Be the browser's mentor, not its micromanager

WDC 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

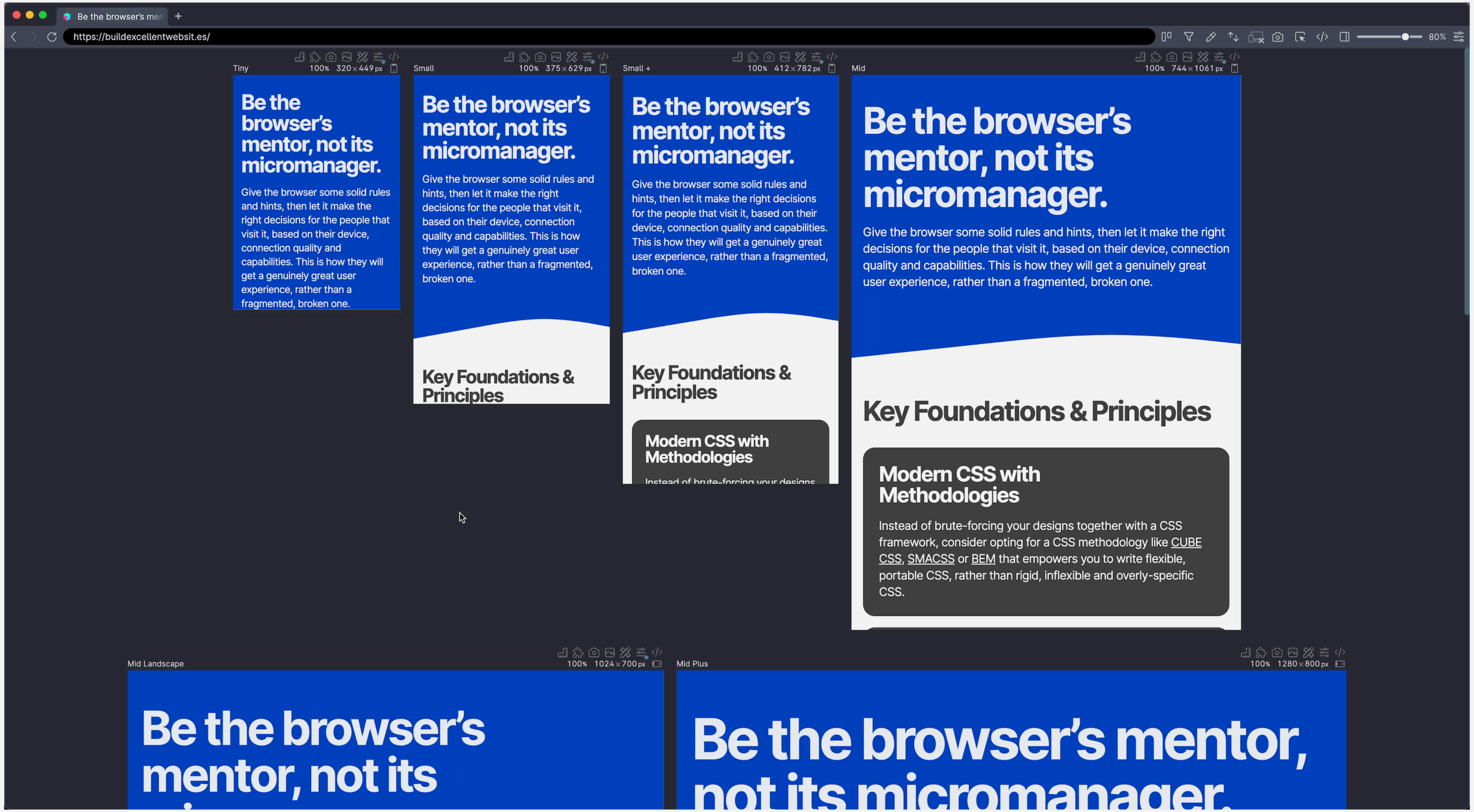
Let’s take a look at what we are building

buildexcellentwebsit.es

#WebDevConf

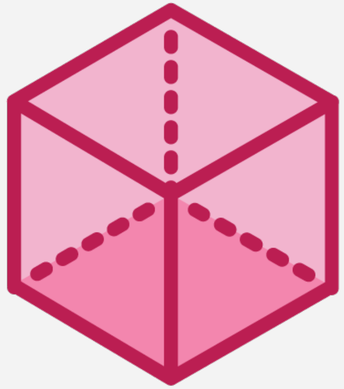


buildexcellentwebsit.es


Be the browser's men
https://buildexcellentwebsit.es/
80%
Tiny
100% 320 × 449 px
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.
Small
100% 375 × 629 px
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
Small +
100% 412 × 782 px
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
Mid
100% 744 × 1061 px
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.
Mid Landscape
100% 1024 × 700 px
Be the browser's mentor, not its
Mid Plus
100% 1280 × 800 px
Be the browser's mentor, not its micromanager.

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

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

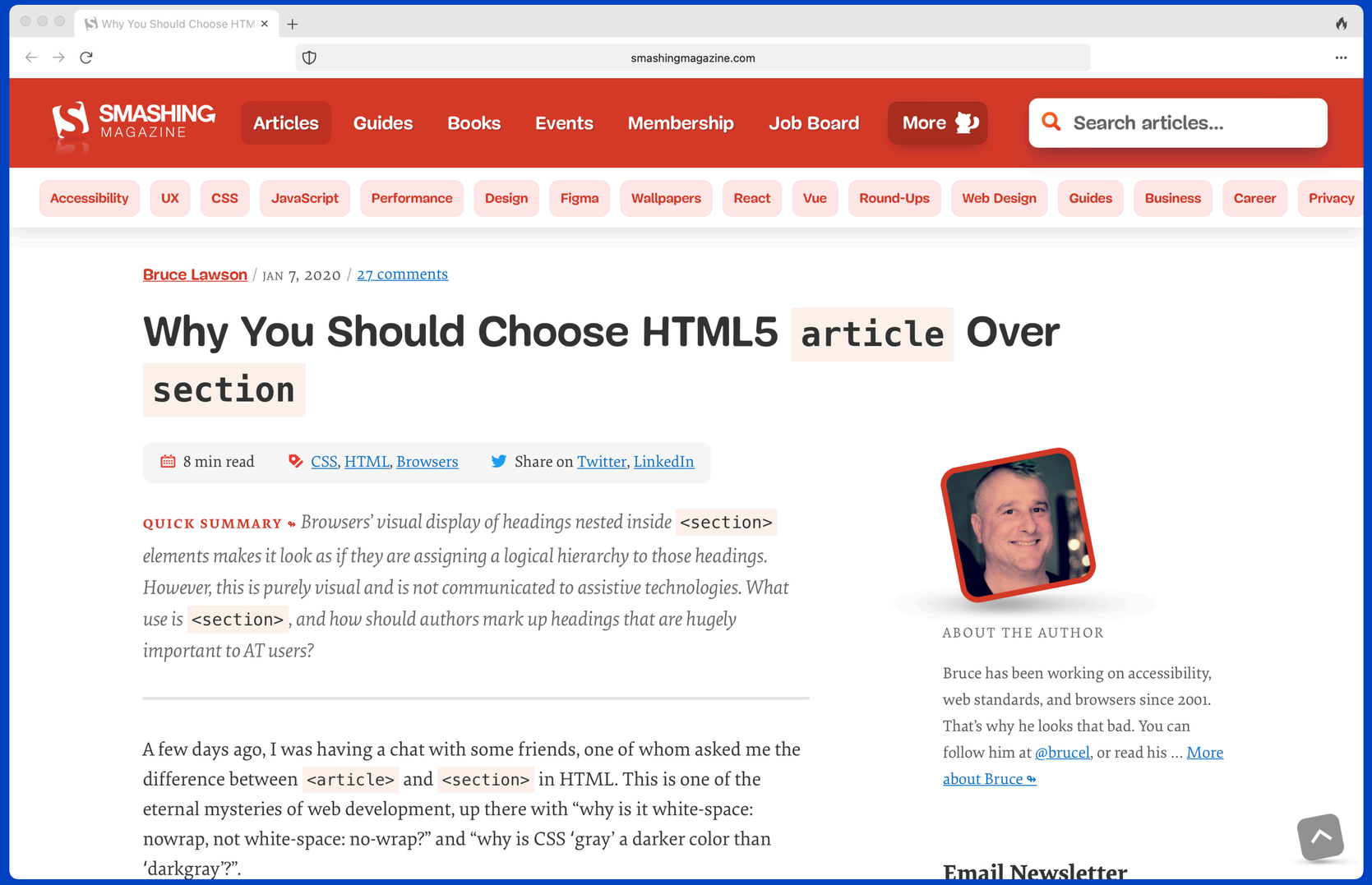
Get the HTML right and you've built a solid foundation. If not, you're building on sand.



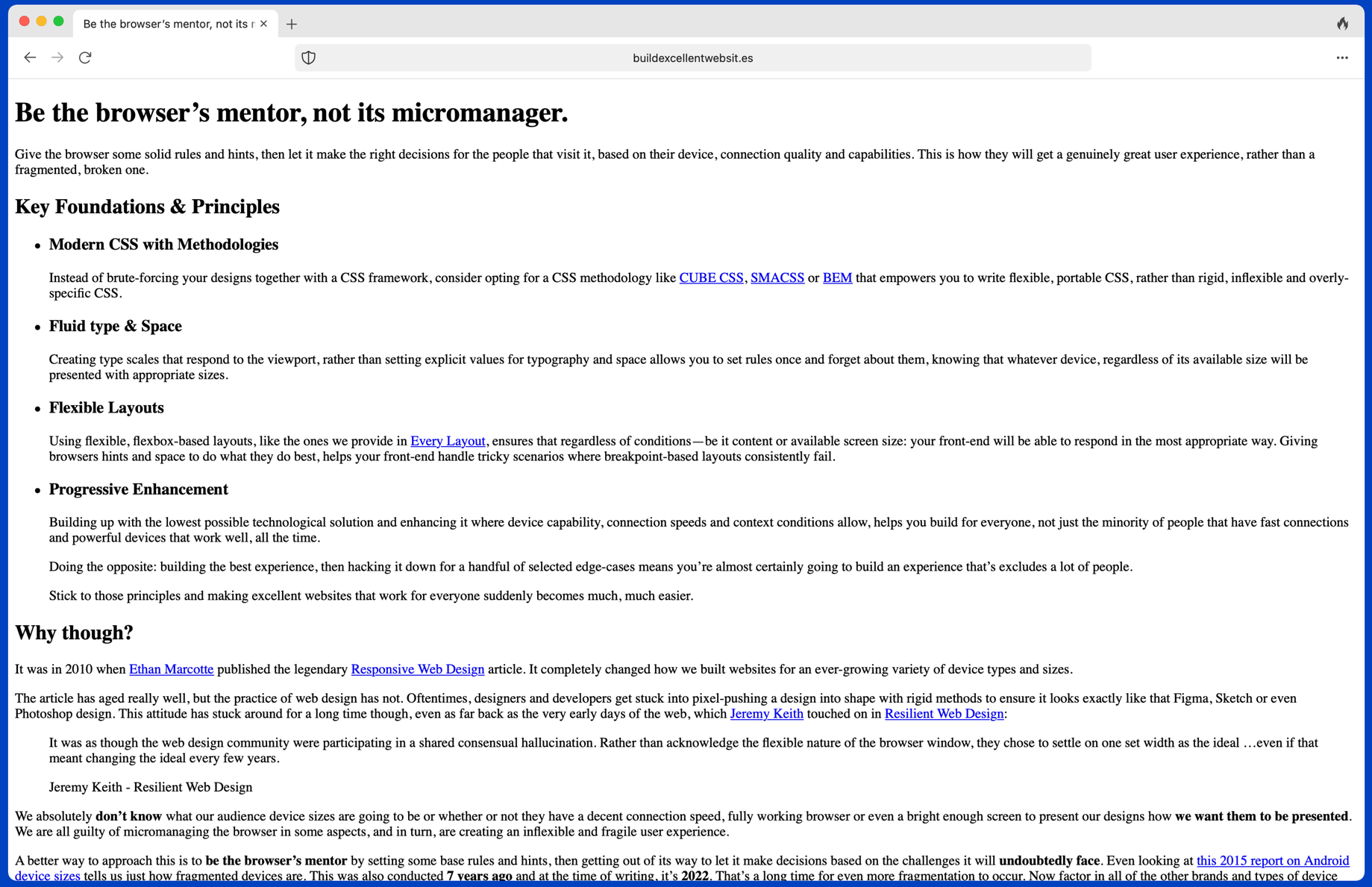
Stefan Rousseau/AP

```
<!DOCTYPE html>
<html lang="en">
  <head> </head>
  <body>
    <main class="flow">
      <header class="section spot-color-primary">...</header>
      <article class="region flow">...</article>
      <article class="region">...</article>
      <article class="section spot-color-primary">...</article>
      <article class="signoff region">...</article>
    </main>
  </body>
</html>

```



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



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

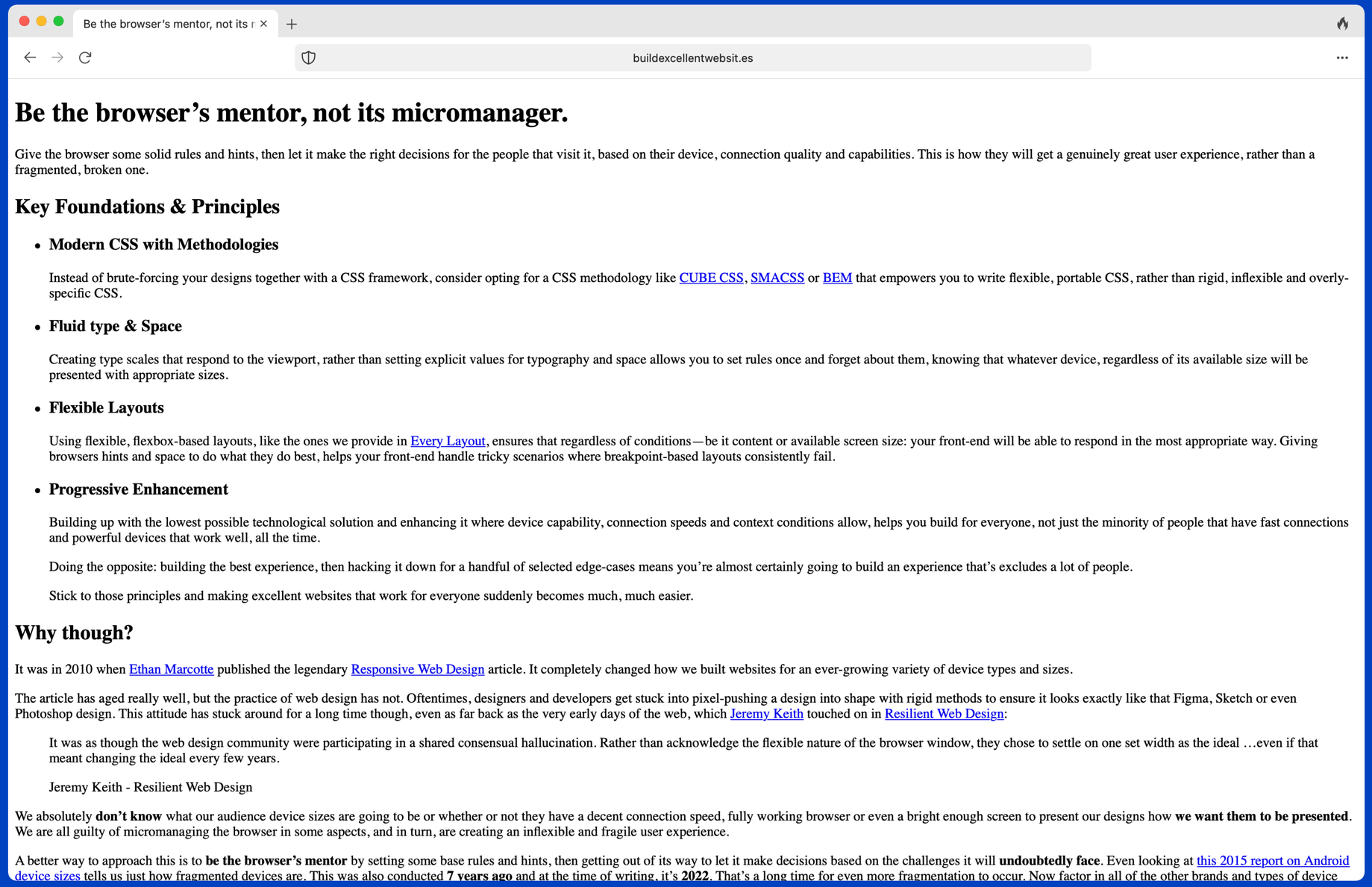
We absolutely **don't know** what our audience device sizes are going to be or whether or not they have a decent connection speed, fully working browser or even a bright enough screen to present our designs how **we want them to be presented**. We are all guilty of micromanaging the browser in some aspects,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



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

```css
*::before,
*::after {
  box-sizing: border-box;
}

body, h1, h2, h3, h4,
p, figure, blockquote, dl, dd {
  margin: 0;
}

ul[role='list'],
ol[role='list'] {
  list-style: none;
}

html {
  text-size-adjust: none;
  -webkit-text-size-adjust: none;
}

html:focus-within {
  scroll-behavior: smooth;
}

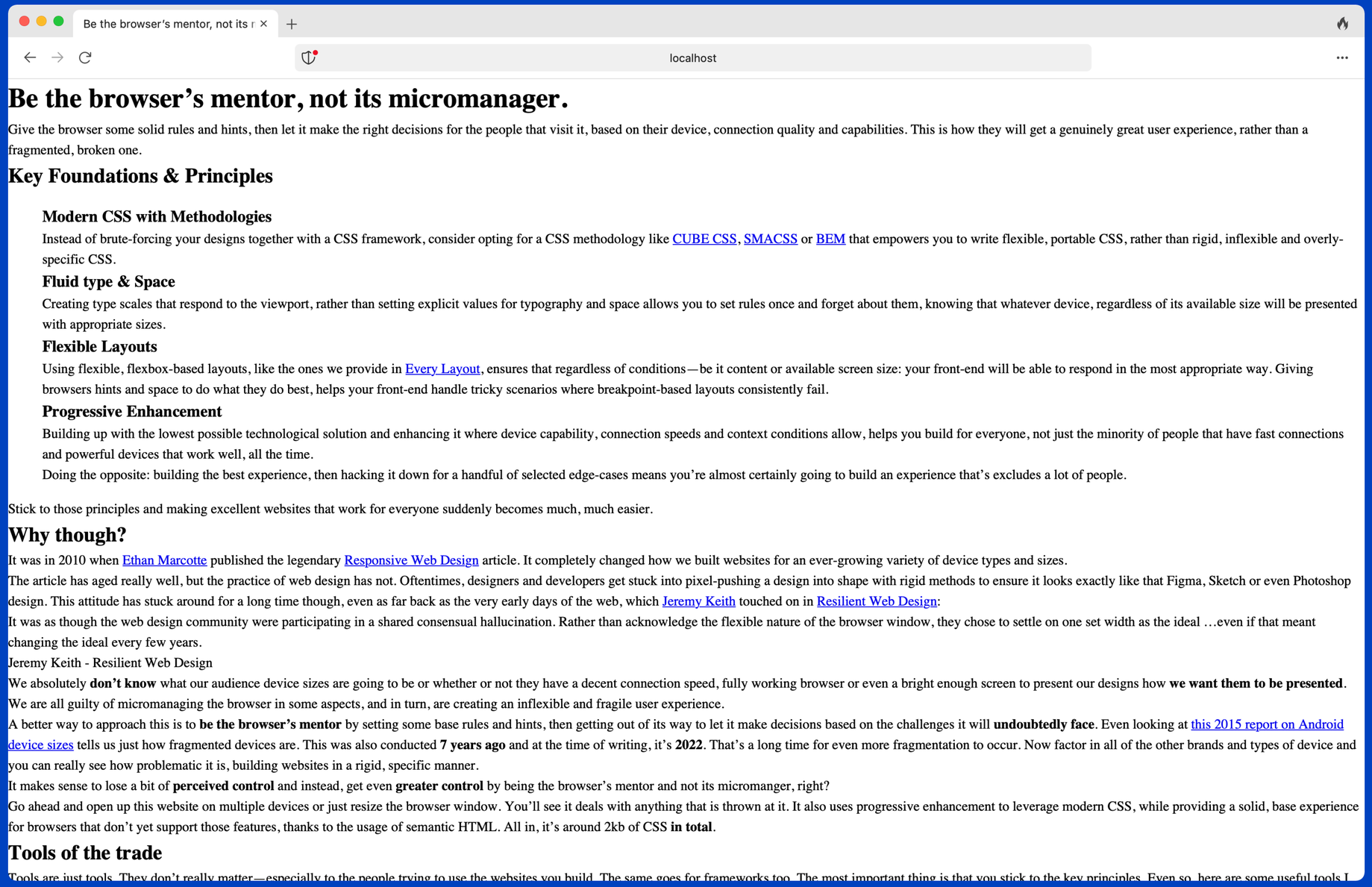
body {
  min-height: 100vh;
  text-rendering: optimizeSpeed;
  line-height: 1.5;
}

a:not([class]) {
  text-decoration-skip-ink: auto;
}

img, picture {
  max-width: 100%;
  display: block;
}

input, button, textarea, select {
  font: inherit;
}

