

F(X) TEAM · 午夜识堂

CSS新特性

@大漠#2021-05-12#杭州

FED × 铁 × F(X)TEAM 联合出品

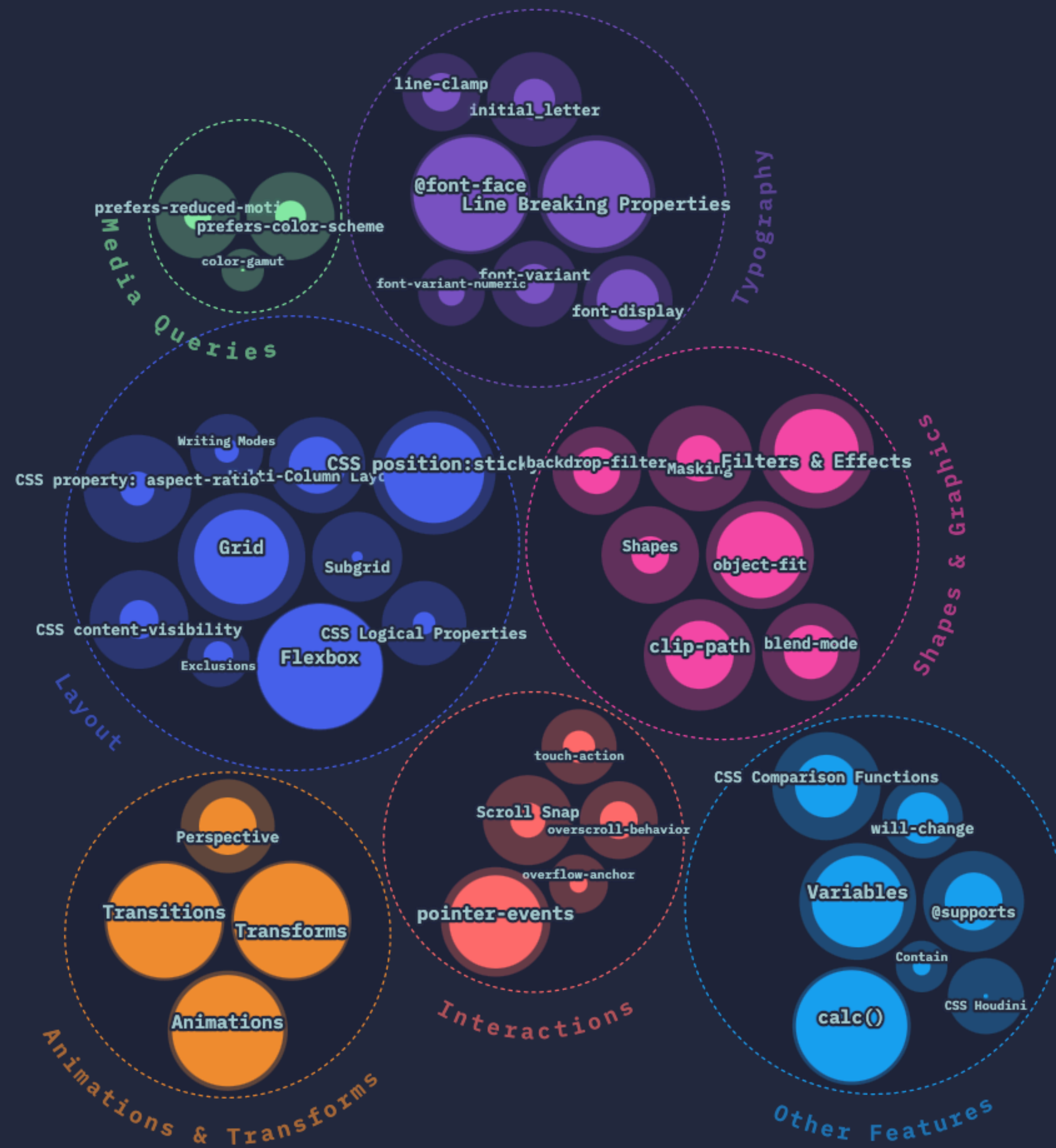
CSS is simple, but not easy!

**SIMPLE ≠ EASY**

CSS IS  
AWESOME

...BUT!

# CSS新特性



# 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()

 **Elad Shechter**  
@eladsc

🧐 CSS Quiz

What will be the text color of this <p> HTML element:  
<main class="main">  
  <p class="p">..  
</main>

These are the styles:

```
.header .p,  
.main .p{ color: green; }  
  
:is(.header, .main) .p{ color: purple; }  
  
:where(.header, .main) .p{ color: red; }  
  
.p{ color: blue;}
```

green	30%
purple	26.7%
red	21.1%
blue	22.2%

90 votes · Final results  
10:03 PM · Mar 16, 2021

11 3 Copy link to Tweet

[Tweet →](#)

[CodePen →](#)



# CSS 伪类选择器 — :is() vs. :where()

```
Modern CSS Pseudo-Class Selectors

1 /* Before */
2 .header p,
3 .main p {
4     color: purple;
5 }
6
7 /* After */
8 :is(.header, .main) .p {
9     color: purple;
10 }
```

```
:is(.header, .main) .p { ExWxbKe:formatted:43
  color: purple;
}
.header .p, .main .p { ExWxbKe:formatted:39
  color: green;
}
.p { ExWxbKe:formatted:51
  color: blue;
}
:where(.header, .main) .p { ExWxbKe:formatted:47
  color: red;
}
```

