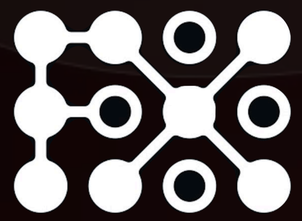


F(X) TEAM · 午夜识堂

CSS新特性 @大漠#2021-05-12#杭州

FED × 掘金 × F(X)TEAM 联合出品

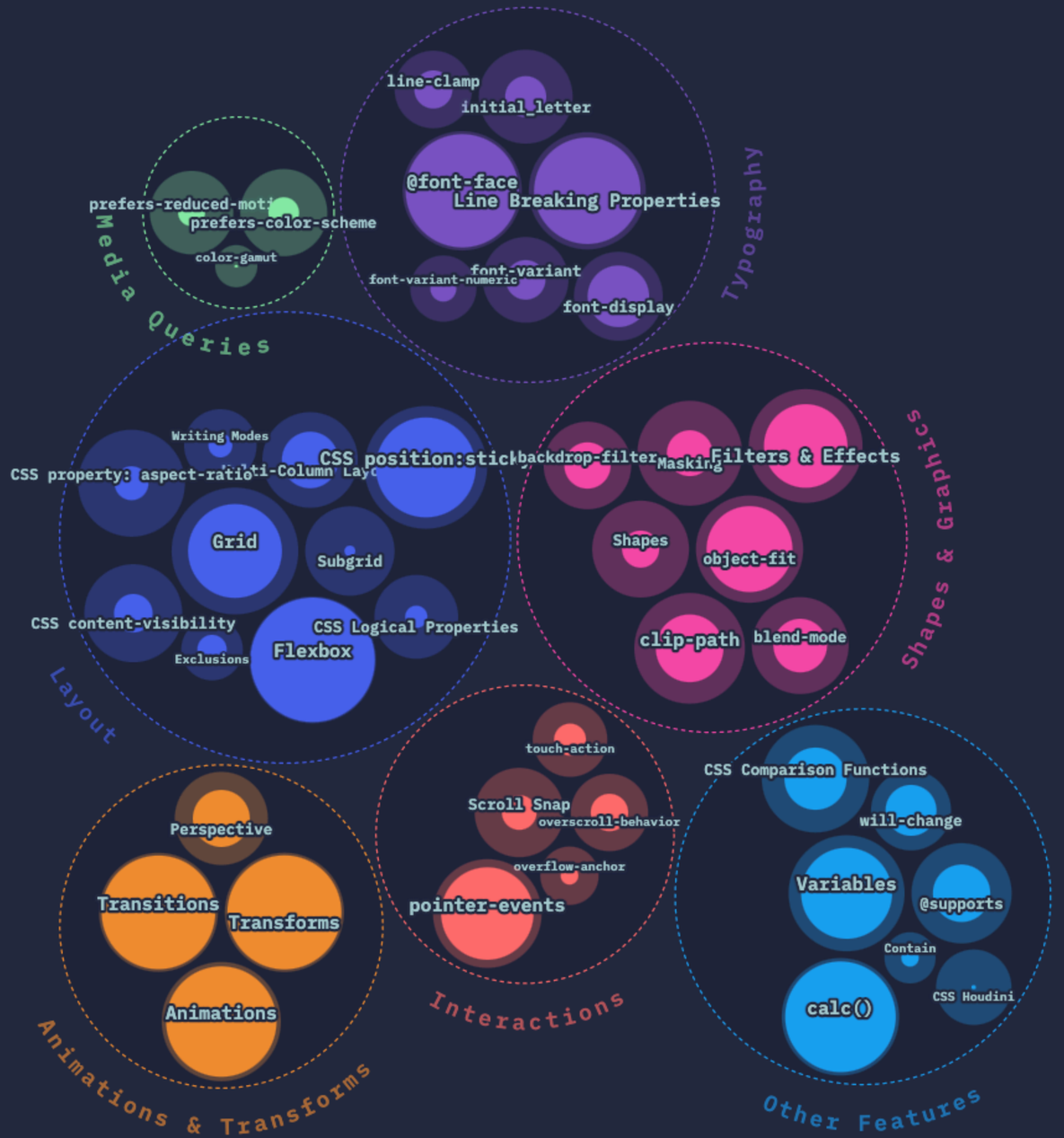
CSS is simple, but not easy!

SIMPLE ≠ EASY

...BUT!

CSS IS AWESOME

CSS新特性


line-clamp
initial_letter
@font-face
Line Breaking Properties
font-variant
font-variant-numeric
font-display
Typography
prefers-reduced-moti
prefers-color-scheme
color-gamut
Media Queries
Writing Modes
CSS position:stick
CSS property: aspect-ratio
Grid
Subgrid
CSS content-visibility
CSS Logical Properties
Exclusions
Flexbox
Layout
backdrop-filter
Maskin
Filters & Effects
Shapes
object-fit
clip-path
blend-mode
Shapes & Graphics
touch-action
Scroll Snap
overscroll-behavior
overflow-anchor
pointer-events
Interactions
CSS Comparison Functions
will-change
Variables
@supports
Contain
CSS Houdini
calc()
Other Features
Perspective
Transitions
Transforms
Animations
Animations & Transforms

CSS新特性

① CSS伪类选择器
② CSS颜色
③ CSS背景
④ CSS蒙层和剪切
⑤ CSS混合模式
⑥ CSS自定义属性
⑦ CSS等比缩放
⑧ CSS滚动捕捉
⑨ CSS Gap(沟槽)
⑩ CSS逻辑属性
⑪ CSS媒体查询
⑫ CSS比较函数
⑬ CSS内容可见性
⑭ CSS外在尺寸和内在尺寸
⑮ CSS的Display

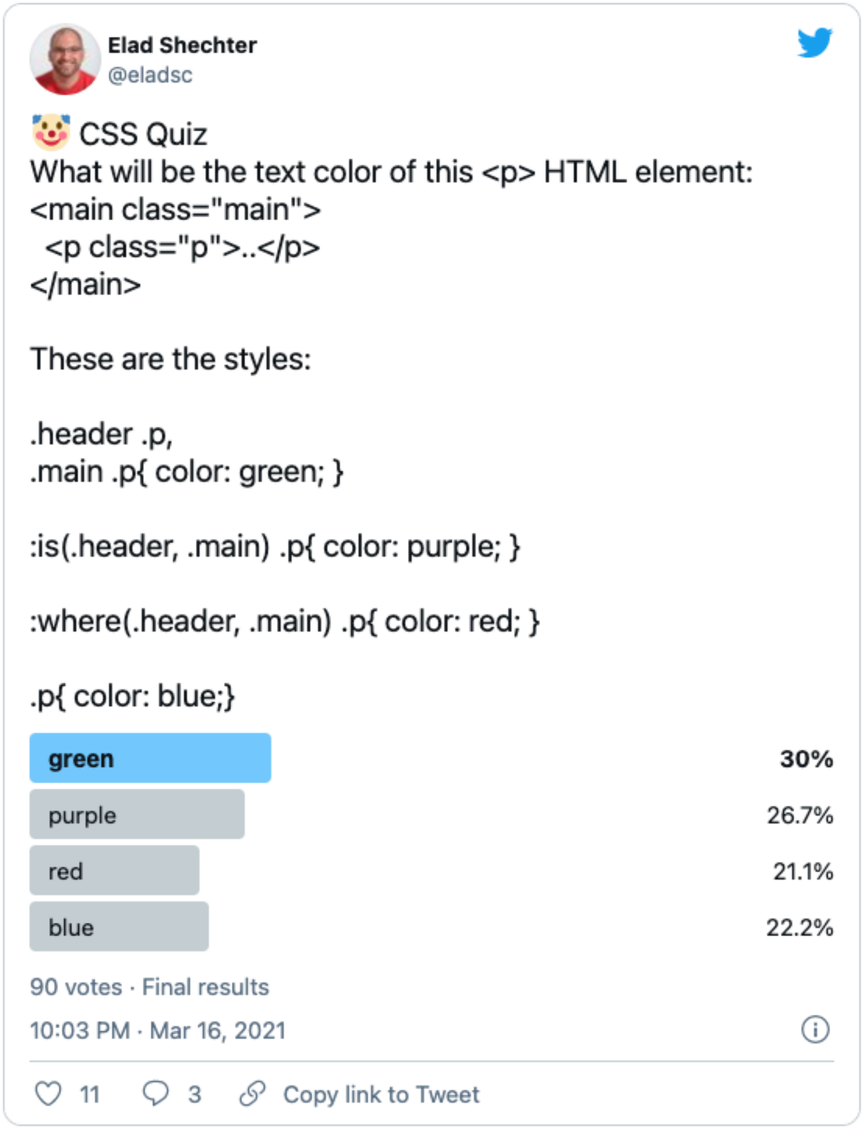
CSS新特性

① CSS伪类选择器
② CSS颜色
③ CSS背景
④ CSS蒙层和剪切
⑤ CSS混合模式
⑥ CSS自定义属性
⑦ CSS等比缩放
⑧ CSS滚动捕捉
⑨ CSS Gap(沟槽)
⑩ CSS逻辑属性
⑪ CSS媒体查询
⑫ CSS比较函数
⑬ CSS内容可见性
⑭ CSS外在尺寸和内在尺寸
⑮ CSS的Display

CSS 伪类选择器

① :is() vs. :where()
② :not() vs. :has()
③ :empty vs. :blank
④ :focus-visible vs. :focus-within

CSS 伪类选择器 — :is() vs. :where()

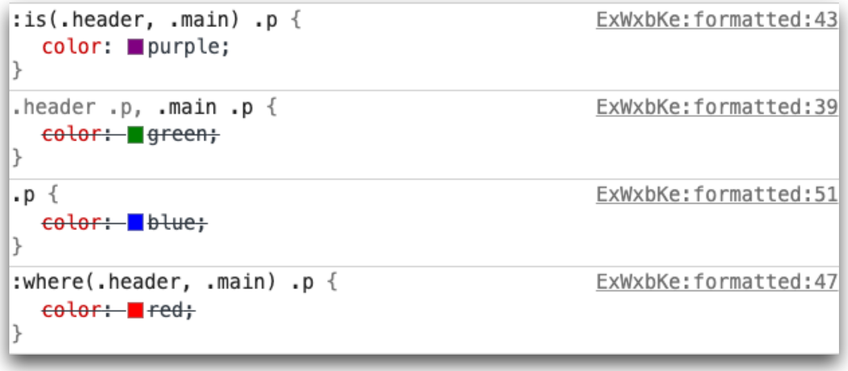


Tweet →| CodePen →|

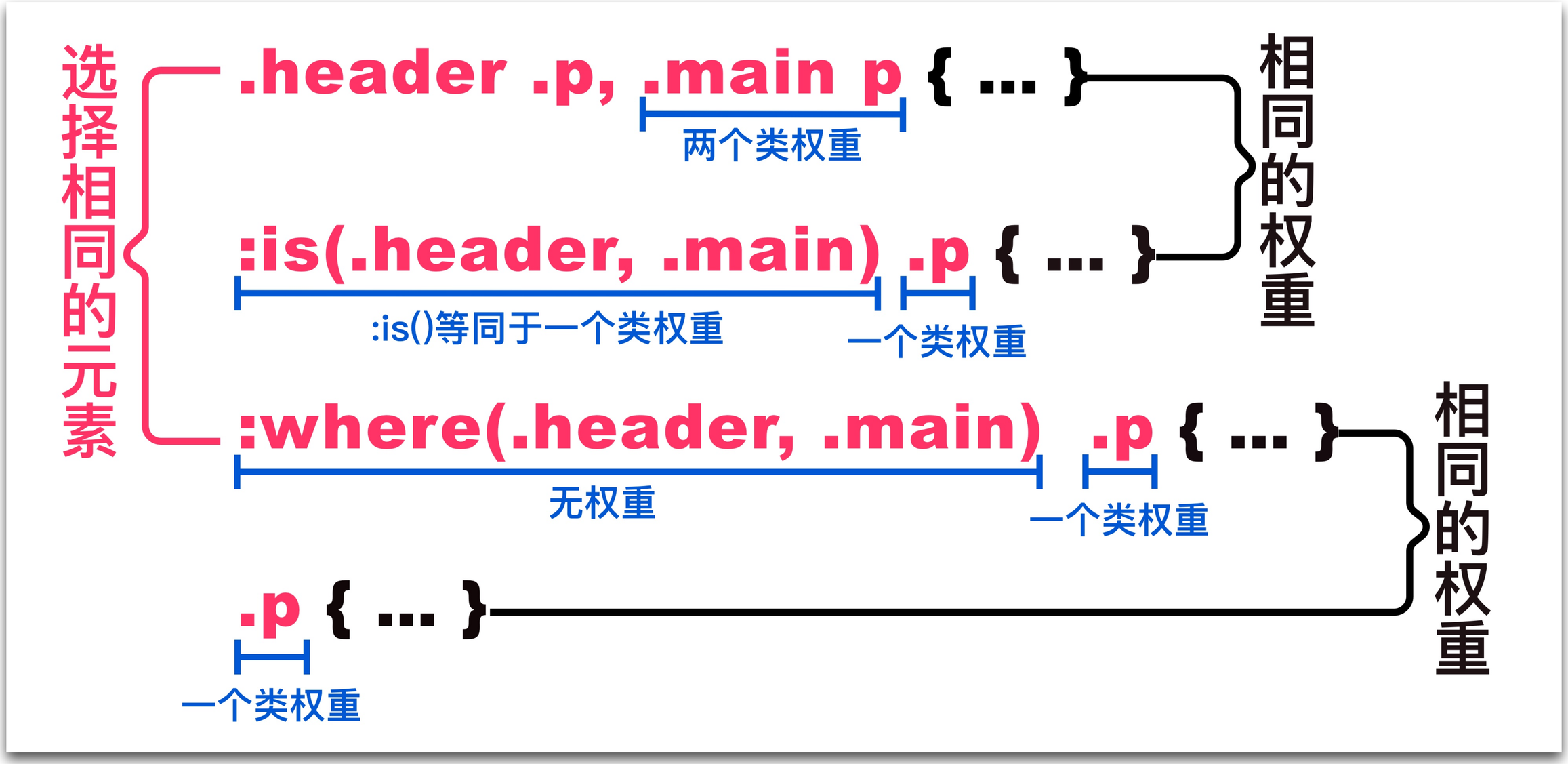
CSS 伪类选择器 — :is() vs. :where()

Modern CSS Pseudo-Class Selectors

```
/* Before */
.header p,
.main p {
    color: purple;
}

/* After */
:is(.header, .main) .p {
  color: purple;
}
```



CSS 伪类选择器 — :is() vs. :where()



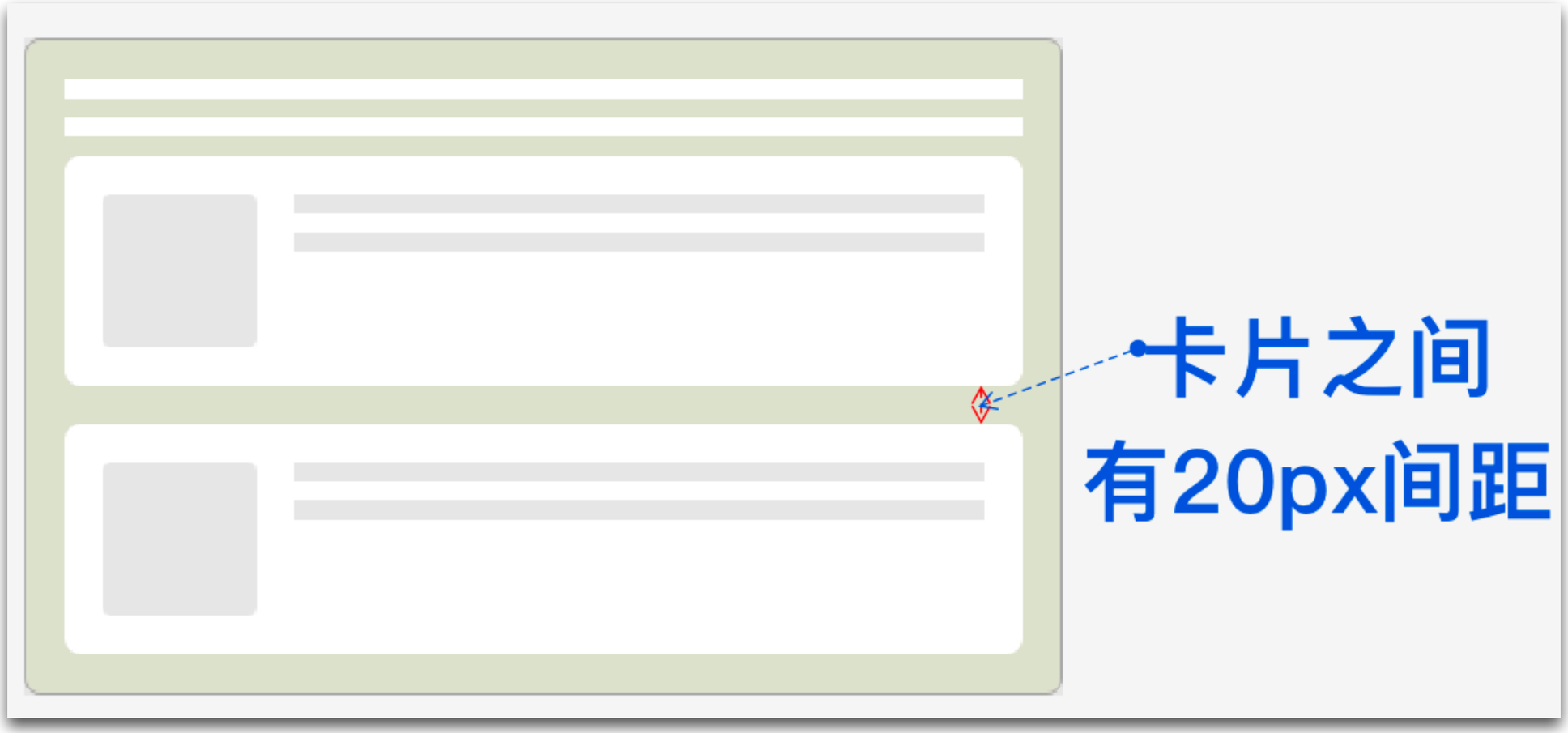
CSS 伪类选择器 — :is() vs. :where()

```html
<main class="main">
  <p class="p">What is the text color?</p>
</main>
```

```css
:where(.header, .main) .p {
  color: red;
}

.p {
  color: blue;
}

:is(.header, .main) .p {
  color: purple;
}

.header .p,
.main .p {
  color: green;
}
```

CodePen →

CSS 伪类选择器 — :not() vs. :has()



CSS 伪类选择器 — :not() vs. :has()

```css
/* Before */
li + li {
    margin-top: 20px;
}

// 或
li {
    margin-bottom: 20px;
}

li:last-child {
    margin-bottom: 0
}
```

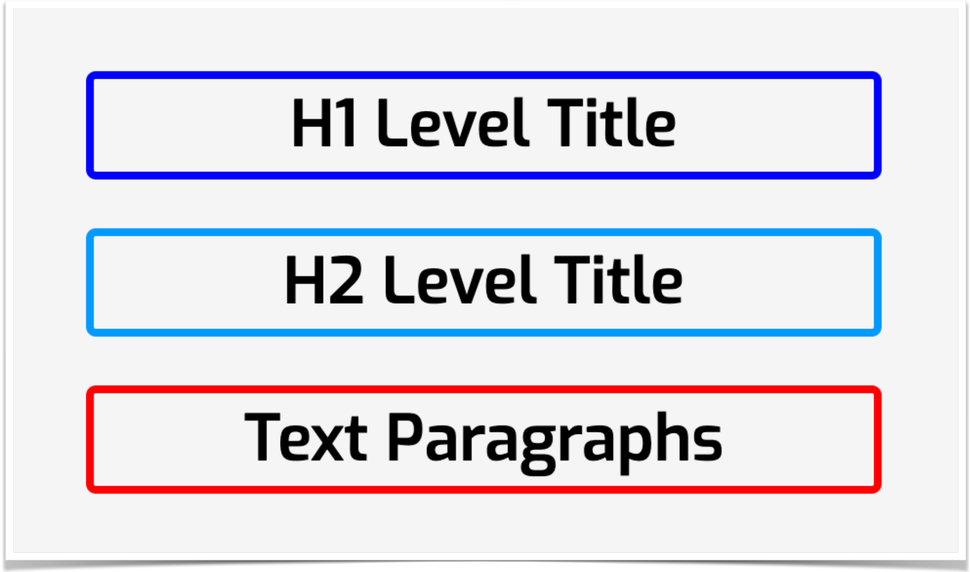
```css
/* After */
li:not(:last-child) {
    margin-top: 20px;
}
```

CodePen ⇥

CSS 伪类选择器 — :not() vs. :has()

```html
<!-- HTML -->
<section>
  <h1>H1 Level Title</h1>
</section>
<section>
  <h2>H2 Level Title</h2>
</section>
<section>
  <p>Text Paragraphs</p>
</section>
```

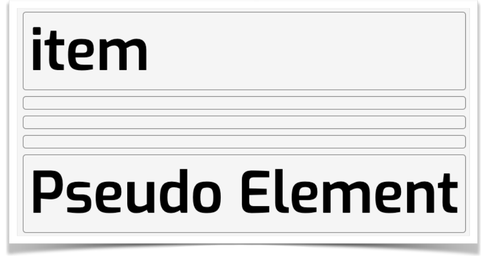
```css
/* CSS */
section:not(:has(p)) {
  margin-bottom: 30px;
}

section:has(p) {
  border-color: red;
}

section:has(h1) {
  border-color: blue;
}

section:has(h2) {
  border-color: #09f;
}
```



CSS 伪类选择器 — :empty vs. :blank

```html
<!-- HTML -->
<ul>
  <li>item</li>
  <li> </li><!-- 中间有下空格 -->
  <li><!-- 这里有一个注释 --></li>
  <li></li>
  <li></li>
</ul>
```

```css
/* CSS */
li {
  padding: 10px;
  font-size: 5vw;
  border: 1px solid rgb(0 0 0 / 0.5);
  border-radius: 5px;
}

li:last-child::after {
  content: "Pseudo Element";
}
```



CSS 伪类选择器 — :empty vs. :blank

```css
/* CSS */
li:empty {
  border: 0 !important;
  clip: rect(1px, 1px, 1px, 1px) !important;
  clip-path: inset(50%) !important;
  height: 1px !important;
  margin: -1px !important;
  overflow: hidden !important;
  padding: 0 !important;
  position: absolute !important;
  width: 1px !important;
  white-space: nowrap !important;
}
```

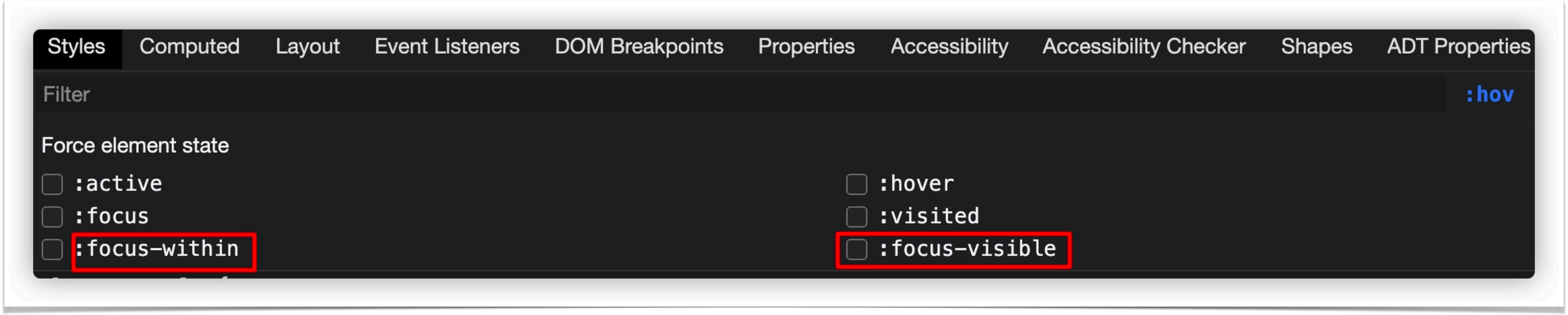
CodePen →

CSS 伪类选择器 — :empty vs. :blank

```css
/* CSS */
li:blank {
  border: 0 !important;
  clip: rect(1px, 1px, 1px, 1px) !important;
  clip-path: inset(50%) !important;
  height: 1px !important;
  margin: -1px !important;
  overflow: hidden !important;
  padding: 0 !important;
  position: absolute !important;
  width: 1px !important;
  white-space: nowrap !important;
}
```

CodePen ⇥

CSS 伪类选择器 — :focus-visible vs. :focus-within



CSS 伪类选择器 — :focus-visible vs. :focus-within

```css
button:focus:not(:focus-visible) {
  outline: 2px dotted #416dea;
  outline-offset: 2px;
  box-shadow: 0px 1px 1px #416dea;
}

button:focus-visible {
  outline: 2px solid #416dea;
  outline-offset: 2px;
  box-shadow: 0px 1px 1px #416dea;
}
```

CodePen →

CSS 伪类选择器 — :focus-visible vs. :focus-within

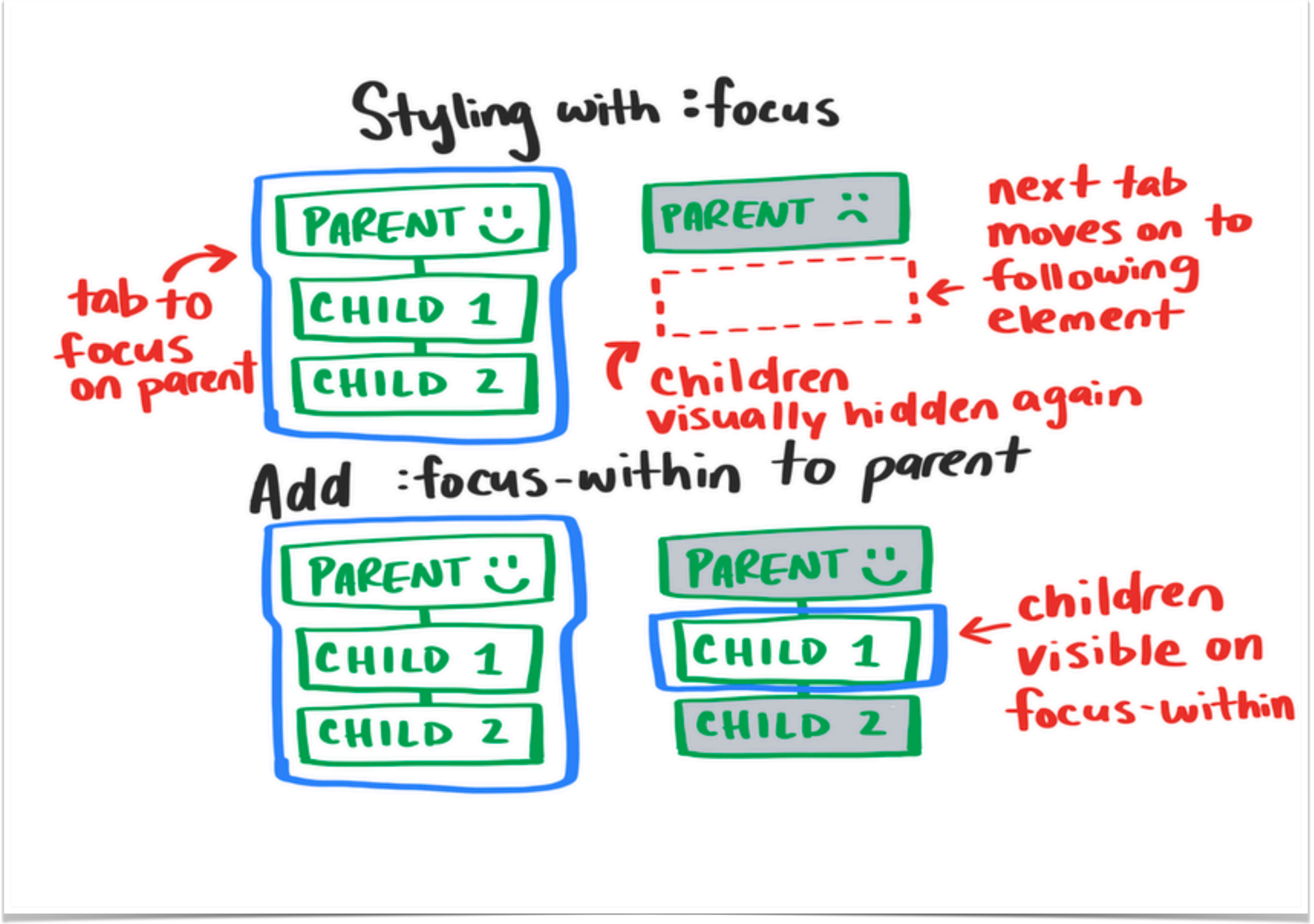
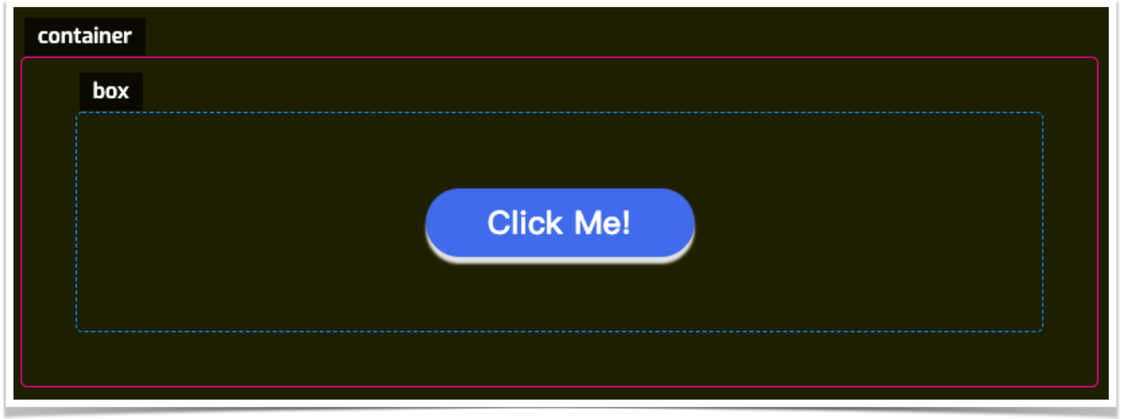