```



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

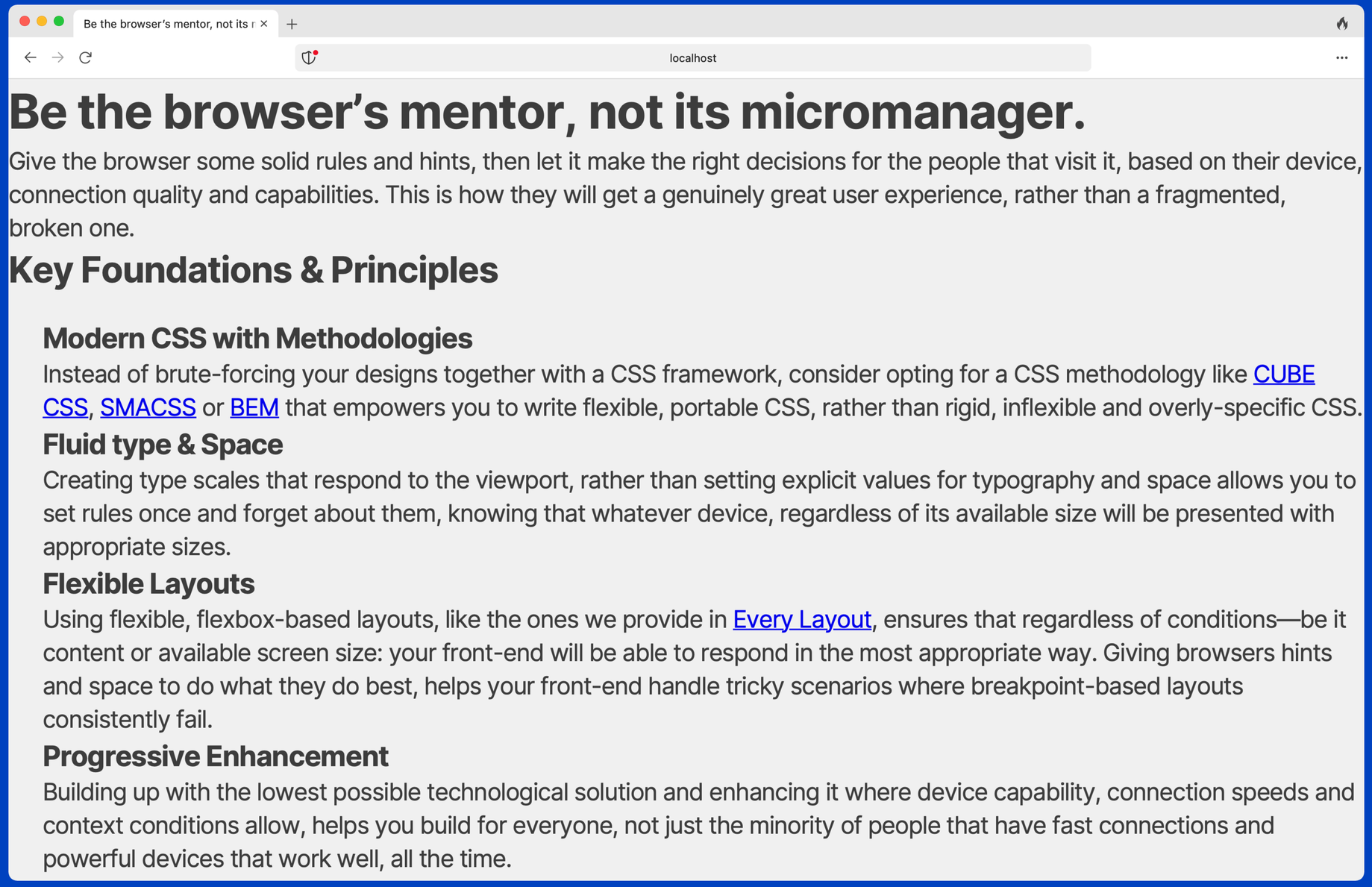
Blocks

Exceptions

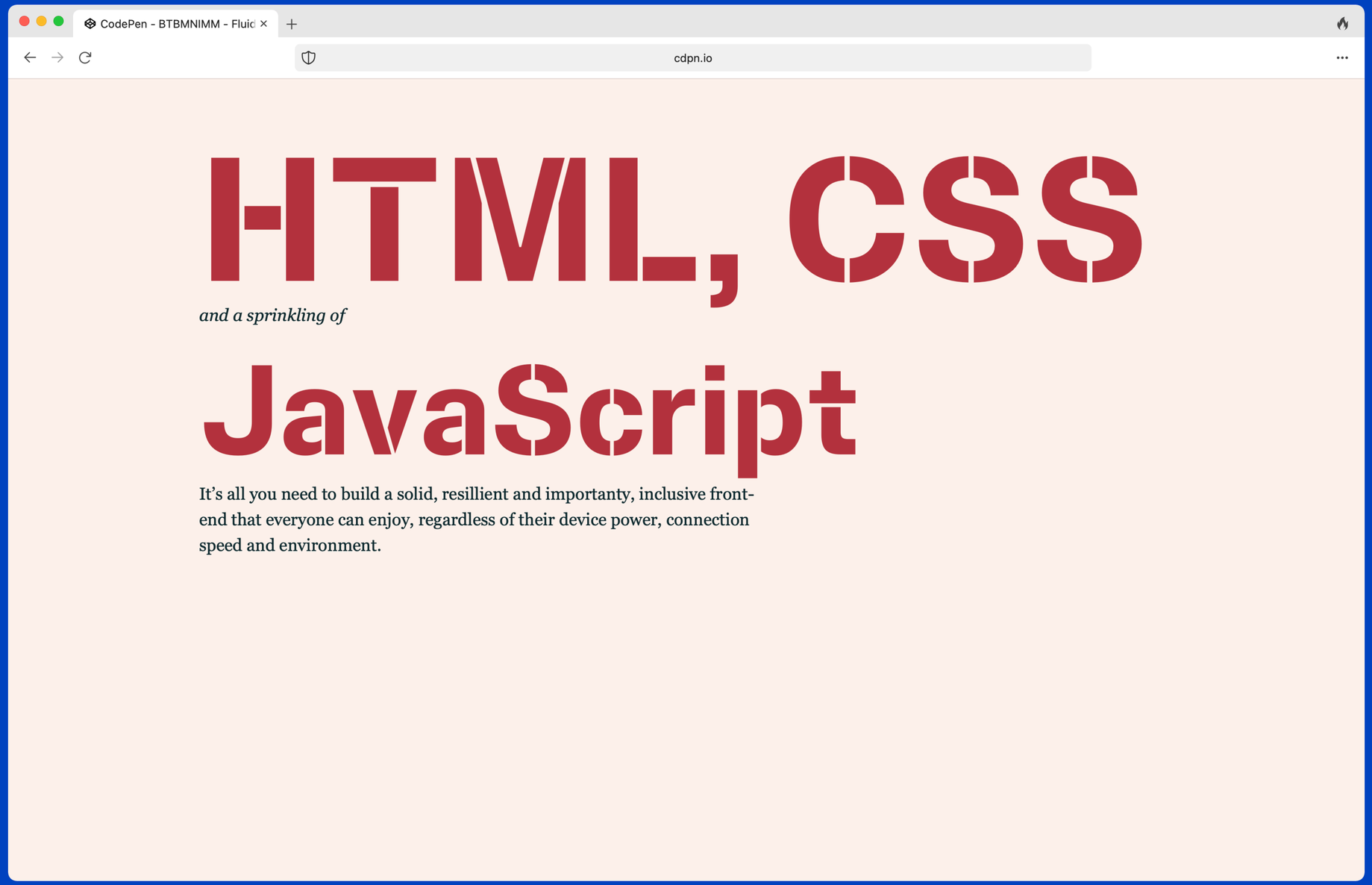
```css
:root {
  --color-primary: #0042bf;
  --color-primary-glare: #d8e2f4;
  --color-secondary: #ee5141;
  --color-secondary-glare: #ffe3e5;
  --space-s: clamp(1rem,0.92rem + 0.39vw,1.25rem);
  --space-m: clamp(1.5rem,1.38rem + 0.58vw,1.875rem);
  --space-l: clamp(2rem,1.84rem + 0.78vw,2.5rem);
  --size-step-1: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

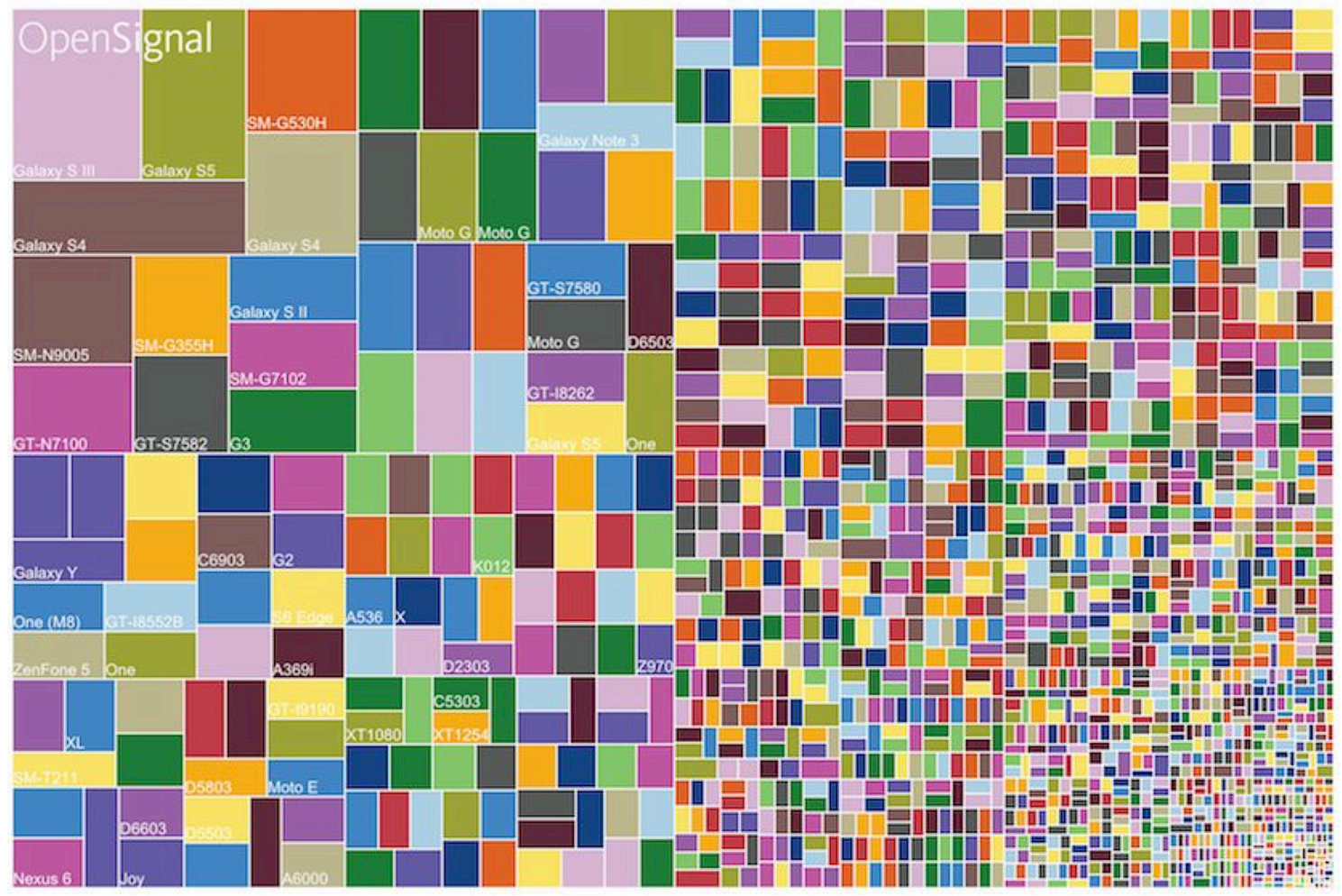
```css
:root {
  --gutter: var(--space-s-m);
  --border-radius: var(--size-step-1);
  --transition-base: 250ms ease;
  --transition-movement: 200ms linear;
  --transition-fade: 200ms ease;
  --transition-bounce: 500ms cubic-bezier(0.5, 0.05, 0.2, 1.5);
  --tracking: -0.05ch;
  --tracking-s: -0.075ch;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

```css
body {
  color: var(--color-dark);
  background: var(--color-light);
  font-size: var(--size-step-1);
  font-family: var(--font-base);
  line-height: 1.4;
  letter-spacing: var(--tracking);
}
```

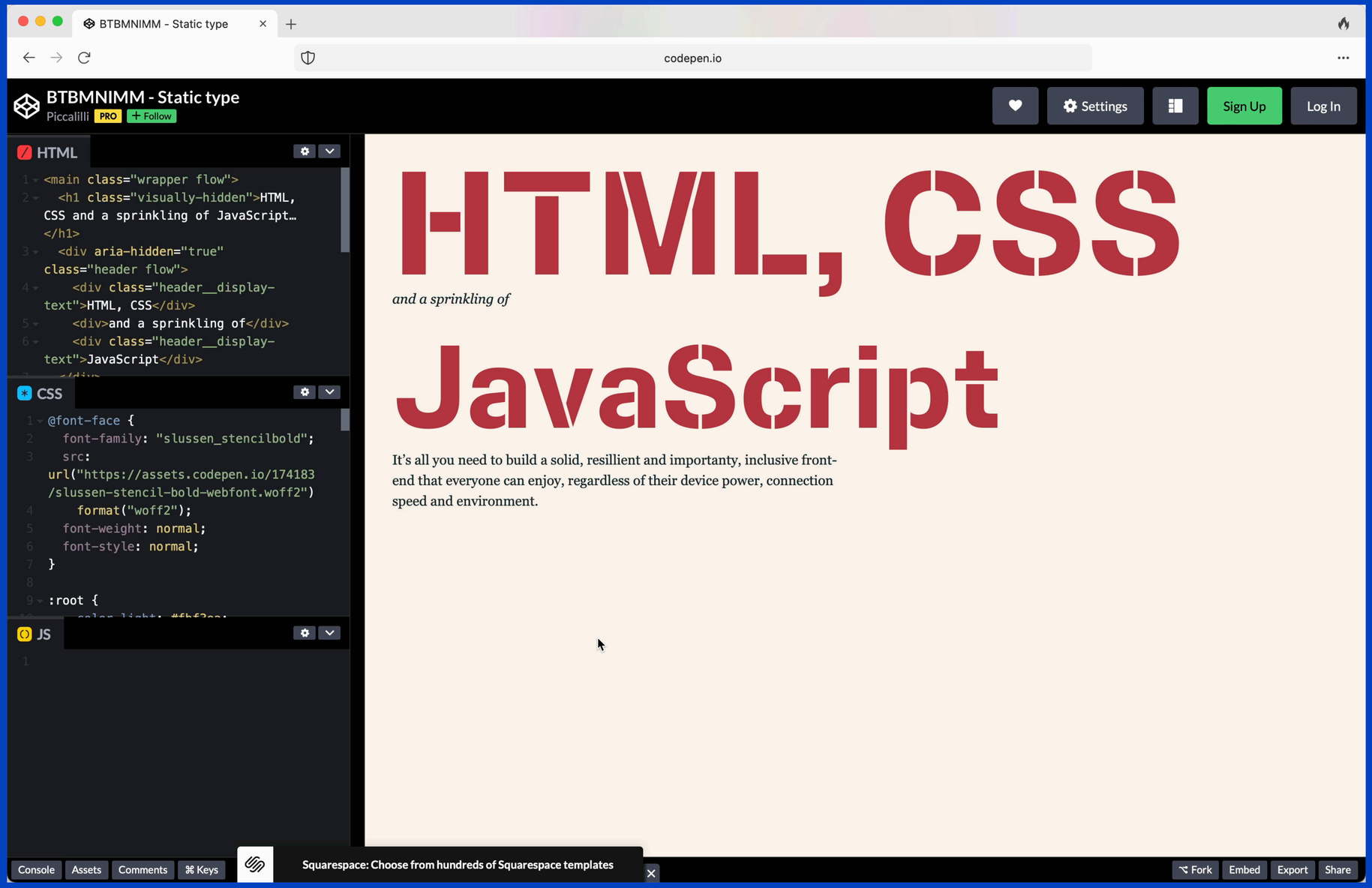

Be the browser's mentor, not its r
localhost
Be the browser's mentor, not its micromanager.
Give the browser some solid rules and hints, then let it make the right decisions for the people that visit it, based on their device, connection quality and capabilities. This is how they will get a genuinely 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


CodePen - BTBMNIMM - Fluic
cdpn.io
HTML, CSS
and a sprinkling of
JavaScript
It's all you need to build a solid, resillient and importanty, inclusive front-end that everyone can enjoy, regardless of their device power, connection speed and environment.