# CSS 伪类选择器 — :is() vs. :where()

选择相同的元素

**.header .p, .main p** { ... }

两个类权重

**:is(.header, .main) .p** { ... }

:is() 等同于一个类权重      一个类权重

**:where(.header, .main) .p** { ... }

无权重      一个类权重

**.p** { ... }

一个类权重

相同的权重

相同的权重

# CSS 伪类选择器 — :is() vs. :where()



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

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

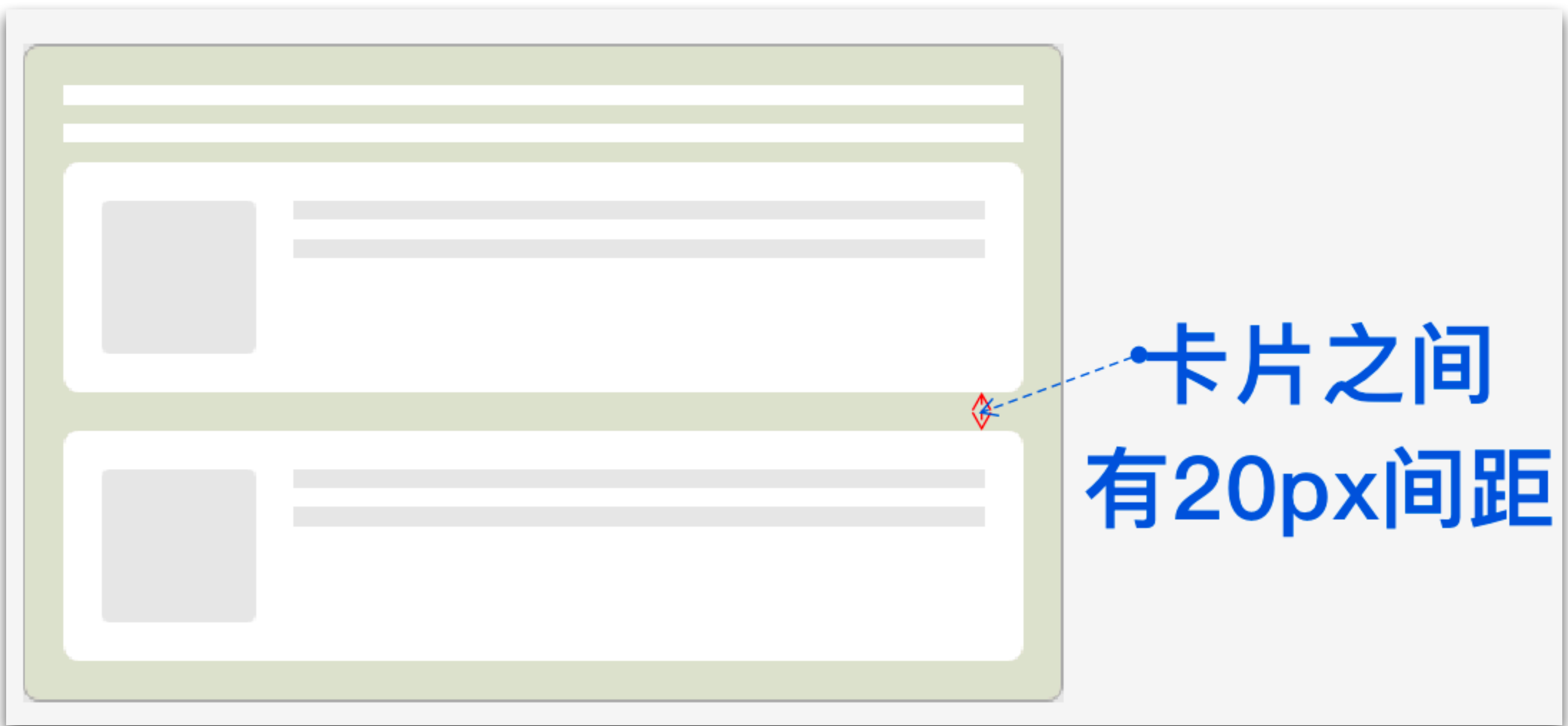
```
.p {
  color: blue;
}
```

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

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

[CodePen](#) →

# CSS 伪类选择器 — :not() vs. :has()



卡片之间  
有20px间距

# CSS 伪类选择器 — :not() vs. :has()

```
• • •  
  
/* Before */  
li + li {  
    margin-top: 20px;  
}  
  
// 或  
li {  
    margin-bottom: 20px;  
}  
  
li:last-child {  
    margin-bottom: 0  
}
```

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

[CodePen →](#)

# CSS 伪类选择器 — :not() vs. :has()

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

```
/* 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;
}
```

H1 Level Title

H2 Level Title

Text Paragraphs

# CSS 伪类选择器 — :empty vs. :blank



```
<!-- 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 */
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";
}
```

item

Pseudo Element

# CSS 伪类选择器 — :empty vs. :blank

```
/* 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 →](#)



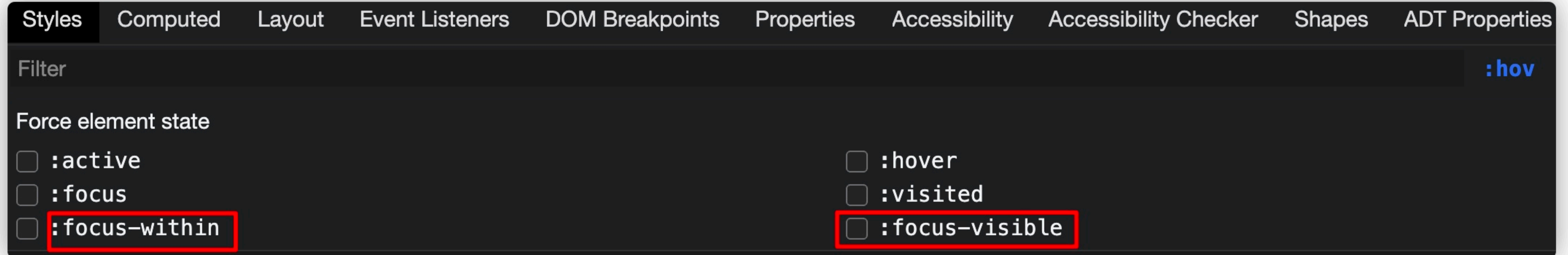
# CSS 伪类选择器 — :empty vs. :blank