Image URL ⇥

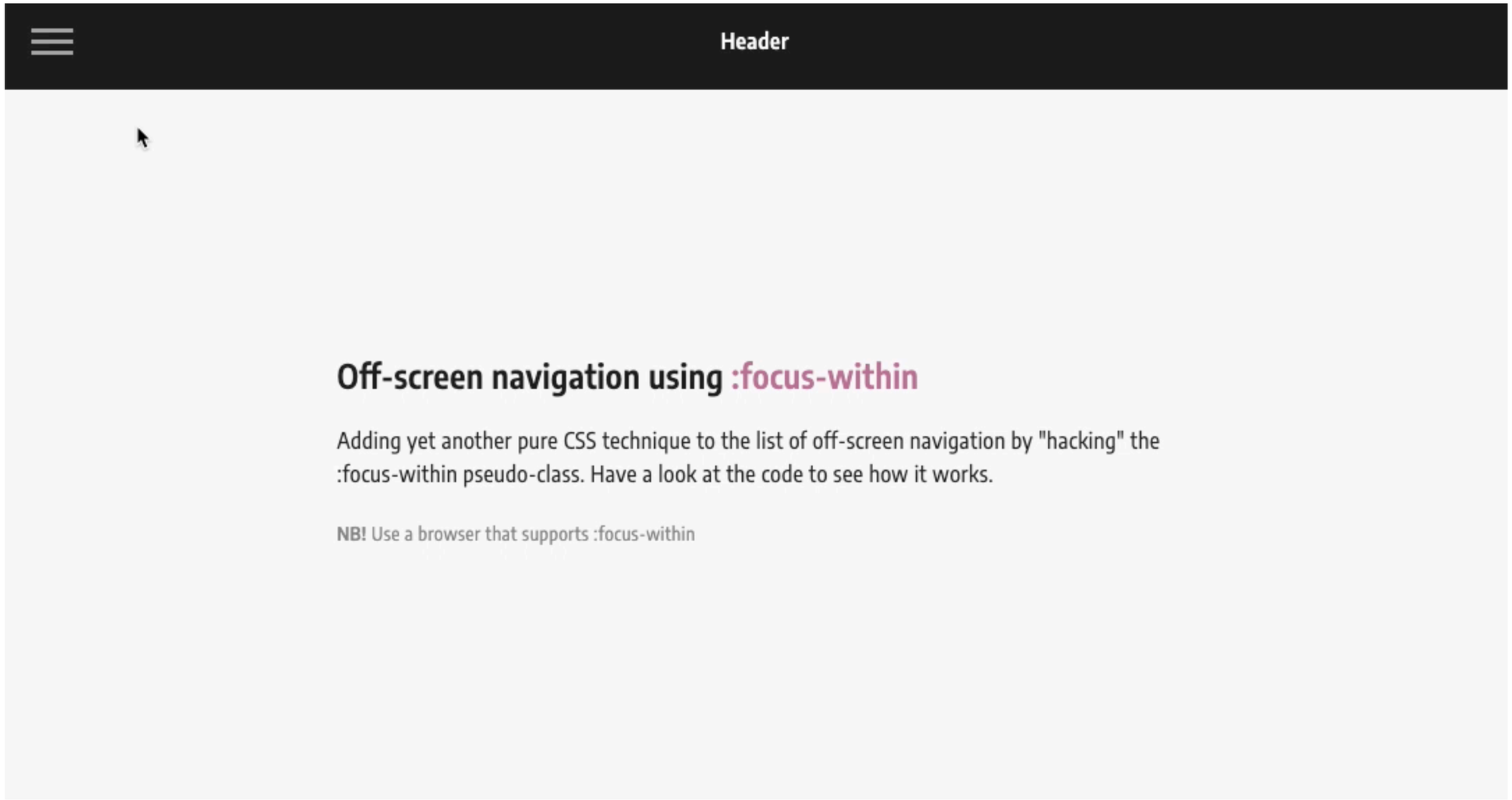
CSS 伪类选择器 — :focus-visible vs. :focus-within

```css
.box:focus-within,
.container:focus-within {
  box-shadow: 0 0 5px 3px rgba(255, 255, 255, 0.65);
}
```



Codepen →

CSS 伪类选择器 — :focus-visible vs. :focus-within



Codepen →

CSS新特性

~~① CSS伪类选择器~~

② CSS颜色

③ CSS背景

④ CSS蒙层和剪切

⑤ CSS混合模式

⑥ CSS自定义属性

⑦ CSS等比缩放

⑧ CSS滚动捕捉

⑨ CSS Gap(沟槽)

⑩ CSS逻辑属性

⑪ CSS媒体查询

⑫ CSS比较函数

⑬ CSS内容可见性

⑭ CSS外在尺寸和内在尺寸

⑮ CSS的Display

CSS 颜色

属性（Property）	旧语法（Old Syntax）		新语法（New Syntax）	
	不带透明通道	带透明通道	不带透明通道	带透明通道
rgb()	rgb(255, 255, 255)	✘	rgb(255 255 255)	rgb(255 255 255 / .5)
rgba()	✘	rgba(255, 255, 255, .5)	rgba(255 255 255)	rgba(255 255 255 / .5)
hsl()	hsl(200, 30%, 30%)	✘	hsl(200 30% 30%)	hsl(200 30% 30% / .5)
hsla()	✘	hsla(200, 30%, 30%, .5)	hsla(200 30% 30%)	hsla(200 30% 30% / .5)
hwb()	✘	✘	hwb(300 40% 20%)	hwb(300 40% 20% / .8)
lab()	✘	✘	lab(29% 39 20)	lab(29% 39 20 / .5)
lch()	✘	✘	lch(29% 44 27)	lch(29% 44 27 / .5)

Codepen →

CSS 颜色

CSS Colors

```
1 rgb (255 255 255 / .5) = rgb (255, 255, 255, .5)
2 rgba(255 255 255)      = rgba(255, 255, 255)
3 rgba(255 255 255)      = rgba(255  255  255 / 1)
4 rgba(255 255 255 / .5) = rgba(255, 255, 255, .5)
5 hsl (200 30% 30% / .5) = hsl (200, 30%, 30%, .5)
6 hsla(200 30% 30%)      = hsla(200, 30%, 30%)
7 hsla(200 30% 30%)      = hsla(200  30%  30% / 1)
8 hsla(200 30% 30% / .5) = hsla(200, 30%, 30%, .5)
```

CSS 颜色

```
                                    keep same h and s
                                          ↓  ↓
background: hsl(from var(--theme-primary) h  s  30%);
                          ↑                      ↑
                   original color          change l
```

Primary

hsl(274, 61%, 50%)

Modified

hsl(274, 61%, 30%)

CSS 颜色

CSS Colors

```css
.bg-primary-100 {
  background-color: hsl(from var(--theme-primary) h s 90%);
}
.bg-primary-200 {
  background-color: hsl(from var(--theme-primary) h s 80%);
}
.bg-primary-300 {
  background-color: hsl(from var(--theme-primary) h s 70%);
}
```

CSS 颜色

```
color: white;
background-color: color(display-p3 0 1 0.052);
z-index: 1;
top: -30rem;
padding-top: 45rem;
margin-bottom: -30rem;
```

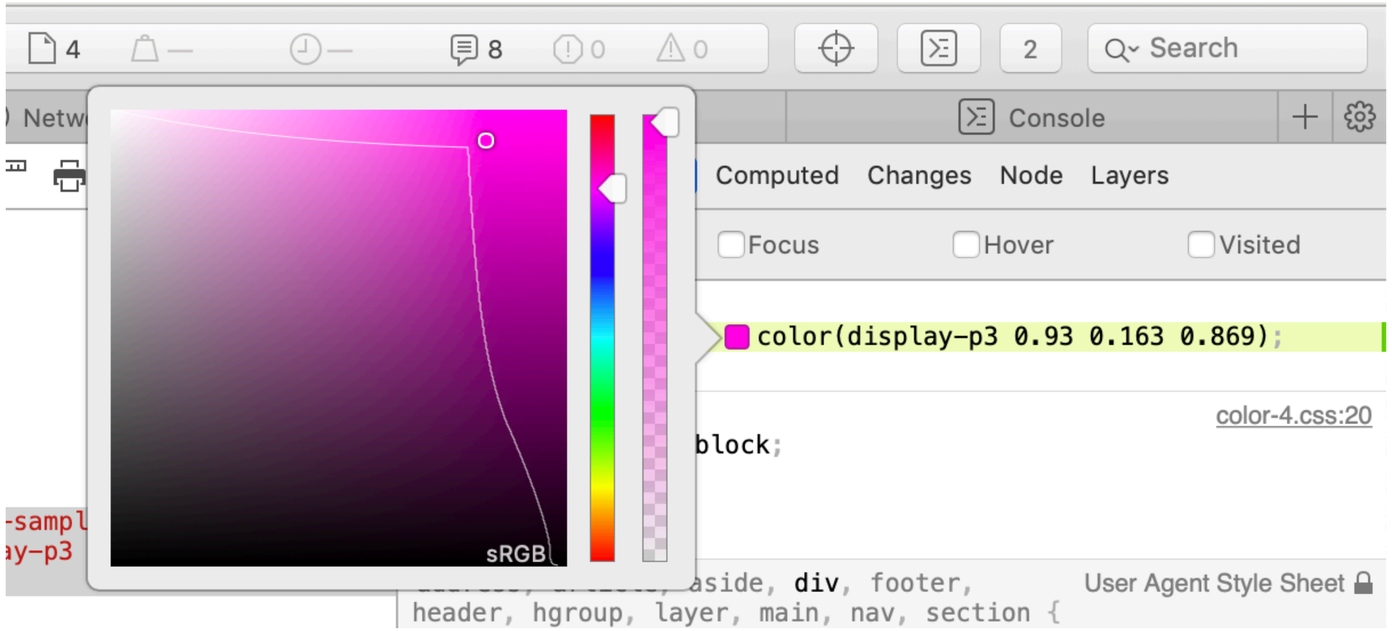
Clamp to sRGB

```
color: white;
background-color: hsl(203.6, 100%, 12%);
z-index: 1;
top: -30rem;
padding-top: 45re
margin-bottom: -3
}
```

Format: Short Hex
Format: HSL
Format: RGB
Format: Color Function

Convert to Display-P3

```
html, body, div, sp
h1, h2, h3, h4, h5,
abbr, acronym, addr
dfn, em, img, ins, kbd, q, samp, small, strik,
strong, sub, sup, tt, var, dl, dt, dd, ol, ul,
```



CSS新特性

~~① CSS伪类选择器~~

~~② CSS颜色~~

③ CSS背景

④ CSS蒙层和剪切

⑤ CSS混合模式

⑥ CSS自定义属性

⑦ CSS等比缩放

⑧ CSS滚动捕捉

⑨ CSS Gap(沟槽)

⑩ CSS逻辑属性

⑪ CSS媒体查询

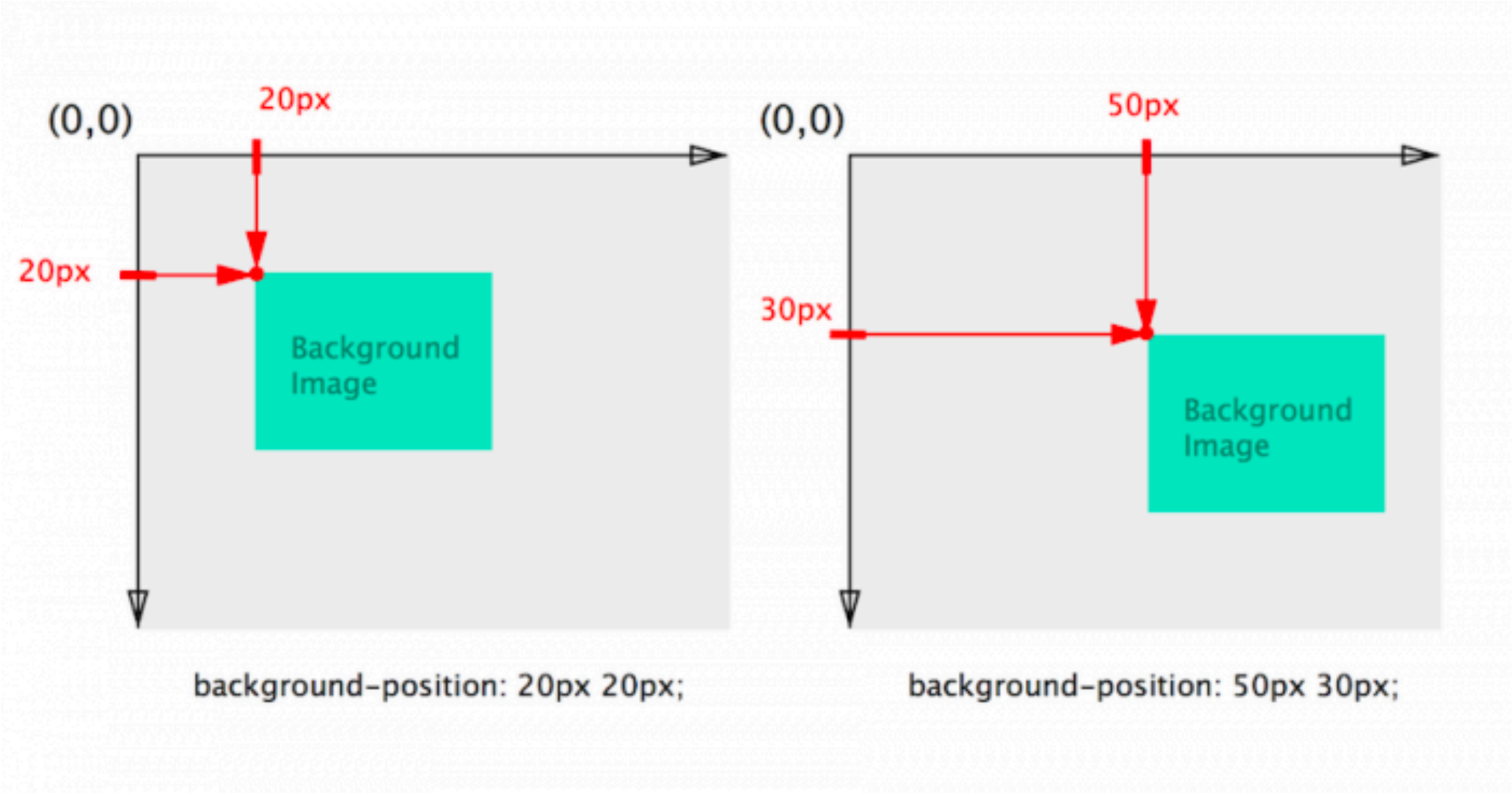
⑫ CSS比较函数

⑬ CSS内容可见性

⑭ CSS外在尺寸和内在尺寸

⑮ CSS的Display

CSS 背景 — background-position



CSS 背景 — background-position

CSS 背景 — background-position

Before

```css
:root {
    --xPosition: 50px;
    --yPosition: 50px;
}
.container {
    background-position:
      calc(100% - var(--xPosition))
      calc(100% - var(--yPosition));
}
```

After

```css
:root {
    --xPosition: 50px;
    --yPosition: 50px;
}
.container {
    background-position:
      right  var(--xPosition)
      bottom  var(--yPosition);
}
```

Codepen →

CSS 背景 — background-repeat

background-repeat

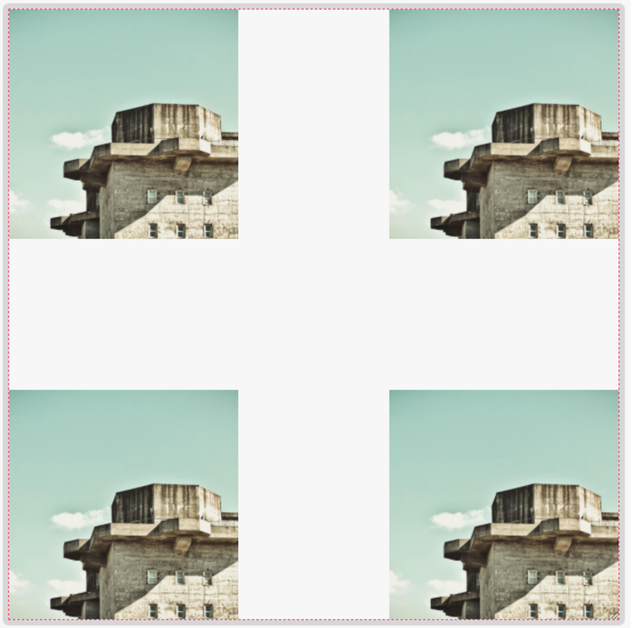
```
.container {
    background-repeat:
        no-repeat |
        repeat-x  |
        repeat-y  |
        repeat    |
        round     |
        space     ;
}
```

CSS 背景 — background-repeat

background-image

repeat

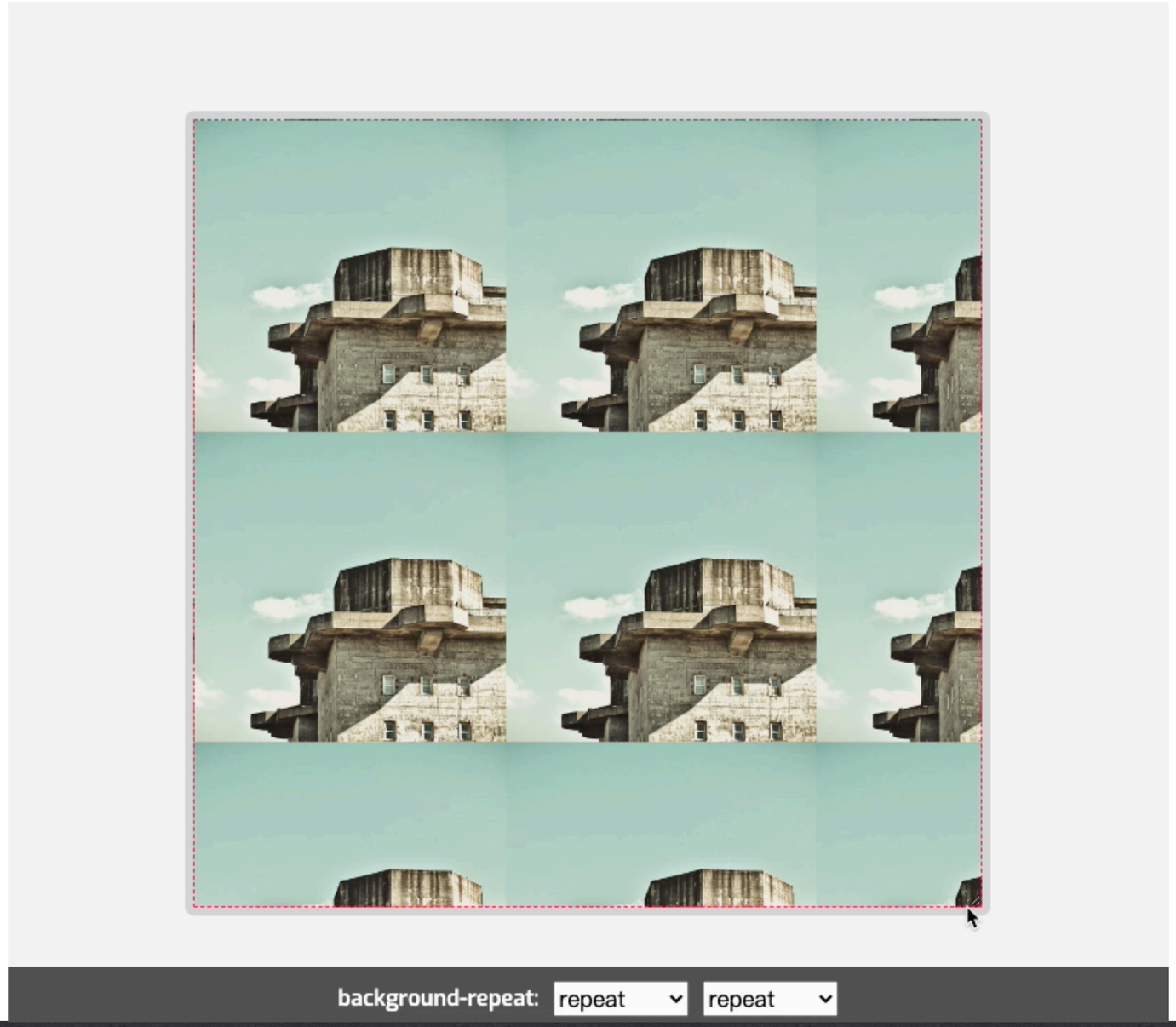
round



space

Codepen →

CSS 背景 — background-repeat



Codepen →

CSS新特性

~~① CSS伪类选择器~~

~~② CSS颜色~~

~~③ CSS背景~~

④ CSS蒙层和剪切

⑤ CSS混合模式

⑥ CSS自定义属性

⑦ CSS等比缩放

⑧ CSS滚动捕捉

⑨ CSS Gap(沟槽)

