to set rules once and forget about them, knowing that whatever device, regardless of its available size will be presented with appropriate sizes.
- Flexible Layouts**

Using flexible, flexbox-based layouts, like the ones we provide in [Every Layout](#), ensures that regardless of conditions—be it content or available screen size: your front-end will be able to respond in the most appropriate way. Giving browsers hints and space to do what they do best, helps your front-end handle tricky scenarios where breakpoint-based layouts consistently fail.
- Progressive Enhancement**

Building up with the lowest possible technological solution and enhancing it where device capability, connection speeds and context conditions allow, helps you build for everyone, not just the minority of people that have fast connections and powerful devices that work well, all the time.

Doing the opposite: building the best experience, then hacking it down for a handful of selected edge-cases means you're almost certainly going to build an experience that's excludes a lot of people.

A screenshot of a web browser window titled "Be the browser's mentor, not its奴" showing four dark gray callout boxes with white text on a light gray background. The browser has a standard OS X-style interface with red, yellow, and green window controls.

The browser address bar shows "localhost".

- Modern CSS with Methodologies**

Instead of brute-forcing your designs together with a CSS framework, consider opting for a CSS methodology like [CUBE CSS](#), [SMACSS](#) or [BEM](#) that empowers you to write flexible, portable CSS, rather than rigid, inflexible and overly-specific CSS.
- Fluid type & Space**

Creating type scales that respond to the viewport, rather than setting explicit values for typography and space allows you to set rules once and forget about them, knowing that whatever device, regardless of its available size will be presented with appropriate sizes.
- Flexible Layouts**

Using flexible, flexbox-based layouts, like the ones we provide in [Every Layout](#), ensures that regardless of conditions—be it content or available screen size: your front-end will be able to respond in the most appropriate way. Giving browsers hints and space to do what they do best, helps your front-end handle tricky scenarios where breakpoint-based layouts consistently fail.
- Progressive Enhancement**

Building up with the lowest possible technological solution and enhancing it where device capability, connection speeds and context conditions allow, helps you build for everyone, not just the minority of people that have fast connections and powerful devices that work well, all the time.

Doing the opposite: building the best experience, then hacking it down for a handful of selected edge-cases means you're almost certainly going to build an experience that's excludes a lot of people.



```
1 <ul class="grid" role="list" data-rows="masonry" data-layout="50-50">
2   ...
3 </ul>
```



```
1 .grid[data-rows='masonry'] {  
2   grid-template-rows: masonry;  
3   align-items: start;  
4 }
```

Be the browser's mentor, not its奴 ×

localhost

Modern CSS with Methodologies

Instead of brute-forcing your designs together with a CSS framework, consider opting for a CSS methodology like [CUBE CSS](#), [SMACSS](#) or [BEM](#) that empowers you to write flexible, portable CSS, rather than rigid, inflexible and overly-specific CSS.

Fluid type & Space

Creating type scales that respond to the viewport, rather than setting explicit values for typography and space allows you to set rules once and forget about them, knowing that whatever device, regardless of its available size will be presented with appropriate sizes.

Flexible Layouts

Using flexible, flexbox-based layouts, like the ones we provide in [Every Layout](#), ensures that regardless of conditions—be it content or available screen size: your front-end will be able to respond in the most appropriate way. Giving browsers hints and space to do what they do best, helps your front-end handle tricky scenarios where breakpoint-based layouts consistently fail.

Progressive Enhancement

Building up with the lowest possible technological solution and enhancing it where device capability, connection speeds and context conditions allow, helps you build for everyone, not just the minority of people that have fast connections and powerful devices that work well, all the time.

Doing the opposite: building the best experience, then hacking it down for a handful of selected edge-cases means you're almost certainly going to build an experience that's excludes a lot of people.

A screenshot of a web browser window titled "Be the browser's mentor, not its slave" with the URL "localhost:8080". The browser interface includes standard controls like back/forward, search, and a zoom level of 67%. The main content area features four dark gray rounded rectangular boxes, each containing a title and a descriptive paragraph.

- Modern CSS with Methodologies**

Instead of brute-forcing your designs together with a CSS framework, consider opting for a CSS methodology like [CUBE CSS](#), [SMACSS](#) or [BEM](#) that empowers you to write flexible, portable CSS, rather than rigid, inflexible and overly-specific CSS.
- Fluid type & Space**

Creating type scales that respond to the viewport, rather than setting explicit values for typography and space allows you to set rules once and forget about them, knowing that whatever device, regardless of its available size will be presented with appropriate sizes.
- Progressive Enhancement**

Building up with the lowest possible technological solution and enhancing it where device capability, connection speeds and context conditions allow, helps you build for everyone, not just the minority of people that have fast connections and powerful devices that work well, all the time.