```
/* CSS */
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



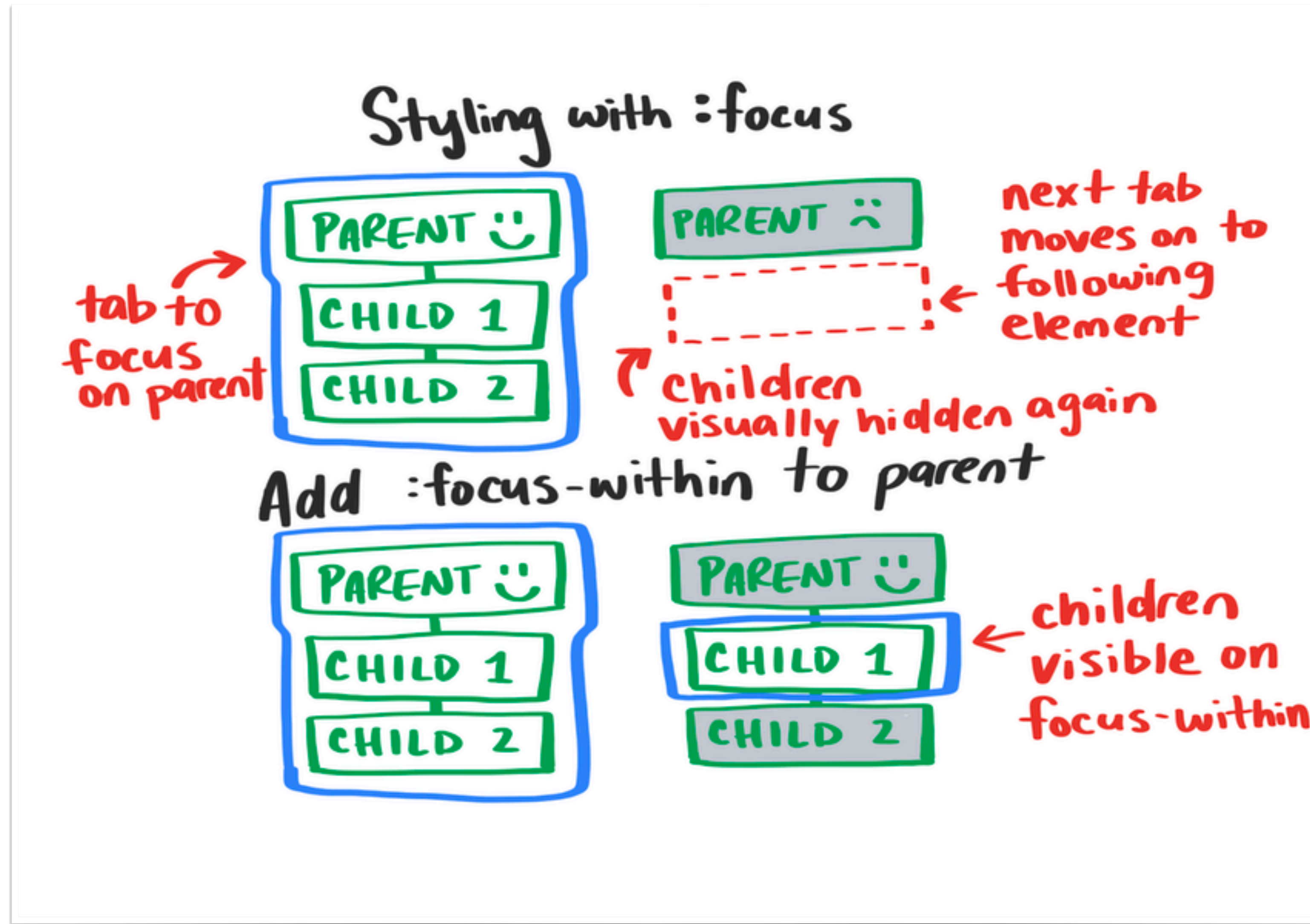
# CSS 伪类选择器 — :focus-visible vs. :focus-within

```
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

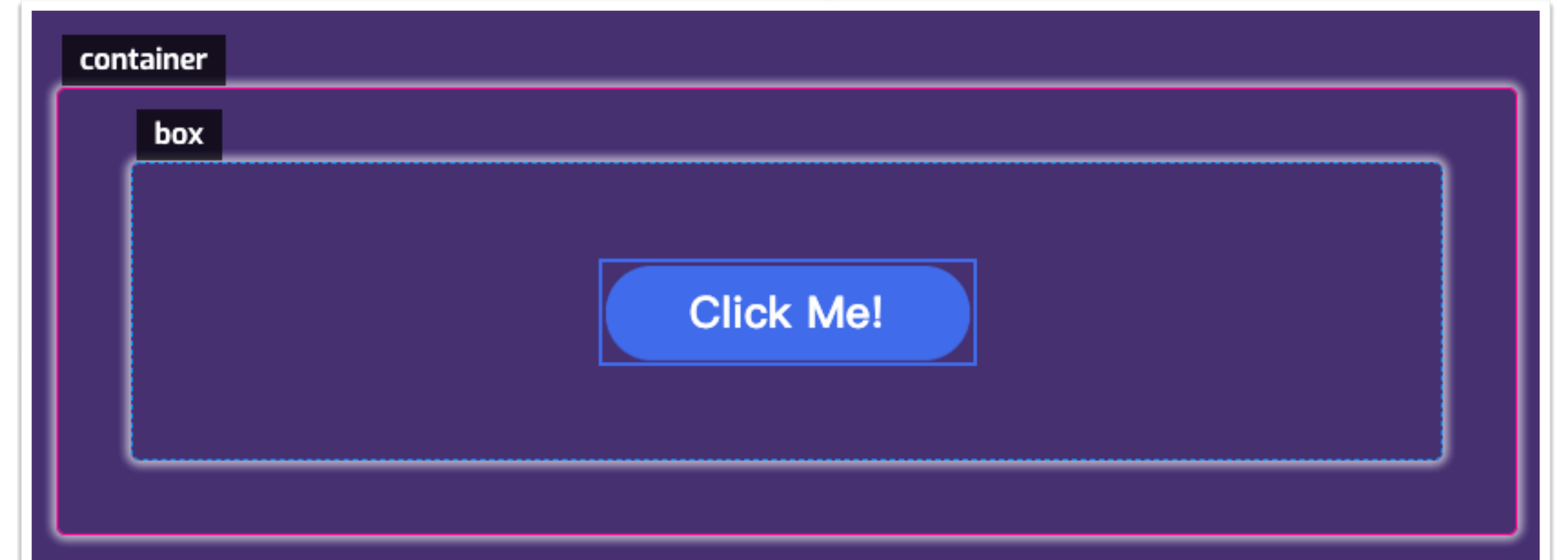
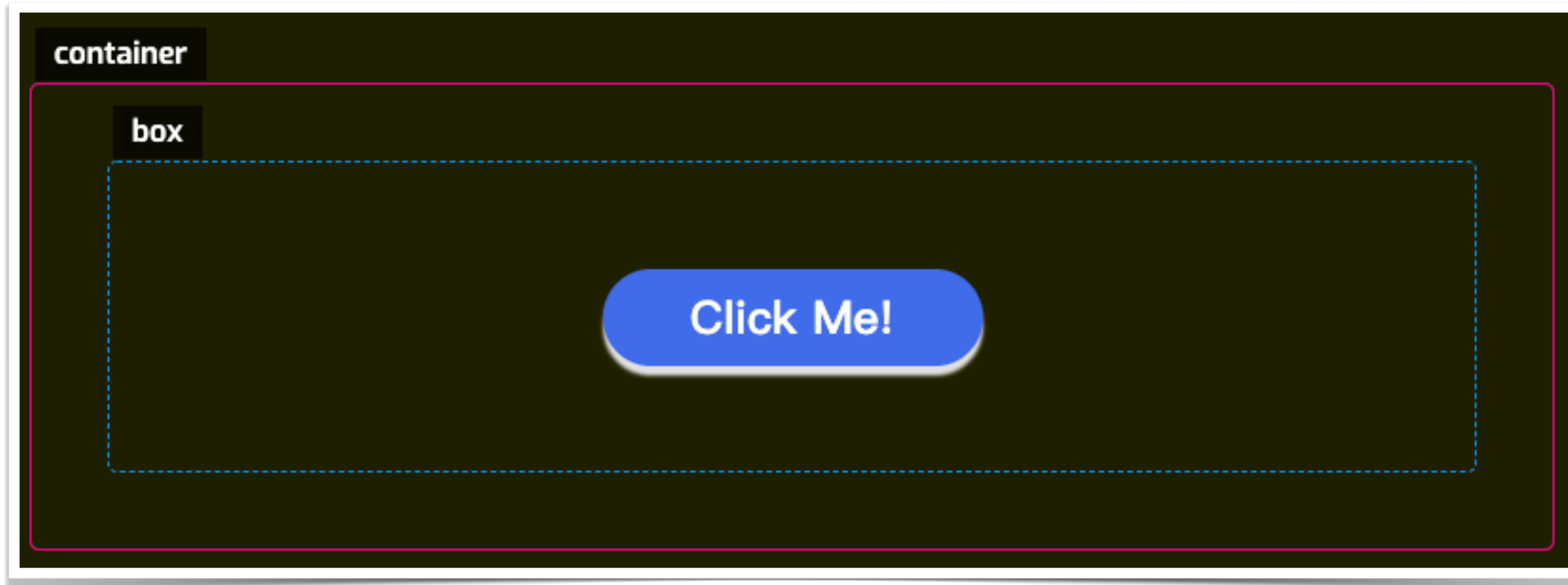


[Image URL](#) →

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

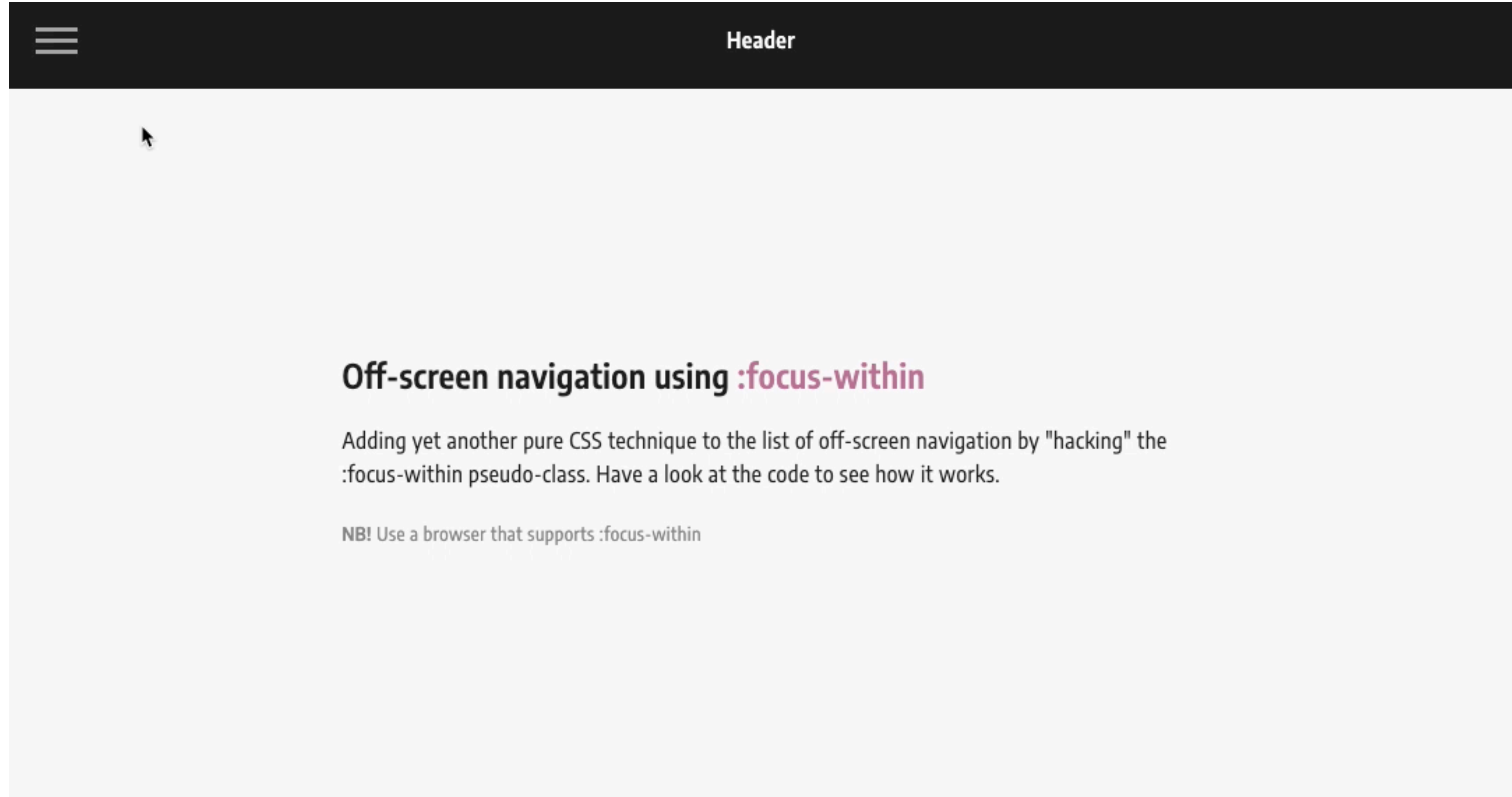