⑩ CSS逻辑属性

⑪ CSS媒体查询

⑫ CSS比较函数

⑬ CSS内容可见性

⑭ CSS外在尺寸和内在尺寸

⑮ CSS的Display

CSS masking & clipping

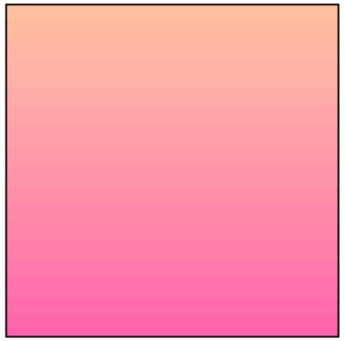
剪切
（Cliping）

源

剪切路径

剪切后对象

遮罩
（Masking）

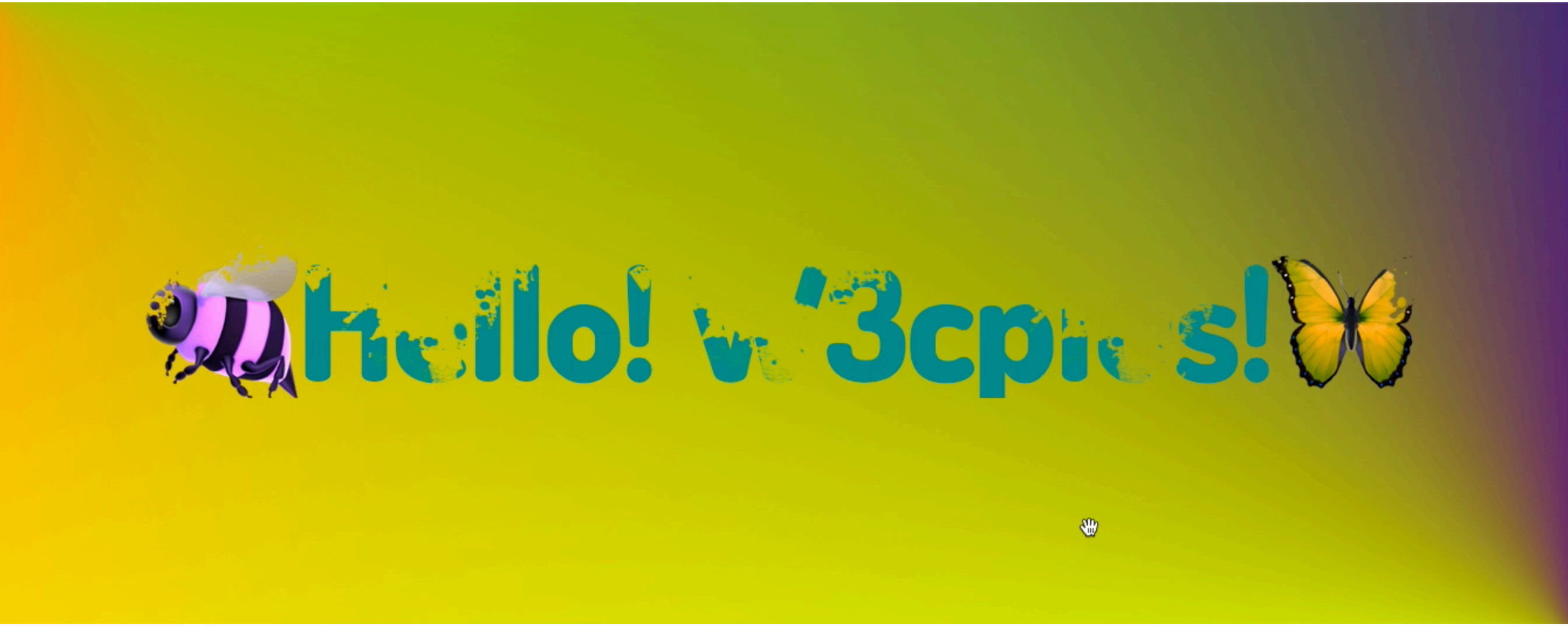


源

遮罩层
（高亮遮罩层）

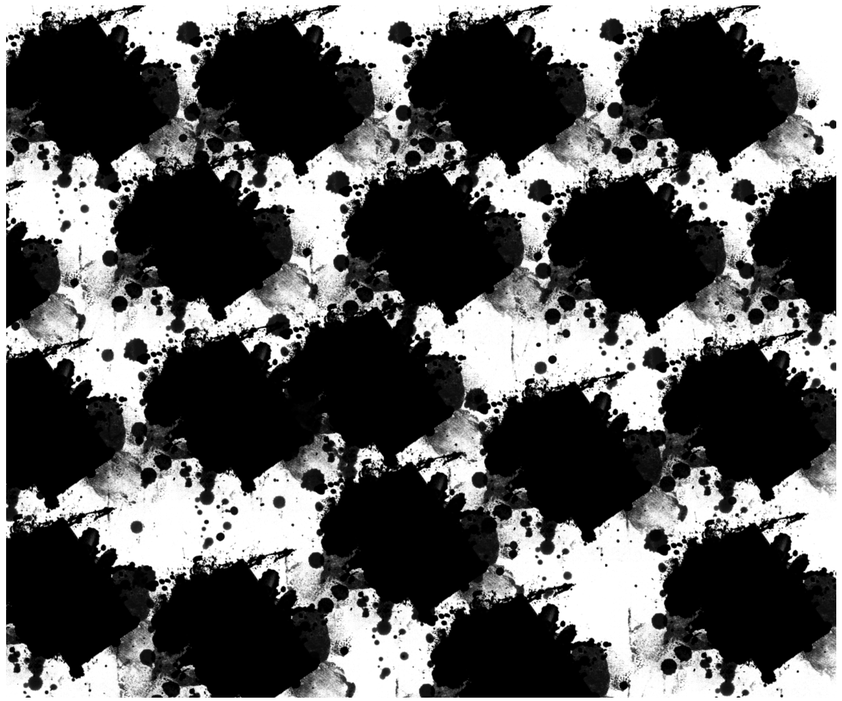
遮罩后对象

CSS masking & clipping



Codepen →

CSS masking & clipping



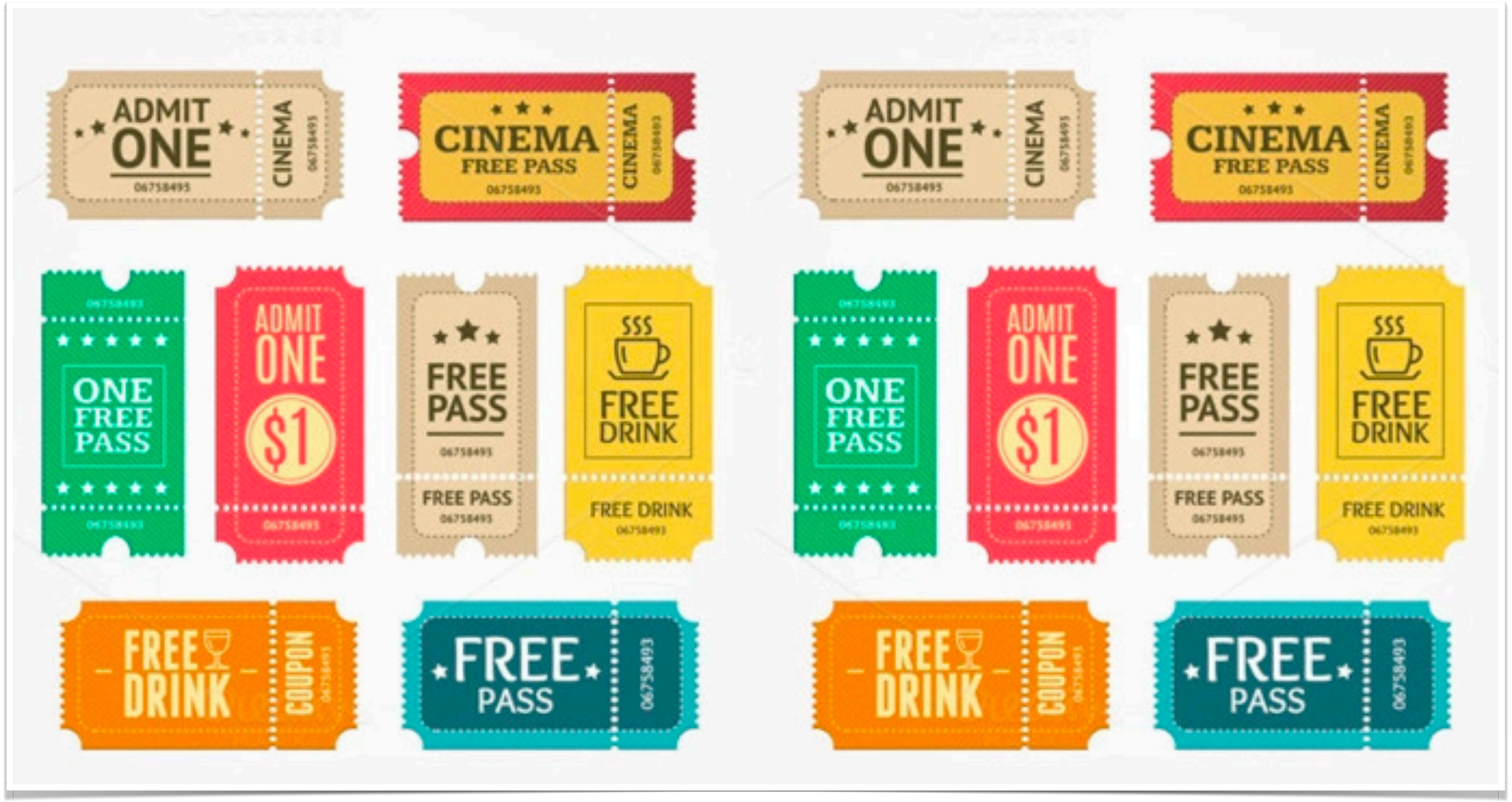
mask-image

masking

```
1 div {
2   mask-image: url('masking.png')
3 }
```

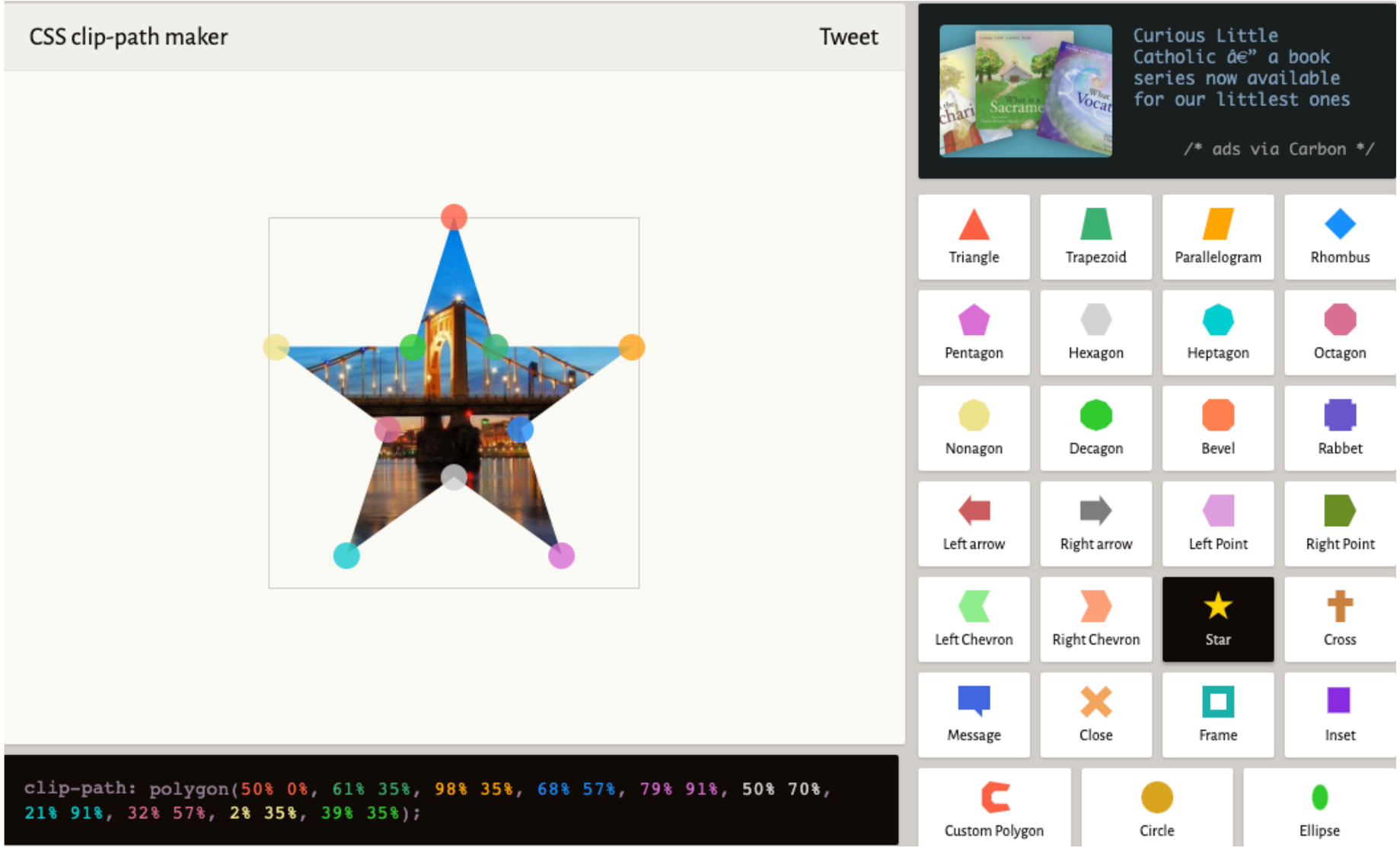
Codepen →

CSS masking & clipping



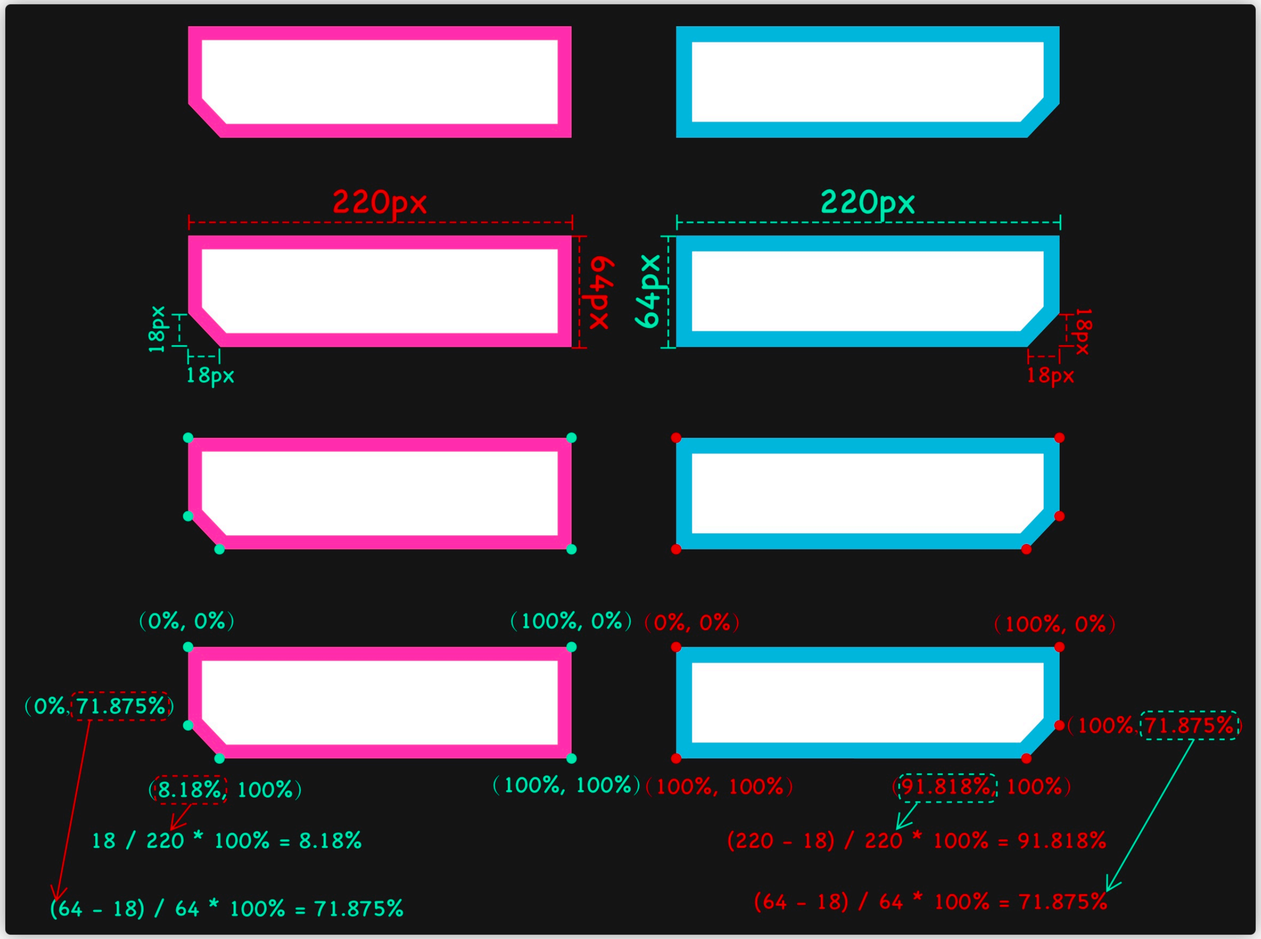
Codepen →

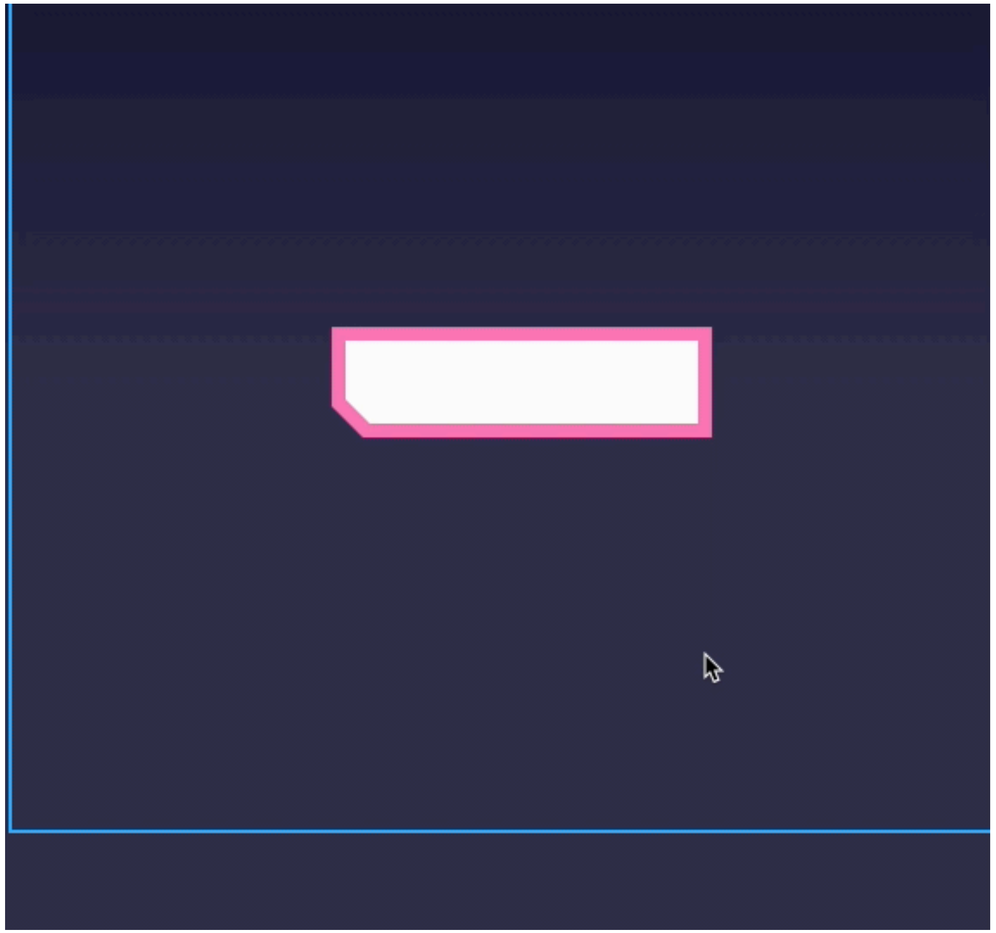
CSS masking & clipping



Clippy →

CSS masking & clipping





CodePen →

CSS新特性

~~① CSS伪类选择器~~

~~② CSS颜色~~

~~③ CSS背景~~

~~④ CSS蒙层和剪切~~

⑤ CSS混合模式

⑥ CSS自定义属性

⑦ CSS等比缩放

⑧ CSS滚动捕捉

⑨ CSS Gap(沟槽)

⑩ CSS逻辑属性

⑪ CSS媒体查询

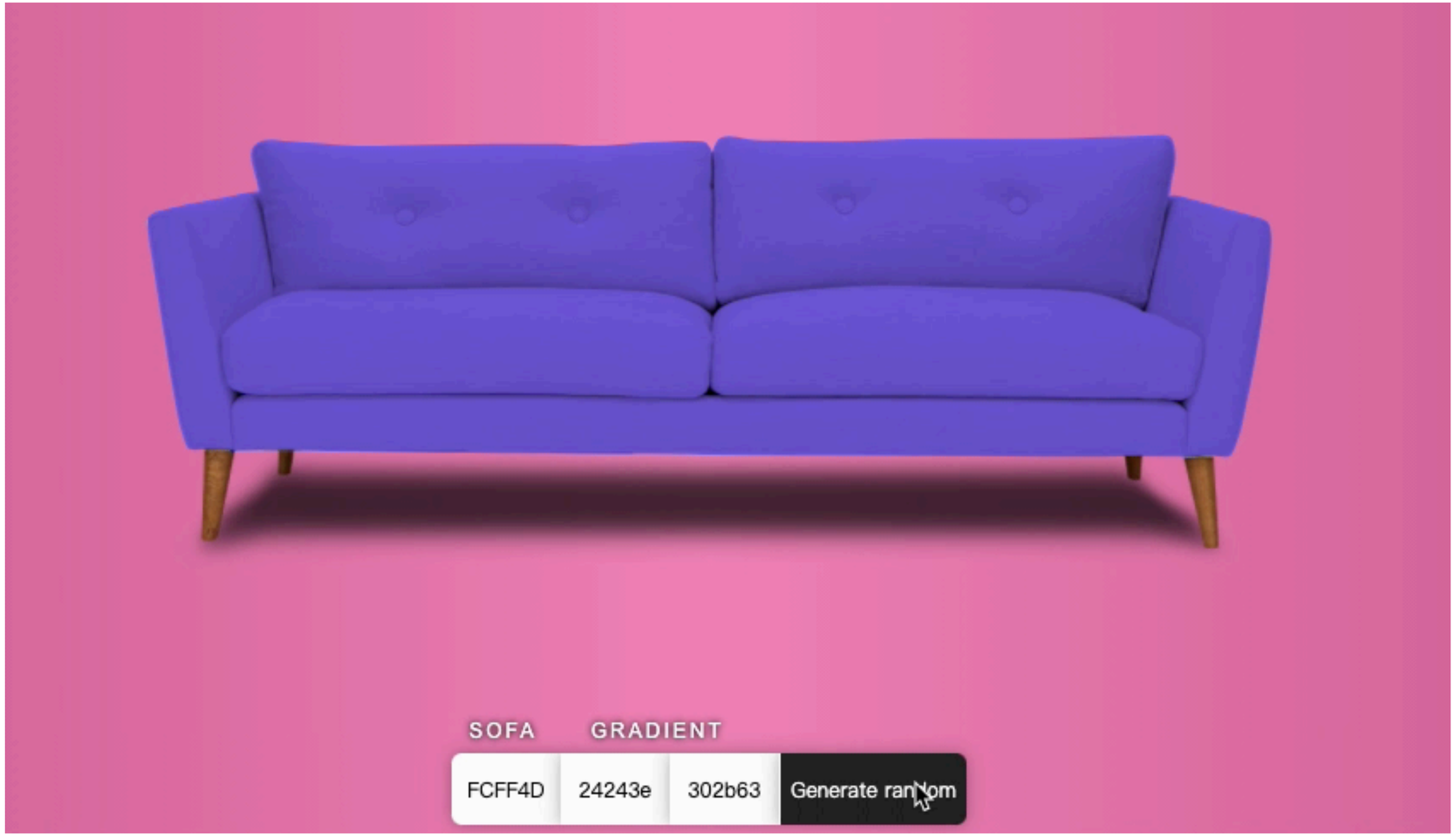
⑫ CSS比较函数

⑬ CSS内容可见性

⑭ CSS外在尺寸和内在尺寸

⑮ CSS的Display

CSS 混合模式



CodePen ⇥

CSS 混合模式

原图

SVG Path

mix-blend-mode

```
:root {
  --fill: #fcff4d
}

svg {
    fill: var(--fill);
    mix-blend-mode: multiply;
}
```

CodePen →

CSS新特性

~~① CSS伪类选择器~~

~~② CSS颜色~~

~~③ CSS背景~~

~~④ CSS蒙层和剪切~~

~~⑤ CSS混合模式~~

⑥ CSS自定义属性

⑦ CSS等比缩放

⑧ CSS滚动捕捉

⑨ CSS Gap(沟槽)

⑩ CSS逻辑属性

⑪ CSS媒体查询

⑫ CSS比较函数

⑬ CSS内容可见性

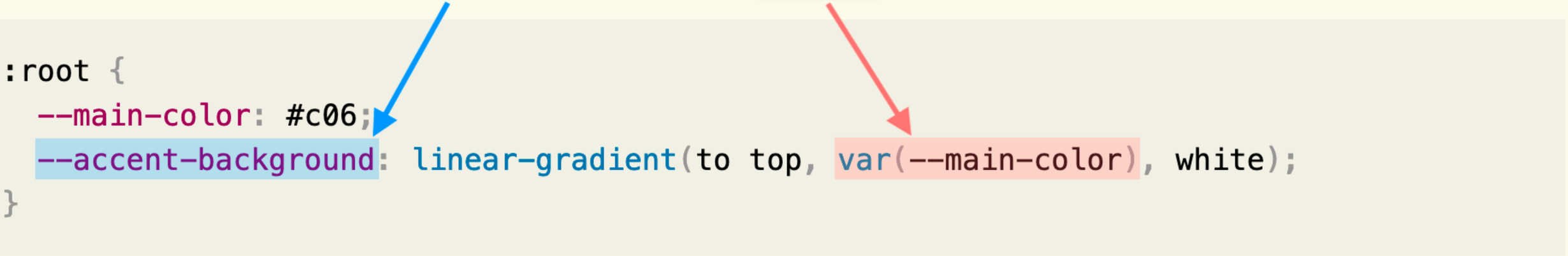
⑭ CSS外在尺寸和内在尺寸

⑮ CSS的Display

CSS 自定义属性

EXAMPLE 7

This example shows a custom property safely using a variable:



```
:root {
  --main-color: #c06;
  --accent-background: linear-gradient(to top, var(--main-color), white);
}
```

The '--accent-background' property (along with any other properties that use 'var(--main-color)') will automatically update when the '--main-color' property is changed.

CSS 自定义属性

didate Recommendation

§ 3.1. Invalid Variables

When a custom property has its initial value, 'var()' functions cannot use it for substitution. Attempting to do so makes the declaration invalid at computed-value time, unless a valid fallback is specified.

A declaration can be ***invalid at computed-value time*** if it contains a 'var()' that references a custom property with its initial value, as explained above, or if it uses a valid custom property, but the property value, after substituting its 'var()' functions, is invalid. When this happens, the computed value of the property is either the property's inherited value or its initial value depending on whether the property is inherited or not, respectively, as if the property's value had been specified as the 'unset' keyword.

当一个自定义属性的值是initial时，var()函数不能使用它进行替换。除非指定了一个有效的回退值，否则会使声明在计算值时无效。

W3C →|

CSS 自定义属性

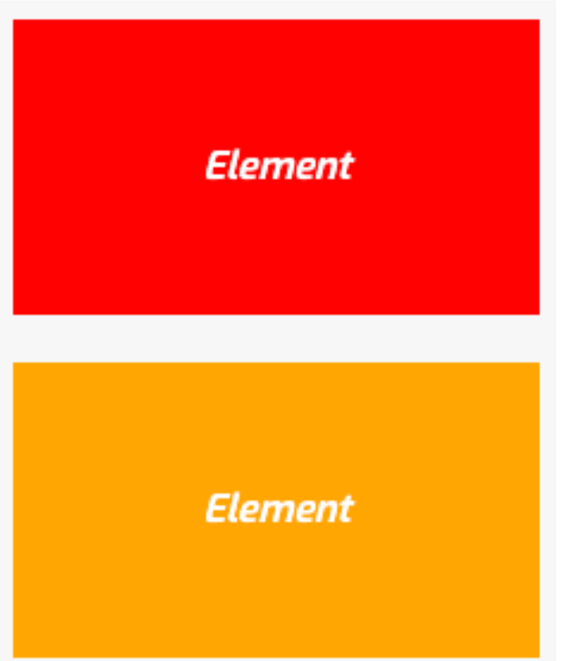
无效变量

```
:root {
    --foo: initial;
}

bar {
    color: var(--foo); // ❎
}

foo {
    color: var(--foo, orange); // ✅
}
```

CSS 自定义属性

无效变量

```html
<!-- HTML -->
<div class="element">
    <i>Element</i>
</div>

<i>Element</i>

/* CSS */
body { color: #fff; }
.element { --color: red; }
i {
  --foo: initial;
  --color: var(--foo);
  background-color: var(--color, orange);
}
```

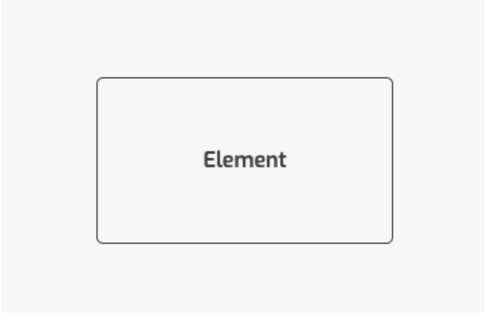


CodePen →

CSS 自定义属性

有效变量

```html
<!-- HTML -->
<section>Element</section>

/* CSS */
section {
  --color: ;
  background-color: var(--color, orange);
}
```

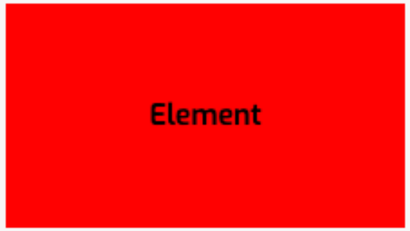


CodePen →

CSS 自定义属性

有效变量 vs. 无效变量

```
:root {
  --initial: initial; //  ❎
  --valid: ;          //  ✅
}

foo {
    background-color: var(--initial, red); // red
    color: var(--valid, red); // 空格
}
```



CodePen →

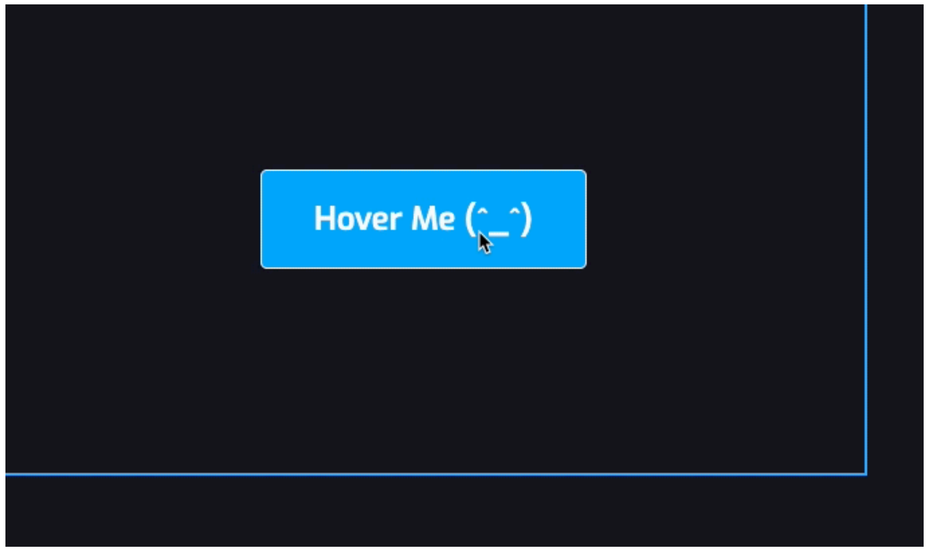
CSS 自定义属性

有效变量 vs. 无效变量

```
:root {
  --initial:; //  ❎ 冒号和分号之间无空格符
  --valid: ;  //  ✅ 冒号和分号之间有空格符
}

foo {
    background-color: var(--initial, red); // red
    color: var(--valid, red); // 空格
}
```