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


BTBMNIMM - Static type
codepen.io
BTBMNIMM - Static type
Piccalilli PRO + Follow
Settings
Sign Up
Log In
HTML
<main class="wrapper flow">
<h1 class="visually-hidden">HTML, CSS and a sprinkling of JavaScript…</h1>

HTML, CSS
and a sprinkling of
JavaScript
CSS
@font-face {
font-family: "slussen_stencilbold";
src:
url("https://assets.codepen.io/174183/slussen-stencil-bold-webfont.woff2")
format("woff2");
font-weight: normal;
font-style: normal;
}
:root {
JS
HTML, CSS
and a sprinkling of
JavaScript
It's all you need to build a solid, resillient and importanty, inclusive front-end that everyone can enjoy, regardless of their device power, connection speed and environment.
Console
Assets
Comments
⌘ Keys
Squarespace: Choose from hundreds of Squarespace templates
Fork
Embed
Export
Share

```css
.header__display-text {
  font-size: 2rem;
}

.header__display-text:first-of-type {
  font-size: 2.3rem;
}

@media (min-width: 768px) {
  .header__display-text {
    font-size: 6rem;
  }

  .header__display-text:first-of-type {
    font-size: 8rem;
  }
}

@media (min-width: 1100px) {
  .header__display-text {
    font-size: 8rem;
  }

  .header__display-text:first-of-type {
    font-size: 10rem;
  }
}
```

```css
@media (min-width: 768px) {
  .header__display-text {
    font-size: 6rem;
  }

  .header__display-text:first-of-type {
    font-size: 8rem;
  }
}

@media (min-width: 856px) {
  .header__display-text {
    font-size: 7rem;
  }

  .header__display-text:first-of-type {
    font-size: 9rem;
  }
}

@media (min-width: 1100px) {
  .header__display-text {
    font-size: 8rem;
  }

  .header__display-text:first-of-type {
    font-size: 10rem;
  }
}
```


BTBMNIMM - Fluid type
codepen.io
BTBMNIMM - Fluid type
Piccalilli PRO + Follow
Settings
Sign Up
Log In
HTML
<main class="wrapper flow">
<h1 class="visually-hidden">HTML, CSS and a sprinkling of JavaScript…</h1>

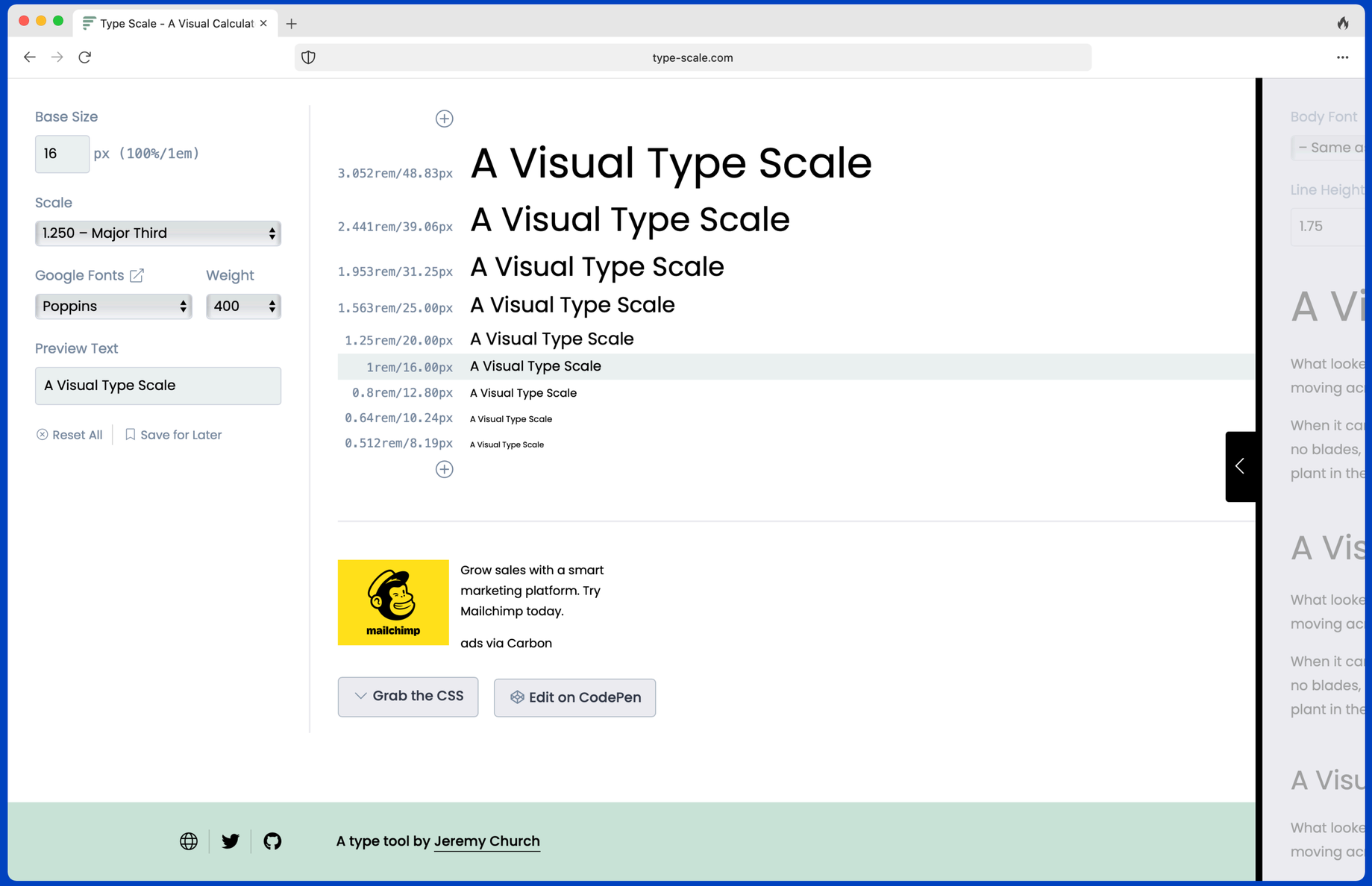
HTML, CSS
and a sprinkling of
JavaScript
CSS
@font-face {
font-family: "slussen_stencilbold";
src:
url("https://assets.codepen.io/174183/slussen-stencil-bold-webfont.woff2")
format("woff2");
font-weight: normal;
font-style: normal;
}
:root {
JS
HTML, CSS
and a sprinkling of
JavaScript
It's all you need to build a solid, resillient and importanty, inclusive front-end that everyone can enjoy, regardless of their device power, connection speed and environment.
Console
Assets
Comments
⌘ Keys
Squarespace: Choose from hundreds of Squarespace templates
Fork
Embed
Export
Share

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

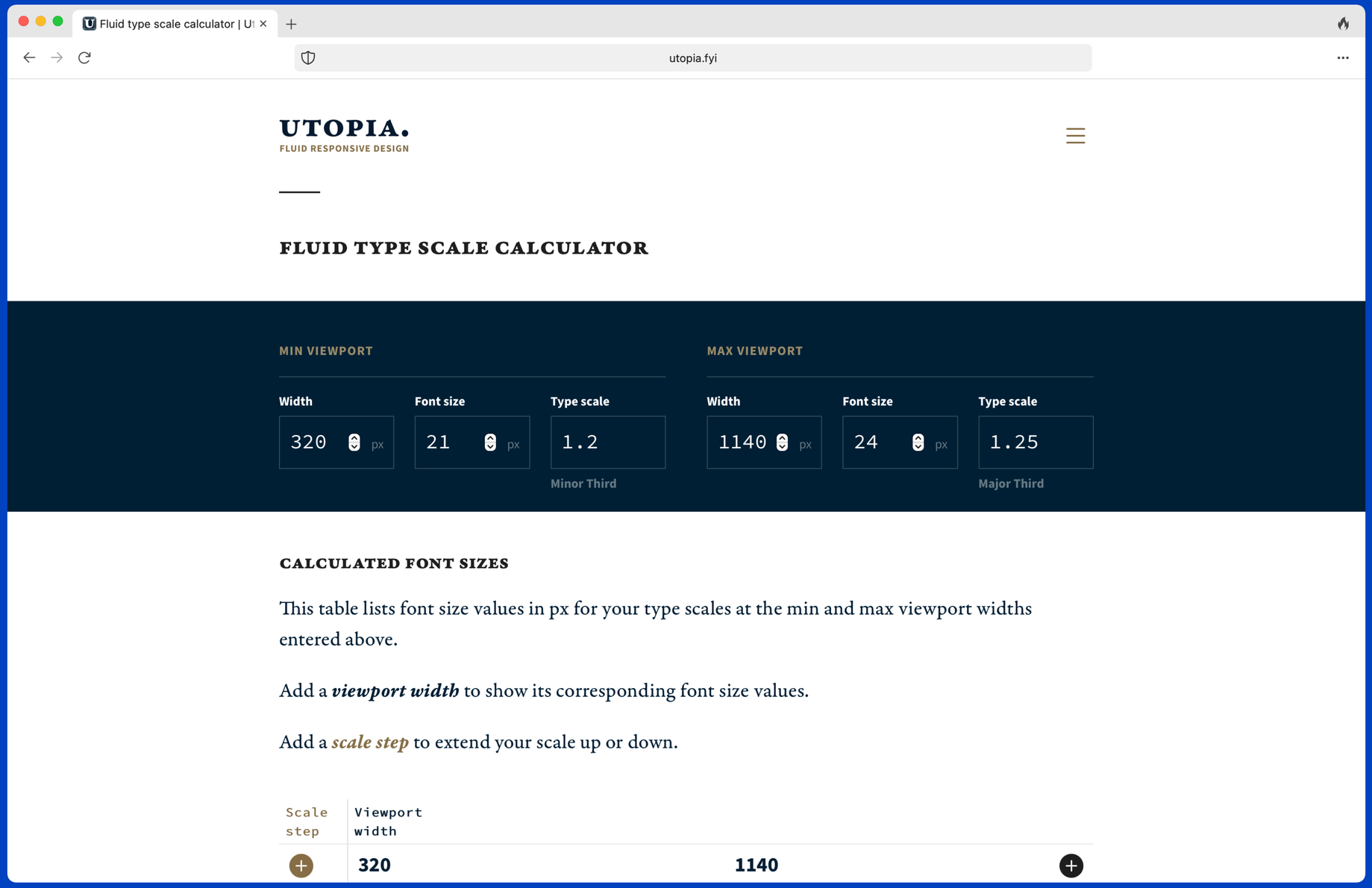

BTBMNIMM - Fluid type
codepen.io
BTBMNIMM - Fluid type
Piccalilli PRO + Follow
Settings
Sign Up
Log In
HTML
<main class="wrapper flow">
<h1 class="visually-hidden">HTML, CSS and a sprinkling of JavaScript…</h1>

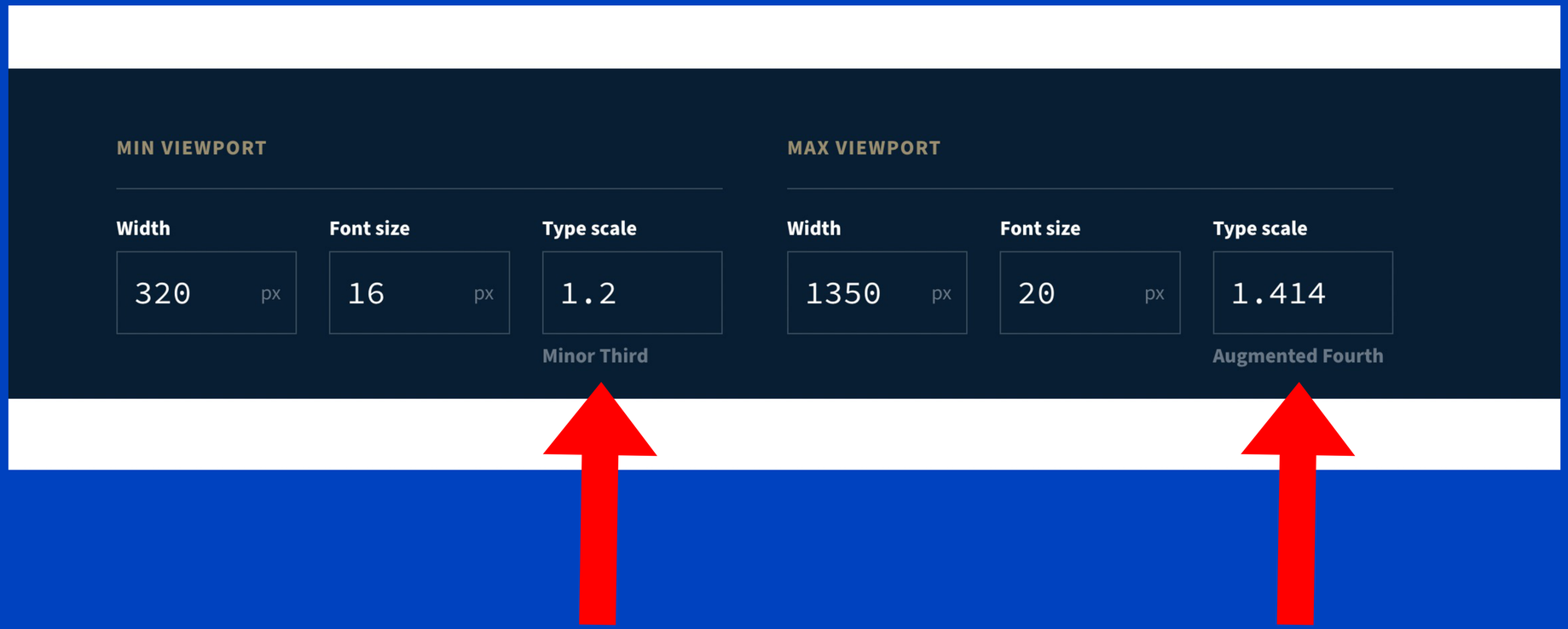
HTML, CSS
and a sprinkling of
JavaScript
CSS
@font-face {
font-family: "slussen_stencilbold";
src:
url("https://assets.codepen.io/174183/slussen-stencil-bold-webfont.woff2")
format("woff2");
font-weight: normal;
font-style: normal;
}
:root {
JS
HTML, CSS
and a sprinkling of
JavaScript
It's all you need to build a solid, resillient and importanty, inclusive front-end that everyone can enjoy, regardless of their device power, connection speed and environment.
Console
Assets
Comments
⌘ Keys
Squarespace: Choose from hundreds of Squarespace templates
Fork
Embed
Export
Share

Size and space scales

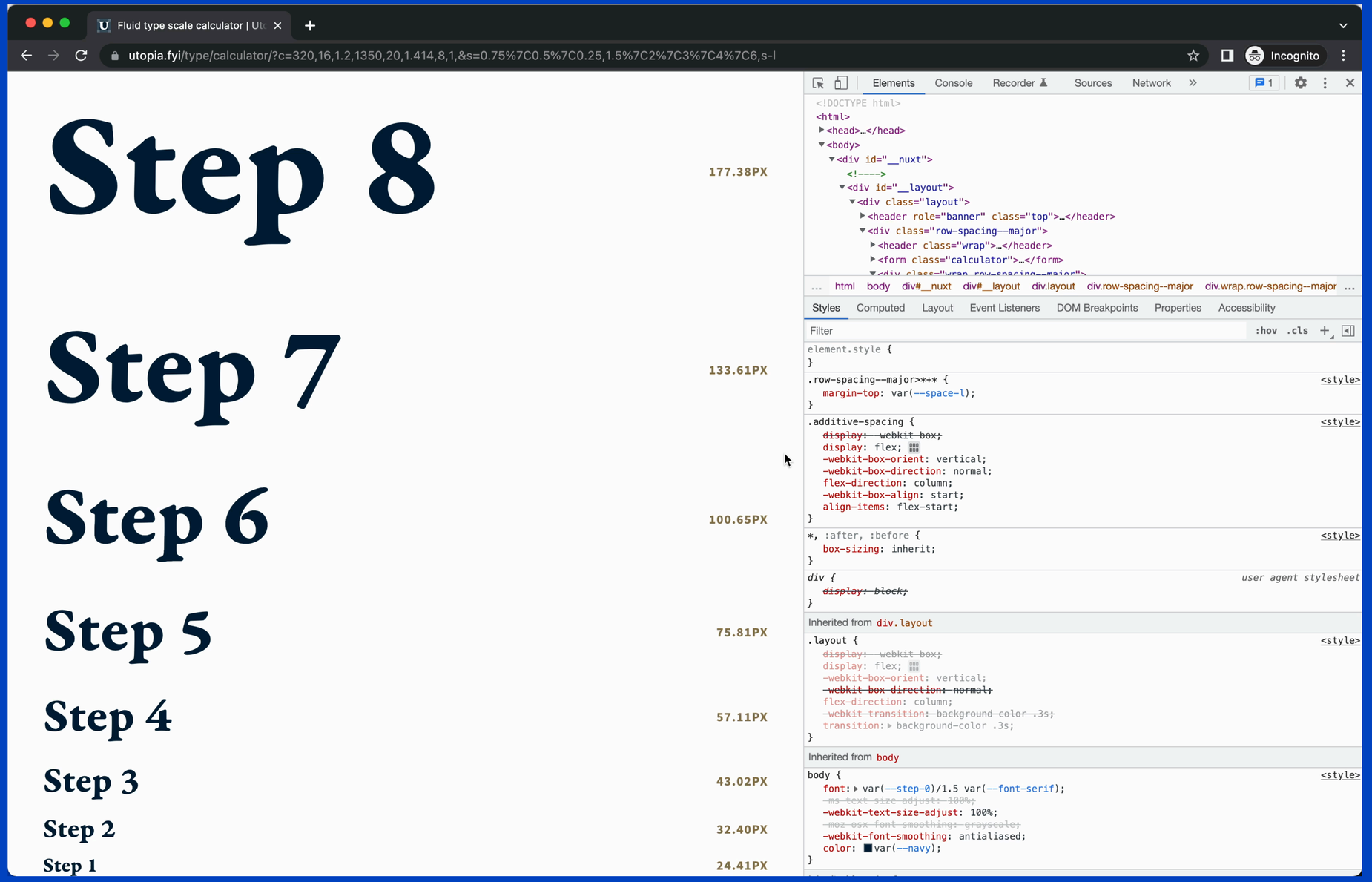

Type Scale - A Visual Calculat
type-scale.com
Base Size
16
px (100%/1em)
Scale
1.250 – Major Third
Google Fonts
Weight
Poppins
400
Preview Text
A Visual Type Scale
Reset All
Save for Later
3.052rem/48.83px A Visual Type Scale
2.441rem/39.06px A Visual Type Scale
1.953rem/31.25px A Visual Type Scale
1.563rem/25.00px A Visual Type Scale
1.25rem/20.00px A Visual Type Scale
1rem/16.00px A Visual Type Scale
0.8rem/12.80px A Visual Type Scale
0.64rem/10.24px A Visual Type Scale
0.512rem/8.19px A Visual Type Scale
mailchimp
Grow sales with a smart marketing platform. Try Mailchimp today.
ads via Carbon
Grab the CSS
Edit on CodePen
A type tool by Jeremy Church
Body Font
– Same a
Line Height
1.75

```
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)
```


Fluid type scale calculator | Ut
utopia.fyi
UTOPIA.
FLUID RESPONSIVE DESIGN
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
Width
320 px
Font size
16 px
Type scale
1.2
Minor Third
MAX VIEWPORT
Width
1350 px
Font size
20 px
Type scale
1.414
Augmented Fourth

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


Fluid type scale calculator | Ut
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
Incognito
Step 8
177.38PX
Step 7
133.61PX
Step 6
100.65PX
Step 5
75.81PX
Step 4
57.11PX
Step 3
43.02PX
Step 2
32.40PX
Step 1
24.41PX
Elements
Console
Recorder
Sources
Network
<!DOCTYPE html>
<html>
<head>…</head>
<body>

<!---->

<header role="banner" class="top">…</header>

<header class="wrap">…</header>
<form class="calculator">…</form>
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 {
}
.row-spacing--major>*+* {
margin-top: var(--space-l);
}
.additive-spacing {
display: -webkit-box;
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: -webkit-box;
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);
}