```
.box:focus-within,  
.container:focus-within {  
  box-shadow: 0 0 5px 3px rgba(255, 255, 255, 0.65);  
}
```



[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)	x	rgb(255 255 255)	rgb(255 255 255 / .5)
rgba()	x	rgba(255, 255, 255, .5)	rgba(255 255 255)	rgba(255 255 255 / .5)
hsl()	hsl(200, 30%, 30%)	x	hsl(200 30% 30%)	hsl(200 30% 30% / .5)
hsla()	x	hsla(200, 30%, 30%, .5)	hsla(200 30% 30%)	hsla(200 30% 30% / .5)
hwb()	x	x	hwb(300 40% 20%)	hwb(300 40% 20% / .8)
lab()	x	x	lab(29% 39 20)	lab(29% 39 20 / .5)
lch()	x	x	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

background: hsl(**from** **var(--theme-primary)** **h** **s** **30%**);

keep same h and s  
↓ ↓  
original color      change l



**Primary**

hsl(274, 61%, 50%)



**Modified**

hsl(274, 61%, 30%)

# CSS 颜色

```
CSS Colors

1 .bg-primary-100 {
2   background-color: hsl(from var(--theme-primary) h s 90%);
3 }
4 .bg-primary-200 {
5   background-color: hsl(from var(--theme-primary) h s 80%);
6 }
7 .bg-primary-300 {
8   background-color: hsl(from var(--theme-primary) h s 70%);
9 }
```



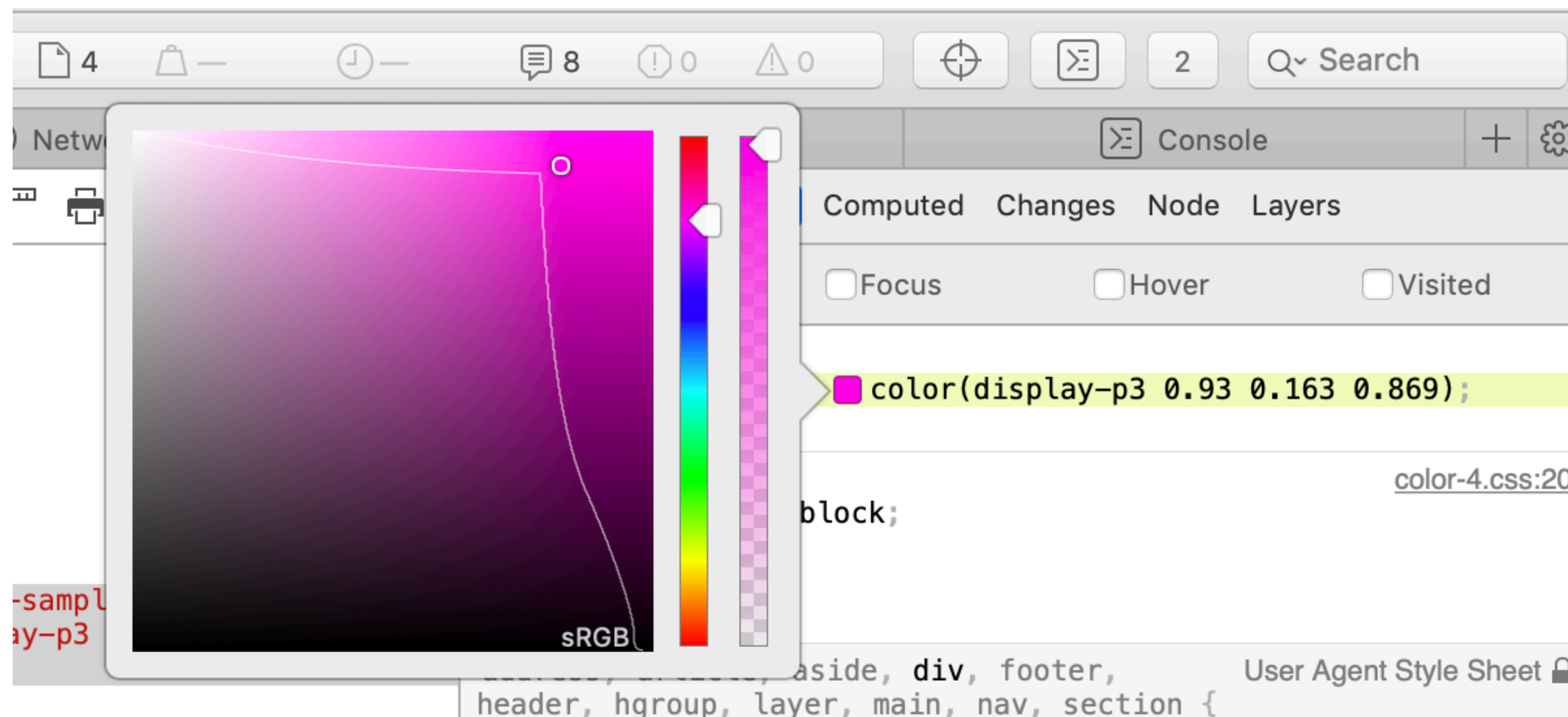
# CSS 颜色