CSS 自定义属性

```
:root {
  --ON: initial; /* ❎ 👉 等同 --ON:; */
  --OFF: ; /* ✅ 👉 注意冒号和分号之间空格 */
}

button {
  --is-raised: var(--OFF);
  --border-color: var(--is-raised, rgb(0 0 0 / 0.1));
  --gradient: var(
    --is-raised,
    linear-gradient(hsl(0 0% 100% / 0.3), transparent)
  );
  --box-shadow: var(
    --is-raised,
    0 1px hsl(0 0% 100% / 0.8) inset,
    0 0.1em 0.1em -0.1em rgb(0 0 0 / 0.2)
  );
  --text-shadow: var(--is-raised, 0 -1px 1px rgb(0 0 0 / 0.3));

  border: 1px solid var(--border-color);
  background: hsl(200 100% 50%) var(--gradient);
  box-shadow: var(--box-shadow);
  text-shadow: var(--text-shadow);
}
button:hover {
  --is-raised: var(--ON);
}
```

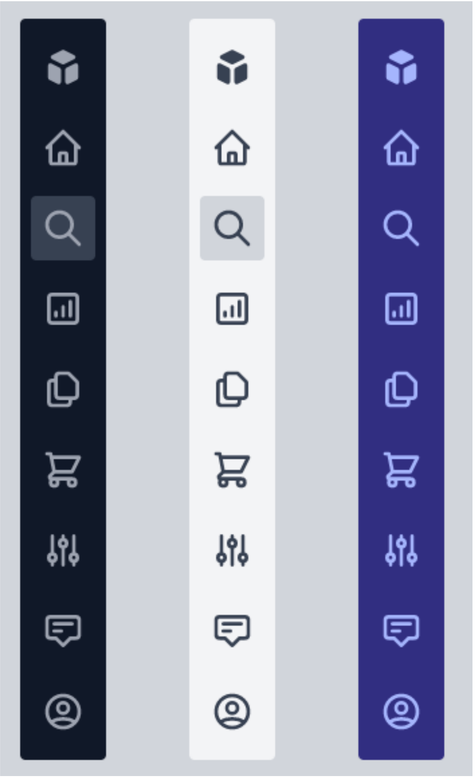


CodePen →

CSS 自定义属性

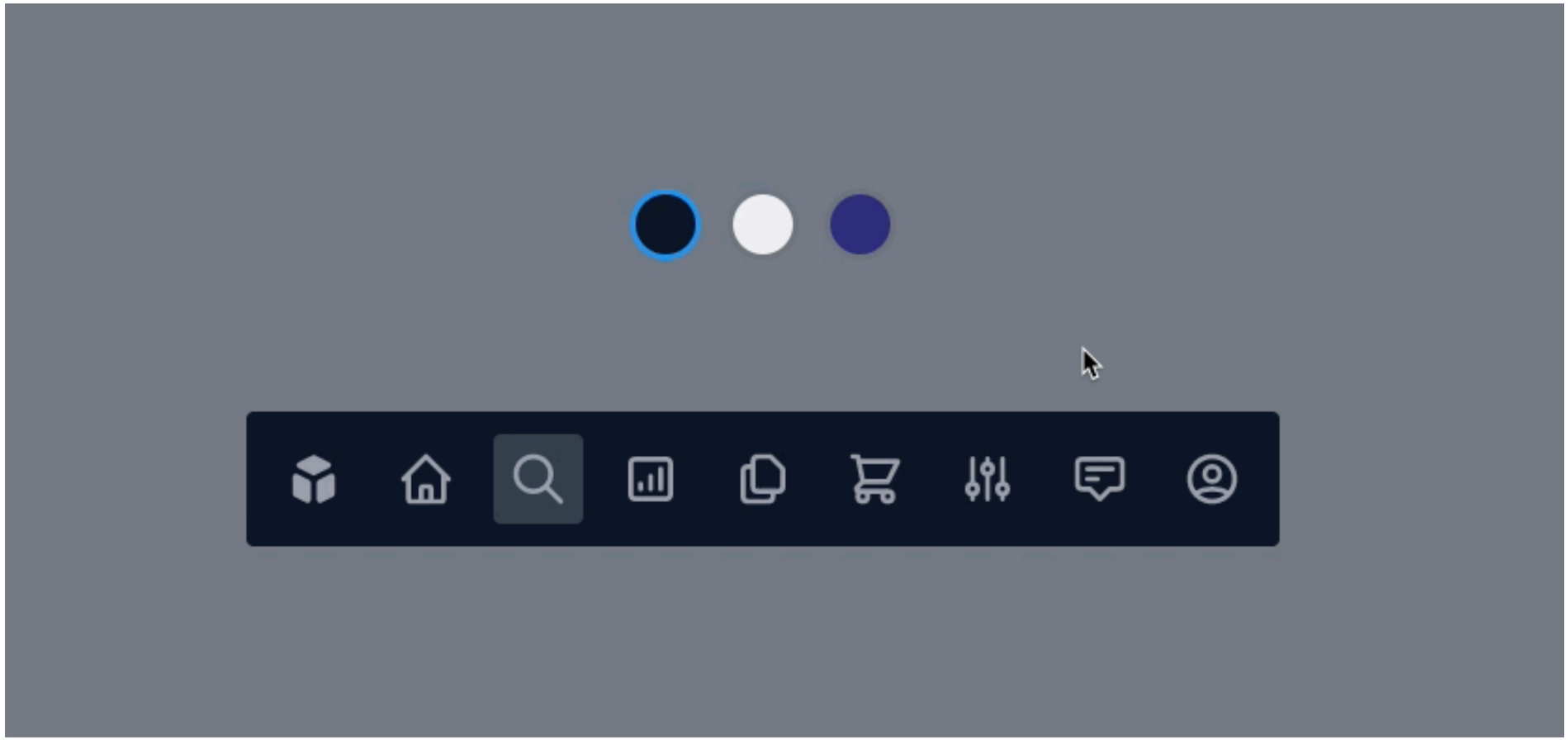
```css
/* 设置切换开关 */
:root {
  --ON: initial; // ❎ 👉 等同 --ON:;
  --OFF: ;       // ✅ 👉 注意冒号和分号之间空格
}
/* 默认为Dark */
.dark {
  --light: var(--OFF);
  --dark: var(--ON);
  --blue: var(--OFF);
}
/* 默认为Light */
.light {
  --light: var(--ON);
  --dark: var(--OFF);
  --blue: var(--OFF);
}
/* 默认为Blue */
.blue {
  --light: var(--OFF);
  --dark: var(--OFF);
  --blue: var(--ON);
}
```

```css
.nav {
  /* Dark */
  --dark-color: rgb(156 163 175);
  --dark-bgcolor: rgb(17 24 39);
  --dark-active-bgcolor: rgb(55 65 81);

  /* Light */
  --light-color: rgb(55 65 81);
  --light-bgcolor: rgb(243 244 246);
  --light-active-bgcolor: rgb(209 213 219);

  /* Blue */
  --blue-color: rgb(165 180 252);
  --blue-bgcolor: rgb(49 46 129);
  --blue-active-bgcolor: rgb(67 56 202);
}
.nav{
  color: var(
    --light, var(
      --light-color)) var(
        --dark, var(
          --dark-color)) var(
            --blue, var(--blue-color)
    );
  background-color: var(
    --light, var(
      --light-bgcolor)) var(
        --dark, var(
          --dark-bgcolor)) var(
            --blue, var(--blue-bgcolor)
  );
}
```



CodePen →

CSS 自定义属性



CodePen →

CSS新特性

~~① CSS伪类选择器~~

~~② CSS颜色~~

~~③ CSS背景~~

~~④ CSS蒙层和剪切~~

~~⑤ CSS混合模式~~

~~⑥ CSS自定义属性~~

⑦ CSS等比缩放

⑧ CSS滚动捕捉

⑨ CSS Gap(沟槽)

⑩ CSS逻辑属性

⑪ CSS媒体查询

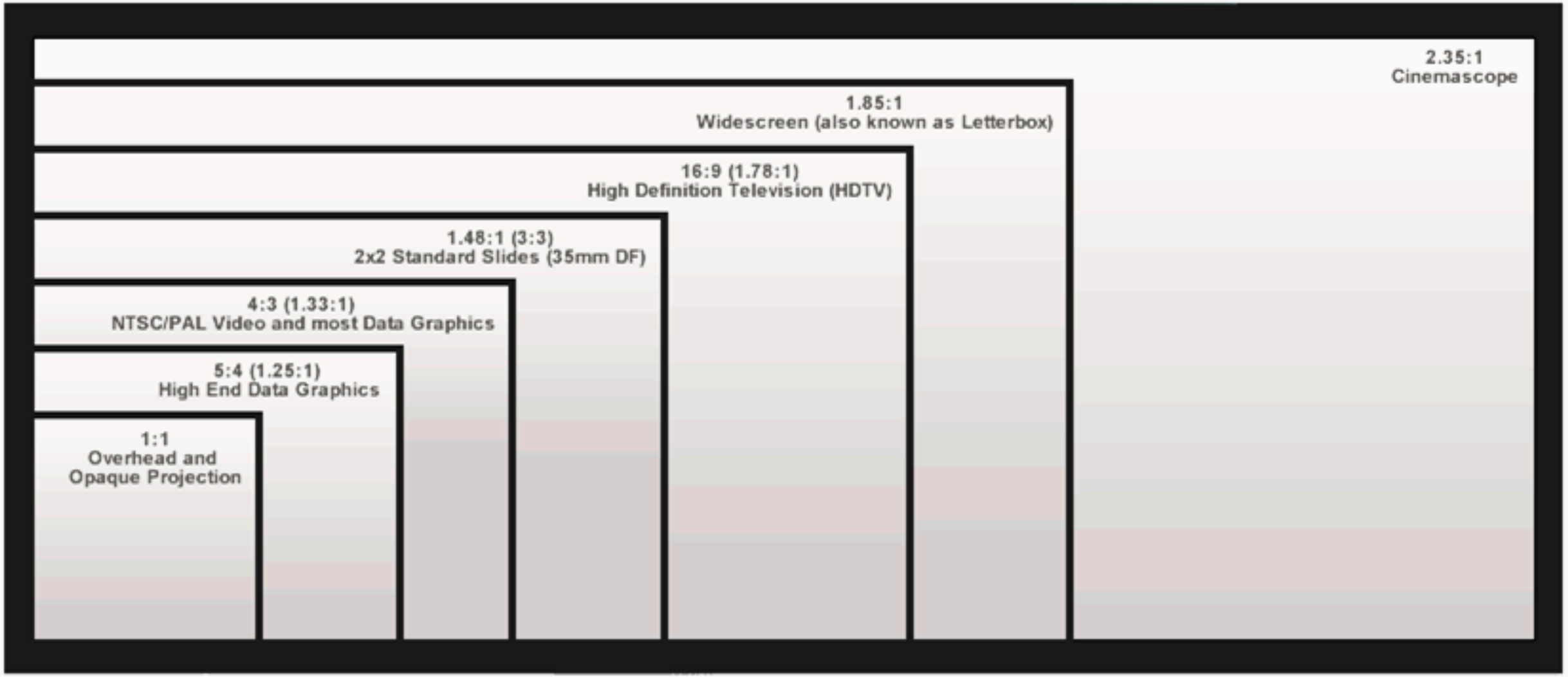
⑫ CSS比较函数

⑬ CSS内容可见性

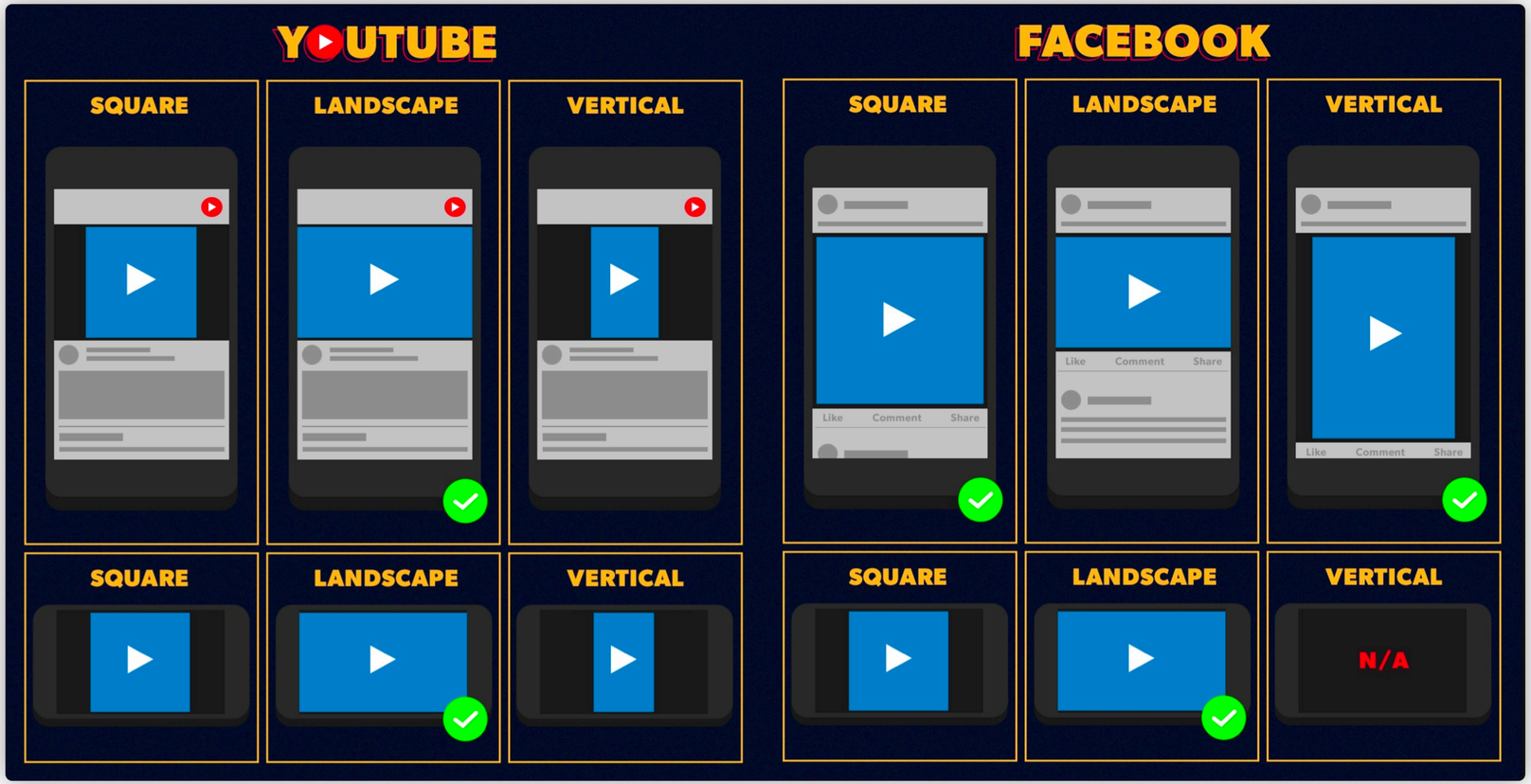
⑭ CSS外在尺寸和内在尺寸

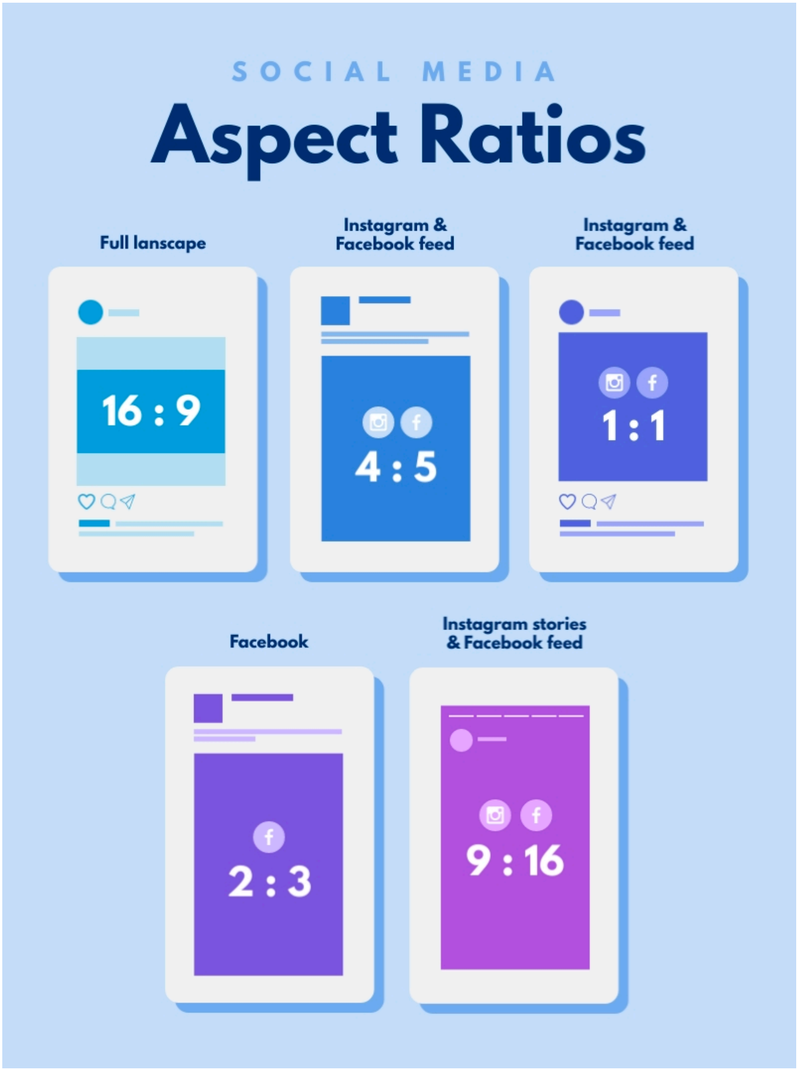
⑮ CSS的Display

CSS 等比缩放（aspect-ratio）

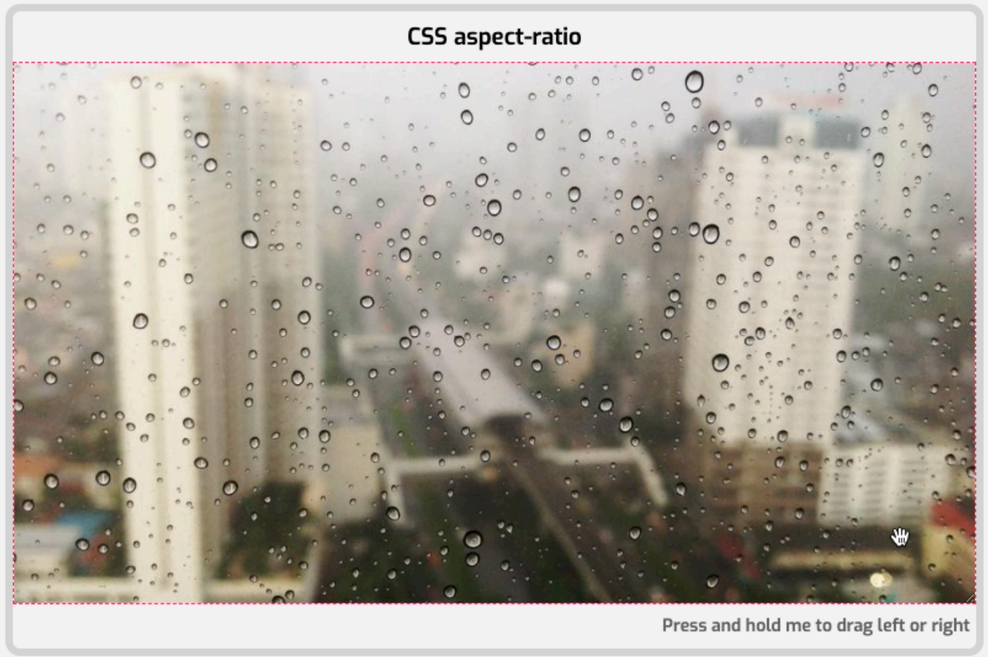


CSS 等比缩放（aspect-ratio）





CSS 等比缩放（aspect-ratio）



CodePen →

CSS 等比缩放（aspect-ratio）

等比缩放：padding-top 或 padding-bottom

```css
/* padding-bottom/top */
.aspect-ratio-container {
    position: relative;
    height: 0;
    padding-top: 56.25%;
    width: 100%;
}
.aspect-ratio-content {
    position: absolute;
    top: 0;
    left: 0;
    width: 100%;
}
```

等比缩放：padding-top 或 padding-bottom

```css
/* CSS Custom Property */
.aspect-ratio-container {
    width: 100%;
}
.aspect-ratio-content {
    --ratio: 16/9;
    height: calc(var(--width) * 1 / (var(--ratio)));
    width: 100%;
}
```

等比缩放：aspect-ratio

```css
/* CSS aspect-ratio */
.aspect-ratio-content {
    width: 100%;
    aspect-ratio: 16 / 9;
}
```

CodePen →

CSS新特性

~~① CSS伪类选择器~~

~~② CSS颜色~~

~~③ CSS背景~~

~~④ CSS蒙层和剪切~~

~~⑤ CSS混合模式~~

~~⑥ CSS自定义属性~~

~~⑦ CSS等比缩放~~

⑧ CSS滚动捕捉

⑨ CSS Gap(沟槽)

⑩ CSS逻辑属性

⑪ CSS媒体查询

⑫ CSS比较函数

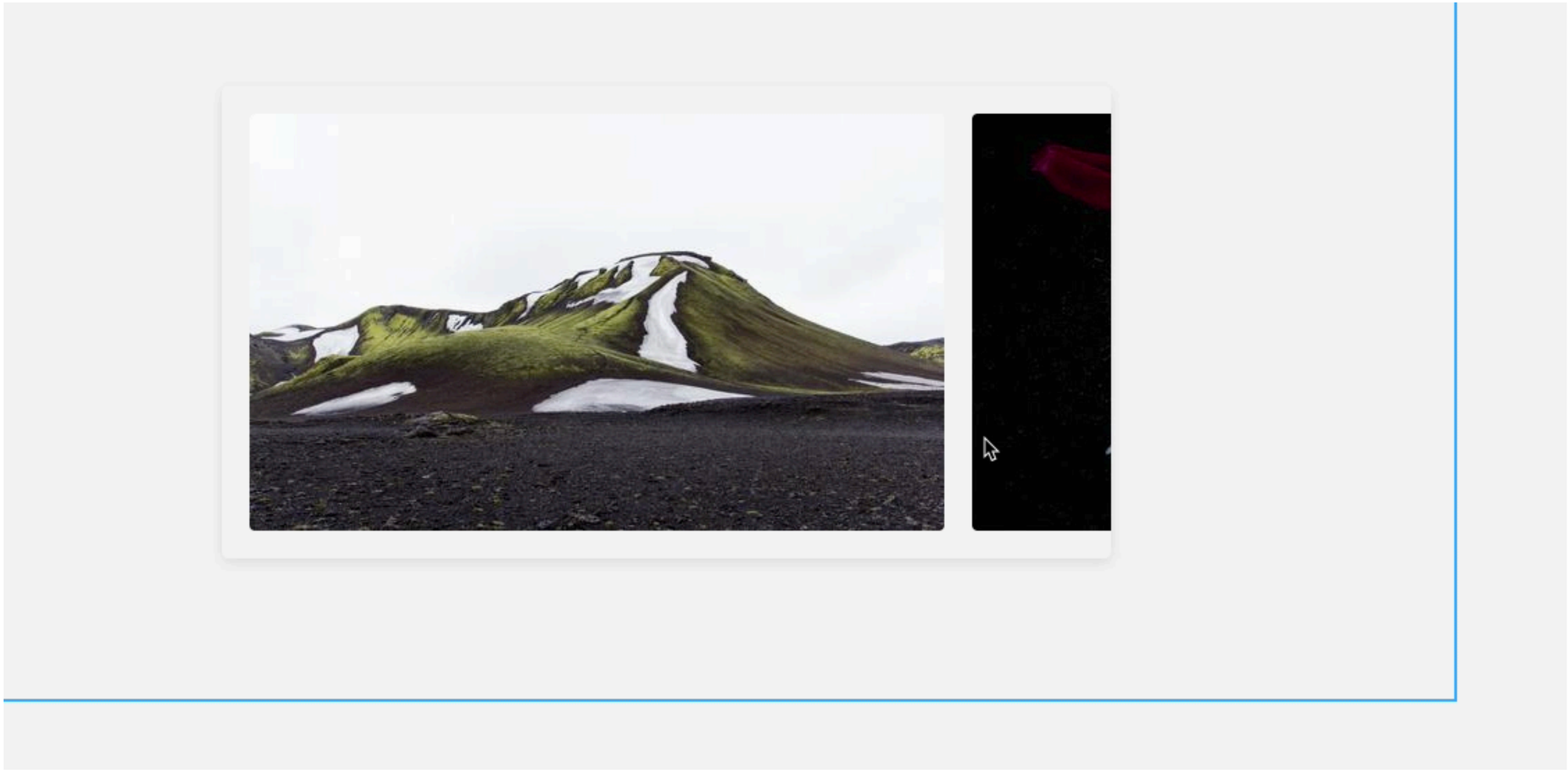
⑬ CSS内容可见性

⑭ CSS外在尺寸和内在尺寸

⑮ CSS的Display

CSS 滚动捕捉

CSS 滚动捕捉



CodePen →

CSS 滚动捕捉

CSS 滚动捕捉

```css
.container {
  scroll-behavior: smooth;
  overflow-x: auto;
  scroll-snap-type: x mandatory;
  scroll-padding: 20px;
}

img {
  scroll-snap-align: center;
  scroll-snap-stop: always;
}
```

CodePen →

CSS 滚动捕捉

CodePen →

CSS新特性

~~① CSS伪类选择器~~

~~② CSS颜色~~

~~③ CSS背景~~

~~④ CSS蒙层和剪切~~

~~⑤ CSS混合模式~~

~~⑥ CSS自定义属性~~

~~⑦ CSS等比缩放~~

~~⑧ CSS滚动捕捉~~

⑨ CSS Gap(沟槽)