Doing the opposite: building the best experience, then hacking it down for a handful of selected edge-cases means you're almost certainly going to build an experience that's excludes a lot of people.
- Flexible Layouts**

Using flexible, flexbox-based layouts, like the ones we provide in [Every Layout](#), ensures that regardless of conditions—be it content or available screen size: your front-end will be able to respond in the most appropriate way. Giving browsers hints and space to do what they do best, helps your front-end handle tricky scenarios where breakpoint-based layouts consistently fail.

Be the browser's mentor, not its奴 +

← → ⌂

buildexcellentwebsites.es

...



CUBE CSS

A CSS methodology that's orientated towards simplicity, pragmatism and consistency.



Utopia

A handy collection of generators and tools that let you build up various fluid type and space scales depending on viewport sizes to help with responsive design.



Every Layout

A series of simple, composable layouts that can be ported to any project. There's also heaps of learning material to help you *really* learn CSS layout.



Design Tokens

Centralised data—almost like a database of design values—that could be consumed by anything that understands a standard, like JSON to help with design consistency.



Polypane

A handy app that lets you spin up multiple viewports in various configurations to help you build truly responsive sites.



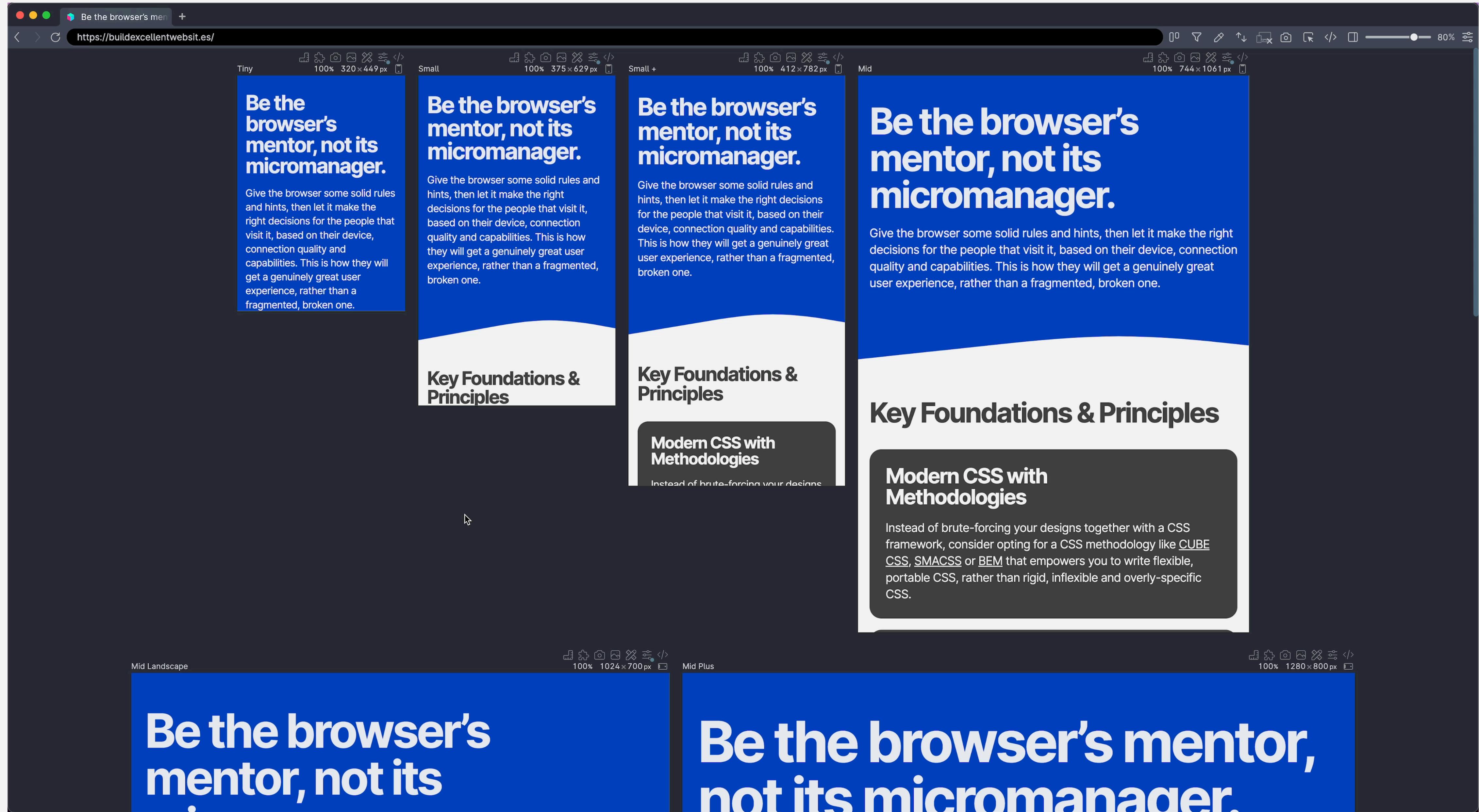
Tailwind

A utility-first CSS framework that is very useful for generating utility classes on demand for CUBE CSS.



```
1 .features {  
2   --grid-placement: auto-fit;  
3   --grid-min-item-size: clamp(16rem, 33%, 20rem);  
4   --gutter: var(--space-l-xl);  
5   --flow-space: var(--space-s);  
6  
7   text-align: center;  
8 }
```

```
1 .features svg {  
2   display: block;  
3   margin-inline: auto;  
4   height: 4em;  
5 }  
6  
7 .features a {  
8   text-decoration: none;  
9 }  
10  
11 .features a:hover {  
12   text-decoration: underline;  
13   text-decoration-thickness: 0.08ex;  
14   text-underline-offset: 0.2ex;  
15 }
```