```
✓ 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: 45rem;  
✓ margin-bottom: -30rem;
```

Format: Short Hex  
Format: HSL  
Format: RGB  
Format: Color Function  
**Convert to Display-P3**



# 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

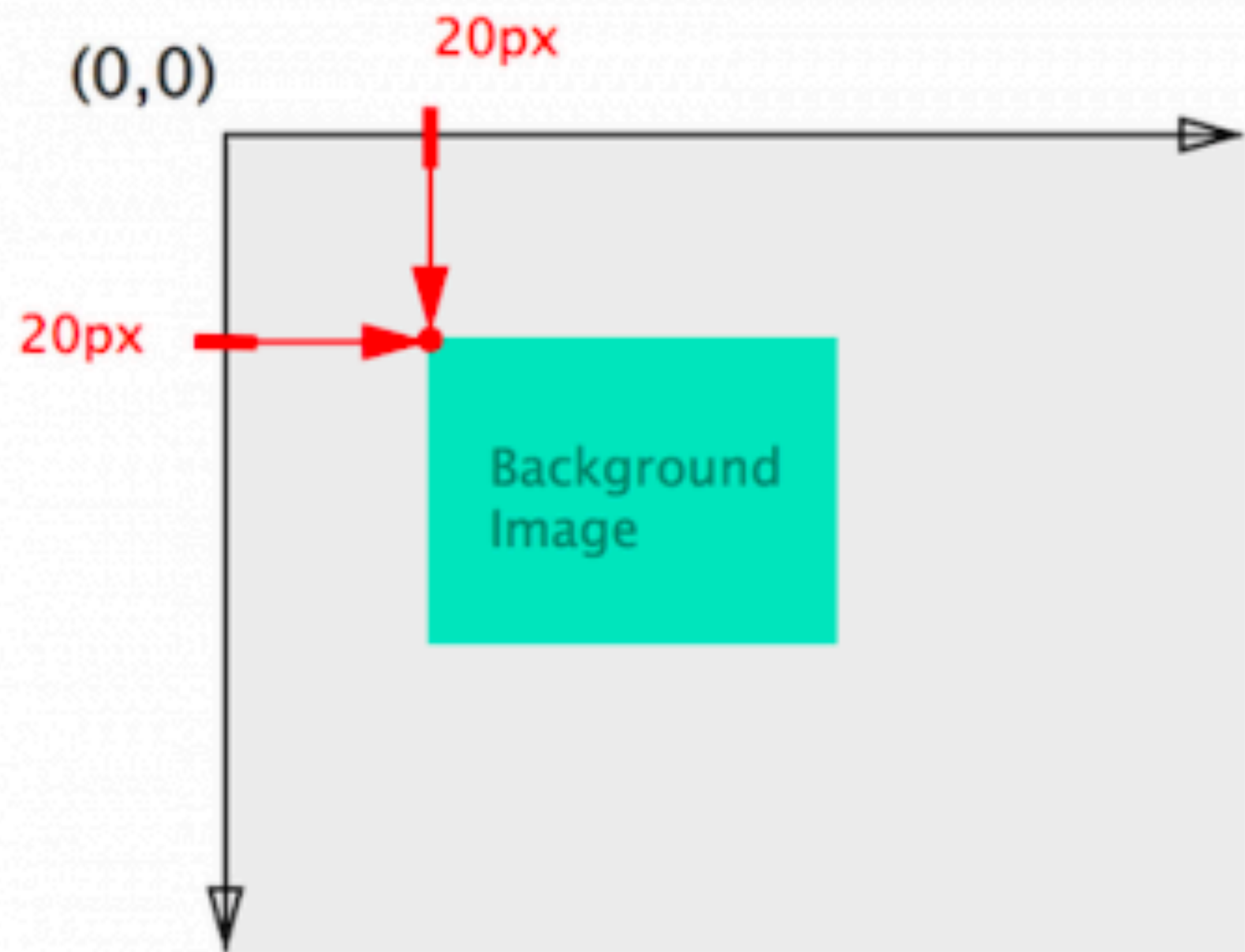
⑫ CSS比较函数

⑬ CSS内容可见性

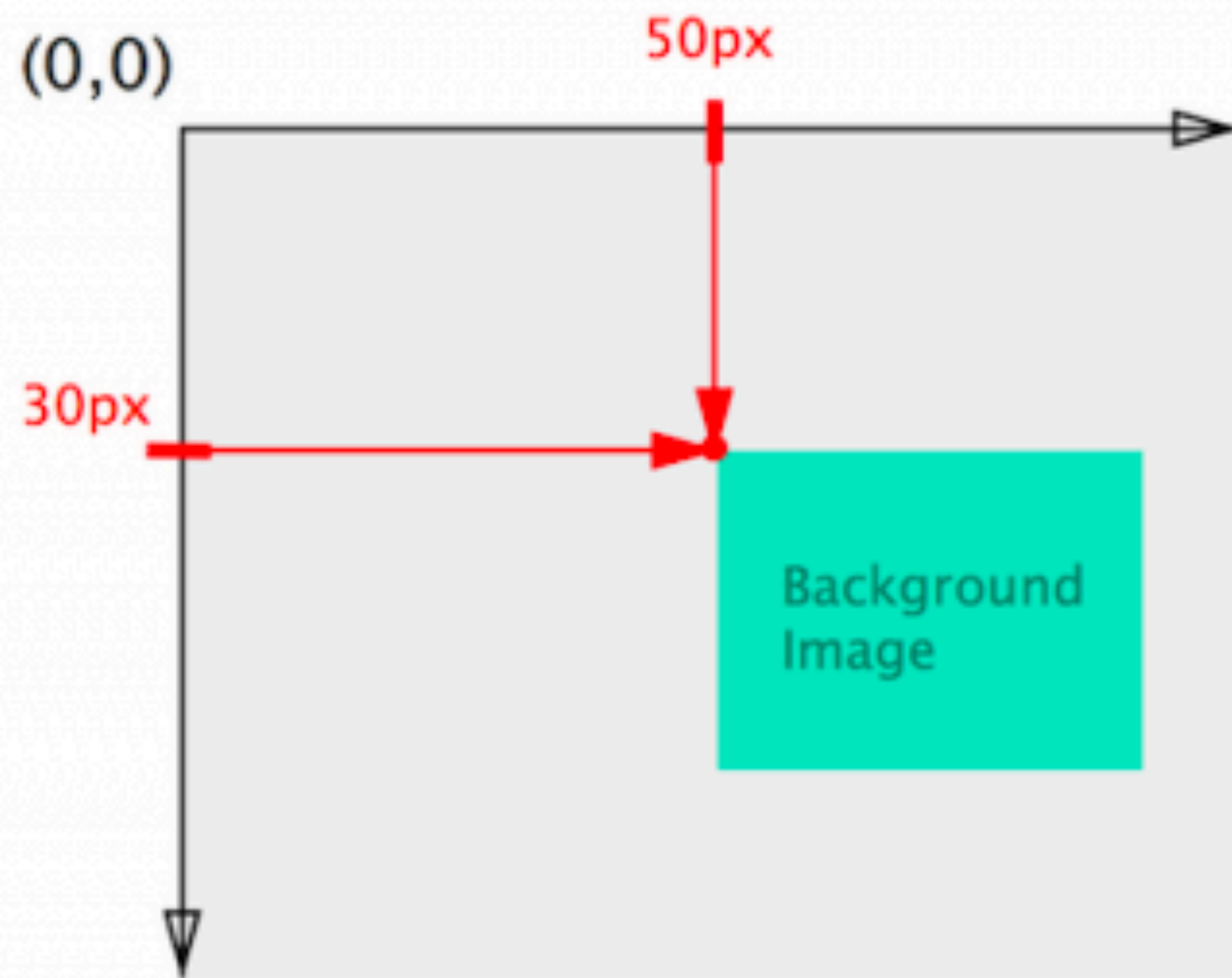
⑭ CSS外在尺寸和内在尺寸

⑮ CSS的Display

# CSS 背景 — background-position



background-position: 20px 20px;



background-position: 50px 30px;

# CSS 背景 — background-position



# CSS 背景 — background-position

```
Before
1 :root {
2   --xPosition: 50px;
3   --yPosition: 50px;
4 }
5 .container {
6   background-position:
7     calc(100% - var(--xPosition))
8     calc(100% - var(--yPosition));
9 }
```

```
After
1 :root {
2   --xPosition: 50px;
3   --yPosition: 50px;
4 }
5 .container {
6   background-position: |
7     right var(--xPosition)
8     bottom var(--yPosition);
9 }
```

[Codepen →](#)



# CSS 背景 — background-repeat