⑩ CSS逻辑属性

⑪ CSS媒体查询

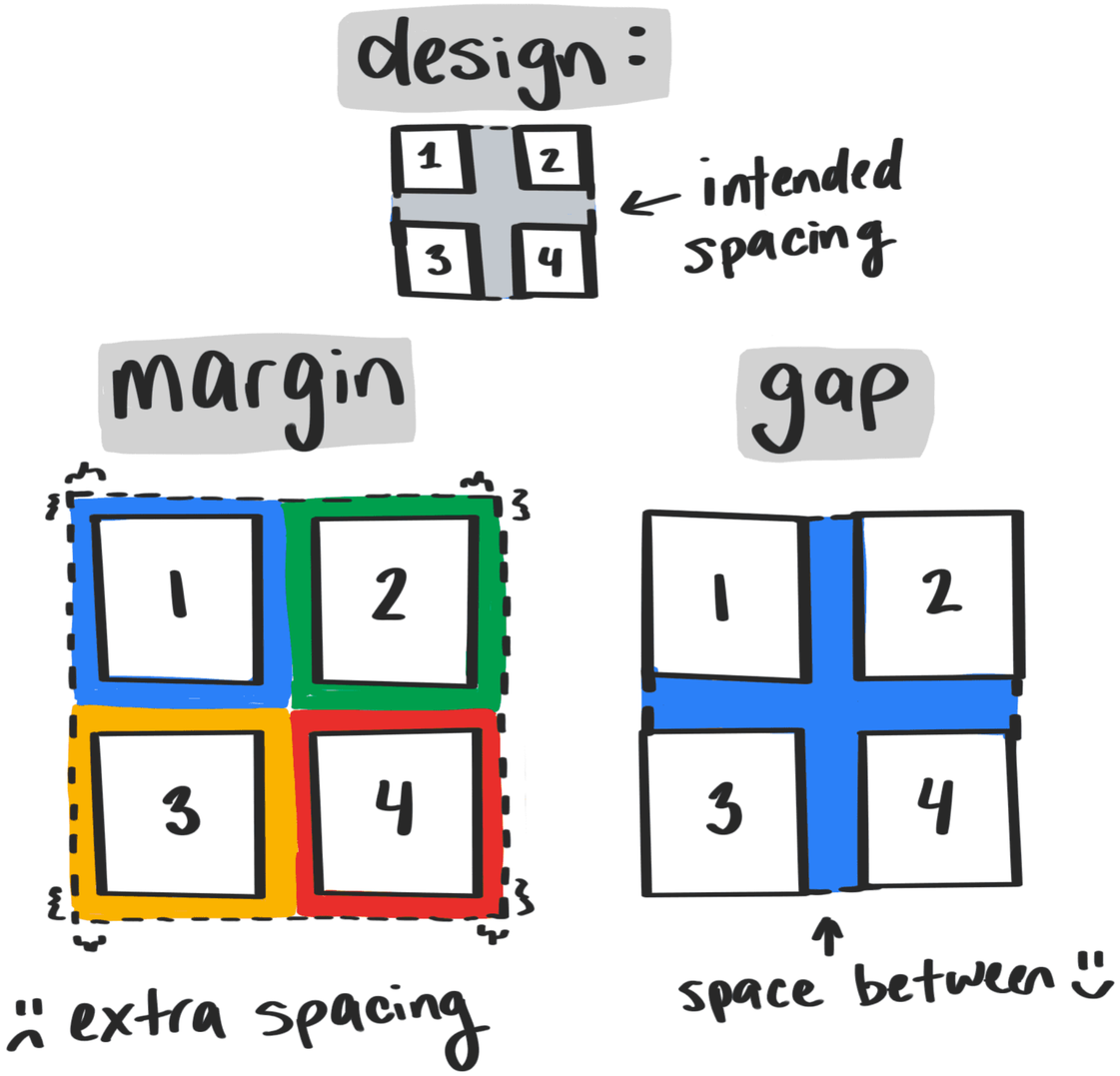
⑫ CSS比较函数

⑬ CSS内容可见性

⑭ CSS外在尺寸和内在尺寸

⑮ CSS的Display

CSS Gap



CSS gap

```css
.container {
  display: flex;
  gap: 10px;
}

.container {
  display: grid;
  gap: 20px
}
```

CSS新特性

~~① CSS伪类选择器~~

~~② CSS颜色~~

~~③ CSS背景~~

~~④ CSS蒙层和剪切~~

~~⑤ CSS混合模式~~

~~⑥ CSS自定义属性~~

~~⑦ CSS等比缩放~~

~~⑧ CSS滚动捕捉~~

~~⑨ CSS Gap(沟槽)~~

⑩ CSS逻辑属性

⑪ CSS媒体查询

⑫ CSS比较函数

⑬ CSS内容可见性

⑭ CSS外在尺寸和内在尺寸

⑮ CSS的Display

CSS 逻辑属性

中文 (LTR)

阿拉伯语 (RTL)

CSS 逻辑属性

中文（LTR）

html属性：<body dir="ltr">

开发者添加的CSS：body { direction: ltr; unicode-bidi: embed; }

客户端自带的CSS：

body[Attributes Style] {direction: ltr;unicode-bidi: isolate;}

阿拉伯语（RTL）

html属性：<body dir="rtl">

开发者添加的CSS：body { direction: rtl; unicode-bidi: embed; }

客户端自带的CSS：

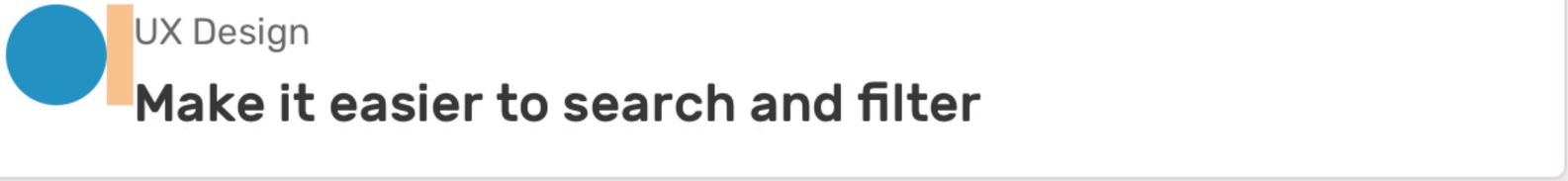
body[Attributes Style] {direction: rtl; unicode-bidi: isolate;}

CSS 逻辑属性

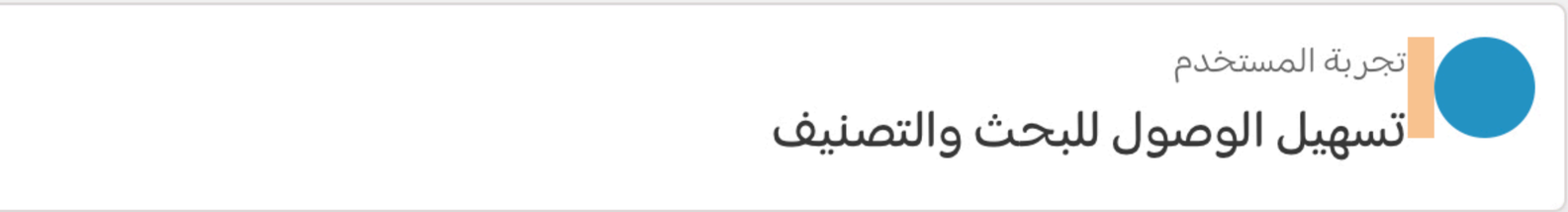

文档流
文档流
writing-mode: vertical-rl
writing-mode: vertical-lr
City Lights in New York
New York, the largest city in the U.S., is an architectural marvel with plenty of historic monuments, magnificent buildings and countless dazzling skyscrapers.
文本流
أضواء المدينة في نيويورك
نيويورك ، أكبر مدينة في الولايات المتحدة ، هي أعجوبة معمارية مع الكثير من المعالم التاريخية والمباني الرائعة وناطحات السحاب الرائعة التي لا تعد ولا تحصى.
文本流
ニューヨークの街の灯
アメリカで最大の都市であるニューヨークは、数多くの歴史的建造物、壮大な建物、無数の見事な超高層ビルが建ち並ぶ建築の驚異です。
文本流
Нью Йорк дахь хотын гэрлүүд
архитектурын гайхамшиг юм. Багануудсан барилга бүхий баршгүй олон тооны тэнгэр барилга байгуулалж, тоолж дурсгалт газруудыг, гайхамшигтай Йорк бол олон тооны түүхийн АНУ-ын хамгийн том хот болох Нью
文本流
阅读方式ltr
阅读方式rtl
文档流
文档流
writing-mode: horizontal-tb
writing-mode: horizontal-tb

CSS 逻辑属性

Left to Right (LTR)

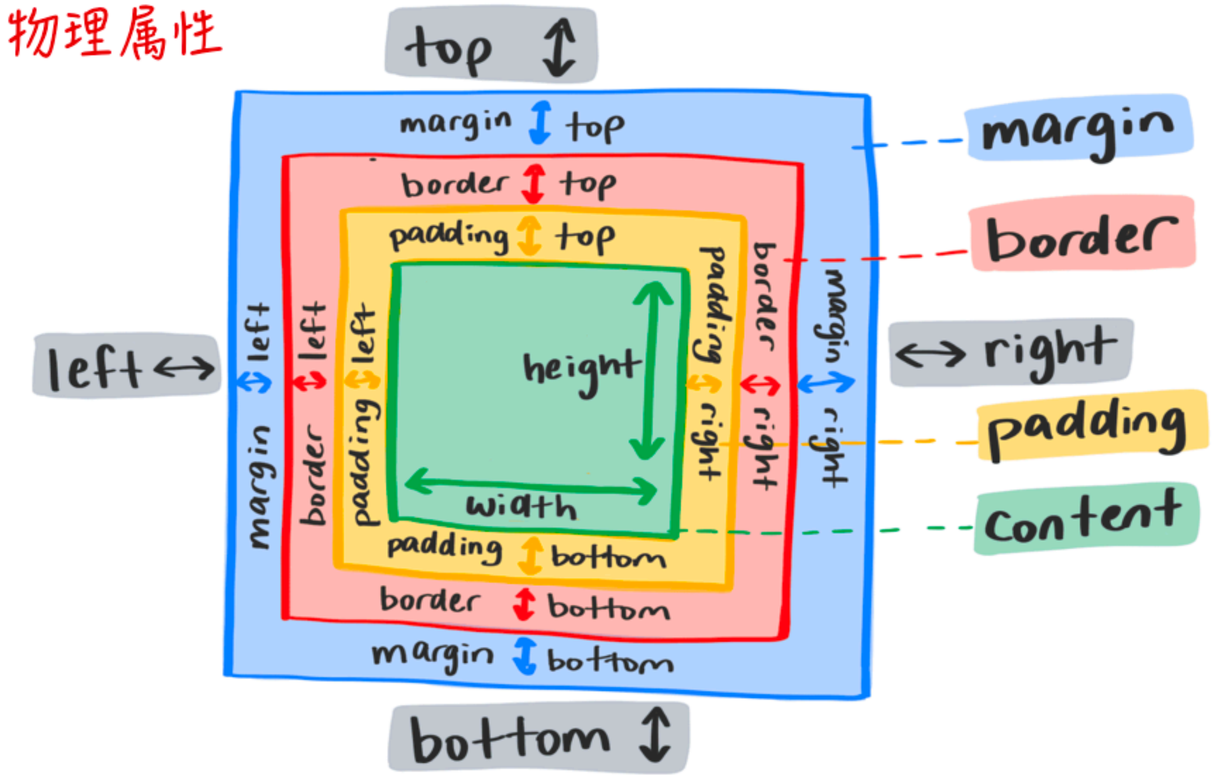


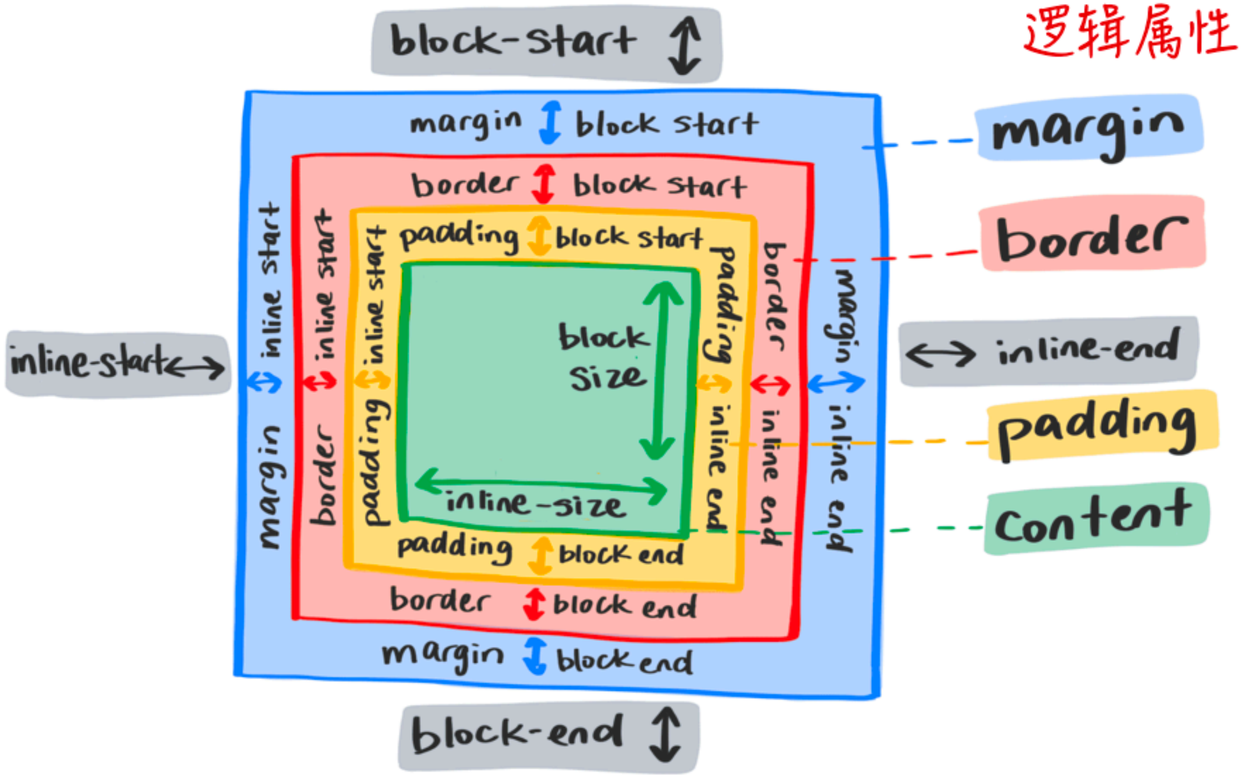
Right to Left (RTL)




```css
.avatar {
  margin-right: 1rem;
}

html[dir="rtl"] .avatar {
  margin-right: 0;
  margin-left: 1rem;
}
```

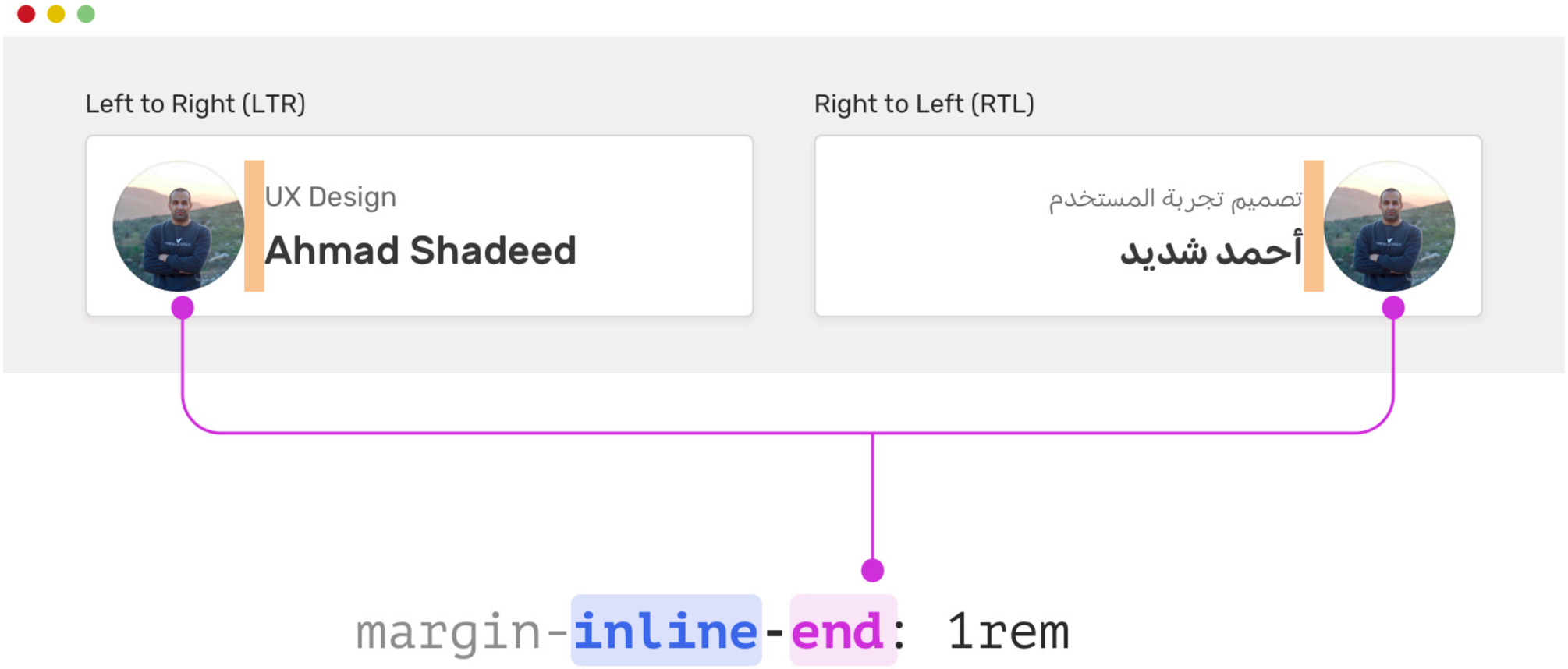
CSS 逻辑属性





CodePen →

CSS 逻辑属性

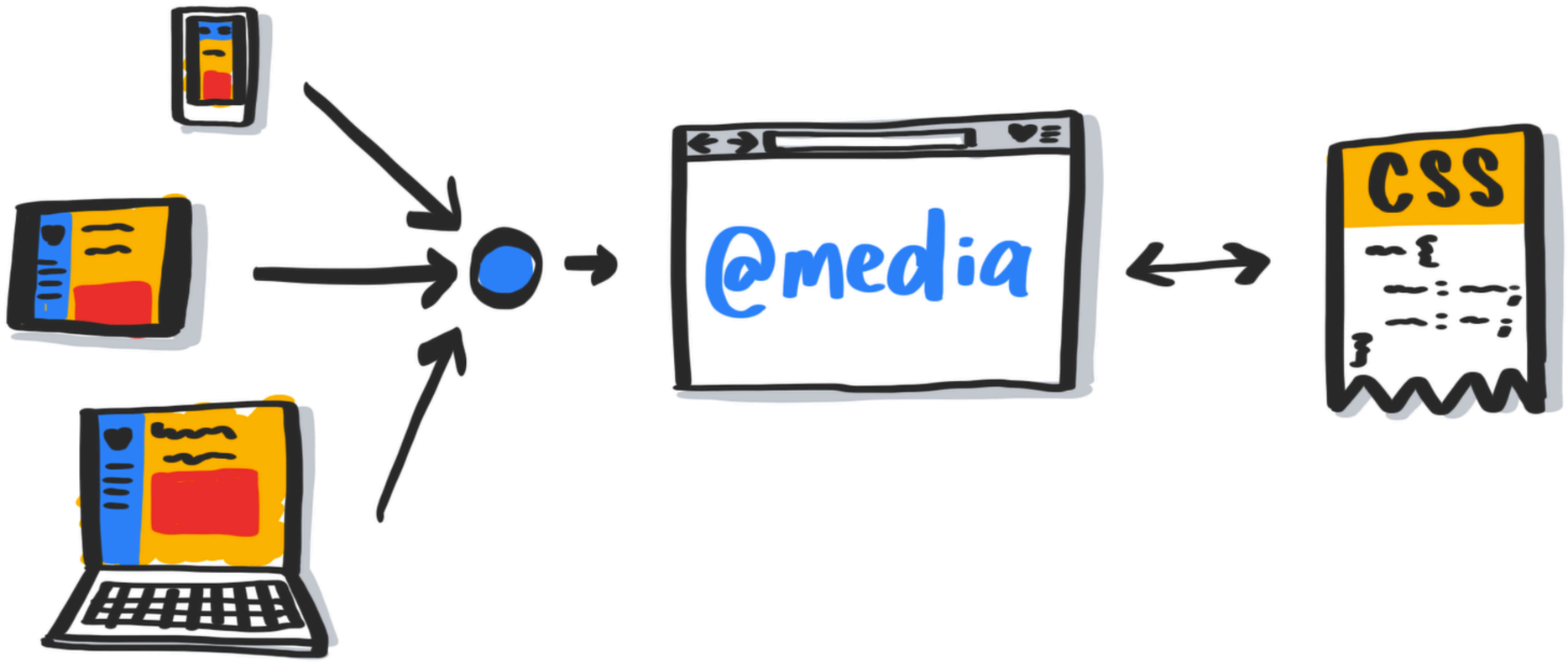


CodePen →

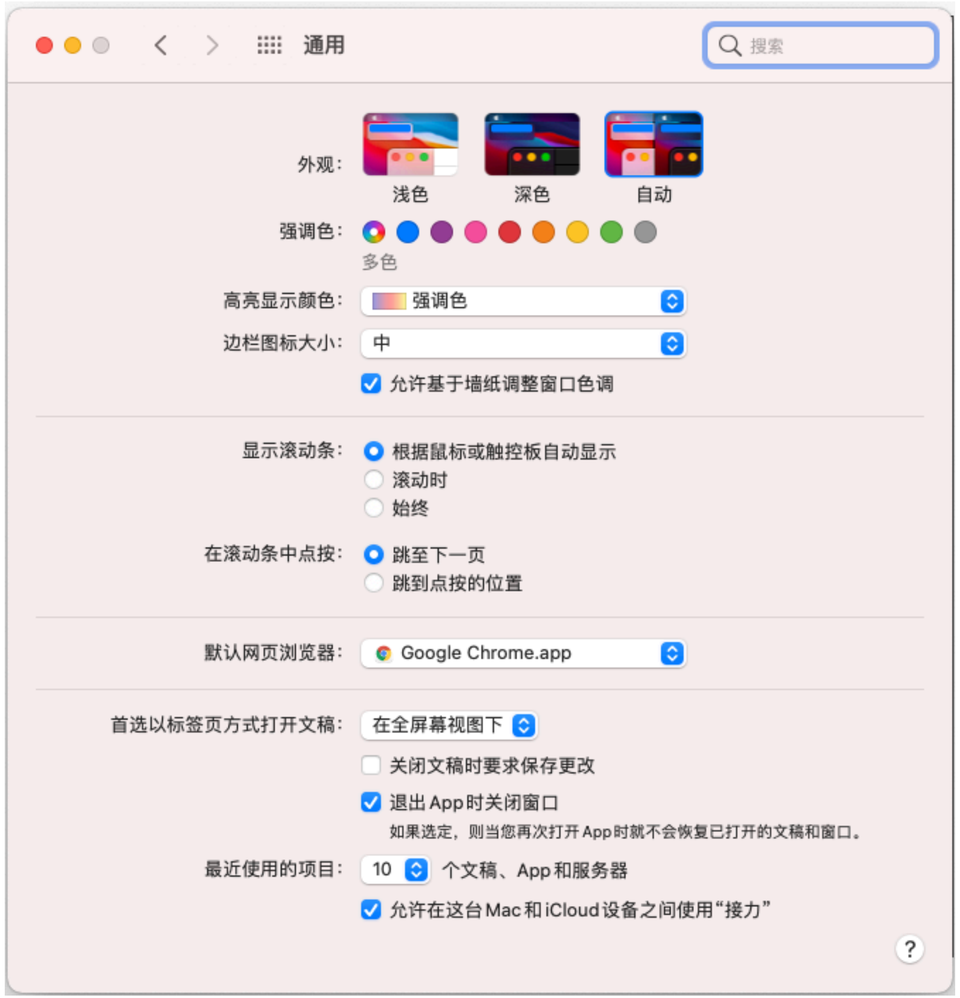
CSS新特性

① CSS伪类选择器
② CSS颜色
③ CSS背景
④ CSS蒙层和剪切
⑤ CSS混合模式
⑥ CSS自定义属性
⑦ CSS等比缩放
⑧ CSS滚动捕捉
⑨ CSS Gap(沟槽)
⑩ CSS逻辑属性
⑪ CSS媒体查询
⑫ CSS比较函数
⑬ CSS内容可见性
⑭ CSS外在尺寸和内在尺寸
⑮ CSS的Display

CSS 媒体查询



CSS 媒体查询



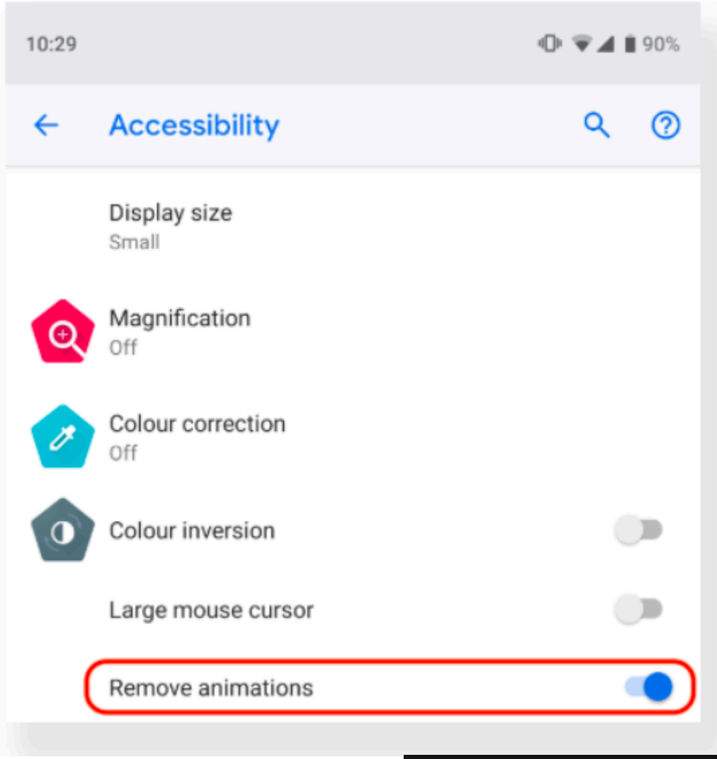

```css
// dark & light mode
:root {
  /* Light theme */
  --c-text: #333;
  --c-background: #fff;
}

body {
  color: var(--c-text);
  background-color: var(--c-background);
}

@media (prefers-color-scheme: dark) {
  :root {
    /* Dark theme */
    --c-text: #fff;
    --c-background: #333;
  }
}
```

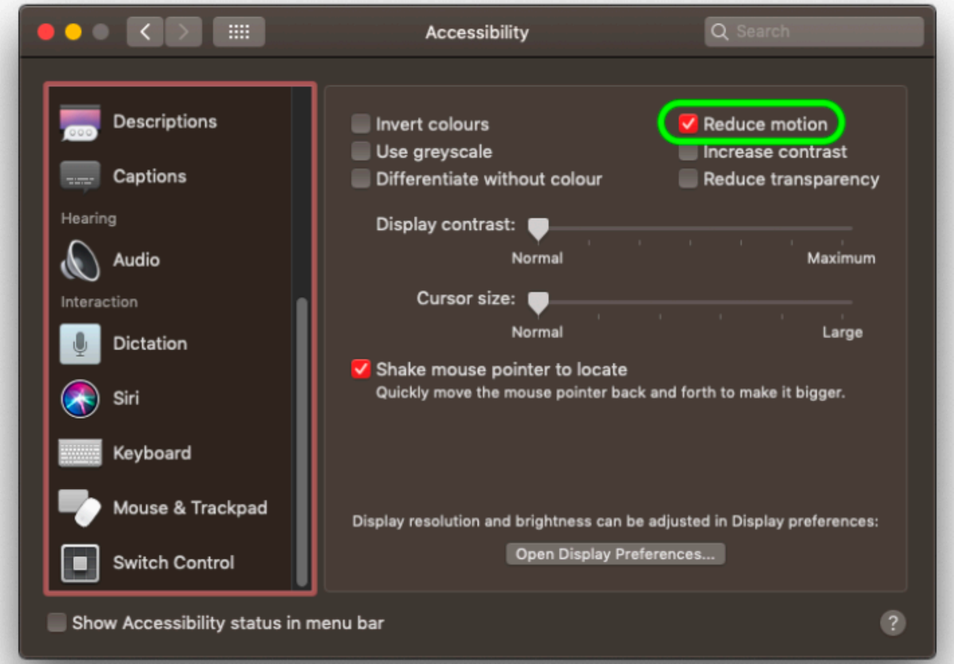
CodePen →

CSS 媒体查询


```css
@media (inverted-colors) {
  img {
    filter: invert(1);
  }
  * {
    box-shadow: none !important; text-shadow:
 none !important;
  }
}
```