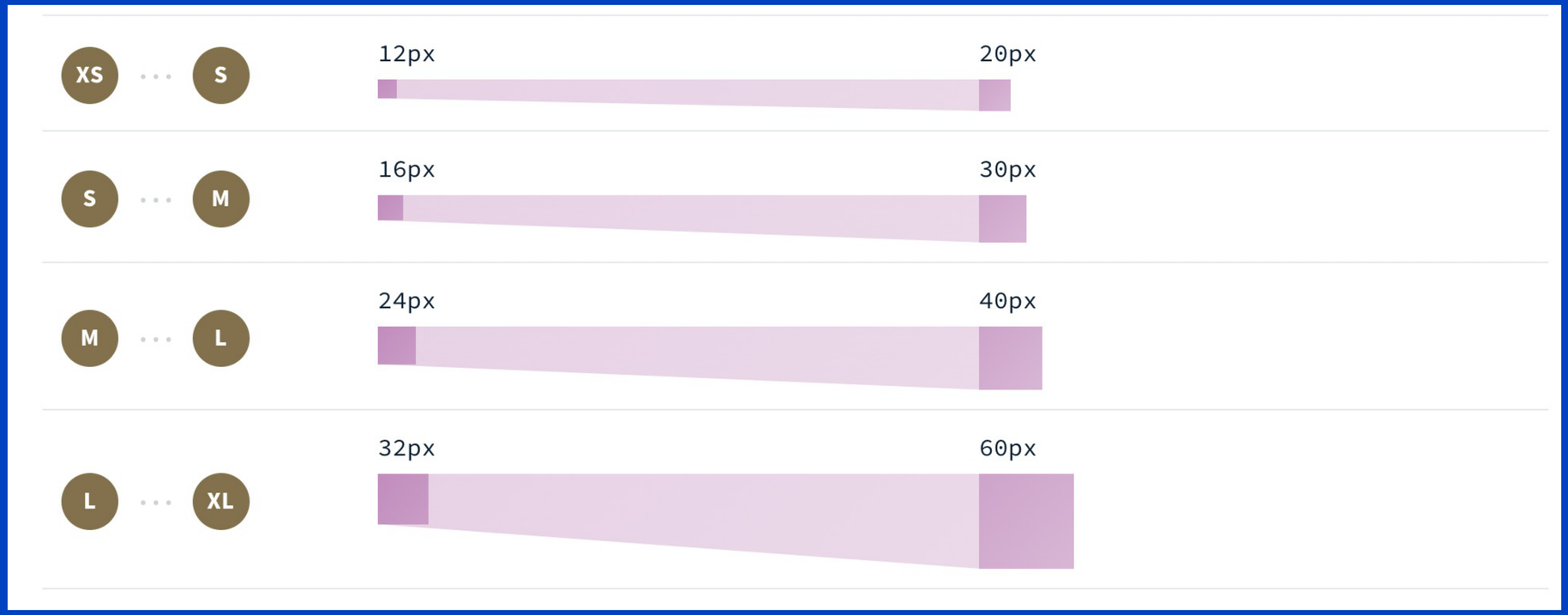
MIN VIEWPORT

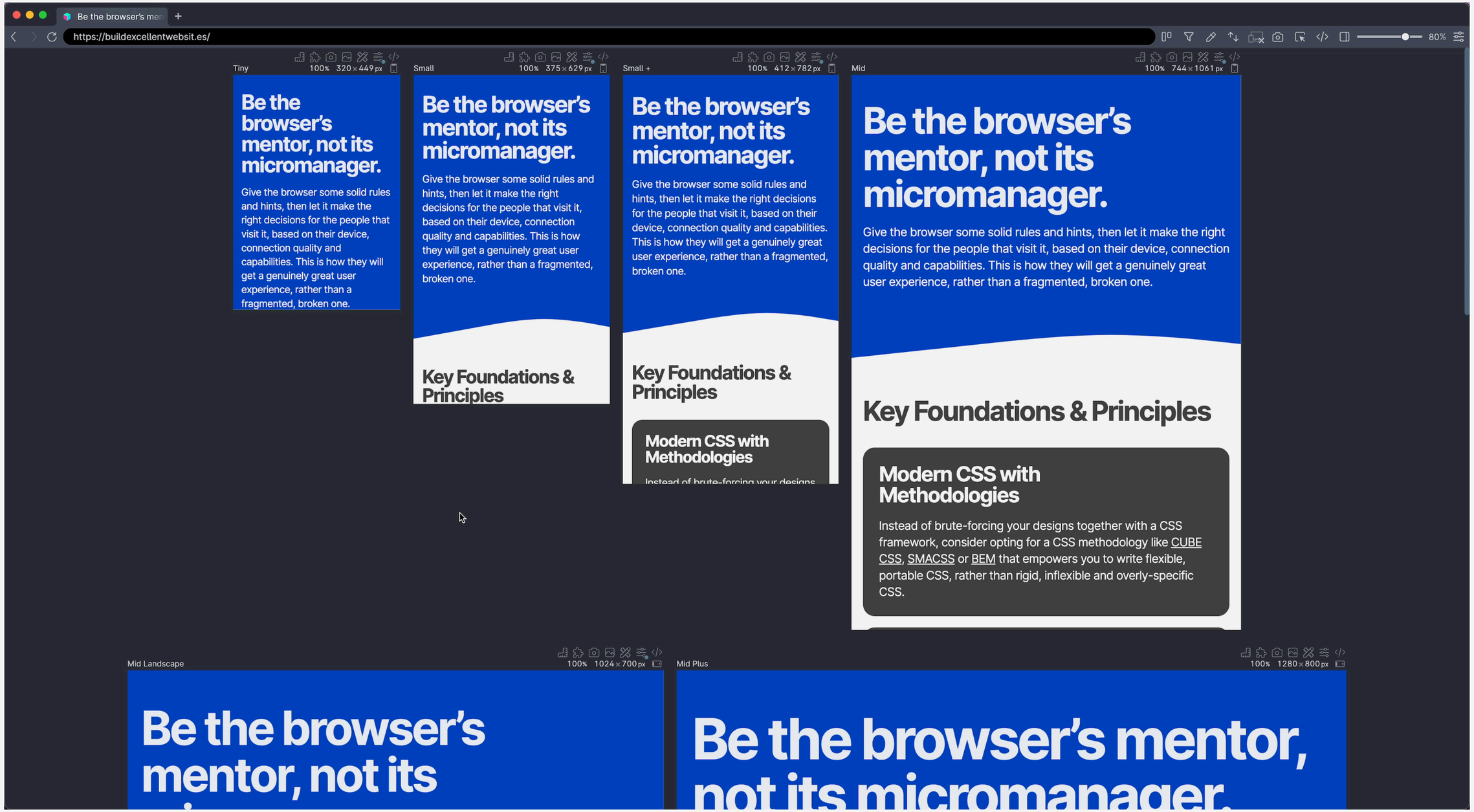
Width	Font size	Type scale
320 px	16 px	1.2
		Minor Third

MAX VIEWPORT

Width	Font size	Type scale
1350 px	20 px	1.414
		Augmented Fourth

	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	


XS
S
12px
20px
S
M
16px
30px
M
L
24px
40px
L
XL
32px
60px


Be the browser's men
https://buildexcellentwebsit.es/
80%
Tiny
100% 320 × 449 px
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.
Small
100% 375 × 629 px
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
Small +
100% 412 × 782 px
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
Mid
100% 744 × 1061 px
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.
Mid Landscape
100% 1024 × 700 px
Be the browser's mentor, not its
Mid Plus
100% 1280 × 800 px
Be the browser's mentor, not its micromanager.

Back to our CSS

```css
body {
  color: var(--color-dark);
  background: var(--color-light);
  font-size: var(--size-step-1);
  font-family: var(--font-base);
  line-height: 1.4;
  letter-spacing: var(--tracking);
}
```

```css
h1, h2, h3 {
  line-height: 1;
  letter-spacing: var(--tracking-s);
}

h1 {
  font-size: var(--size-step-5);
}

h2 {
  font-size: var(--size-step-4);
}

h3 {
  font-size: var(--size-step-3);
}

```

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

h1, h2, h3 {
  max-width: 20ch;
}

```

```css
blockquote:not([class]) {
  font-family: var(--font-serif);
  font-size: var(--size-step-2);
}

blockquote:not([class]) p:last-of-type {
  font-family: var(--font-base);
  font-size: var(--size-step-1);
  font-weight: normal;
}

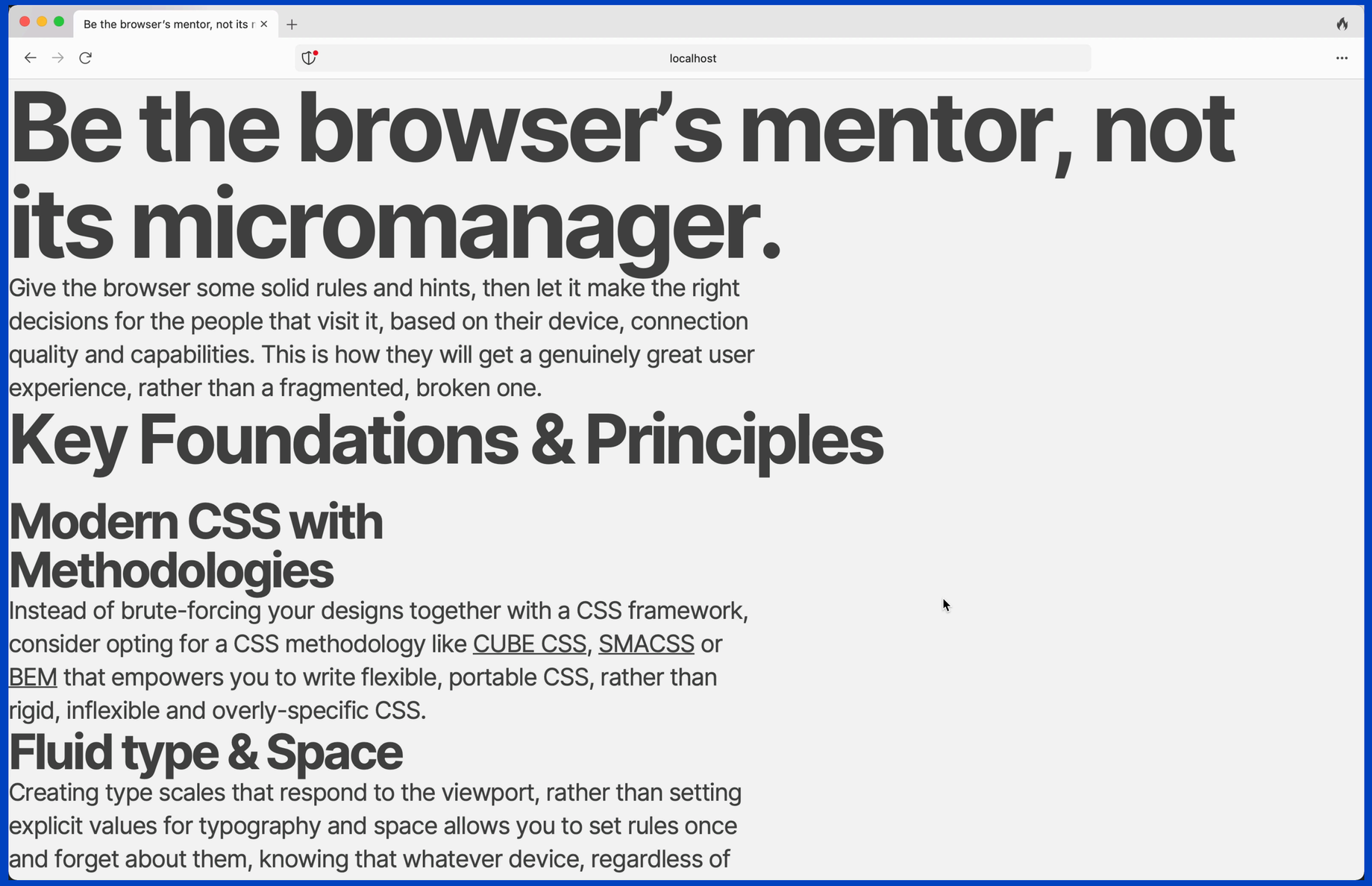
svg {
  height: 2ex;
  width: auto;
  flex: none;
}

a {
  color: currentcolor;
}

a:hover {
  text-decoration: none;
}

:focus {
  outline: 2px solid;
  outline-offset: 0.3ch;
}

:target {
  scroll-margin-top: 2ex;
}
```


Be the browser's mentor, not its r
localhost
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

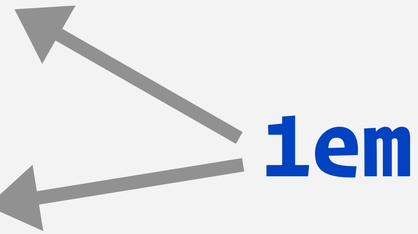
Exceptions


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.