<https://glitch.com/edit/#!/build-excellent-websites>

We've never had it better
with browsers

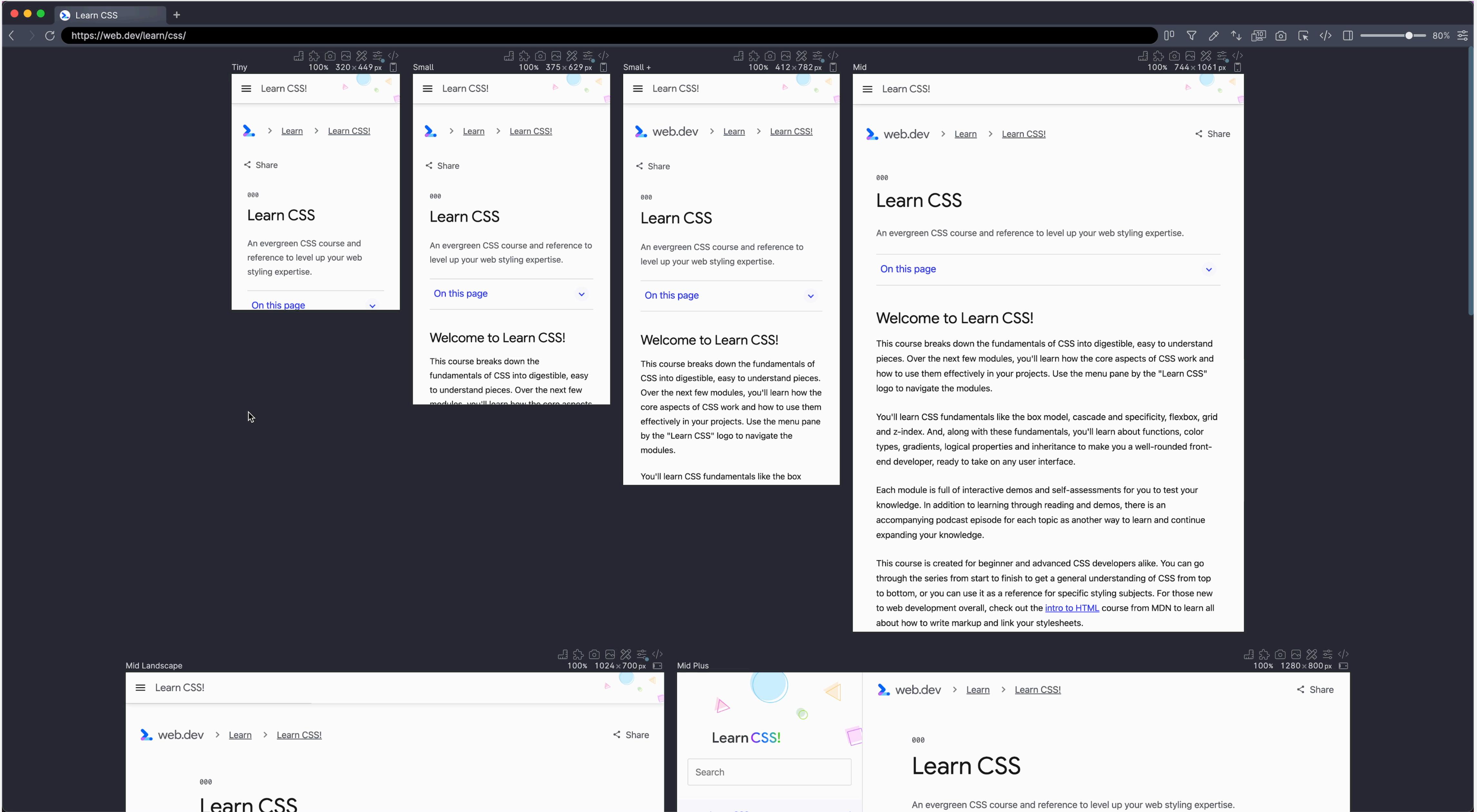


**Build the whole website with
Tailwind**



**Use Tailwind as a utility
generator and lean into CSS**

A couple of examples of this approach in the wild



⚠ CONTENT WARNING ⚠

Glitching and flashing

A Human Future // Home

https://ahumanfuture.co

Tiny 100% 320×449 px

Small 100% 375×629 px

Small + 100% 412×782 px

Mid 100% 744×1061 px

Mid Landscape 100% 1024×700 px

Mid Plus 100% 1280×800 px

A Human Future 2022 Play Audio

Menu [3]

A Human Future is a software development house.

[Welcome]

We build **products**, **platforms** and **prototypes** that help ambitious organisations pull their future towards them, faster.

CAMBRIDGE & LONDON [2]



A Human Future 2022 Play Audio

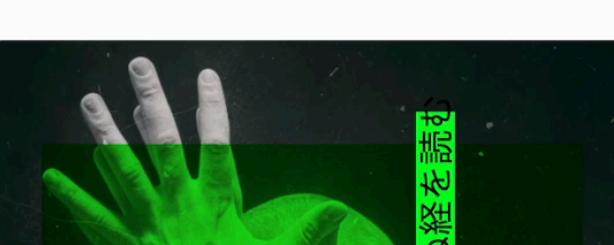
Menu [3]

A Human Future is a software development house.

[Welcome]

We build **products**, **platforms** and **prototypes** that help ambitious organisations pull their future towards them, faster.

CAMBRIDGE & LONDON [2]



A Human Future 2022 Play Audio

Menu [3]

A Human Future is a software development house.

[Welcome]

We build **products**, **platforms** and **prototypes** that help ambitious organisations pull their future towards them, faster.

CAMBRIDGE & LONDON [2]



A Human Future 2022 Play Audio

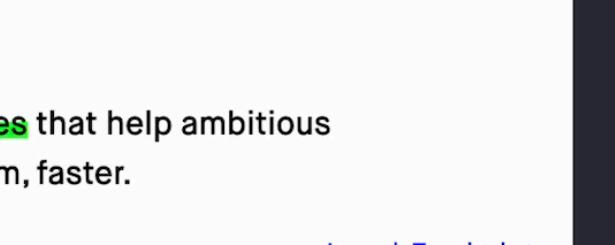
Menu [3]