CSS 媒体查询





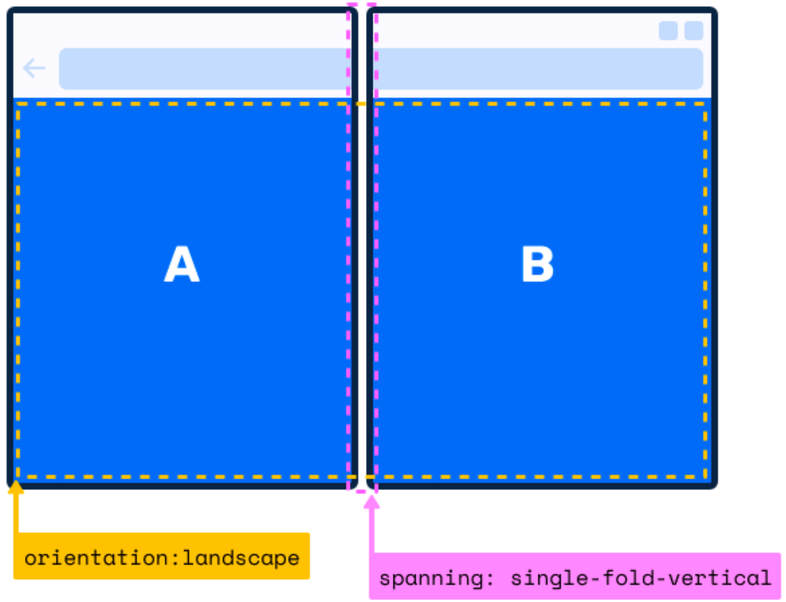

```css
.animation {
  animation: vibrate 0.3s linear infinite both;
}

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

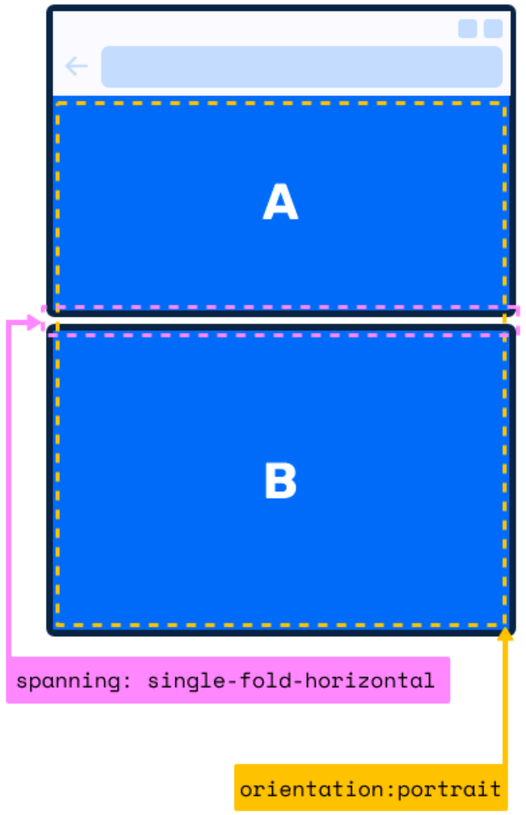
CodePen →

CSS 媒体查询


Samsung Galaxy Z Fold 2
Samsung Galaxy Z Flip
Huawei Mate Xs
Microsoft Surface Duo
Moto Razr 2019
Royole Flexpai
LG G8X ThinkQ
Xiaomi Dual Flex
Apple's Plans For Foldable iPhone
OPPO Foldable Smartphone
TCL
Sony Xperia F
Vivo IQOO

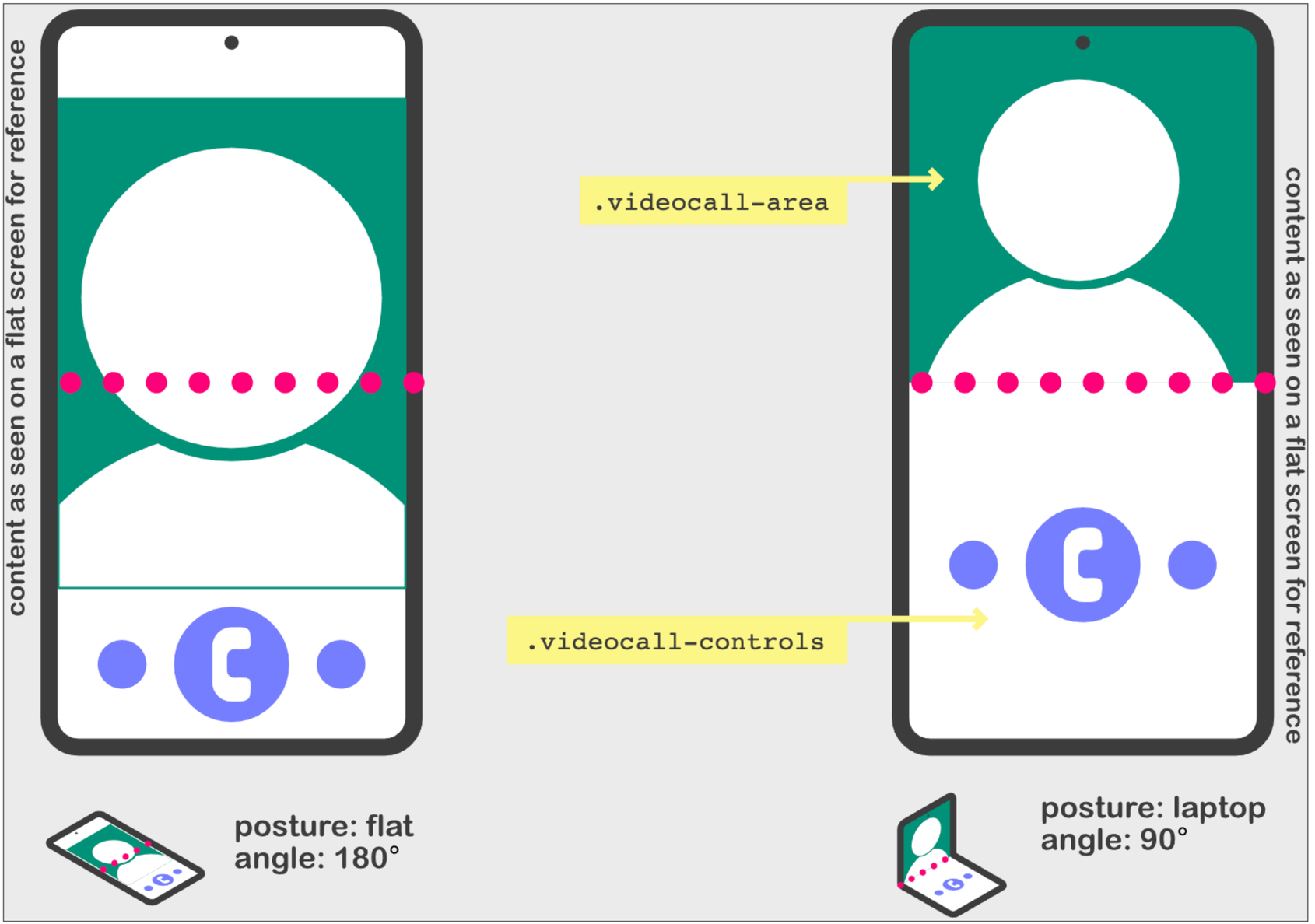
CSS 媒体查询






```
@media (spanning: single-fold-vertical) {
    // css

}
```

CSS 媒体查询




```
1 @media (screen-fold-posture: laptop){
2 // CSS
3
4 }
```

CSS新特性

~~① CSS伪类选择器~~

~~② CSS颜色~~

~~③ CSS背景~~

~~④ CSS蒙层和剪切~~

~~⑤ CSS混合模式~~

~~⑥ CSS自定义属性~~

~~⑦ CSS等比缩放~~

~~⑧ CSS滚动捕捉~~

~~⑨ CSS Gap(沟槽)~~

~~⑩ CSS逻辑属性~~

~~⑪ CSS媒体查询~~

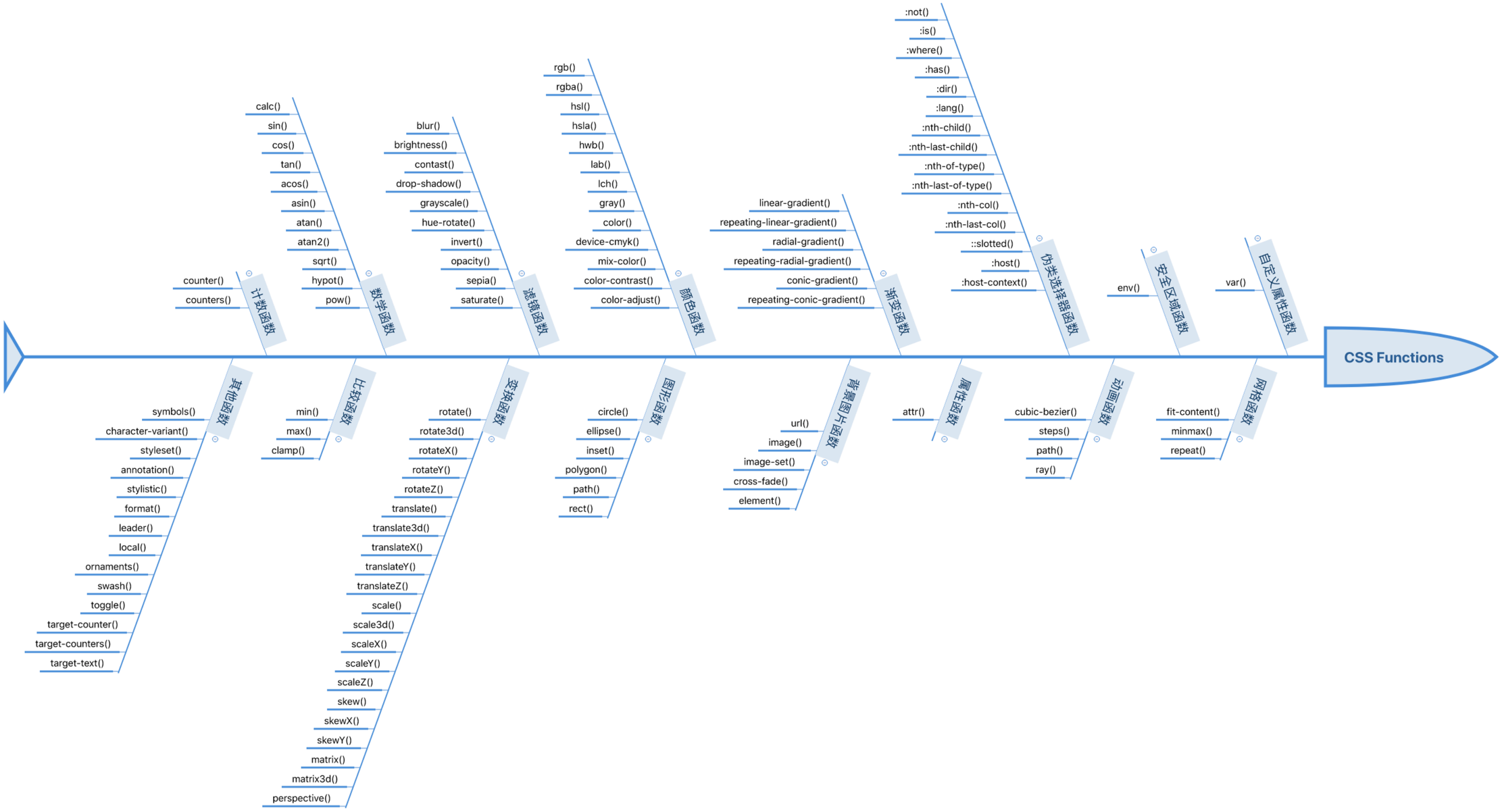
⑫ CSS比较函数

⑬ CSS内容可见性

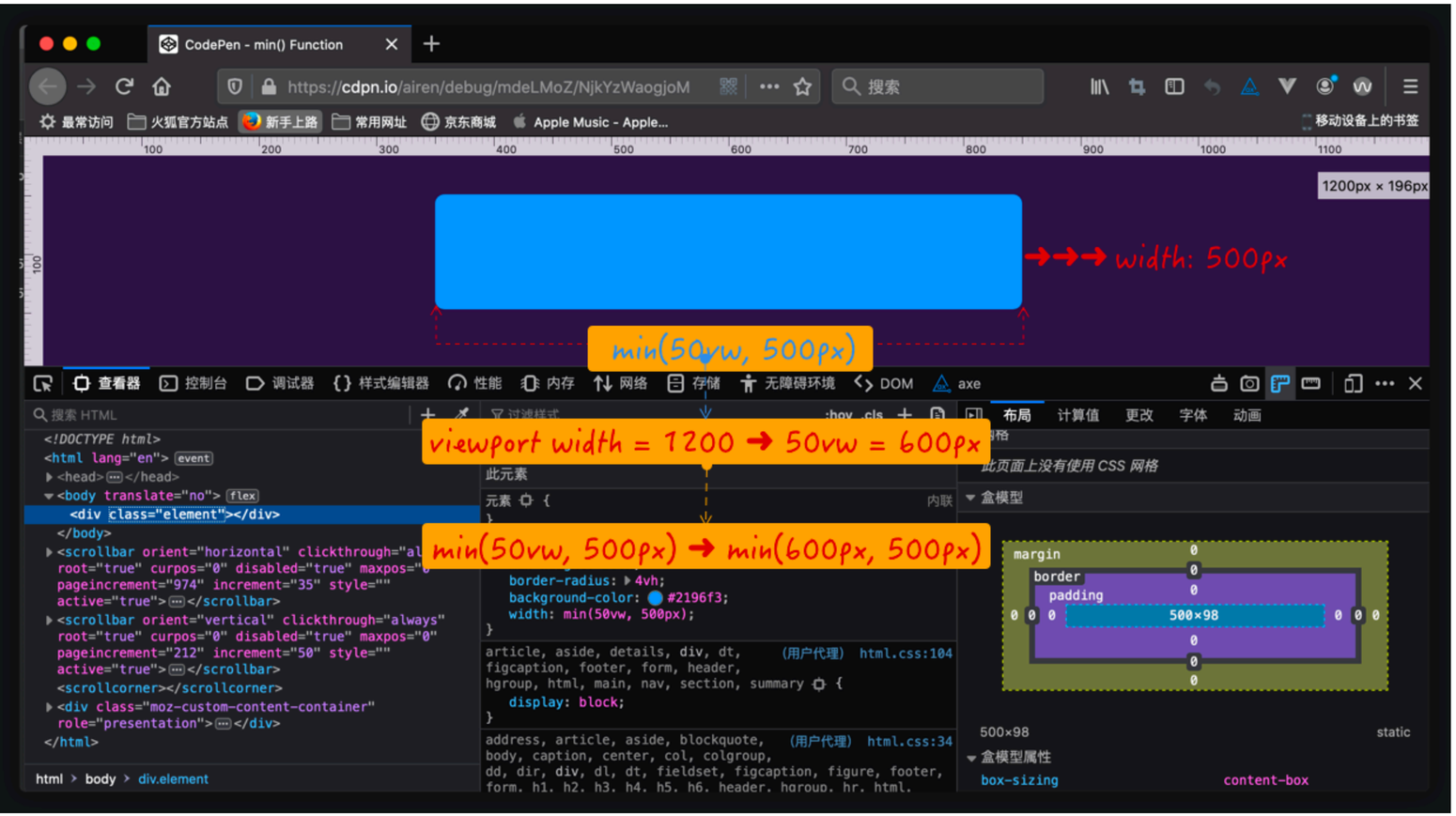
⑭ CSS外在尺寸和内在尺寸

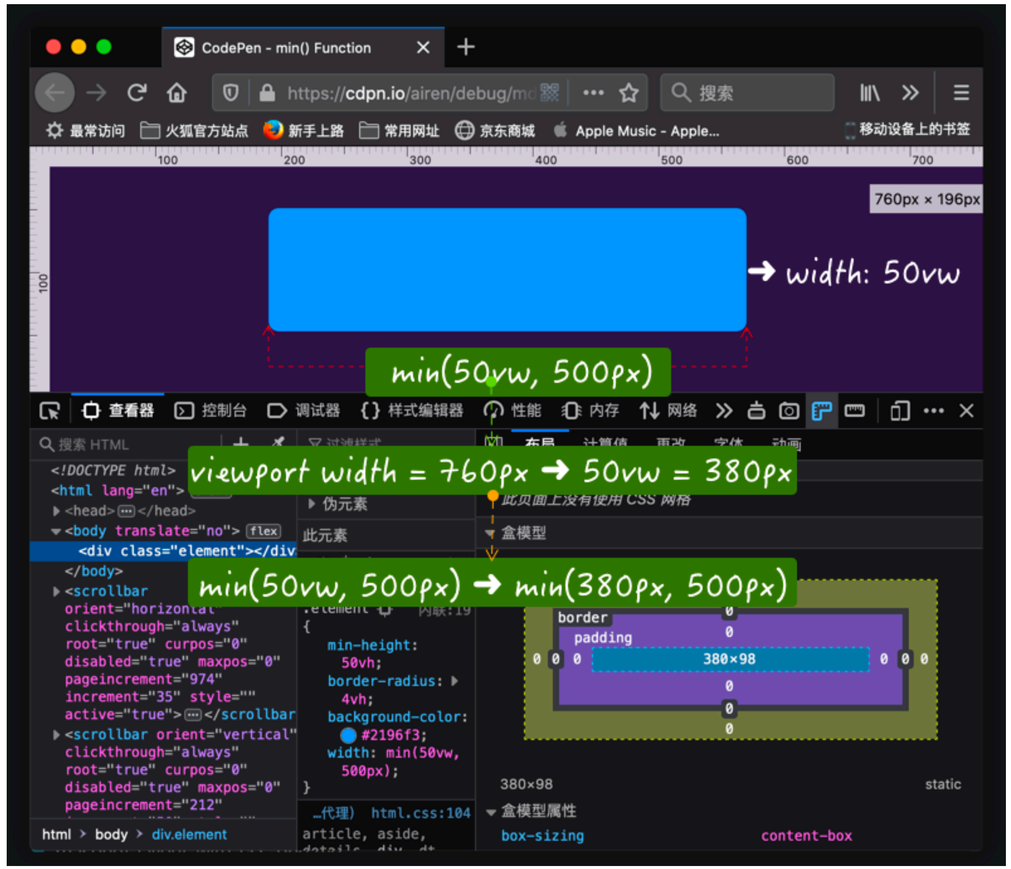
⑮ CSS的Display

CSS 比较函数 min()/max()/clamp()



CSS 比较函数 min()/max()/clamp()

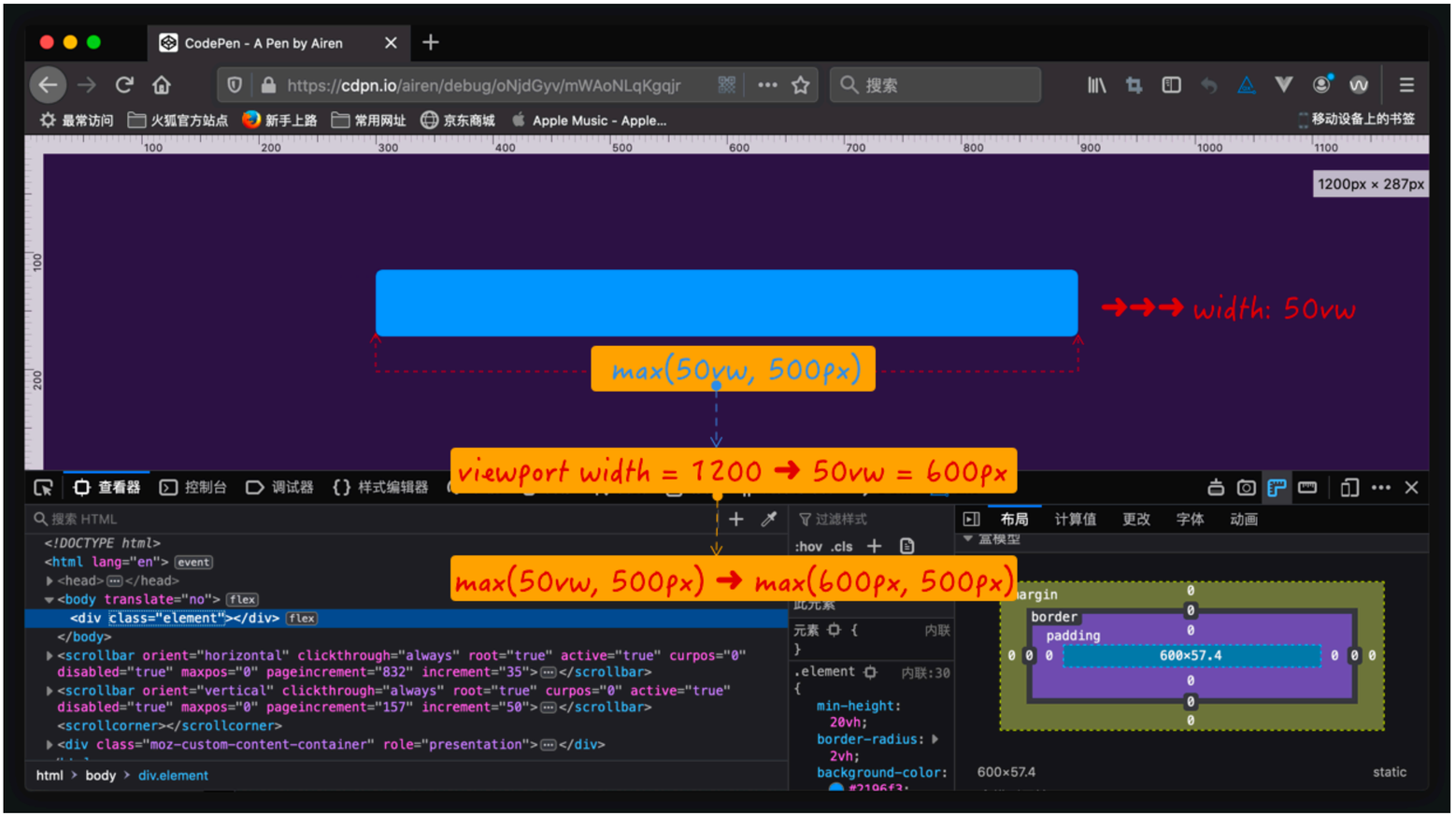


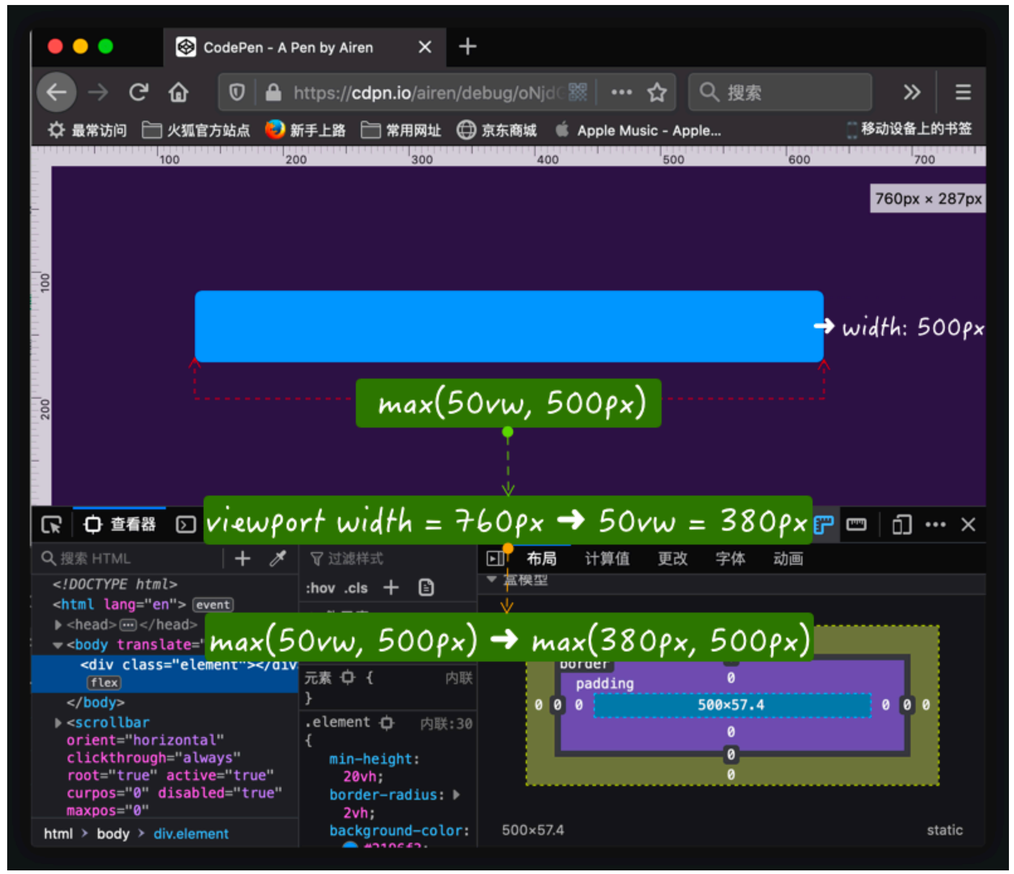



```
1 .container { width: min(50vw, 500px);}
```

CodePen ⇥

CSS 比较函数 min()/max()/clamp()

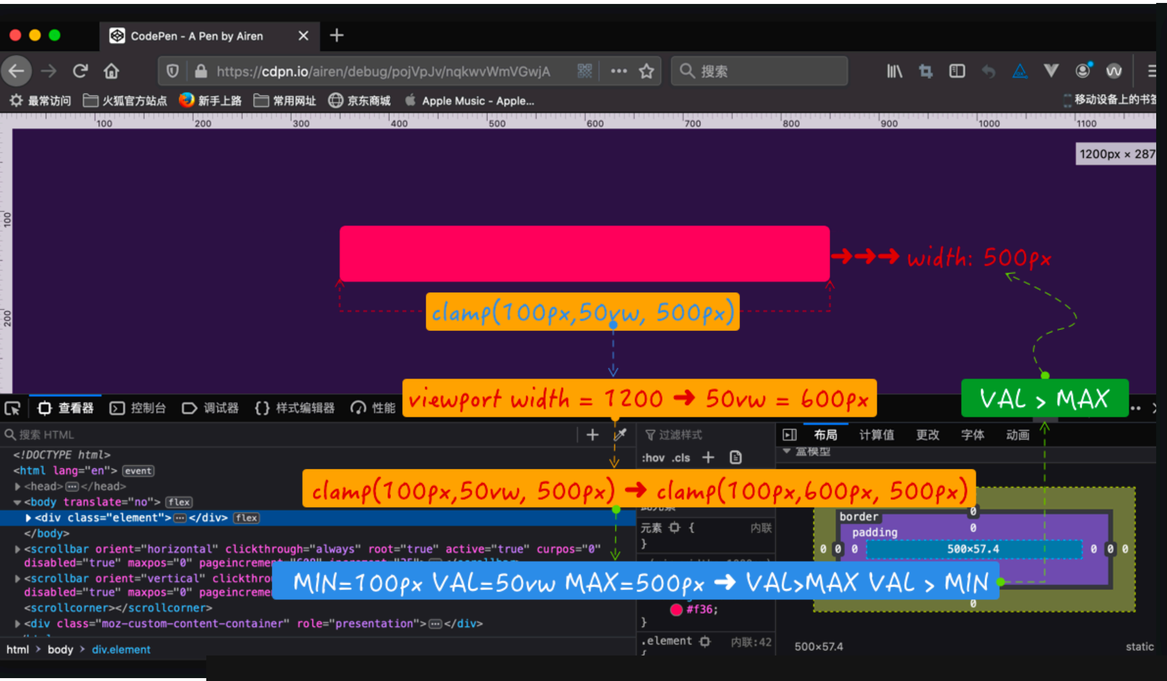


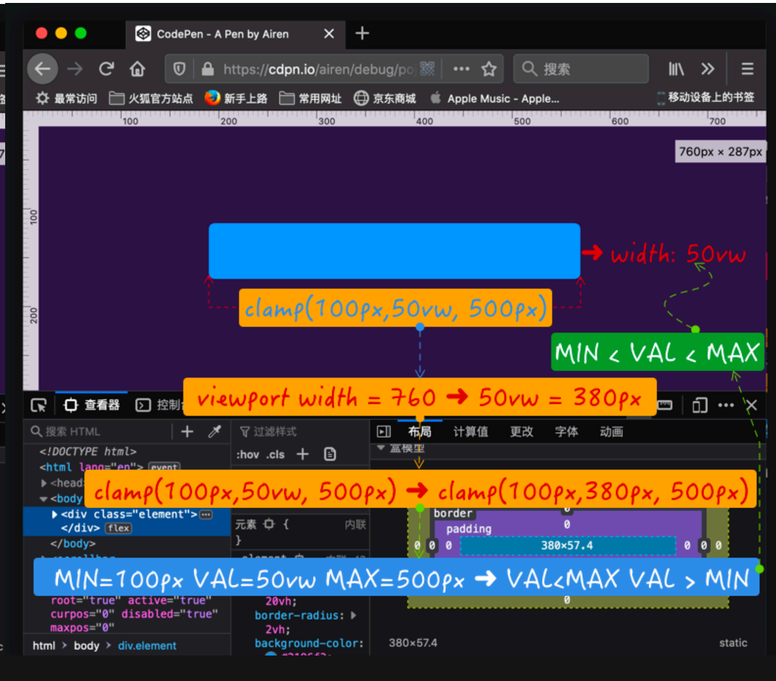