```css
.my-context {
  --flow-space: 10rem;
}

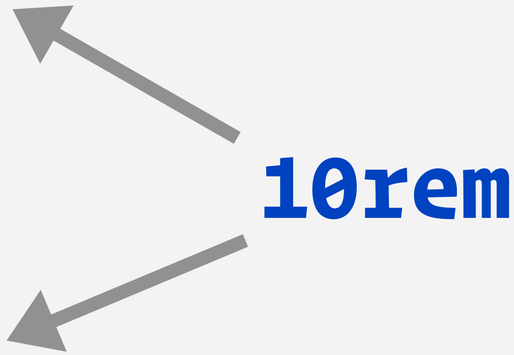
.flow > * + * {
  margin-top: var(--flow-space, 1em);
}
```

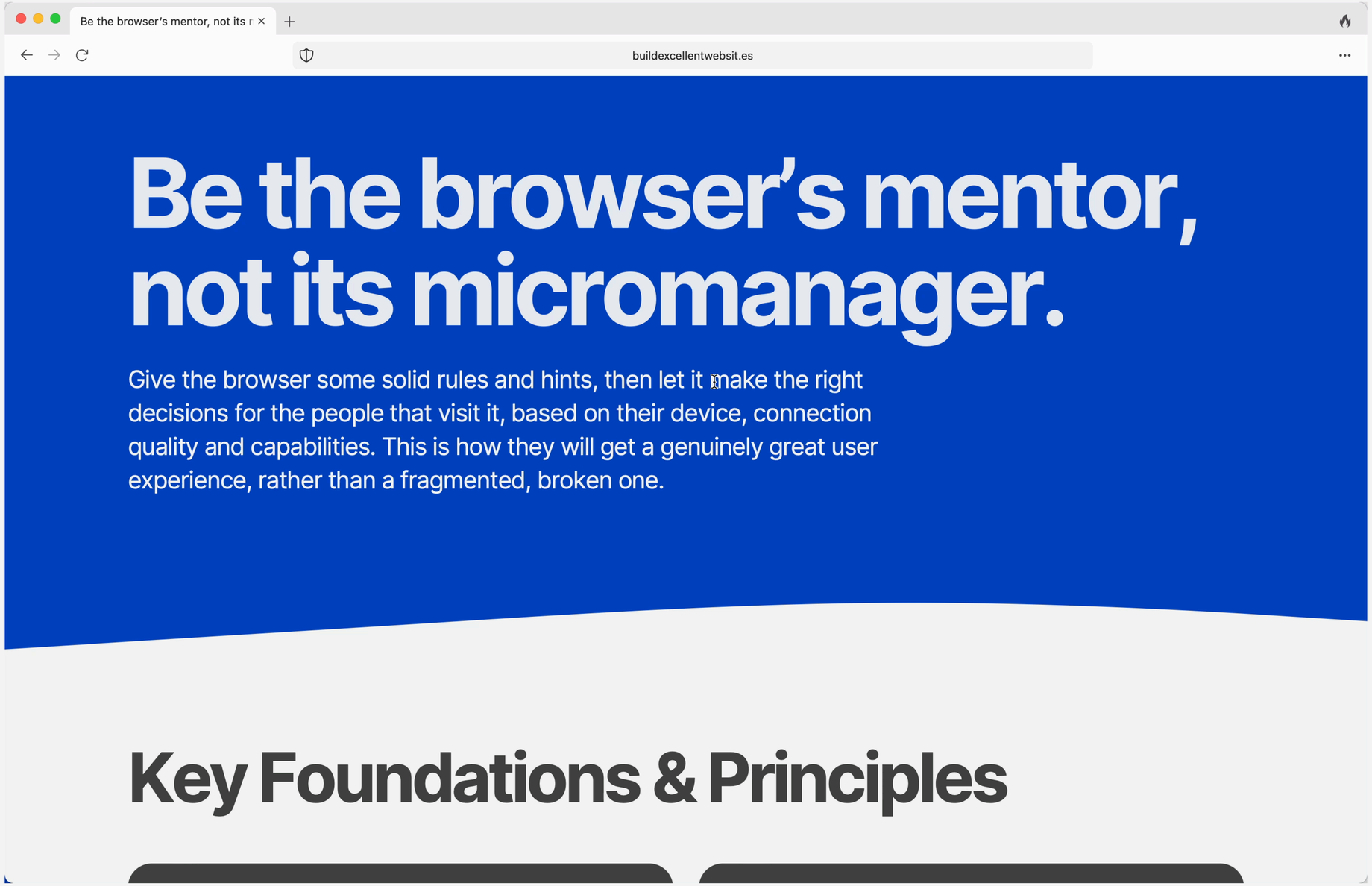
Hello I am a heading

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.

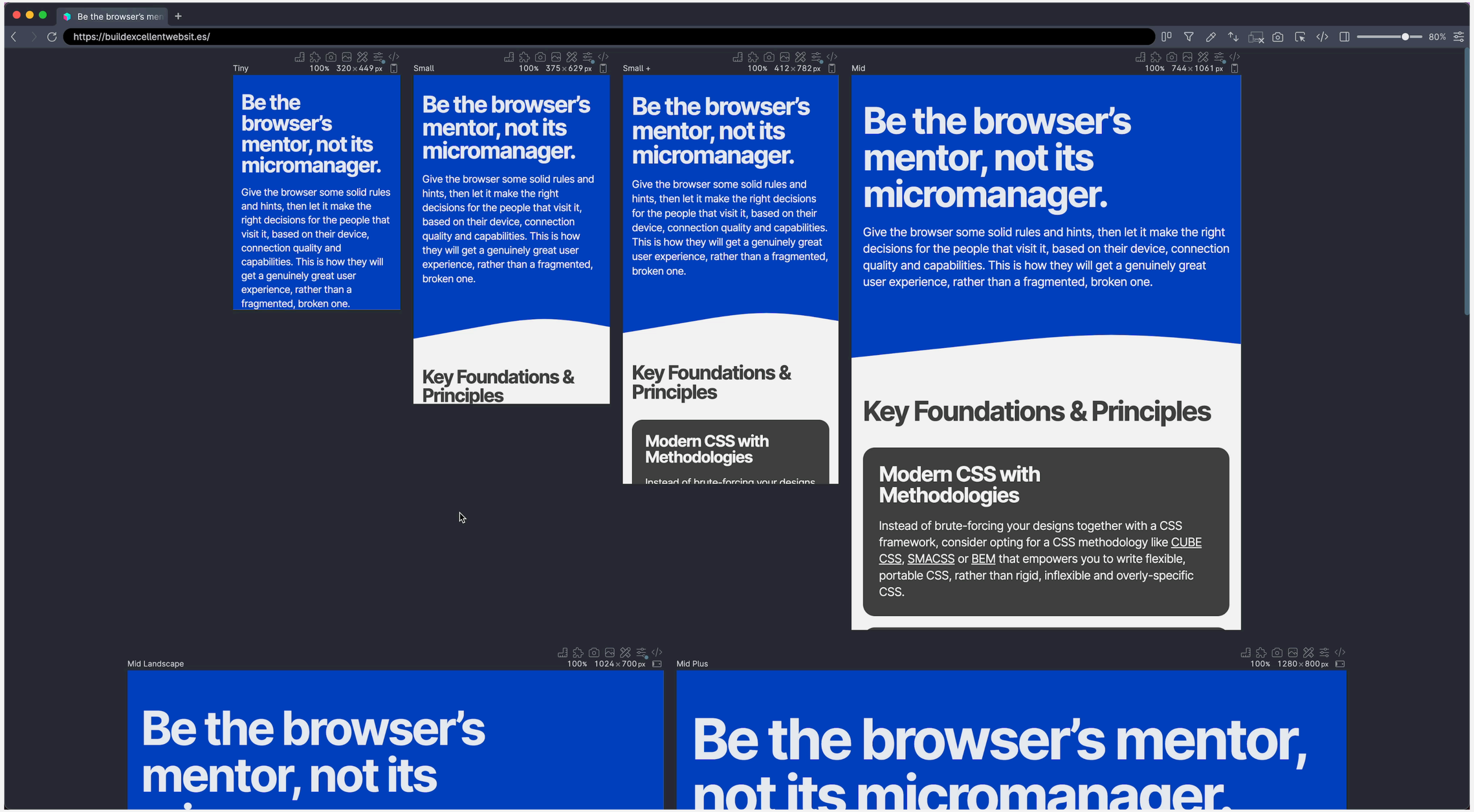
A subheading

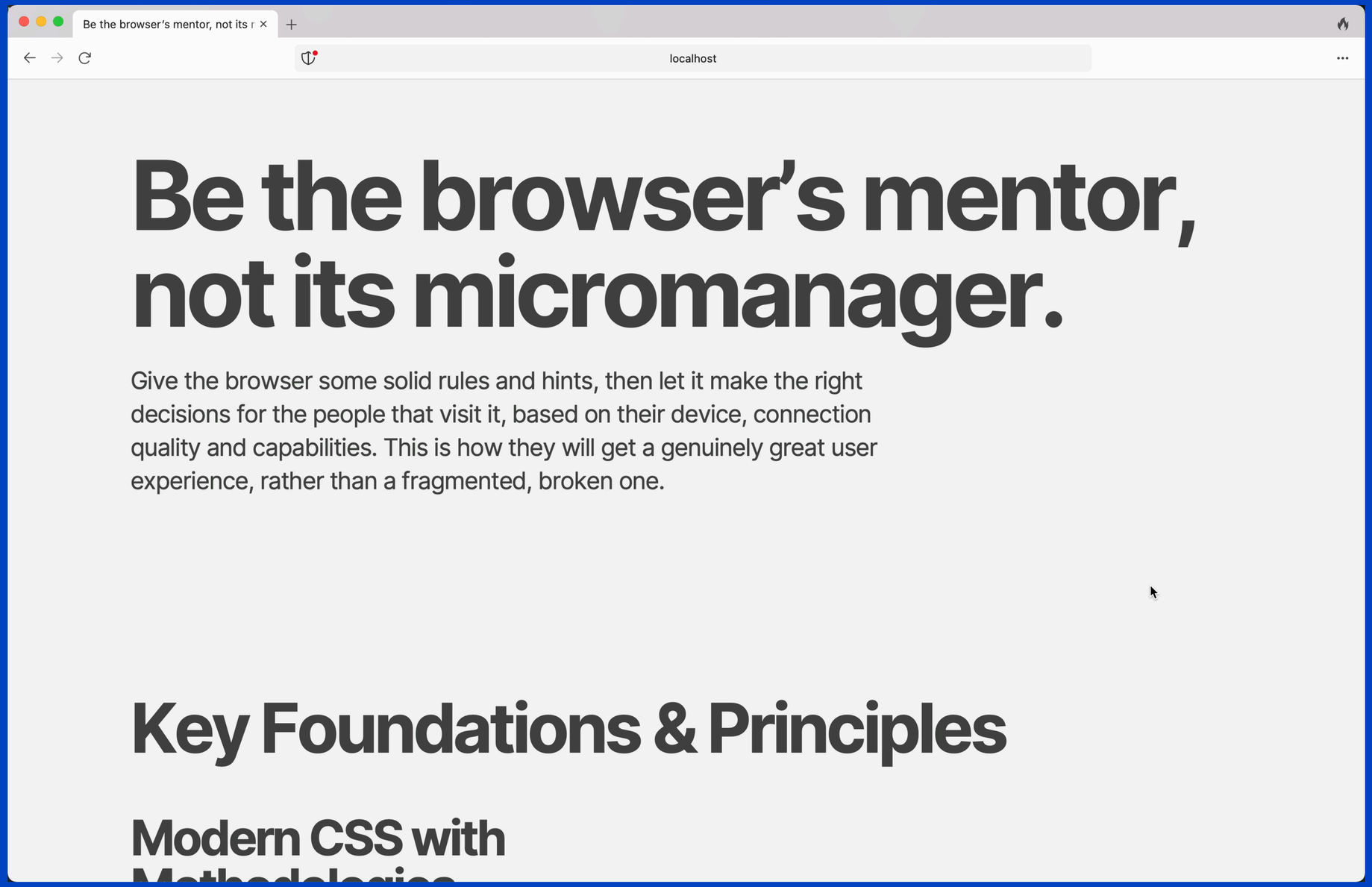
Fusce dapibus, tellus ac cursus commodo, tortor mauris condimentum nibh, ut fermentum massa justo sit amet risus. Donec id elit non mi porta gravida at eget metus.




Be the browser's mentor, not its r
buildexcellentwebsit.es
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

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


Be the browser's men
https://buildexcellentwebsit.es/
80%
Tiny
100% 320 × 449 px
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.
Small
100% 375 × 629 px
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
Small +
100% 412 × 782 px
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
Mid
100% 744 × 1061 px
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.
Mid Landscape
100% 1024 × 700 px
Be the browser's mentor, not its
Mid Plus
100% 1280 × 800 px
Be the browser's mentor, not its micromanager.


Be the browser's mentor, not its r
localhost
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

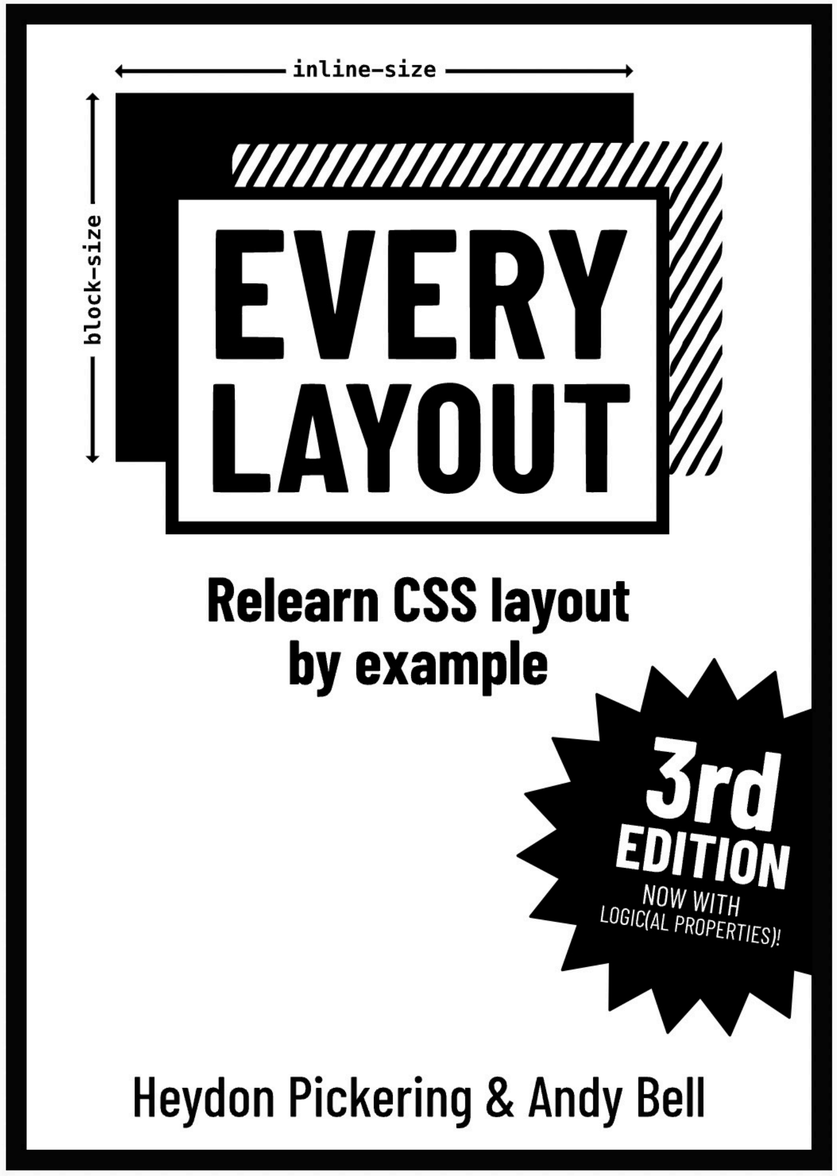
Global CSS

Composition

Utilities

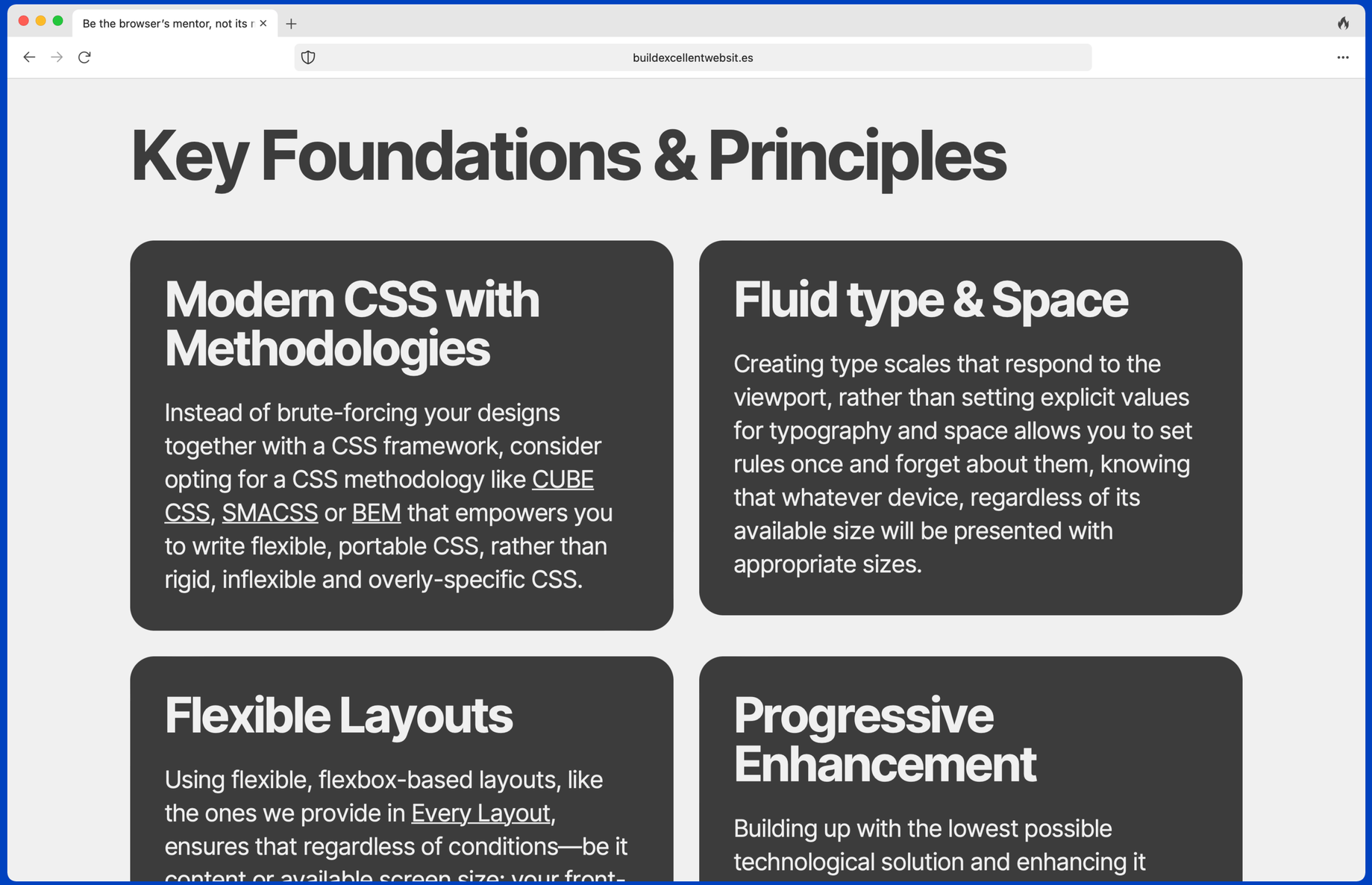
Blocks

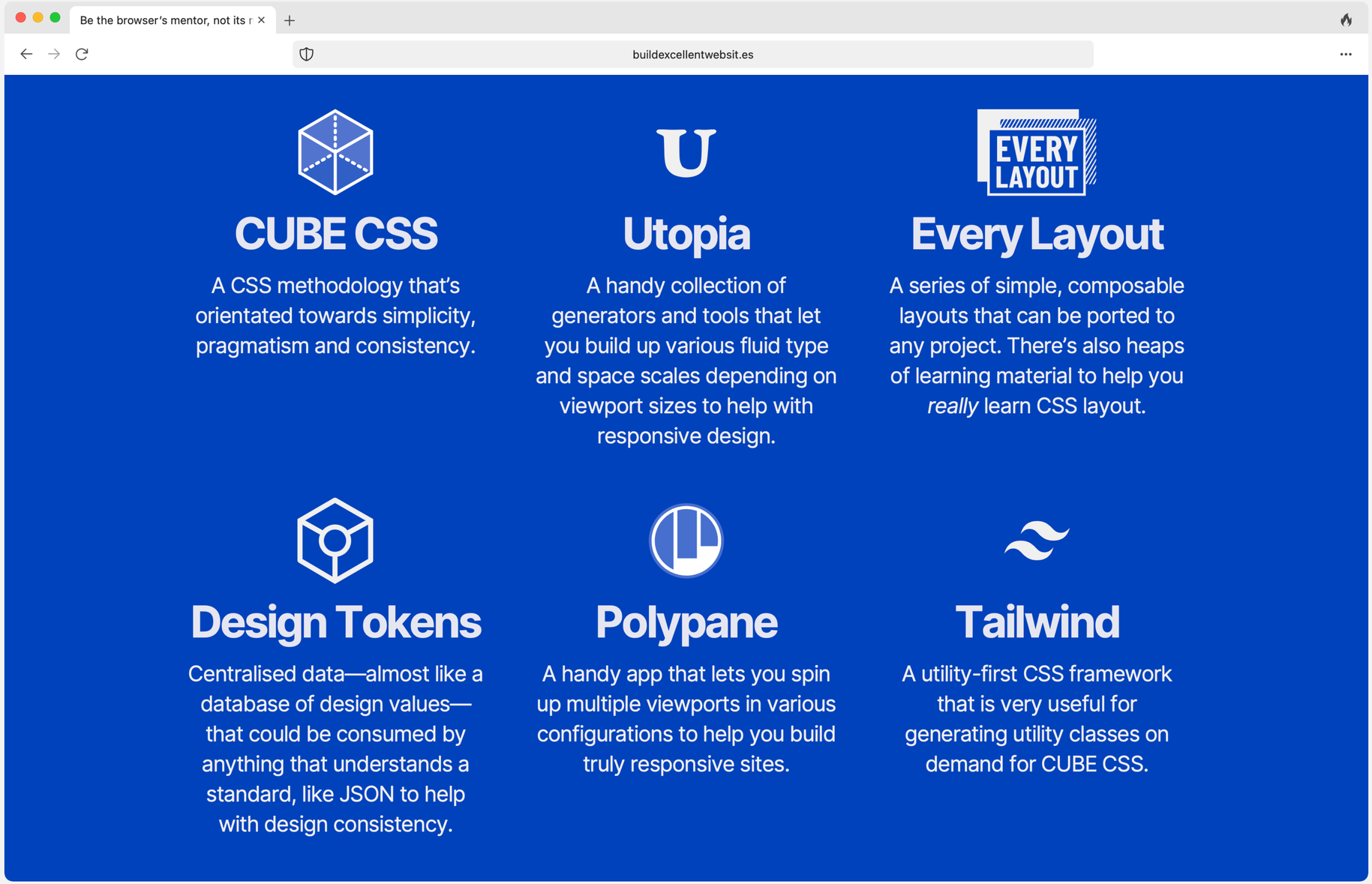
Exceptions



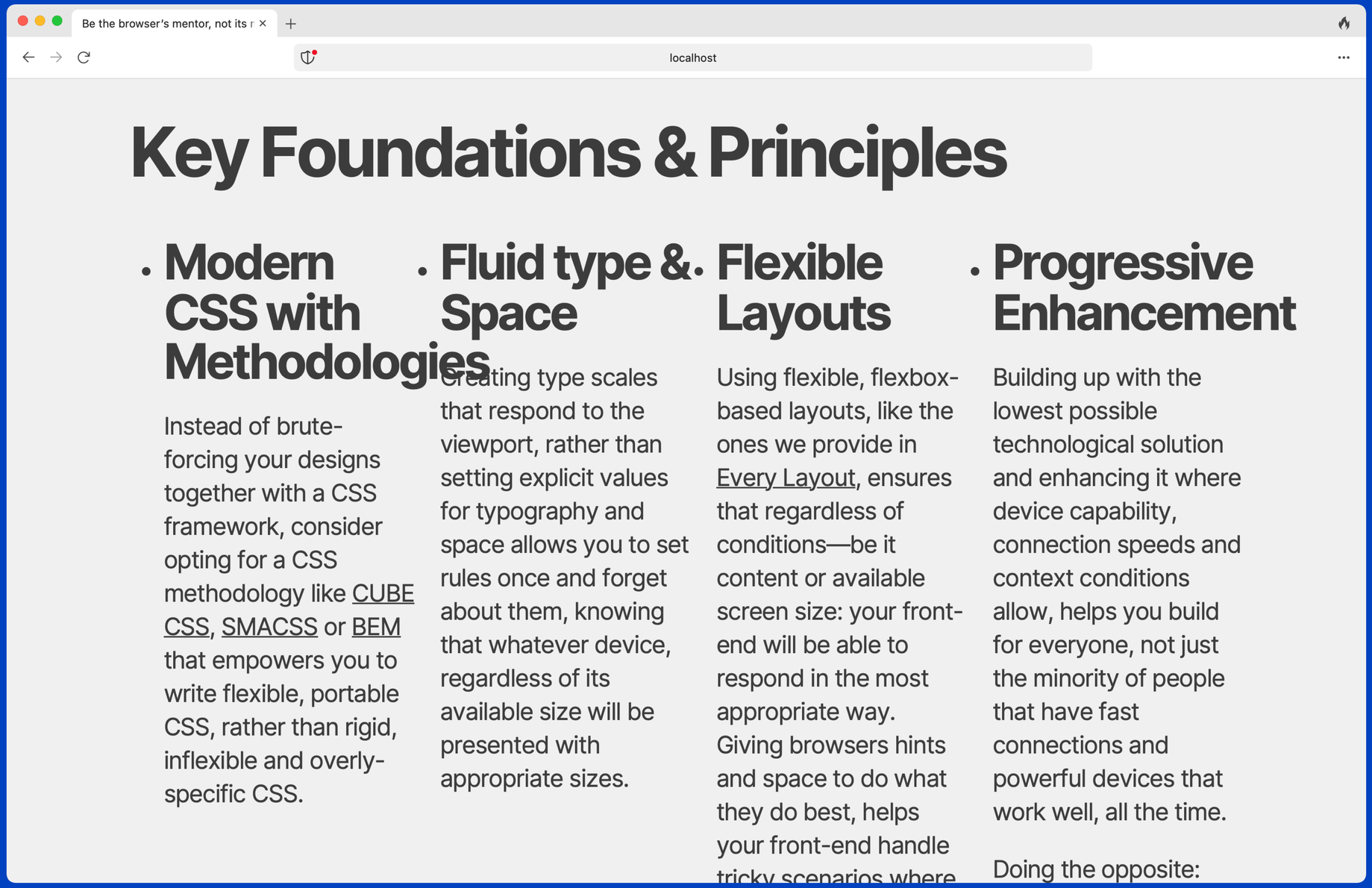
every-layout.dev

40% discount with WDC2022


Be the browser's mentor, not its
buildexcellentwebsit.es
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
Progressive Enhancement
Building up with the lowest possible technological solution and enhancing it

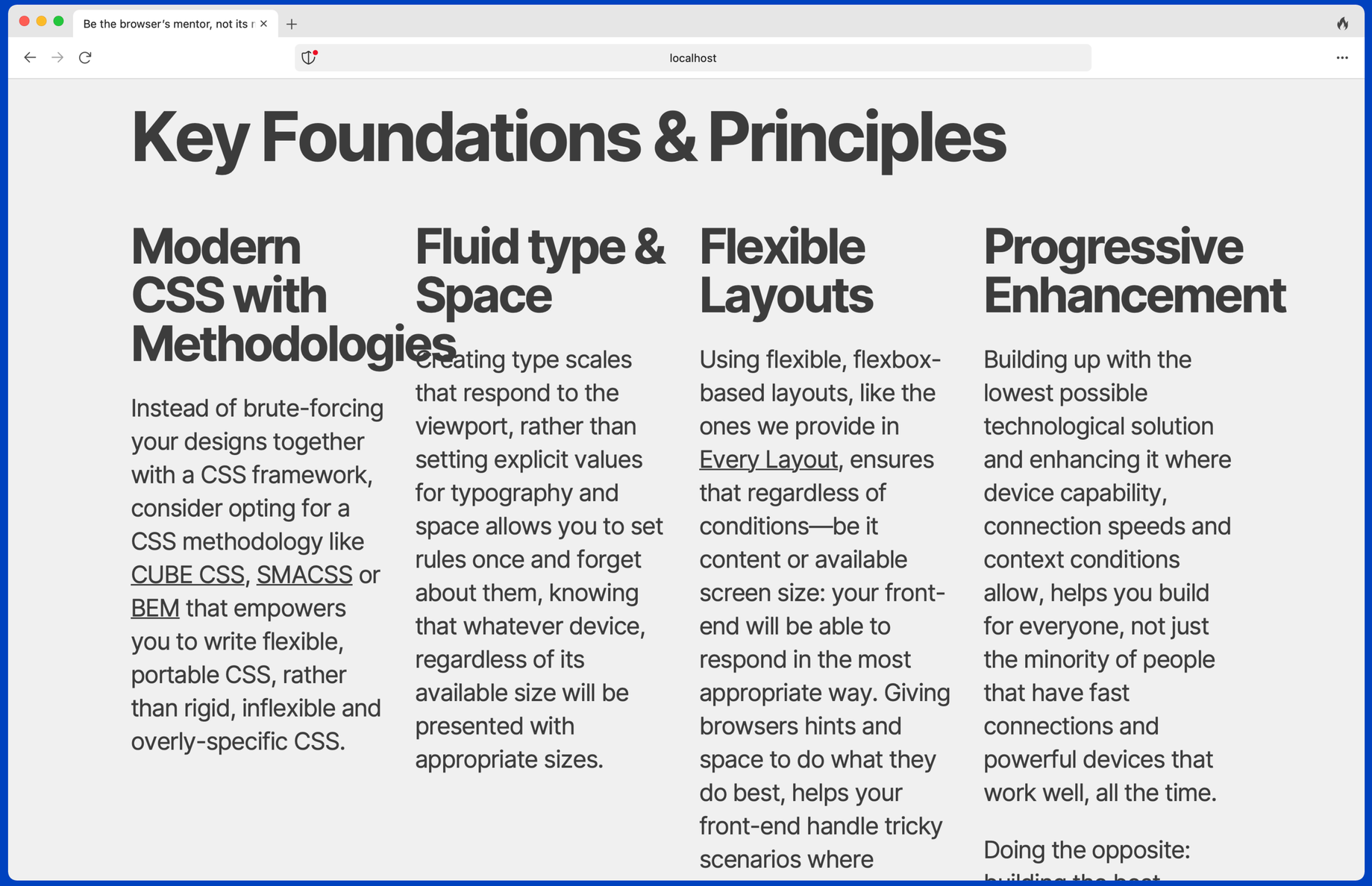

Be the browser's mentor, not its r
buildexcellentwebsit.es
CUBE CSS
A CSS methodology that's orientated towards simplicity, pragmatism and consistency.
Utopia
A handy collection of generators and tools that let you build up various fluid type and space scales depending on viewport sizes to help with responsive design.
EVERY LAYOUT
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.

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

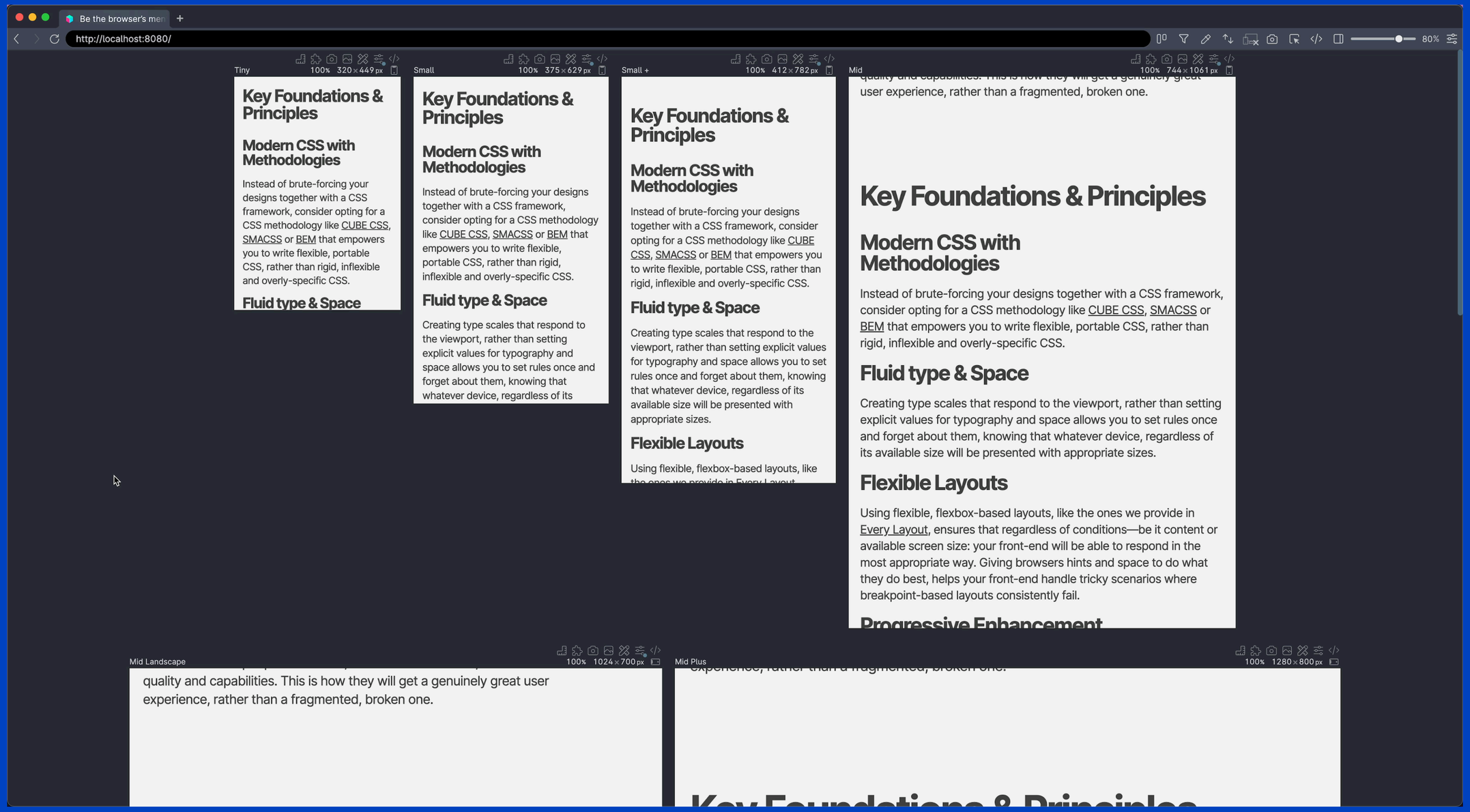

Be the browser's mentor, not its
localhost
Key Foundations & Principles
Modern CSS with Methodologies
Instead of brute-forcing your designs together with a CSS framework, consider opting for a CSS methodology like CUBE CSS, SMACSS or BEM that empowers you to write flexible, portable CSS, rather than 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:


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


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


Be the browser's mentor, not its r
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:

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


Be the browser's men
http://localhost:8080/
80%
Tiny
100% 320 × 449 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
Small
100% 375 × 629 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
Small +
100% 412 × 782 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 available size will be presented with appropriate sizes.
Flexible Layouts
Using flexible, flexbox-based layouts, like
Mid
100% 744 × 1061 px
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
Mid Landscape
100% 1024 × 700 px
quality and capabilities. This is how they will get a genuinely great user experience, rather than a fragmented, broken one.
Mid Plus
100% 1280 × 800 px

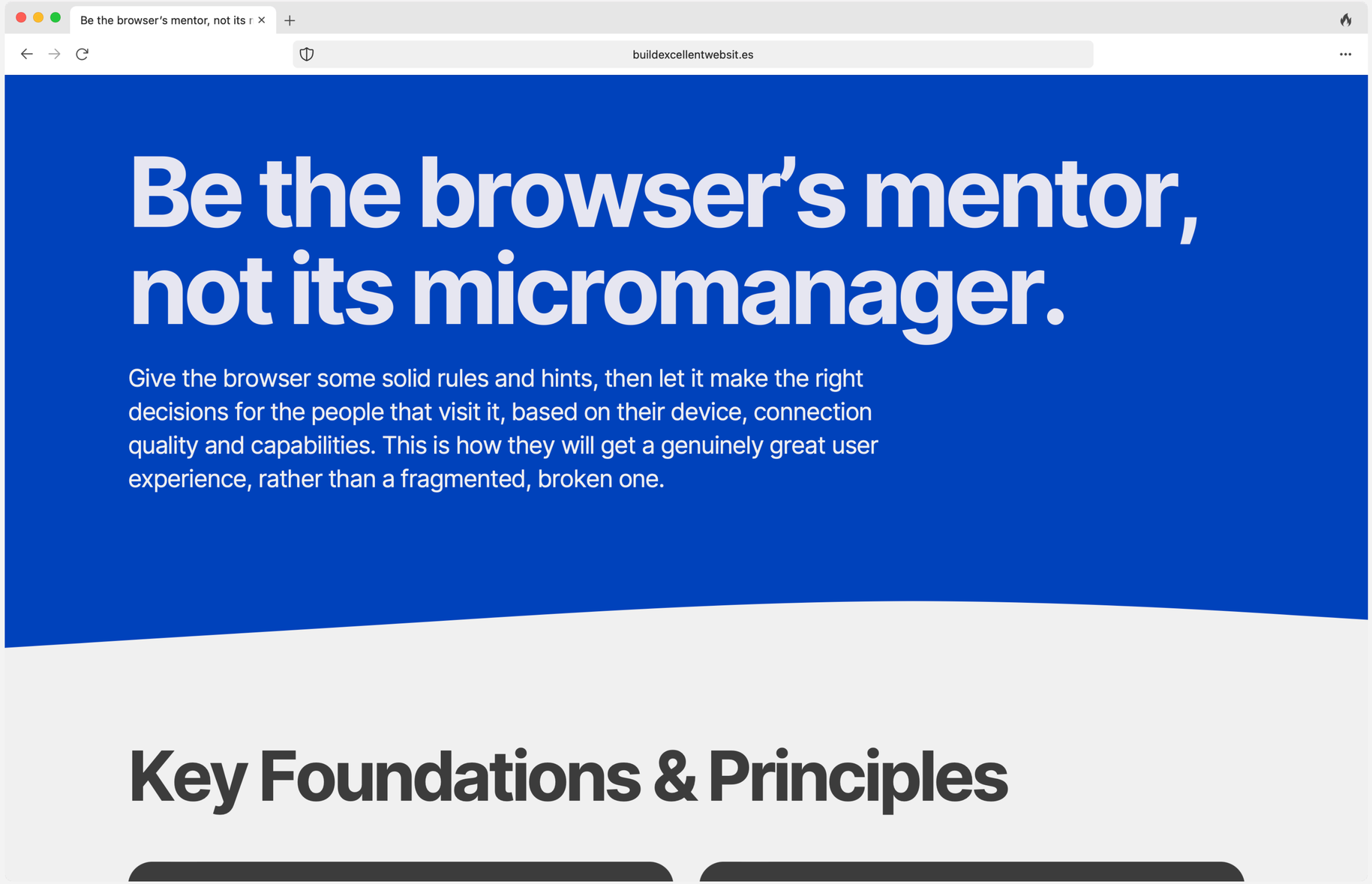
Global CSS

Composition

Utilities

Blocks

Exceptions


Be the browser’s mentor, not its r
buildexcellentwebsit.es
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 <svg class="curve" preserveAspectRatio="none"
  viewBox="0 0 1440 96" fill="none"
  xmlns="http://www.w3.org/2000/svg">
2   <path d="m0 96 80-5.3C160 85 320 75 480 64s320-21
  480-21.3c160 .3 320 10.3 400 16l80 5.3V0H0v96Z"/>
  </svg>
```