A Human Future is a software development house.

[Welcome]

We build **products**, **platforms** and **prototypes** that help ambitious organisations pull their future towards them, faster.

CAMBRIDGE & LONDON [2]



A Human Future 2022 Play Audio

Menu [3]

A Human Future is a software development house.

[Welcome]

We build **products**, **platforms** and **prototypes** that help ambitious organisations pull their future towards them, faster.

CAMBRIDGE & LONDON [2]

[Launch Terminal >](#)

A Human Future 2022 Play Audio

Menu [3]

A Human Future is a software development house.

[Welcome]

We build **products**, **platforms** and **prototypes** that help ambitious organisations pull their future towards them, faster.

CAMBRIDGE & LONDON [2]



A Human Future 2022 Play Audio

Menu [3]

A Human Future is a software development house.

[Welcome]

We build **products**, **platforms** and **prototypes** that help ambitious organisations pull their future towards them, faster.

CAMBRIDGE & LONDON [2]



A Human Future 2022 Play Audio

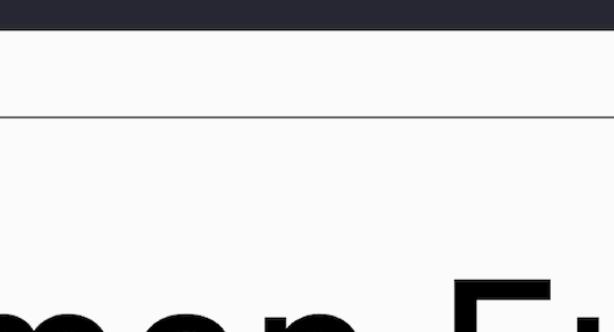
Menu [3]

A Human Future is a software development house.

[Welcome]

We build **products**, **platforms** and **prototypes** that help ambitious organisations pull their future towards them, faster.

CAMBRIDGE & LONDON [2]



**Go forth and build
excellent websites**

Thank you

Andy Bell - @hankchizljaw
<https://set.studio> - @setstudiotweets

<https://swop.link/set-studio>