```
background-repeat

1 .container {
2     background-repeat:
3         no-repeat |
4         repeat-x   |
5         repeat-y   |
6         repeat     |
7         round      |
8         space      ;
9 }
```

# CSS 背景 — background-repeat



background-image



repeat



round



space

[Codepen →](#)

# CSS 背景 — background-repeat



background-repeat: repeat ▼ 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

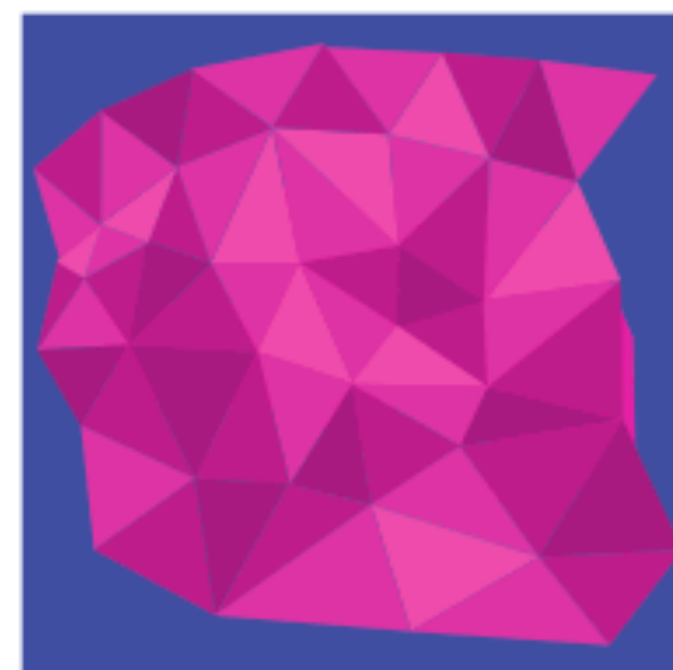
⑬ CSS内容可见性

⑭ CSS外在尺寸和内在尺寸

⑮ CSS的Display

# CSS masking & clipping

剪切  
(Clipping)



源

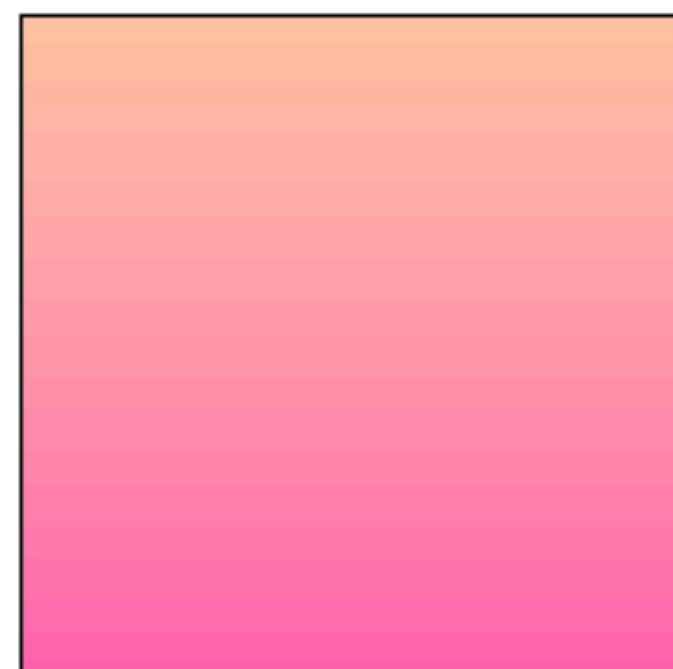


剪切路径



剪切后对象

遮罩  
(Masking)



源



遮罩层  
(高亮遮罩层)



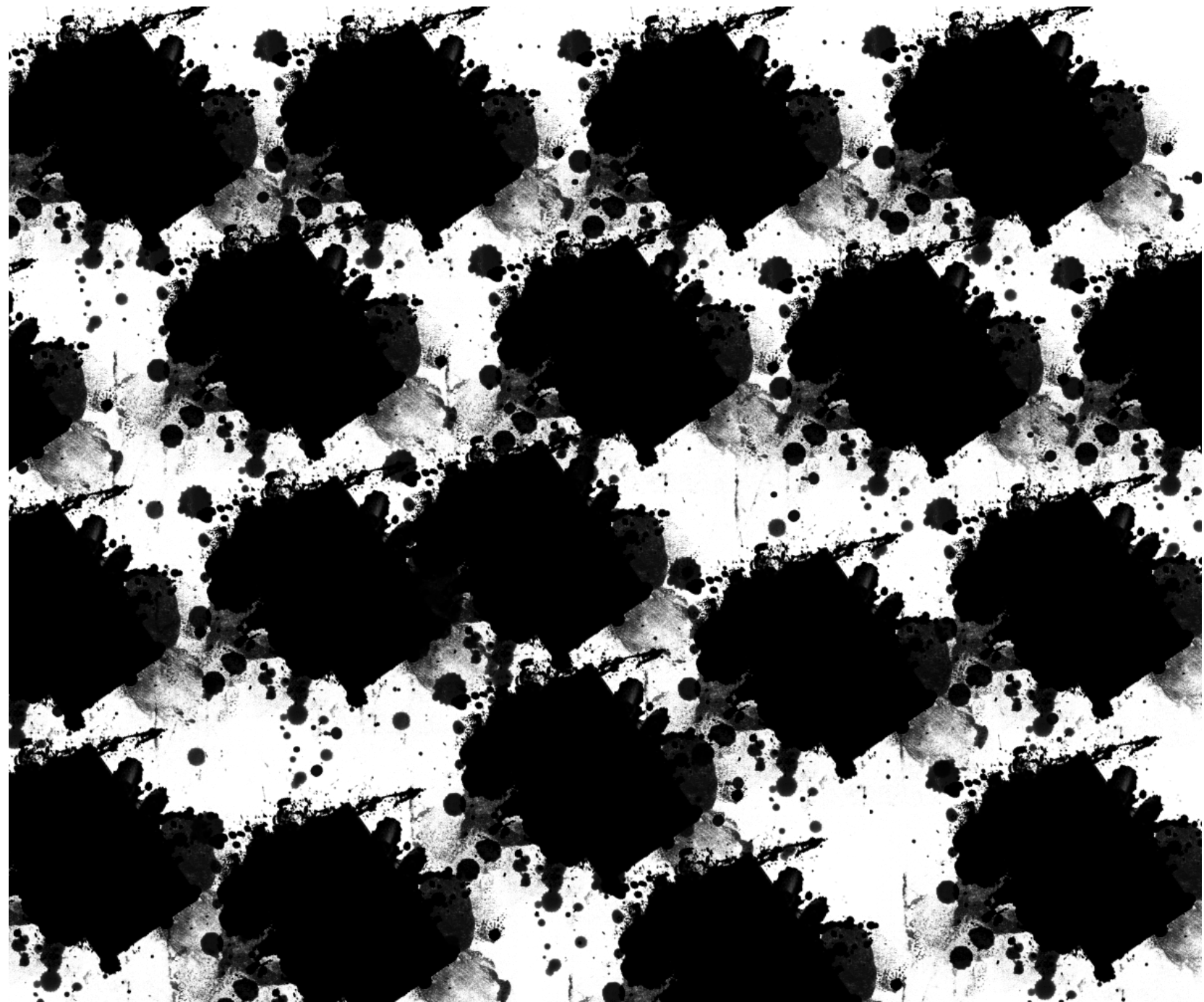
遮罩后对象

# 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

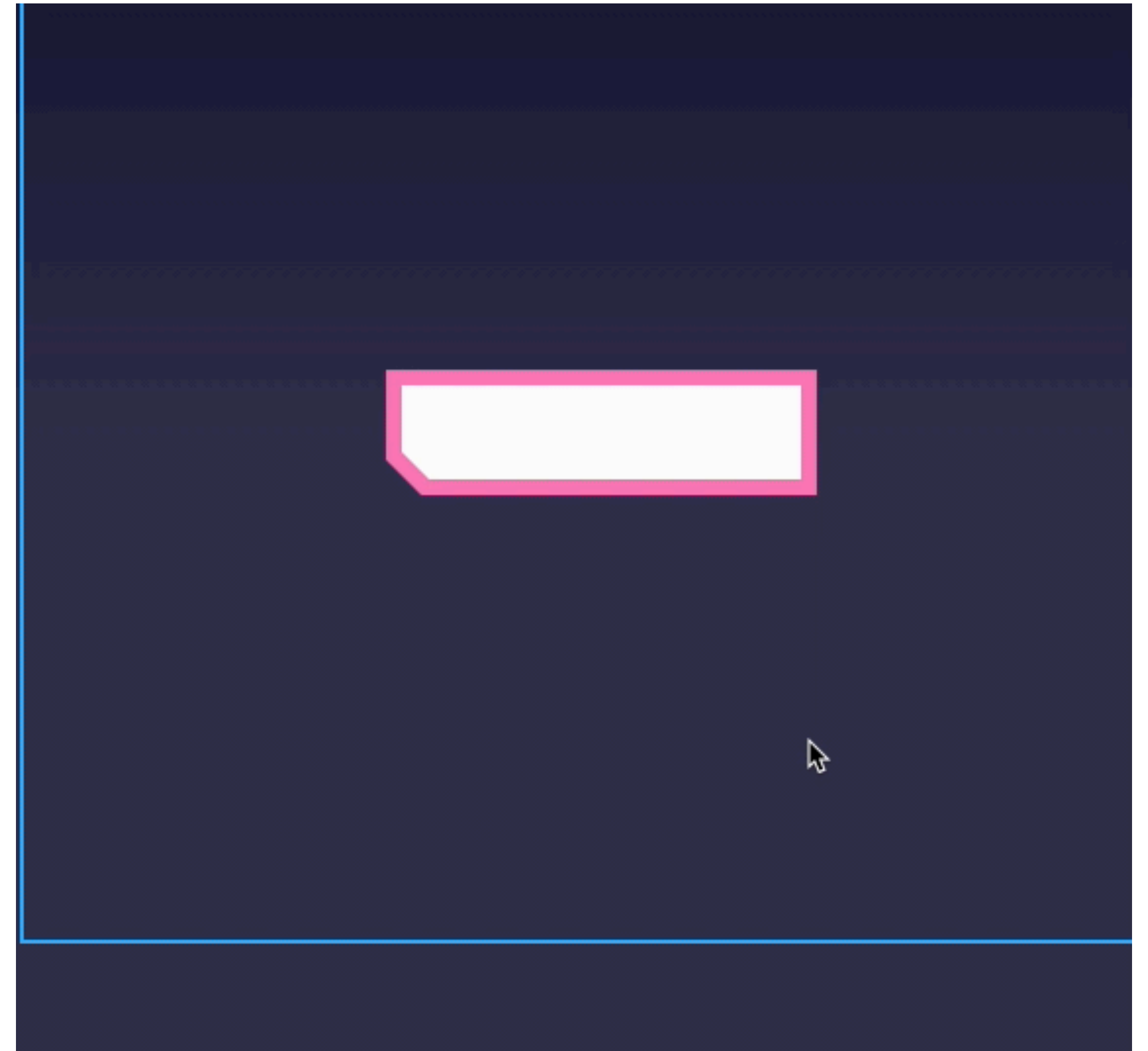
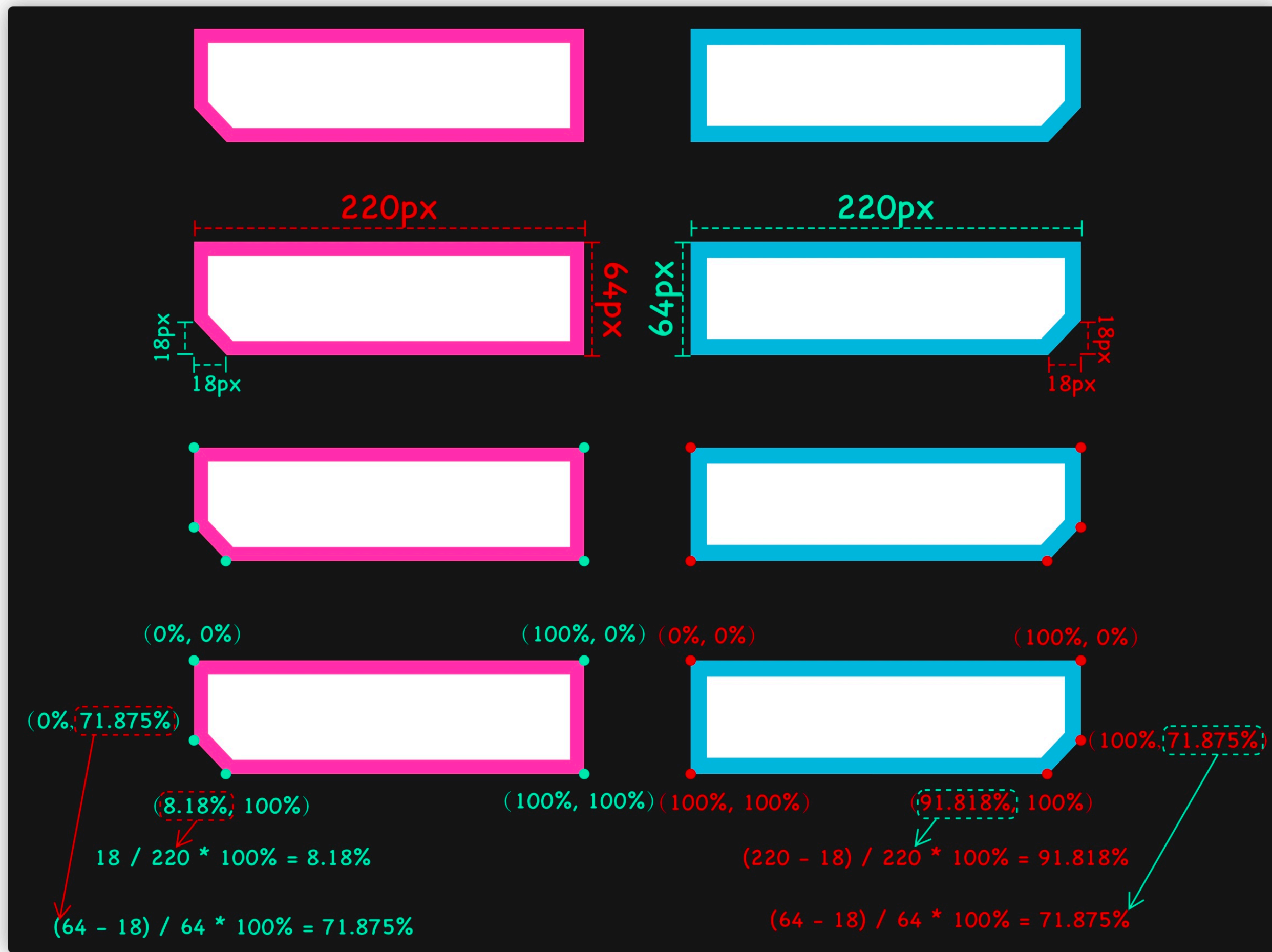
CSS clip-path maker

Tweet