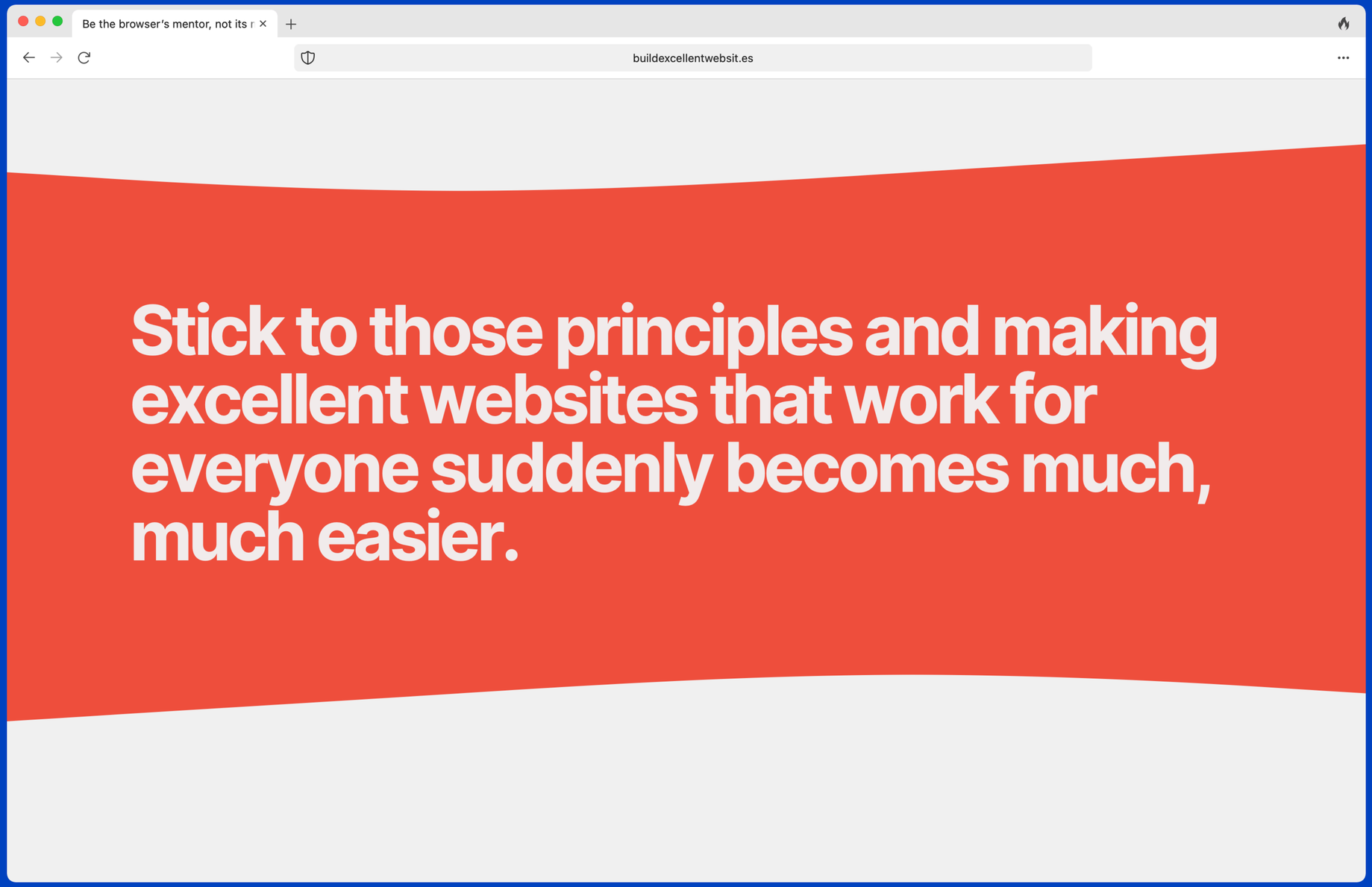
```css
.curve {
  display: block;
  height: 3.5em;
  width: 100%;
  fill: var(--spot-color, var(--color-light));
}
```

```css
.section > .curve:first-child {
  transform: rotate(180deg);
}

.section__inner {
  background: var(--spot-color, var(--color-dark));
  color: var(--color-light);
}

.section blockquote {
  font-weight: 700;
  line-height: 1;
  font-size: var(--size-step-4);
  letter-spacing: var(--tracking-s);
}

.section :is(h1, h2, h3, blockquote) {
  opacity: 95%;
}
```


Be the browser's mentor, not its
buildexcellentwebsit.es
Stick to those principles and making excellent websites that work for everyone suddenly becomes much, much easier.