```
1 .container { width: max(50vw,500px);}
```

CodePen →

CSS 比较函数 min()/max()/clamp()





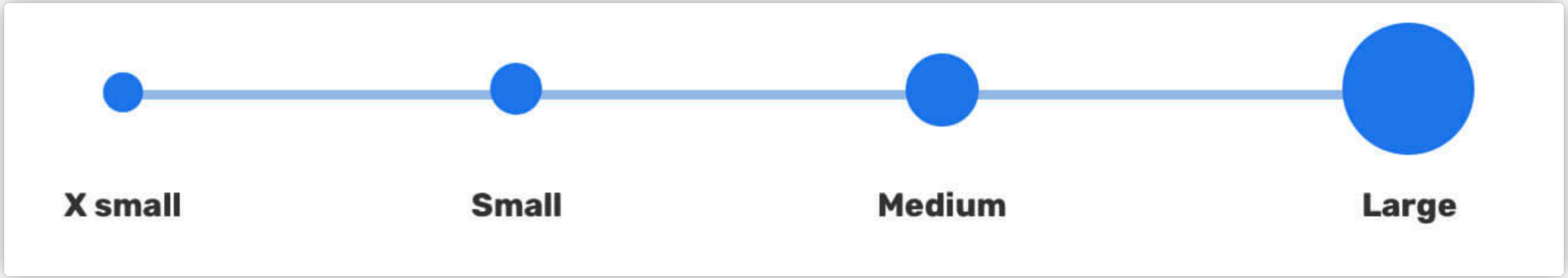

```
1 // clamp(MIN, VAL, MAX)
2 // 如果VAL在MIN和MAX之间，则使用VAL作为函数的返回值；
3 // 如果VAL大于MAX，则使用MAX作为函数的返回值；
4 // 如果VAL小于MIN，则使用MIN作为函数的返回值
5 .container { clamp(100px, 50vw, 500px)}
```

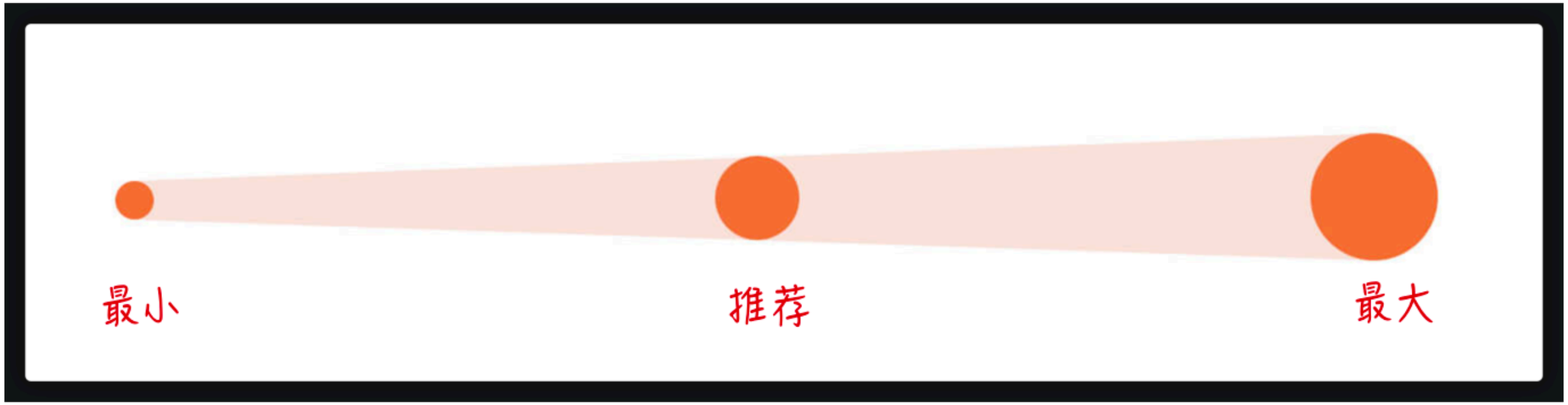
CodePen →

CSS 比较函数 min()/max()/clamp()

```css
.element {
  width: clamp(100px, 50vw, 500px);

// 50vw相当于视窗宽度的一半，
// 如果视窗宽度是760px的话，那么50vw相当等于380px
  width: clamp(100px, 380px, 500px);

// 用min()和max()描述
// clamp(MIN, VAL, MAX) = max(MIN, min(VAL, Max))
  width: max(100px, min(380px, 500px))

// min(380px, 500px)返回的值是380px
  width: max(100px, 380px)

// max(100px, 380px)返回的值是380px
  width: 380px;
}
```

CSS 比较函数 min()/max()/clamp()





CSS 比较函数 min()/max()/clamp()

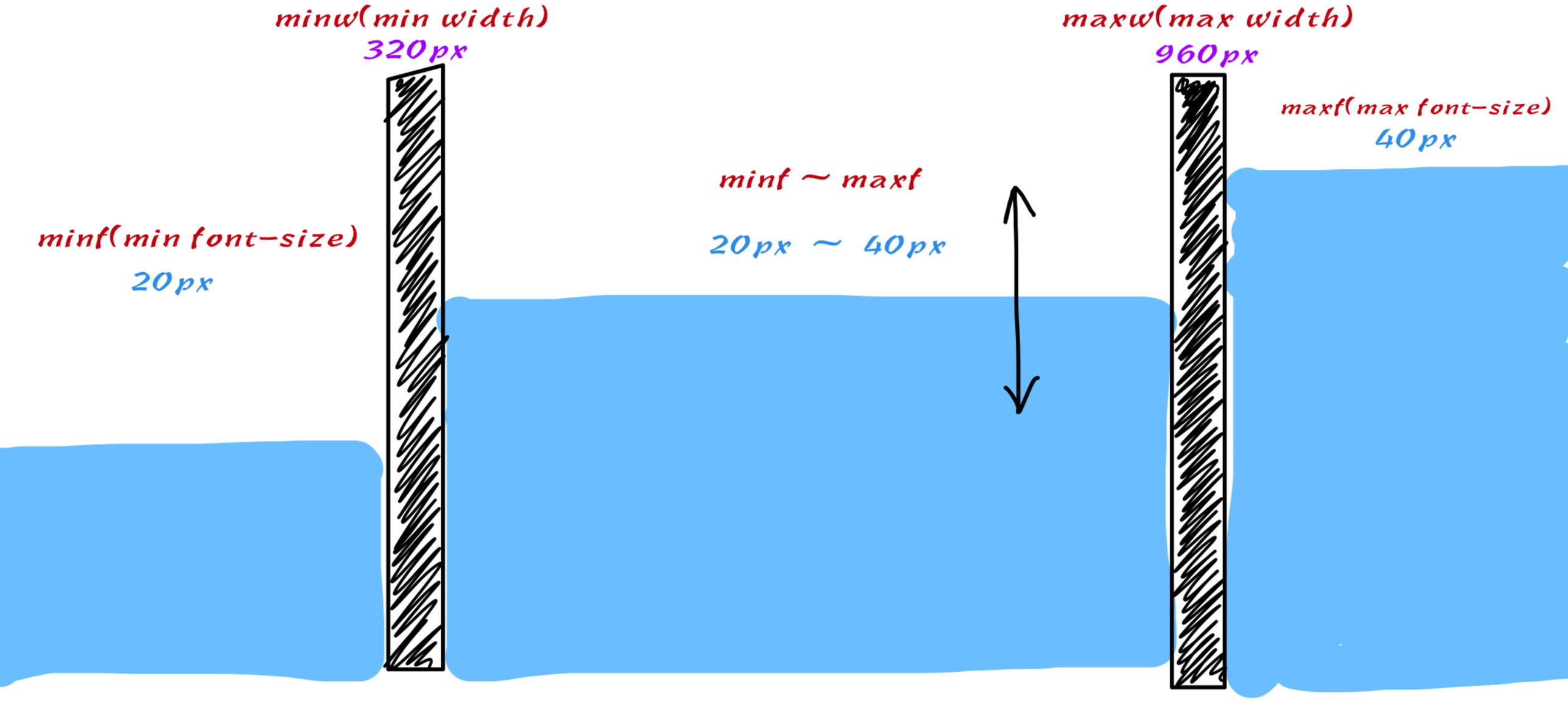
最小 16px —— CSS Comparison Functions

推荐 5vw —— CSS Comparison Functions

最大 50px —— CSS Comparison Functions

```
1 body {
2   font-size: clamp(16px, 5vw, 50px)
3 }
```

CSS 比较函数 min()/max()/clamp()



CSS 比较函数 min()/max()/clamp()

$$\text{minimum font-size} + \left(\frac{\text{current vw} - \text{minimum vw}}{\text{maximum vw} - \text{minimum vw}} \right) * \left(\text{maximum font-size} - \text{minimum font-size} \right)$$

```
font-size:
    calc(
        [minf]px + ([maxf] - [minf]) * ( (100vw - [minw]px) / ([maxw] - [minw]) )
);
```

```
// minf: 20px (min font-size)
// maxf: 40px (max font-size)
// current vw: 100vw
// minw: 320px (min viewport's width)
// maxw: 960px (max viewport's width)
body {
    font-size:
        calc(20px + (40 - 20) * ( (100vw - 320px) /(960 - 320) ));
}
@media only screen and (max-width: 320px) {
  body {
    font-size: 20px;
  }
}
@media only screen and (max-width: 960px) {
  body {
    font-size: 40px;
  }
}
```

CodePen →

CSS 比较函数 min()/max()/clamp()

```css
h1 {
  font-size: clamp(20px, 1rem + 3vw, 40px);
}
```

CodePen →

CSS新特性

~~① CSS伪类选择器~~

~~② CSS颜色~~

~~③ CSS背景~~

~~④ CSS蒙层和剪切~~

~~⑤ CSS混合模式~~

~~⑥ CSS自定义属性~~

~~⑦ CSS等比缩放~~

~~⑧ CSS滚动捕捉~~

~~⑨ CSS Gap(沟槽)~~

~~⑩ CSS逻辑属性~~

~~⑪ CSS媒体查询~~

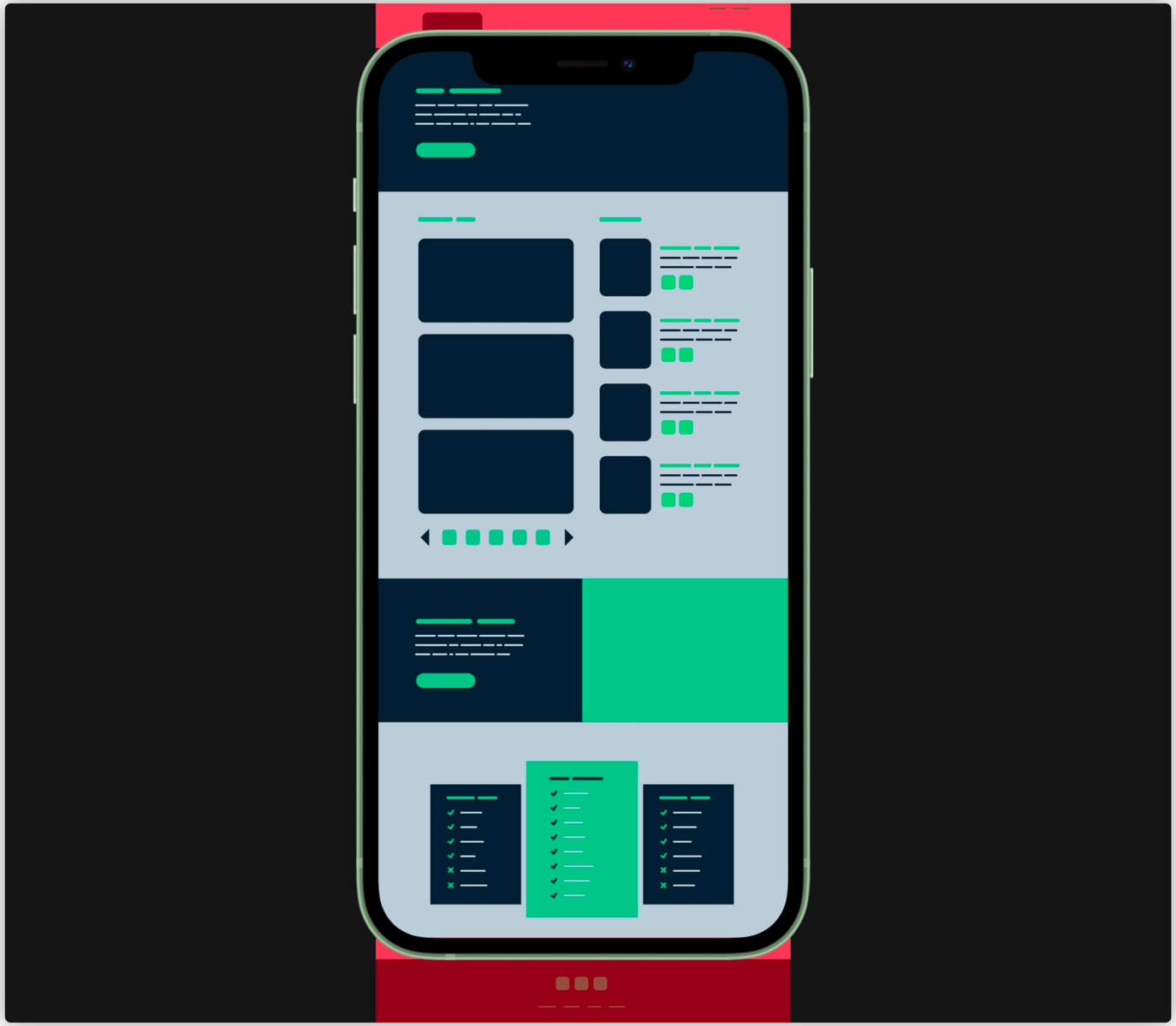
~~⑫ CSS比较函数~~

⑬ CSS内容可见性

⑭ CSS外在尺寸和内在尺寸

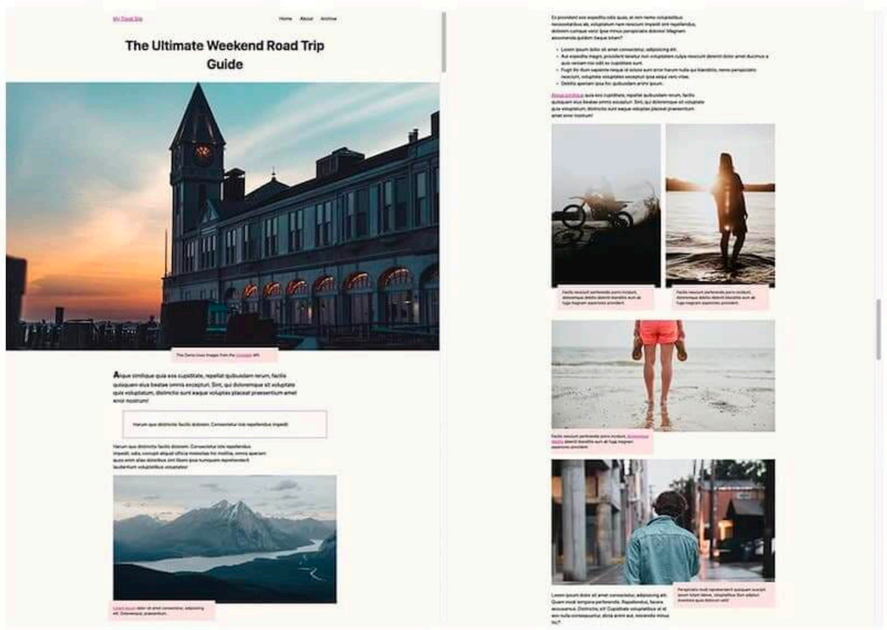
⑮ CSS的Display

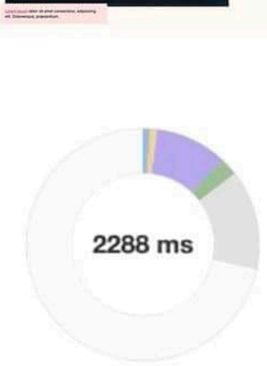
CSS 内容可见性content-visibility



CSS 内容可见性content-visibility

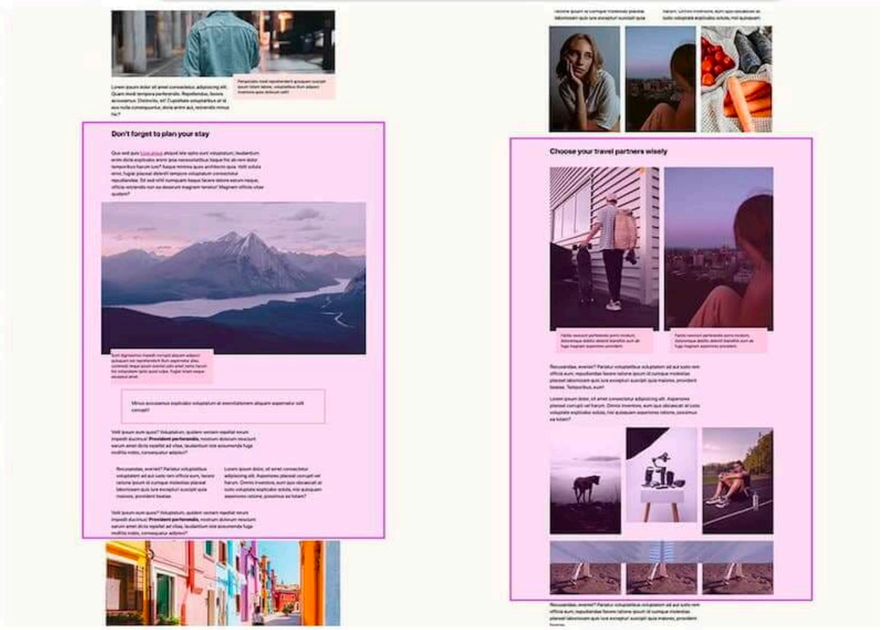
baseline

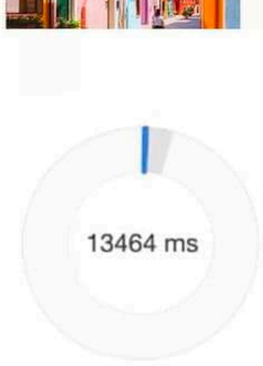




23 ms	Loading
22 ms	Scripting
232 ms	Rendering
55 ms	Painting
320 ms	System
1637 ms	Idle
2288 ms	**Total**

chunks with content-visibility: auto





23 ms	Loading
25 ms	Scripting
30 ms	**Rendering**
34 ms	Painting
476 ms	System
12877 ms	Idle
13464 ms	Total

CSS新特性

① CSS伪类选择器

② CSS颜色

③ CSS背景

④ CSS蒙层和剪切

⑤ CSS混合模式

⑥ CSS自定义属性

⑦ CSS等比缩放

⑧ CSS滚动捕捉

⑨ CSS Gap(沟槽)

⑩ CSS逻辑属性

⑪ CSS媒体查询

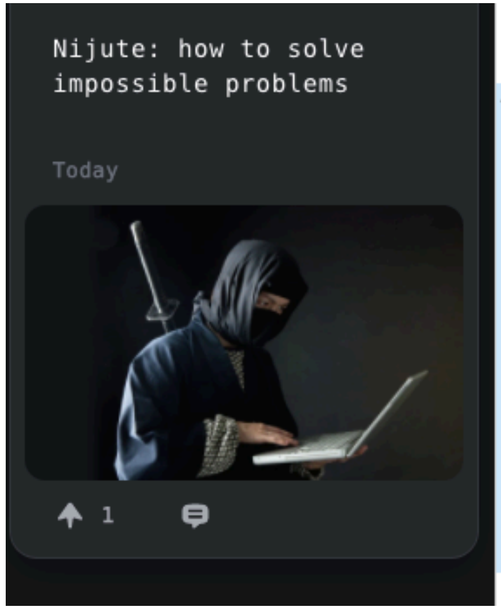
⑫ CSS比较函数

⑬ CSS内容可见性

⑭ CSS外在尺寸和内在尺寸

⑮ CSS的Display

CSS 外在尺寸 vs. 内在尺寸




```
<div class="card__image">
  <!---->
  <img data-src="https://res.cloudinary.com/daily-now/image/upload/f_auto,q_auto/v1/posts/f096be10252e9cac1e9554798e2d114d" data-lowsrc="
```

252 × 158 px (intrinsic: 800 × 533 px)

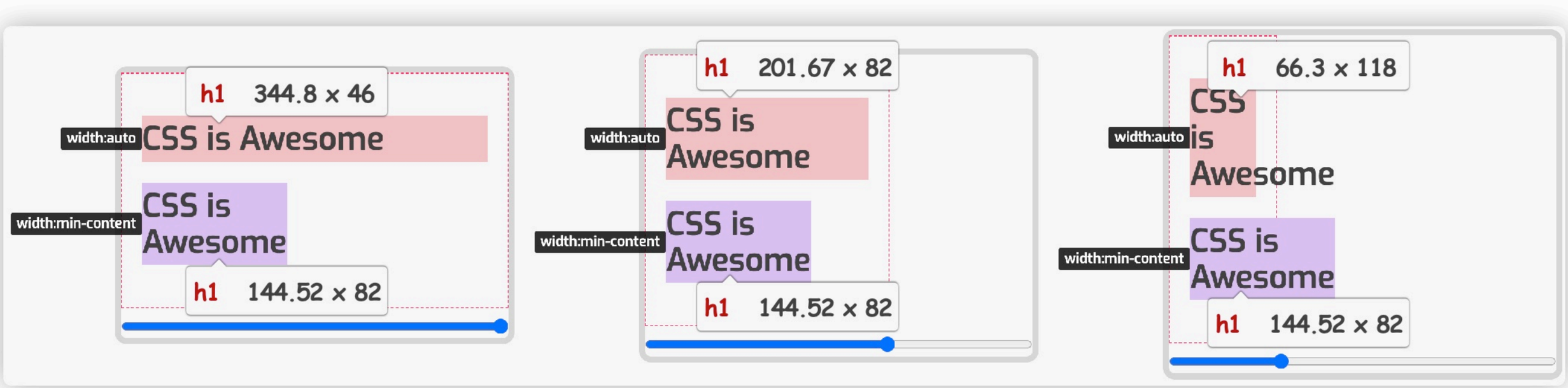
```
src="https://res.cloudinary.com/daily-now/image/upload/f_auto,q_auto/v1/posts/f096be1…">…</img> == $0
</div>
```

CSS 外在尺寸 vs. 内在尺寸

由元素内部内容来决定:

①min-content
②max-content
③fit-content

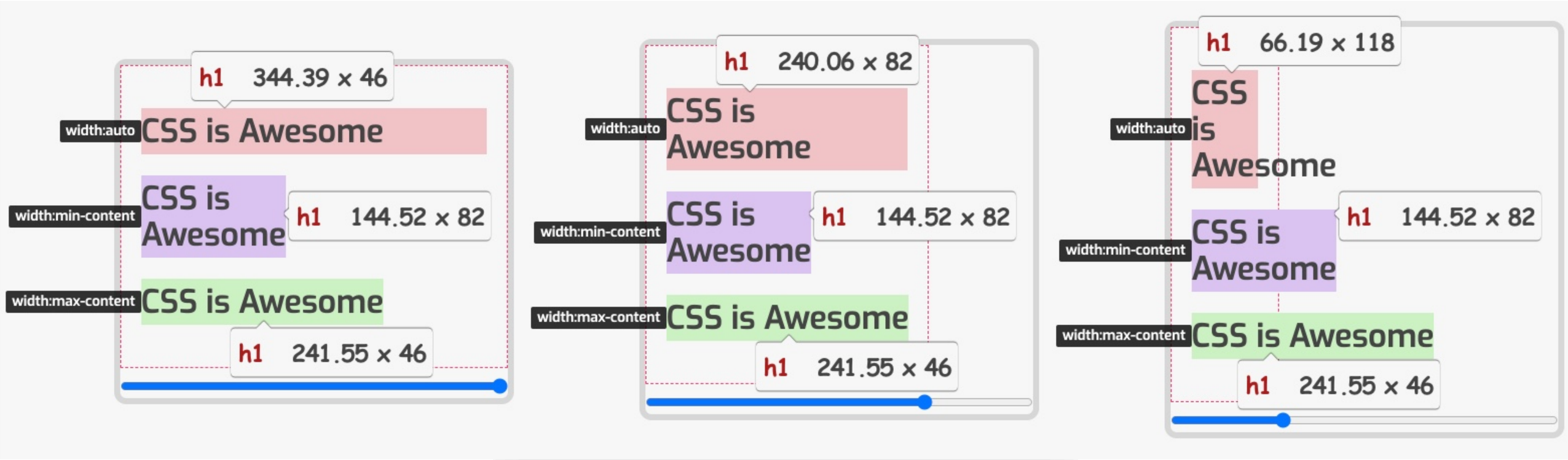
CSS 外在尺寸 vs. 内在尺寸