```
clip-path: polygon(50% 0%, 61% 35%, 98% 35%, 68% 57%, 79% 91%, 50% 70%, 21% 91%, 32% 57%, 2% 35%, 39% 35%);
```

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

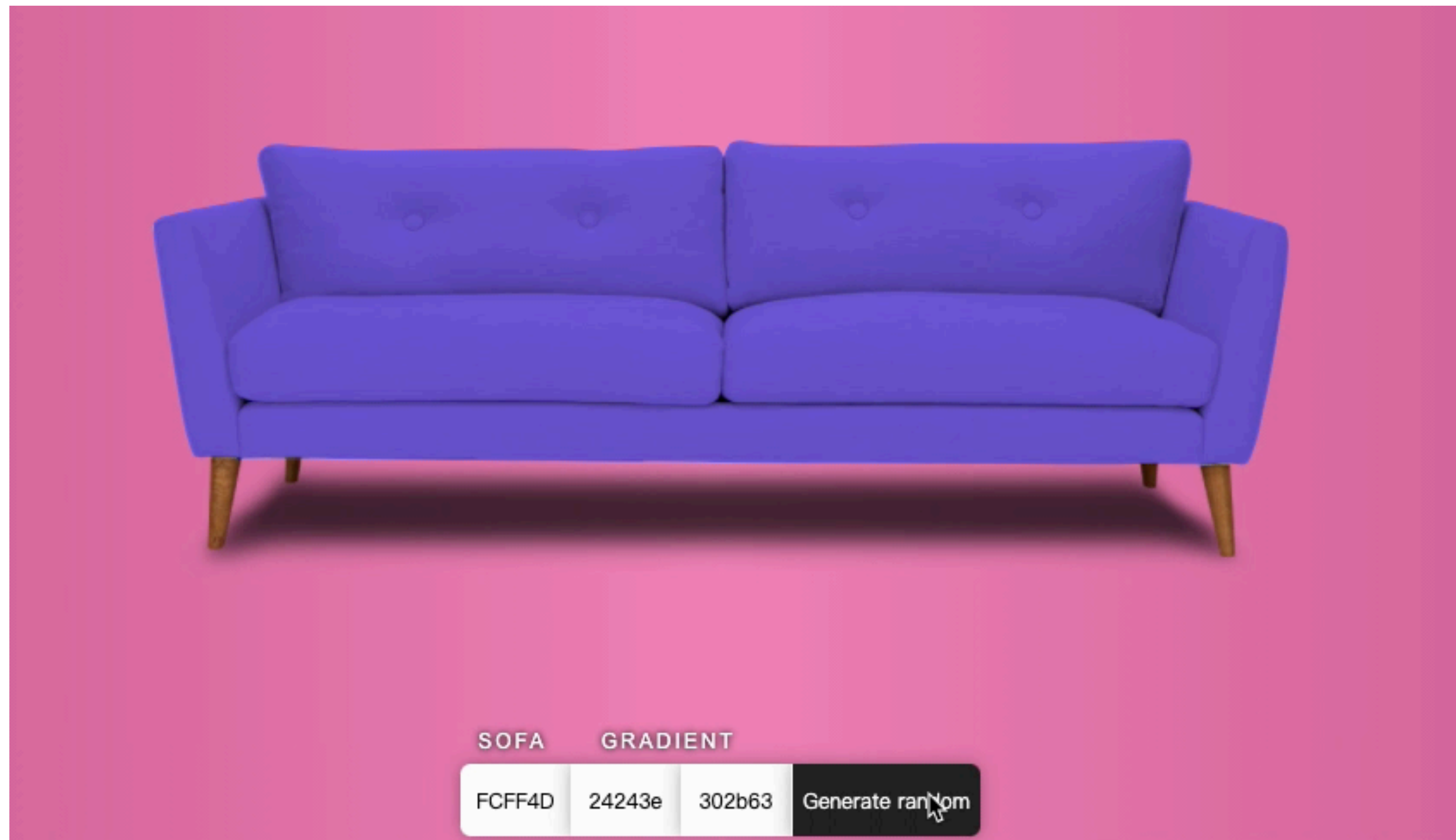
⑫ CSS比较函数

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

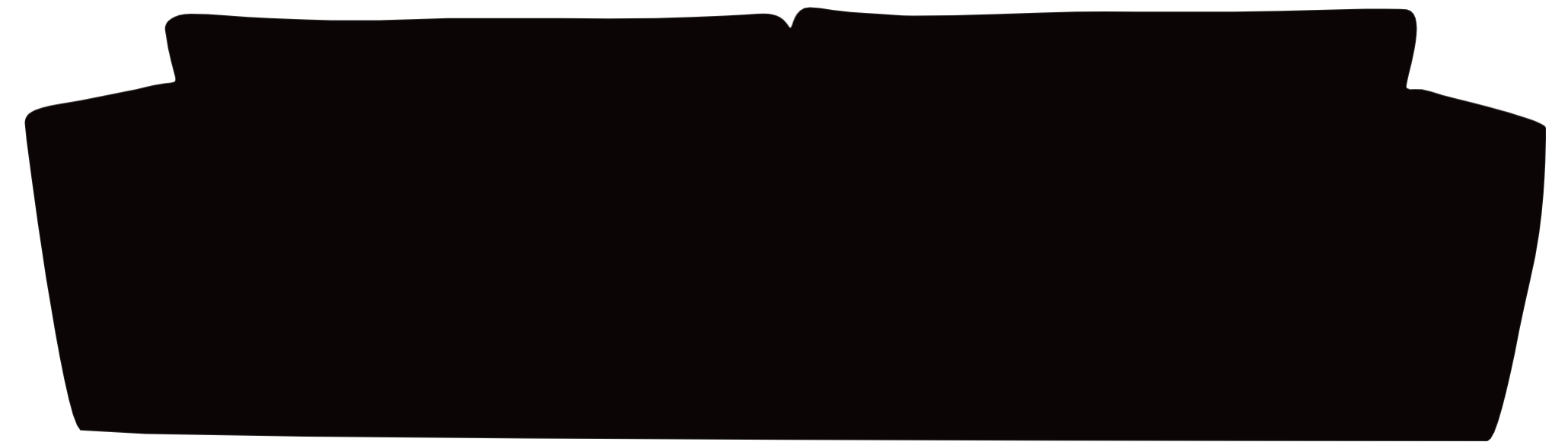


[CodePen →](#)

# CSS 混合模式



原图



SVG Path