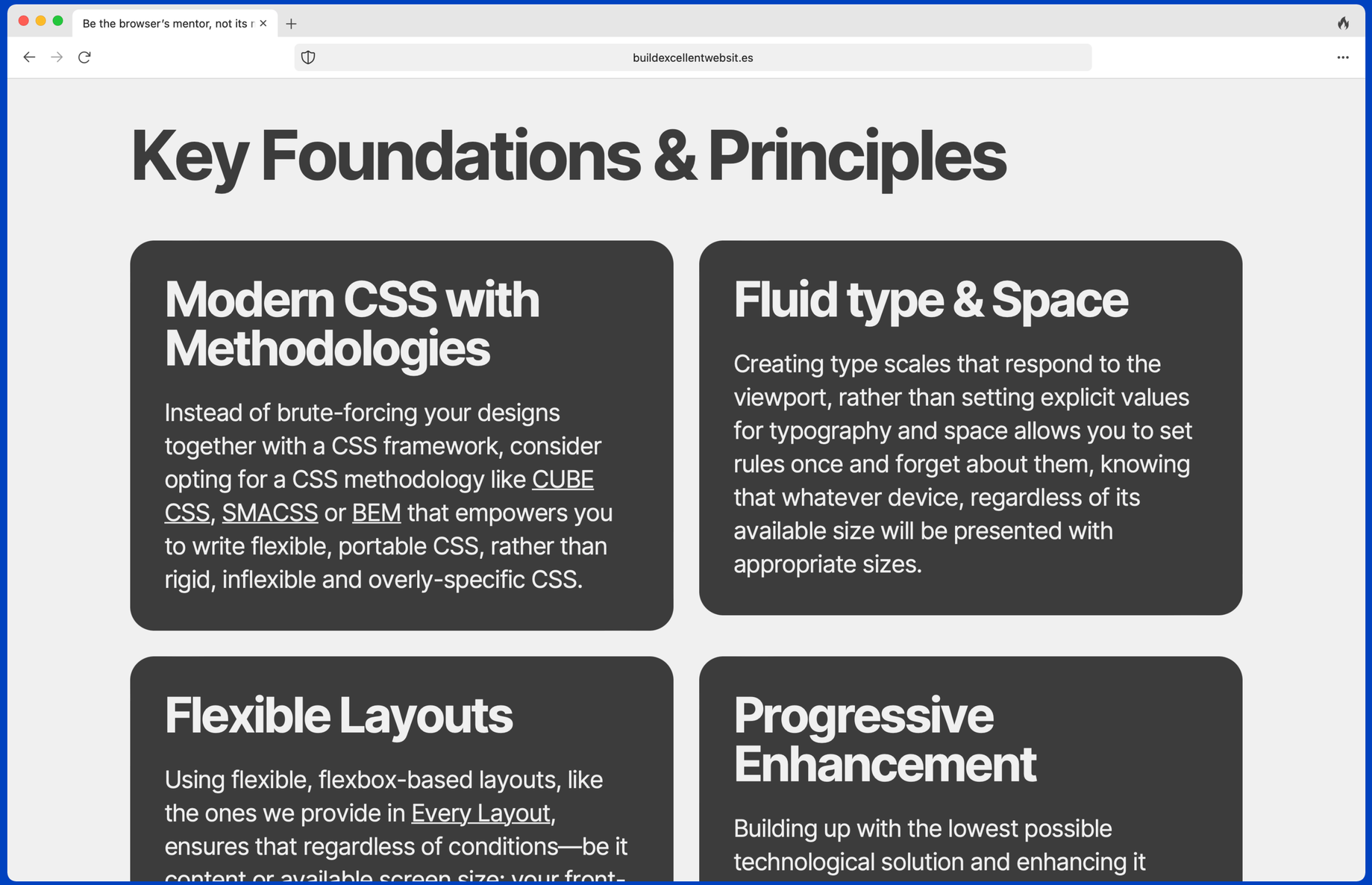
```
plugin(function ({addUtilities, config}) {
  const currentConfig = config();
  const customUtilities = [
    {key: 'spacing', prefix: 'flow-space', property: '--flow-space'},
    {key: 'colors', prefix: 'spot-color', property: '--spot-color'}
  ];

  customUtilities.forEach(({key, prefix, property}) => {
    const group = currentConfig.theme[key];

    if (!group) {
      return;
    }

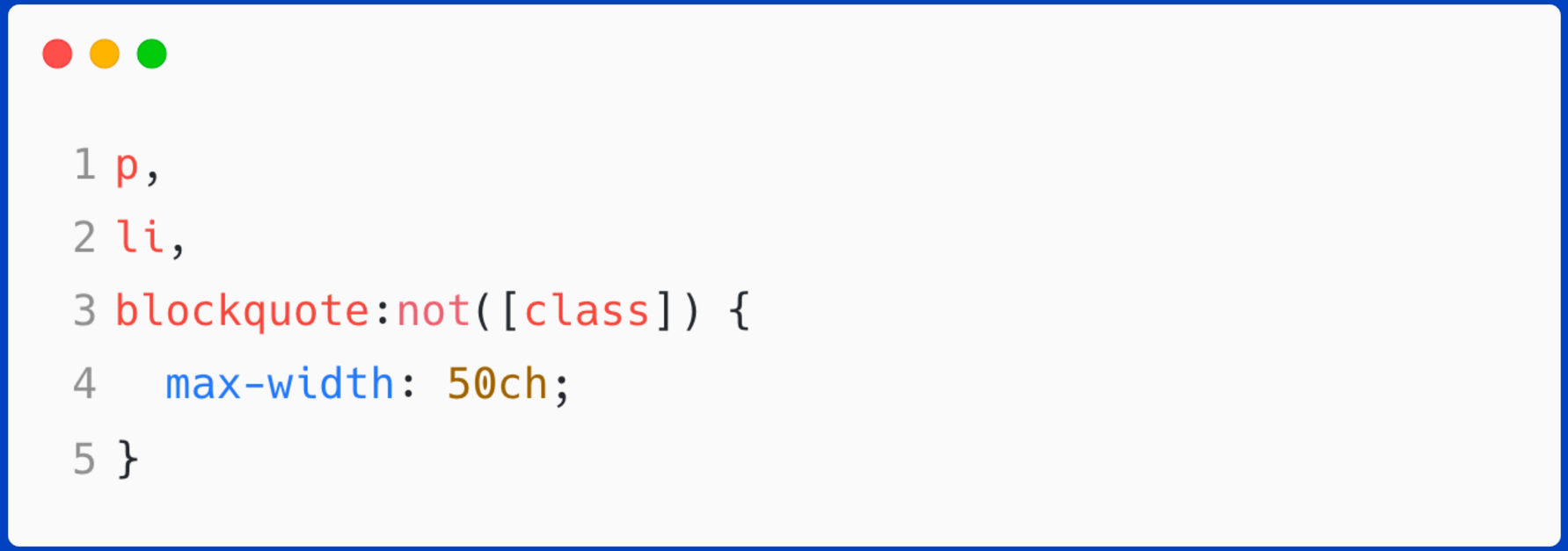
    Object.keys(group).forEach(key => {
      addUtilities({
        [`.${prefix}-${key}`]: postcssJs.objectify(
          postcss.parse(`${property}: ${group[key]}`)
        )
      });
    });
  })
})
```

```
.spot-color-secondary {
    --spot-color: var(--color-secondary);
}
```


Be the browser's mentor, not its
buildexcellentwebsit.es
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
Progressive Enhancement
Building up with the lowest possible technological solution and enhancing it



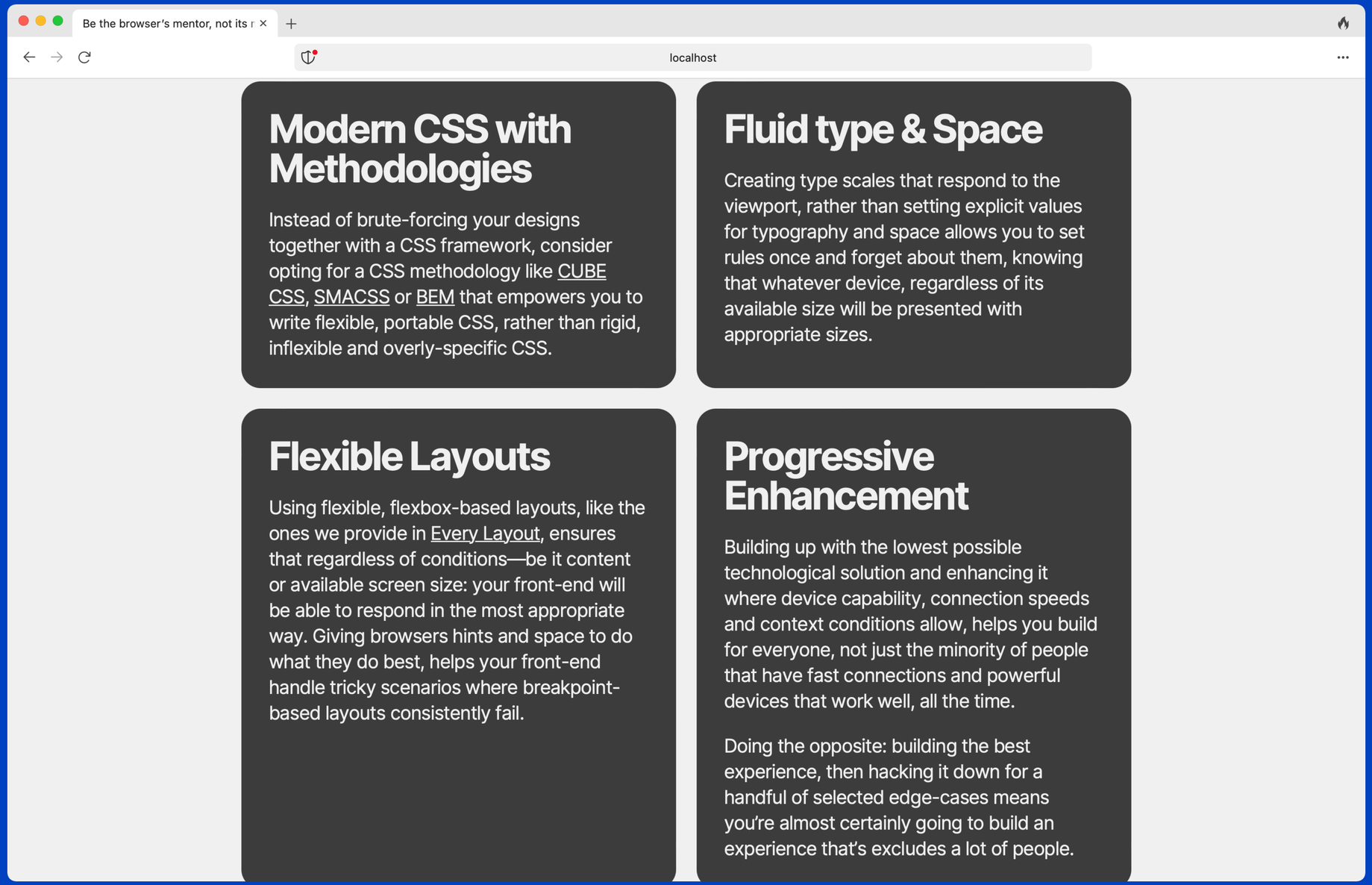
```css
.card {
  background: var(--color-dark);
  color: var(--color-light);
  padding: var(--space-m-l);
  border-radius: var(--border-radius);
  max-width: unset;
}
```



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

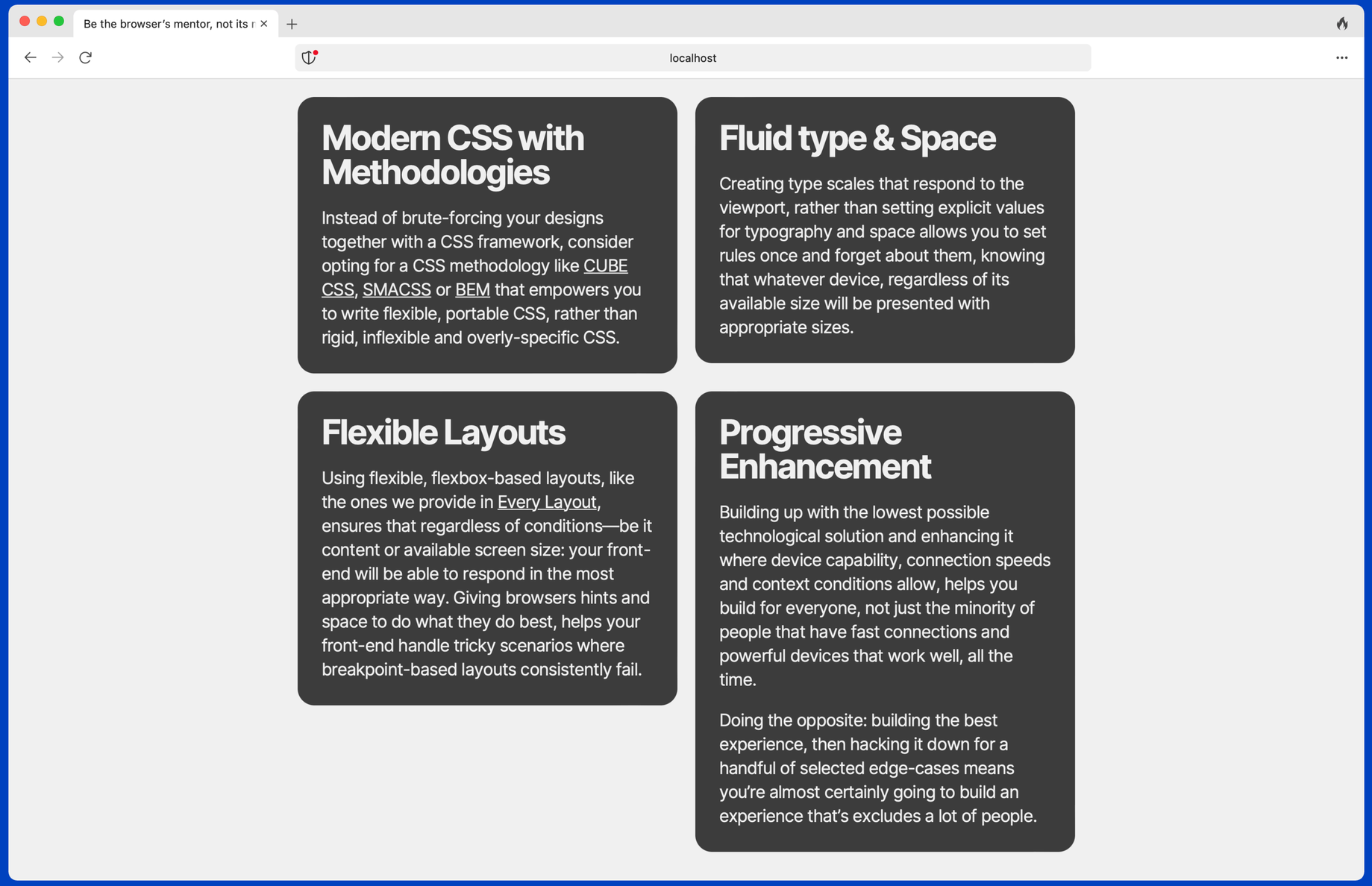


```css
.card {
  background: var(--color-dark);
  color: var(--color-light);
  padding: var(--space-m-l);
  border-radius: var(--border-radius);
  max-width: unset;
}
```


Be the browser's mentor, not its r
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.

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

```css
.grid[data-rows='masonry'] {
  grid-template-rows: masonry;
  align-items: start;
}
```


Be the browser's mentor, not its r
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.



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.

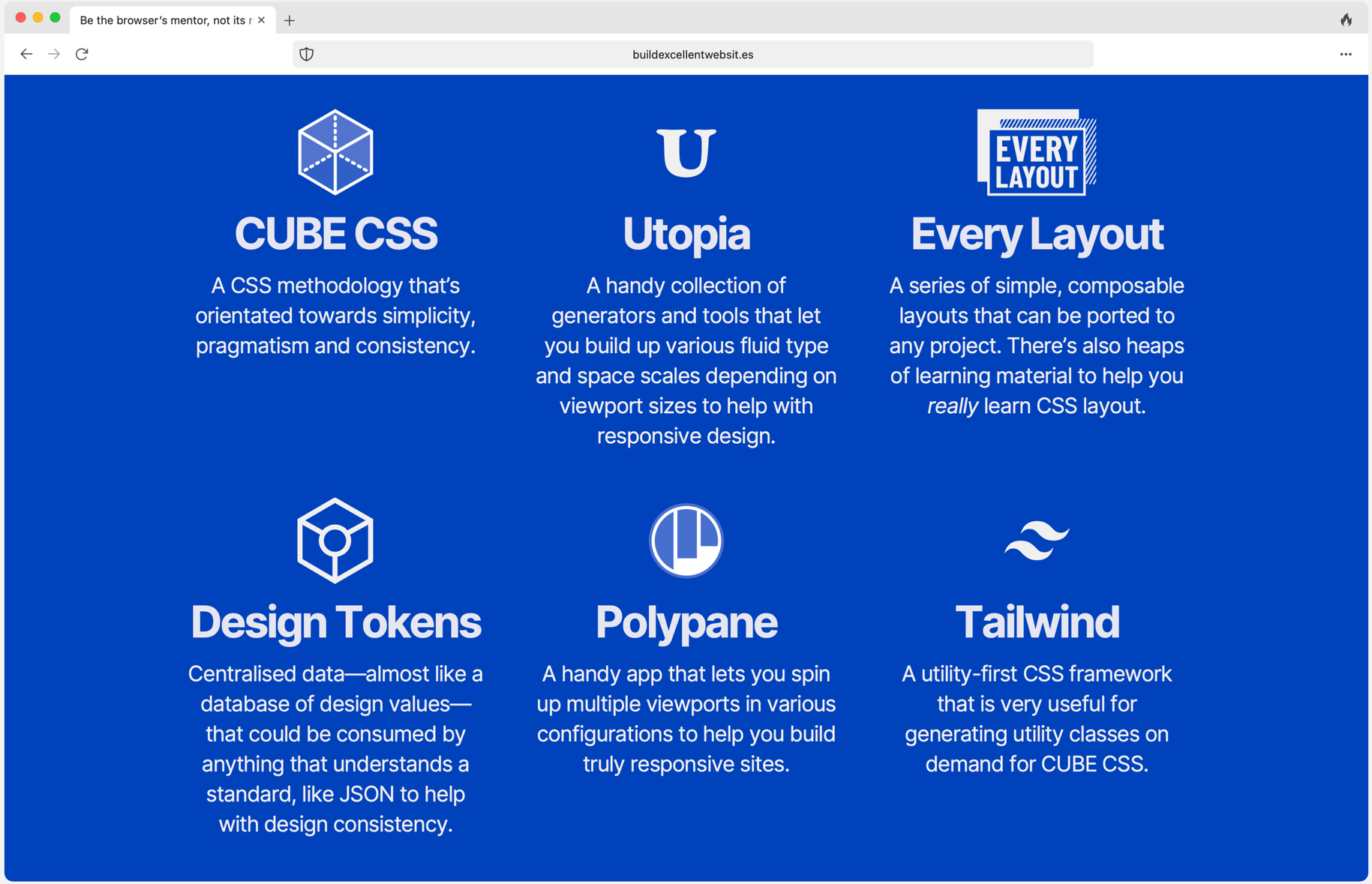
Progressive Enhancement

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 r
buildexcellentwebsit.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
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.