```
<h1>CSS is Awesome</h1>

h1 {
    width: auto; // 外在尺寸
}

h1 {
    width: min-content; // 内在尺寸
}
```

CodePen →

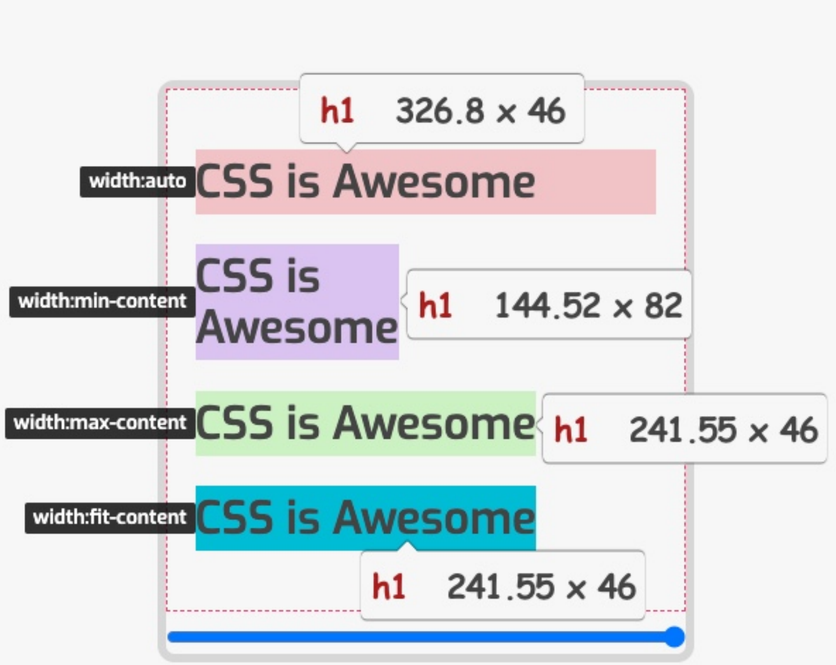
CSS 外在尺寸 vs. 内在尺寸

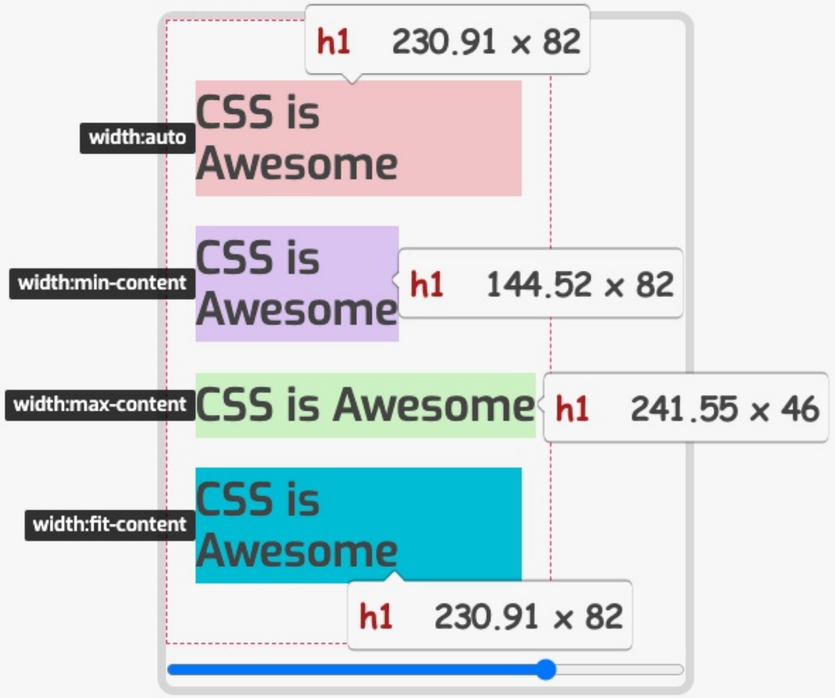


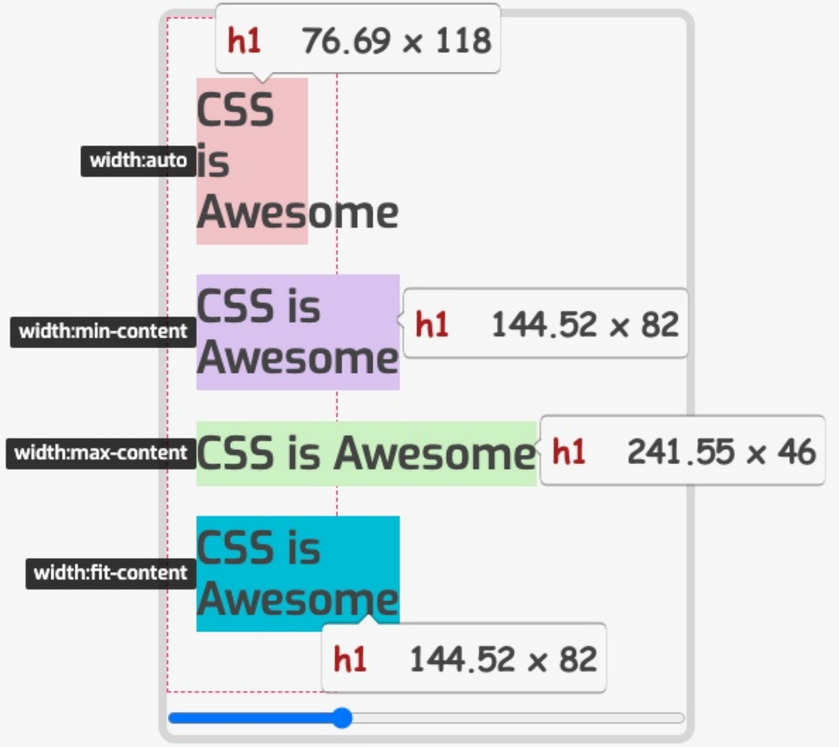

```
// 外在尺寸
h1 { width: auto; }

// 内在尺寸
h1 { width: min-content; }

h1 { width: max-content; }
```

CodePen →

CSS 外在尺寸 vs. 内在尺寸








```
// 外在尺寸
h1 { width: auto; }

// 内在尺寸
h1 { width: min-content; }
h1 { width: max-content; }
h1 { width: fit-content; }

```

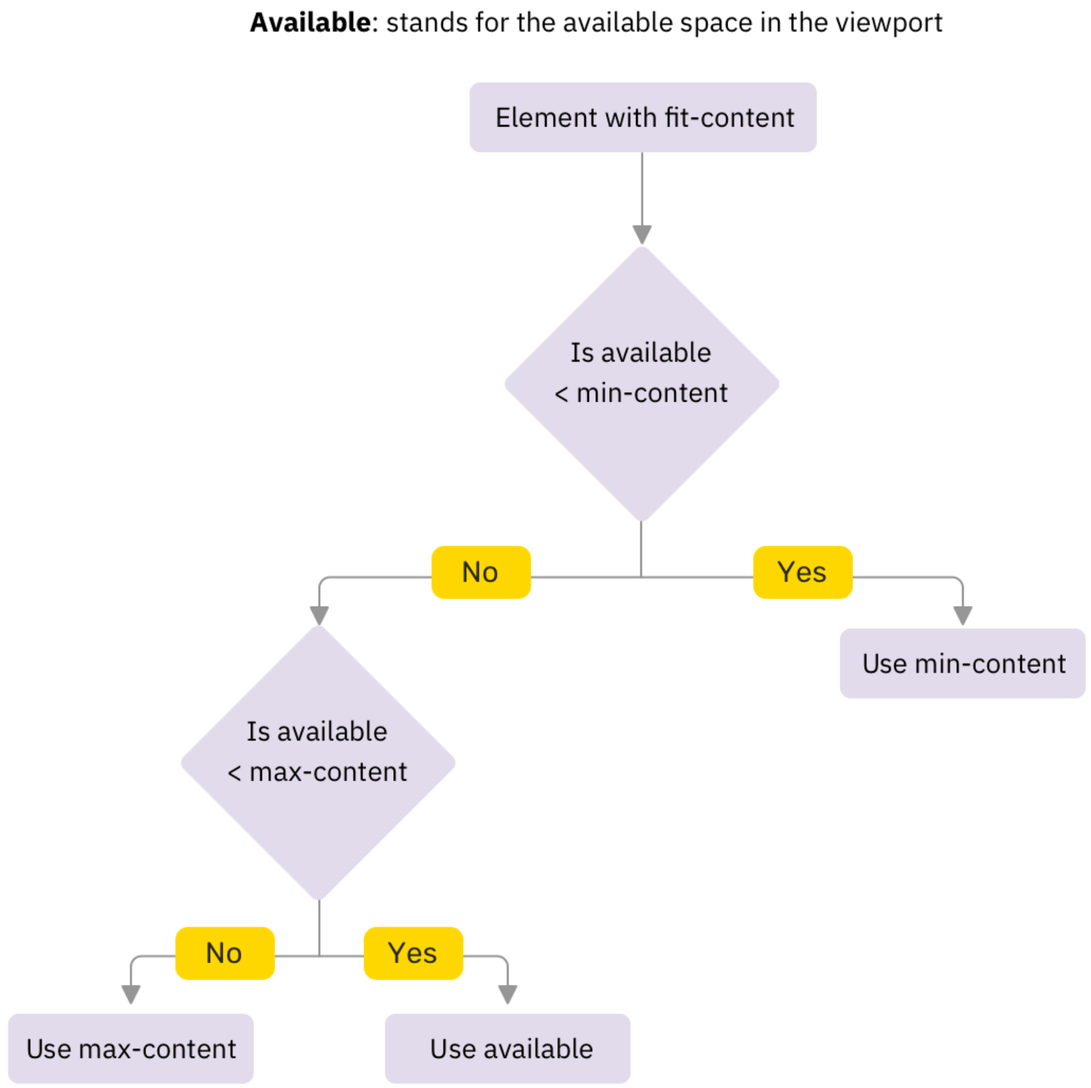
CodePen →

CSS 外在尺寸 vs. 内在尺寸

```css
h1 { width: fit-content; }

// 等同于
h1 {
  width: auto;
  min-width: min-content;
  max-width: max-content;
}

```

CodePen →

CSS 外在尺寸 vs. 内在尺寸



fit-content 相当于min-content 和 max-content,其取值:

① 如果元素的可用空间（Available）充足，fit-content 将使用 max-content

② 如果元素的可用空间（Available）不够充足，比 max-content 小点，那就是用可用空间的值，不会导致内容溢出

③ 如果元素的可用空间（Available）很小，比 min-content 还小,那就使用 min-content

imgURL →|

CSS 外在尺寸 vs. 内在尺寸

min/max/fit-content使用:

① min/max/fit-content 用于 flex-basis 无效

② fit-content 用于设置网格轨道尺寸的属性上无效

③ 网格项目或Flex项目上显式设置width:fit-content也无效,因为它们的默认宽度是min-content

④ 最好不要在 min-width 或 max-width 上使用 fit-content，易造成混乱，建议在 width 上使用fit-content

CSS新特性

~~① CSS伪类选择器~~

~~② CSS颜色~~

~~③ CSS背景~~

~~④ CSS蒙层和剪切~~

~~⑤ CSS混合模式~~

~~⑥ CSS自定义属性~~

~~⑦ CSS等比缩放~~

~~⑧ CSS滚动捕捉~~

~~⑨ CSS Gap(沟槽)~~

~~⑩ CSS逻辑属性~~

~~⑪ CSS媒体查询~~

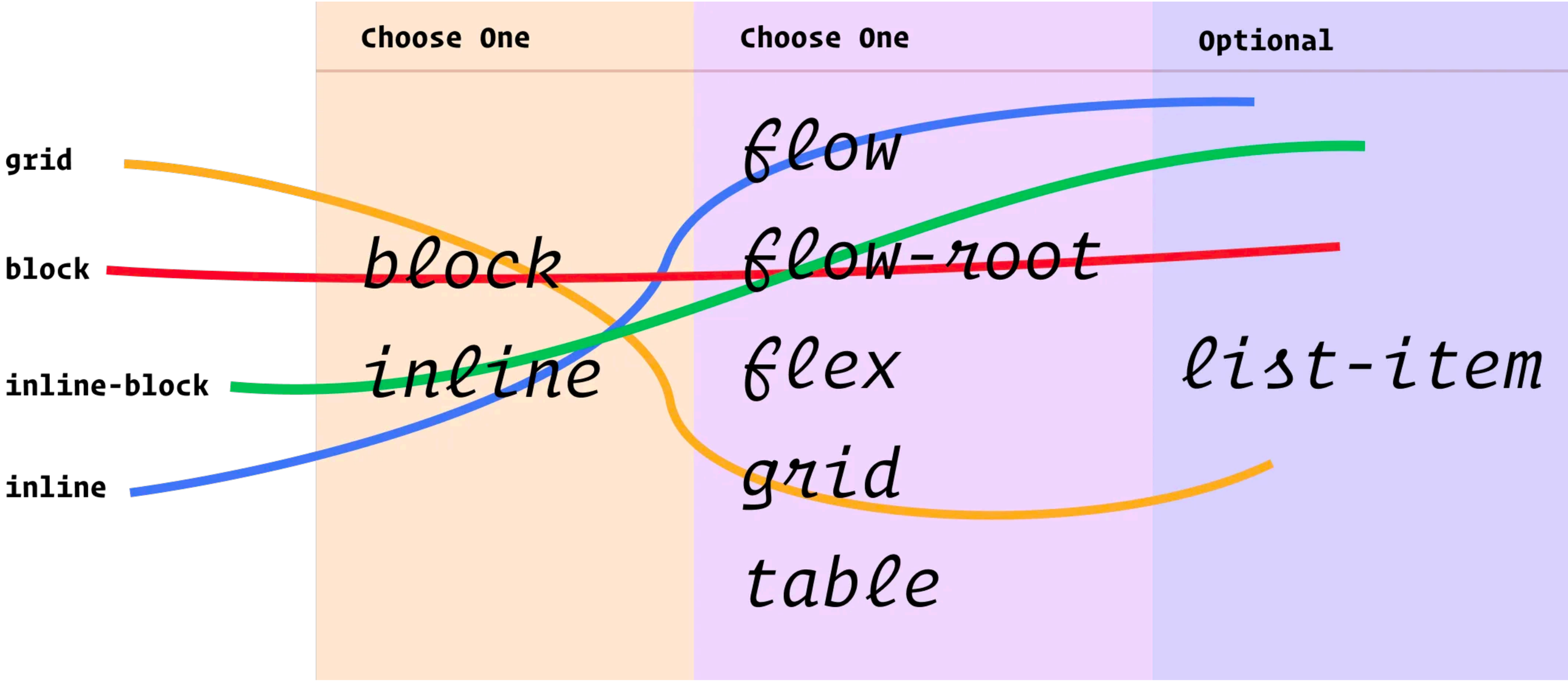
~~⑫ CSS比较函数~~

~~⑬ CSS内容可见性~~

~~⑭ CSS外在尺寸和内在尺寸~~

⑮ CSS的Display

CSS 的display



CSS 的display

单个值	两个值	描述
block	block flow	正常流内的块级盒子
inline	inline flow	正常流内的内联级盒子
inline-block	inline flow-root	定义一个BFC的内联级盒子
list-item	block flow list-item	正常文档流和带有附加标记的块级盒子
flow-root	block flow-root	定义一个BFC的块级盒子
flex	block flex	带有内部伸缩布局的块级盒子
inline-flex	inline flex	带有内部伸缩布局的内联级盒子
grid	block grid	带有内部网格布局的块级盒子
inline-grid	inline grid	带有内部网格布局的内联级盒子
table	block table	带有内部表格布局的块级盒子
inline-table	inline table	带有内部表格布局的内联级盒子
none		从盒子树中移除，包括其所有后代元素
contents		元素替换为框树中的内容

CSS 的display

'contents'

The element itself does not generate any boxes, but its children and pseudo-elements still generate boxes and text runs as normal. For the purposes of box generation and layout, the element must be treated as if it had been replaced in the element tree by its contents (including both its source-document children and its pseudo-elements, such as '::before' and '::after' pseudo-elements, which are generated before/after the element's children as normal).

设置了display: contents的元素自身将不会产生任何盒子，但是它的子元素能正常展示

https://www.w3.org/TR/css-display-3/#box-generation

CSS 的display

```html
<!-- HTML -->
<div class="outer">
  I'm some content
  <div class="inner">I'm some inner content</div>
</div>

/* CSS */
.outer {
  border: 2px solid lightcoral;
  background-color: lightpink;
  padding: 20px;
}

.inner {
  background-color: #ffdb3a;
  padding: 20px;
}
```

CSS 的display

I'm some content

I'm some inner content

Set display: initial(block) contents none (to the element with the name of the outer class)

```
<!-- HTML -->
<div class="outer">
  I'm some content
  <div class="inner">I'm some inner content</div>
</div>
```

I'm some content

I'm some inner content

Set display: initial(block) contents none (to the element with the name of the outer class)

```
<!-- HTML -->

  I'm some content
  <div class="inner">I'm some inner content</div>

```

Set display: initial(block) contents none (to the element with the name of the outer class)

设置了display: contents的元素本身不会被渲染，但是其子元素能够正常被渲染

CodePen →

CSS 的display

```
<header> flex
  <a href="https://www.imgcook.com" target="_blank"> flex item
    <img src="https://gw.alicdn.com/tfs/TB1YzajyNnaK1RjSZFtXXbC2VXa
    4.png" alt="imgcook A cook who can transform design to code">
  </a>
  <ul> flex item
    <li>
      <a href="https://www.imgcook.com/workspace" target="_blank">
      Workspace</a>
    </li>
    <li>…</li>
    <li>…</li>
    <li>…</li>
    <li>…</li>
  </ul>
  <div class="login"> flex item
    <img src="https://gw.alicdn.com/tfs/TB1v3KJwHvpK1RjSZPiXXbmwXXa
    00.svg" alt="Login">
  </div>
</header>
```

```
<header style="--display:grid;"> grid
  <a href="https://www.imgcook.com" target="_blank"> grid item
    <img src="https://gw.alicdn.com/tfs/TB1YzajyNnaK1RjSZFtXXbC2VXa
    4.png" alt="imgcook A cook who can transform design to code">
  </a>
  <ul> grid item
    <li>
      <a href="https://www.imgcook.com/workspace" target="_blank">
      Workspace</a>
    </li>
    <li>…</li>
    <li>…</li>
    <li>…</li>
    <li>…</li>
  </ul>
  <div class="login"> grid item
    <img src="https://gw.alicdn.com/tfs/TB1v3KJwHvpK1RjSZPiXXbmwXXa
    00.svg" alt="Login">
  </div>
</header>
```

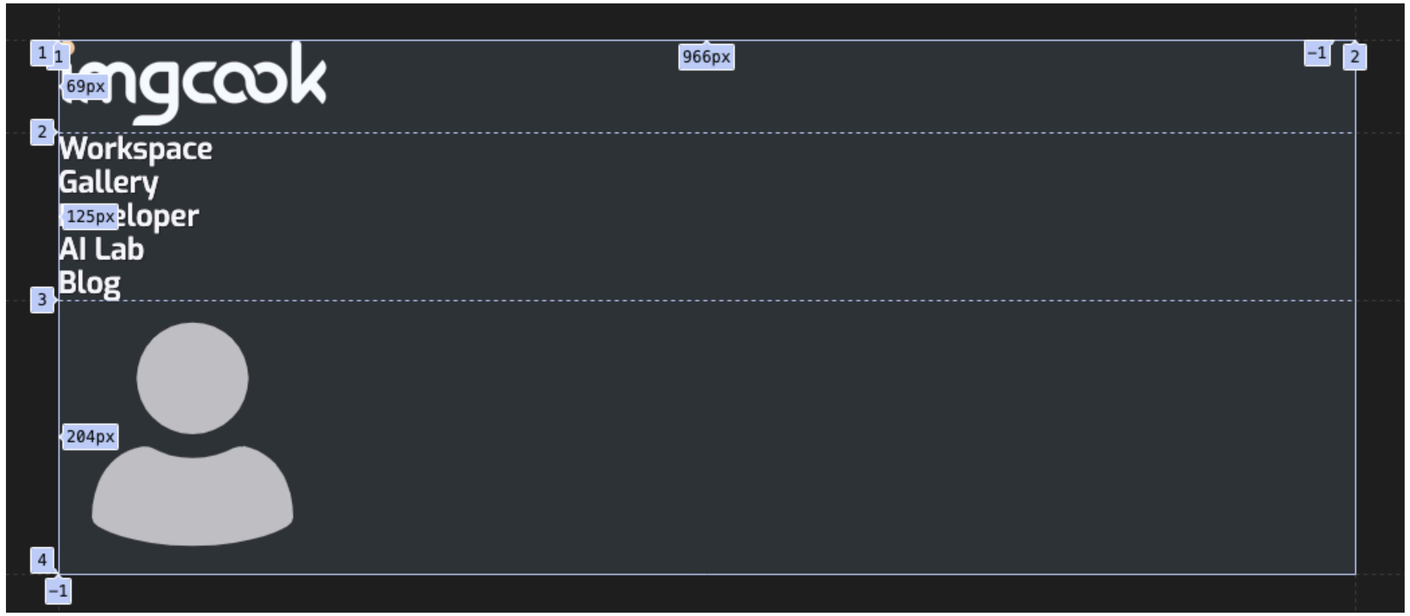
```
1 header {
2     display:
3         flex |
4         inline-flex |
5         grid |
6         inline-grid
7 }
```

可以让非网格容器或Flexbox容器的子元素顺利变成网格项目或Flex项目

CSS 的display

```
<header> flex
  <a href="https://www.imgcook.com" target="_blank"> flex item
    <img src="https://gw.alicdn.com/tfs/TB1YzajyNnaK1RjSZFtXXbC2VXa
    4.png" alt="imgcook A cook who can transform design to code">
  </a>
  <ul> flex item
    <li>
      <a href="https://www.imgcook.com/workspace" target="_blank">
      Workspace</a>
    </li>
    <li>…</li>
    <li>…</li>
    <li>…</li>
    <li>…</li>
  </ul>
  <div class="login"> flex item
    <img src="https://gw.alicdn.com/tfs/TB1v3KJwHvpK1RjSZPiXXbmwXXa
    00.svg" alt="Login">
  </div>
</header>
```


```
<header style="--display:grid;"> grid
  <a href="https://www.imgcook.com" target="_blank"> grid item
    <img src="https://gw.alicdn.com/tfs/TB1YzajyNnaK1RjSZFtXXbC2VXa
    4.png" alt="imgcook A cook who can transform design to code">
  </a>
  <ul> grid item
    <li>
      <a href="https://www.imgcook.com/workspace" target="_blank">
      Workspace</a>
    </li>
    <li>…</li>
    <li>…</li>
    <li>…</li>
    <li>…</li>
  </ul>
  <div class="login"> grid item
    <img src="https://gw.alicdn.com/tfs/TB1v3KJwHvpK1RjSZPiXXbmwXXa
    00.svg" alt="Login">
  </div>
</header>
```



CSS 的display

```
<header> flex
  <a href="https://www.imgcook.com" target="_blank"> flex item
    <img src="https://gw.alicdn.com/tfs/TB1YzajyNnaK1RjSZFtXXbC2VXa
    4.png" alt="imgcook A cook who can transform design to code">
  </a>
  <ul> flex item
    <li>
      <a href="https://www.imgcook.com/workspace" target="_blank">
      Workspace</a>
    </li>
    <li>…</li>
    <li>…</li>
    <li>…</li>
    <li>…</li>
  </ul>
  <div class="login"> flex item
    <img src="https://gw.alicdn.com/tfs/TB1v3KJwHvpK1RjSZPiXXbmwXXa
    00.svg" alt="Login">
  </div>
</header>
```

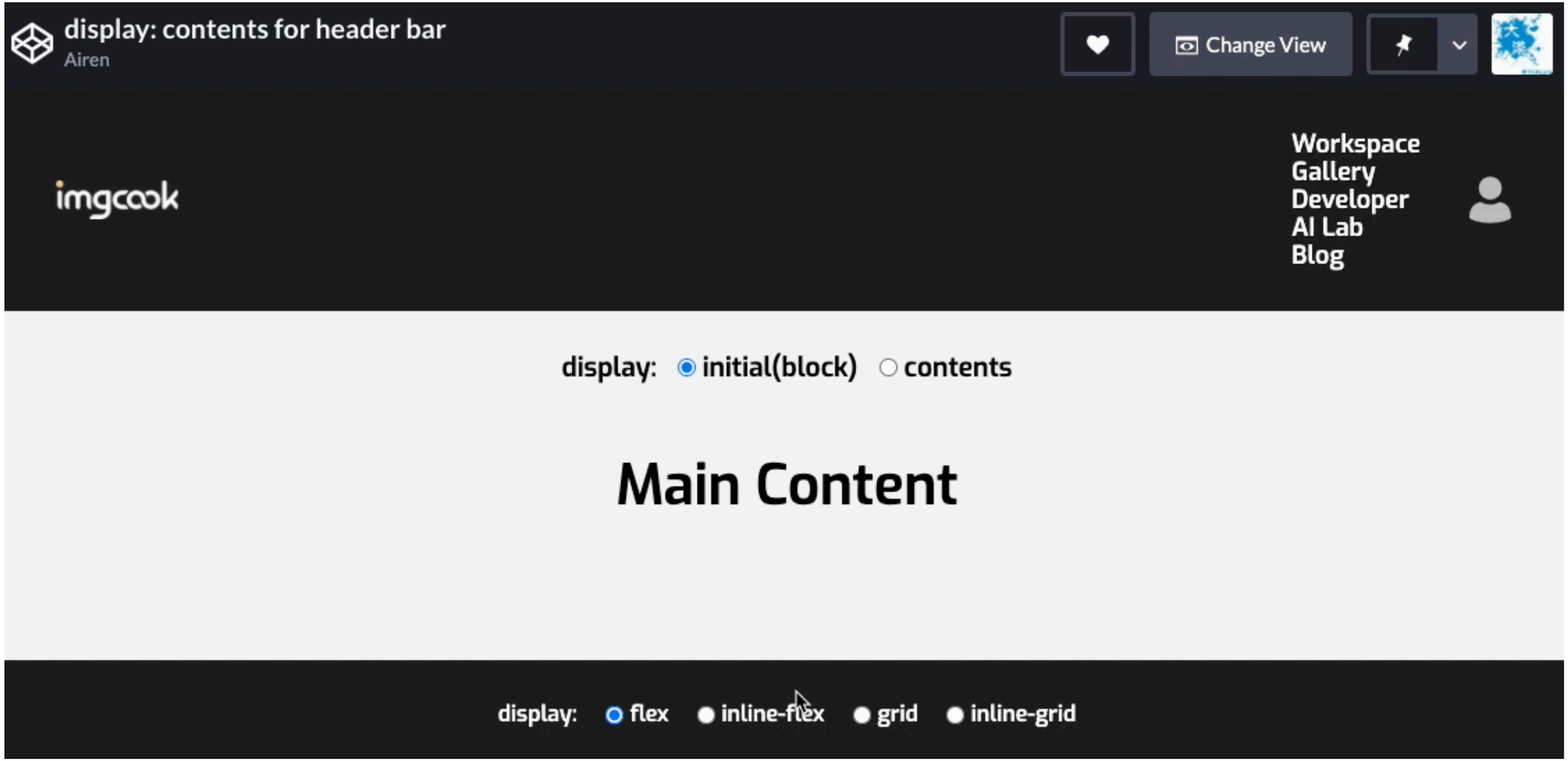
```
<header style="--display:grid;"> grid
  <a href="https://www.imgcook.com" target="_blank"> grid item
    <img src="https://gw.alicdn.com/tfs/TB1YzajyNnaK1RjSZFtXXbC2VXa
    4.png" alt="imgcook A cook who can transform design to code">
  </a>
  <ul> grid item
    <li>
      <a href="https://www.imgcook.com/workspace" target="_blank">
      Workspace</a>
    </li>
    <li>…</li>
    <li>…</li>
    <li>…</li>
    <li>…</li>
  </ul>
  <div class="login"> grid item
    <img src="https://gw.alicdn.com/tfs/TB1v3KJwHvpK1RjSZPiXXbmwXXa
    00.svg" alt="Login">
  </div>
</header>
```

```
<header class="flex" style="--display:flex; grid-template-columns: none;"> flex
  <a href="https://www.imgcook.com" target="_blank" class="logo"> flex item
    <img src="https://gw.alicdn.com/tfs/TB1YzajyNnaK1RjSZFtXXbC2VXa-200-64.png"
    alt="imgcook A cook who can transform design to code">
  </a>
  <ul class="menu" style="display: contents;">
    <li> flex item
      <a href="https://www.imgcook.com/workspace" target="_blank">Workspace</a>
    </li>
    <li>…</li> flex item
    <li>…</li> flex item
    <li>…</li> flex item
    <li>…</li> flex item
  </ul>
  <div class="login"> flex item
    <img src="https://gw.alicdn.com/tfs/TB1v3KJwHvpK1RjSZPiXXbmwXXa-200-200.svg"
    alt="Login">
  </div>
</header>
```

```
<header class="grid" style="--display:grid; grid-template-columns: 1fr repeat(5,
max-content) min-content;"> grid
  <a href="https://www.imgcook.com" target="_blank" class="logo"> grid item
    <img src="https://gw.alicdn.com/tfs/TB1YzajyNnaK1RjSZFtXXbC2VXa-200-64.png"
    alt="imgcook A cook who can transform design to code">
  </a>
  <ul class="menu" style="display: contents;">
    <li> grid item
      <a href="https://www.imgcook.com/workspace" target="_blank">Workspace</a>
    </li>
    <li>…</li> grid item
    <li>…</li> grid item
    <li>…</li> grid item
    <li>…</li> grid item
  </ul>
  <div class="login"> grid item
    <img src="https://gw.alicdn.com/tfs/TB1v3KJwHvpK1RjSZPiXXbmwXXa-200-200.svg"
    alt="Login">
  </div>
</header>
```

CodePen →

CSS 的display



CodePen →

待续...

THANK YOU