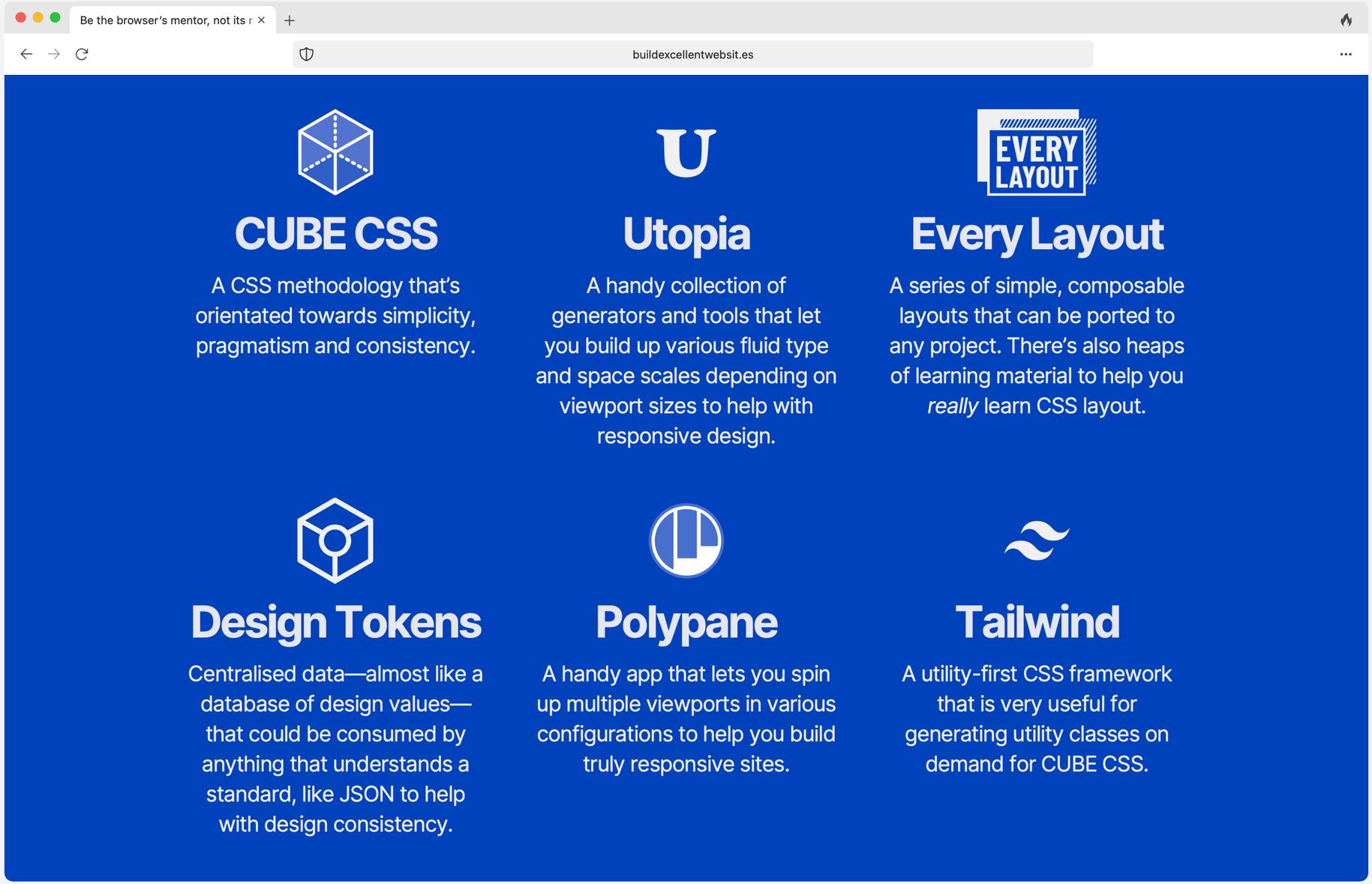
```css
.features {
  --grid-placement: auto-fit;
  --grid-min-item-size: clamp(16rem, 33%, 20rem);
  --gutter: var(--space-l-xl);
  --flow-space: var(--space-s);

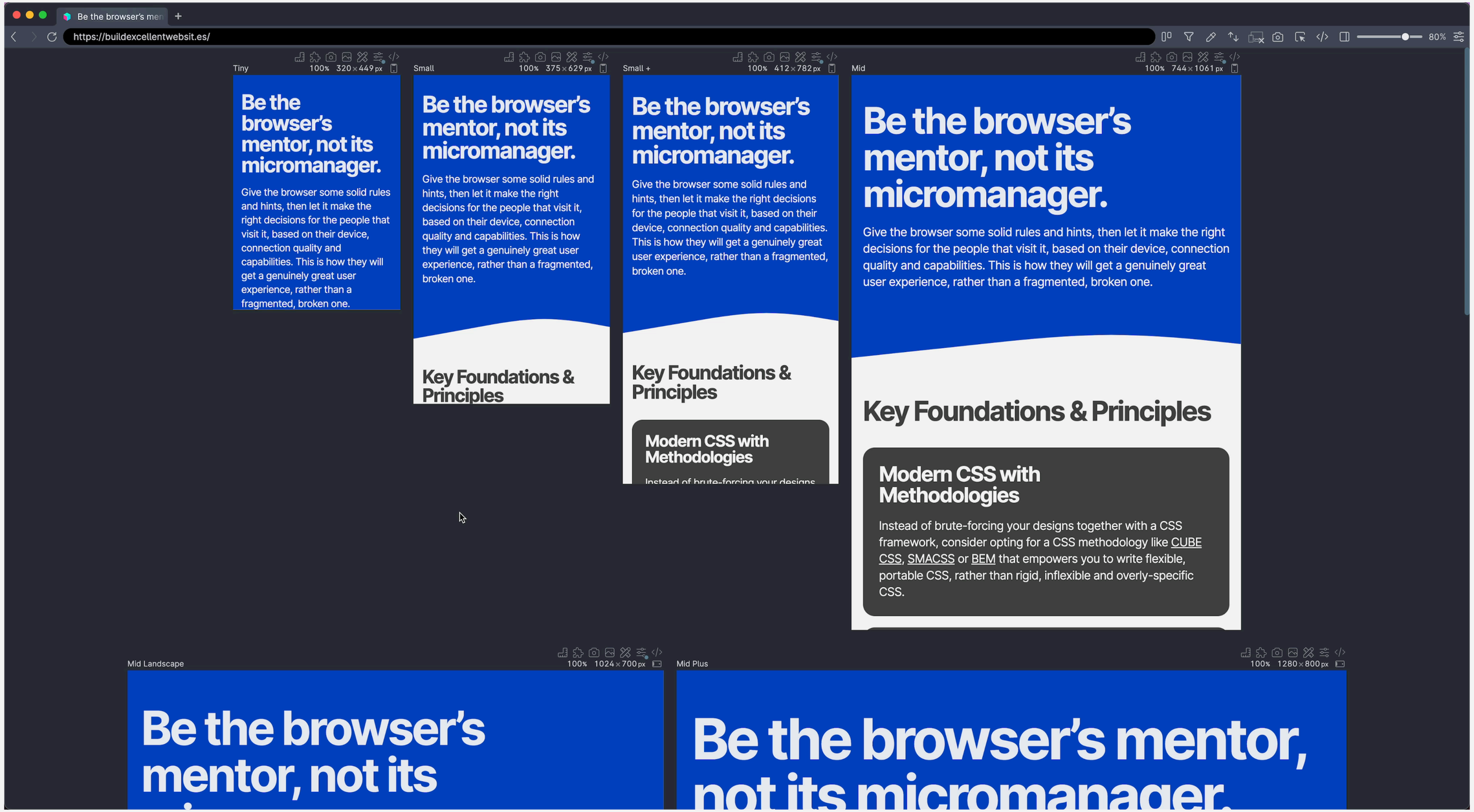
  text-align: center;
}
```

```css
.features svg {
  display: block;
  margin-inline: auto;
  height: 4em;
}

.features a {
  text-decoration: none;
}

.features a:hover {
  text-decoration: underline;
  text-decoration-thickness: 0.08ex;
  text-underline-offset: 0.2ex;
}
```


Be the browser's mentor, not its r
buildexcellentwebsit.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
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.


Be the browser's men
https://buildexcellentwebsit.es/
80%
Tiny
100% 320 × 449 px
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.
Small
100% 375 × 629 px
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
Small +
100% 412 × 782 px
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
Mid
100% 744 × 1061 px
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.
Mid Landscape
100% 1024 × 700 px
Be the browser's mentor, not its
Mid Plus
100% 1280 × 800 px
Be the browser's mentor, not its micromanager.



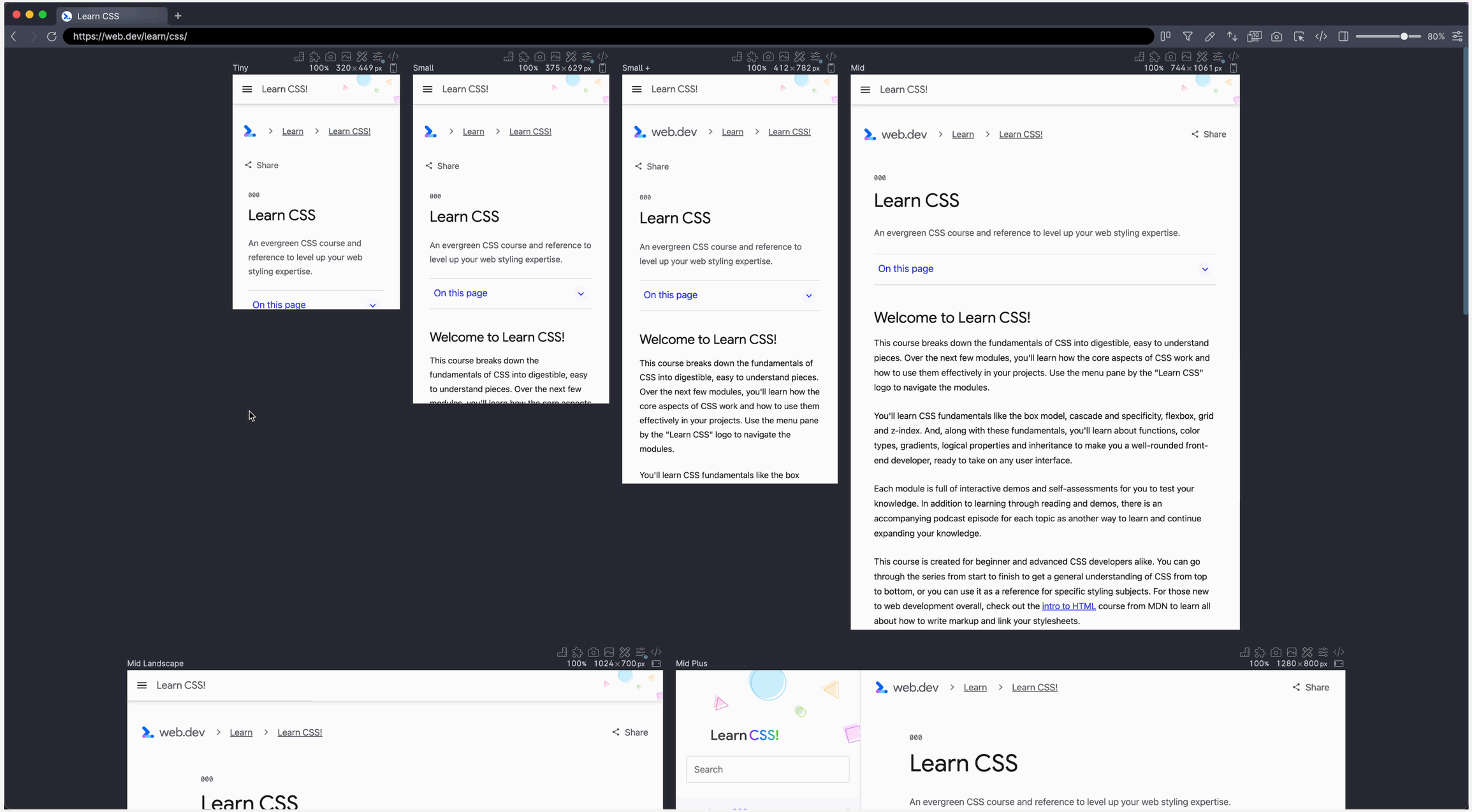
https://glitch.com/edit/#!/build-excellent-websites

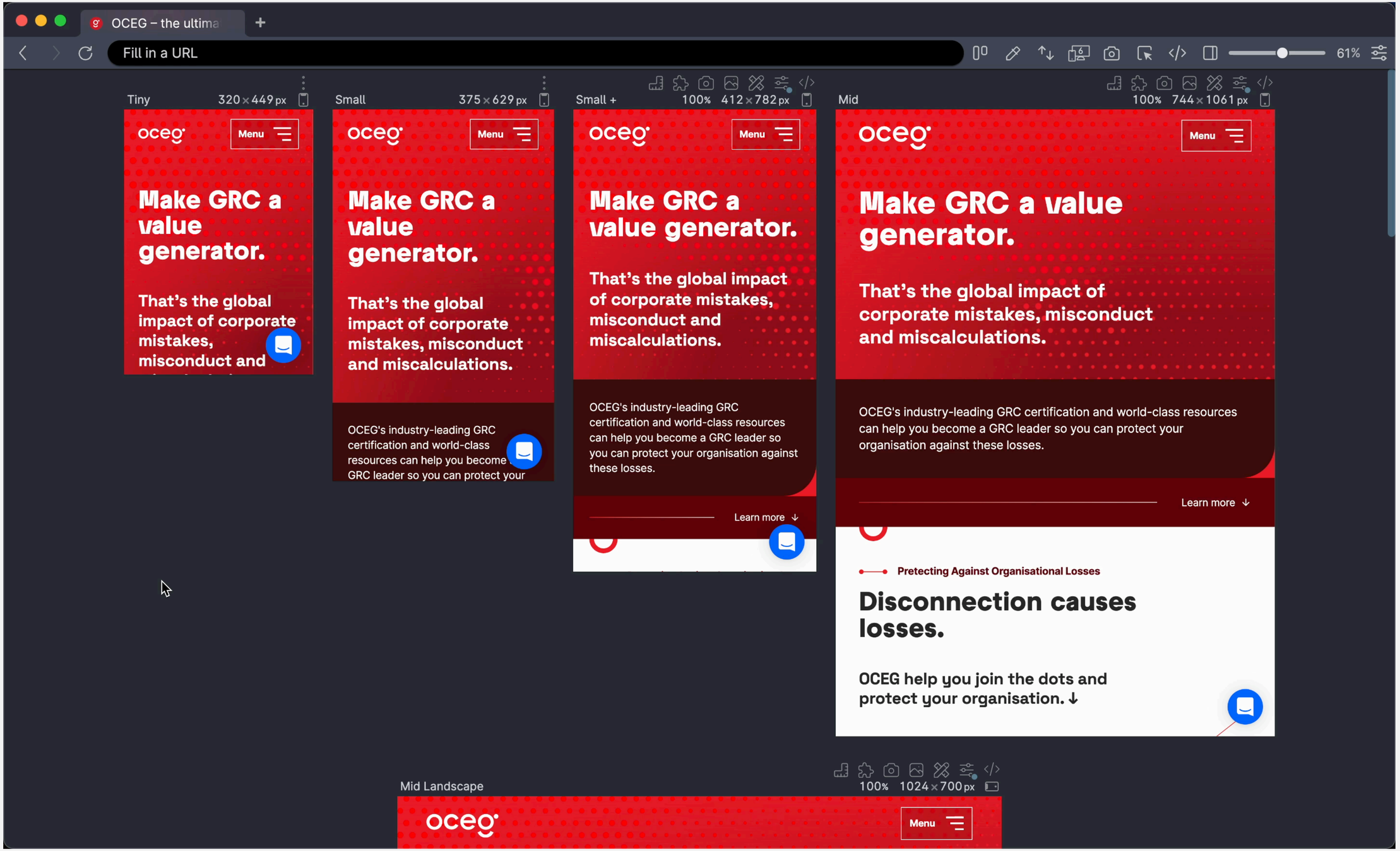
We’ve never had it better with browsers

~~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


Learn CSS
https://web.dev/learn/css/
80%
Tiny
100% 320 × 449 px
Learn CSS!
Learn
Learn CSS!
Share
000
Learn CSS
An evergreen CSS course and reference to level up your web styling expertise.
On this page
Small
100% 375 × 629 px
Learn CSS!
Learn
Learn CSS!
Share
000
Learn CSS
An evergreen CSS course and reference to level up your web styling expertise.
On this page
Welcome to Learn CSS!
This course breaks down the fundamentals of CSS into digestible, easy to understand pieces. Over the next few
Small +
100% 412 × 782 px
Learn CSS!
web.dev
Learn
Learn CSS!
Share
000
Learn CSS
An evergreen CSS course and reference to level up your web styling expertise.
On this page
Welcome to Learn CSS!
This course breaks down the fundamentals of CSS into digestible, easy to understand pieces. Over the next few modules, you'll learn how the core aspects of CSS work and how to use them effectively in your projects. Use the menu pane by the "Learn CSS" logo to navigate the modules.
You'll learn CSS fundamentals like the box
Mid
100% 744 × 1061 px
Learn CSS!
web.dev
Learn
Learn CSS!
Share
000
Learn CSS
An evergreen CSS course and reference to level up your web styling expertise.
On this page
Welcome to Learn CSS!
This course breaks down the fundamentals of CSS into digestible, easy to understand pieces. Over the next few modules, you'll learn how the core aspects of CSS work and how to use them effectively in your projects. Use the menu pane by the "Learn CSS" logo to navigate the modules.
You'll learn CSS fundamentals like the box model, cascade and specificity, flexbox, grid and z-index. And, along with these fundamentals, you'll learn about functions, color types, gradients, logical properties and inheritance to make you a well-rounded front-end developer, ready to take on any user interface.
Each module is full of interactive demos and self-assessments for you to test your knowledge. In addition to learning through reading and demos, there is an accompanying podcast episode for each topic as another way to learn and continue expanding your knowledge.
This course is created for beginner and advanced CSS developers alike. You can go through the series from start to finish to get a general understanding of CSS from top to bottom, or you can use it as a reference for specific styling subjects. For those new to web development overall, check out the intro to HTML course from MDN to learn all about how to write markup and link your stylesheets.
Mid Landscape
100% 1024 × 700 px
Learn CSS!
web.dev
Learn
Learn CSS!
Share
000
Learn CSS
Mid Plus
100% 1280 × 800 px
Learn CSS!
Search
web.dev
Learn
Learn CSS!
Share
000
Learn CSS
An evergreen CSS course and reference to level up your web styling expertise.


OCEG – the ultima
Fill in a URL
61%
Tiny
320 × 449 px
Small
375 × 629 px
Small +
100% 412 × 782 px
Mid
100% 744 × 1061 px
oceg
Menu
Make GRC a value generator.
That's the global impact of corporate mistakes, misconduct and miscalculations.
OCEG's industry-leading GRC certification and world-class resources can help you become a GRC leader so you can protect your organisation against these losses.
Learn more
Pretecting Against Organisational Losses
Disconnection causes losses.
OCEG help you join the dots and protect your organisation.
Mid Landscape
100% 1024 × 700 px

⚠️ CONTENT WARNING ⚠️

Glitching and flashing

A Human Future // Ho

https://ahumanfuture.co

80%

Tiny 100% 320 × 449 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]

Small 100% 375 × 629 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]



Small + 100% 412 × 782 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]



Mid 100% 744 × 1061 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]

Launch Terminal >



Mid Landscape 100% 1024 × 700 px

A Human Future 2022 Play Audio

Menu [3]

A Human Future

Mid Plus 100% 1280 × 800 px

A Human Future 2022 Play Audio

Menu [3]

A Human Future

Go forth and build excellent websites

Thank you

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

https://swop.link/set-studio