```
mix-blend-mode  
1 :root {  
2   --fill: #fcff4d  
3 }  
4  
5 svg {  
6   fill: var(--fill);  
7   mix-blend-mode: multiply;  
8 }
```

[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

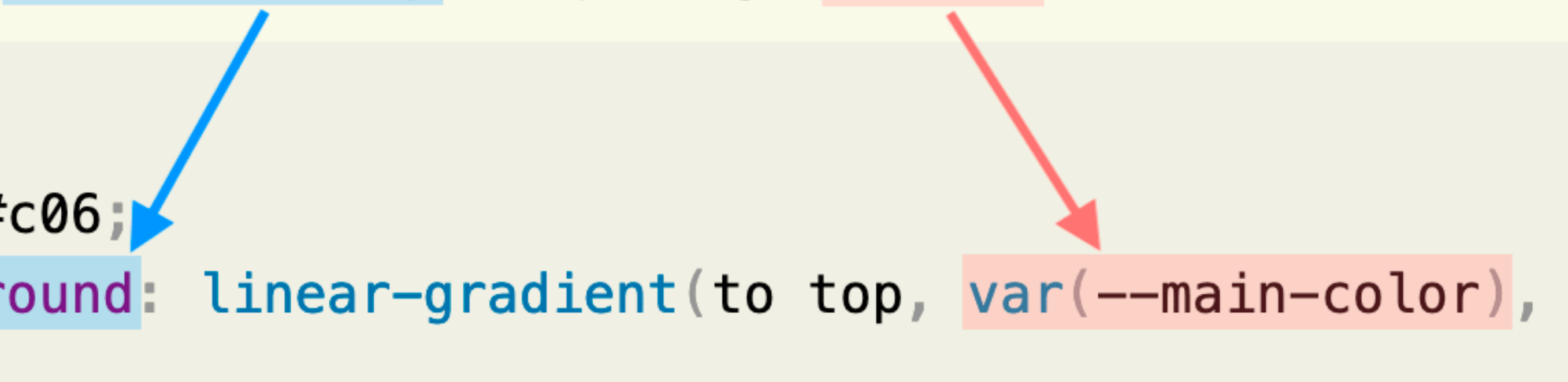
⑮ CSS的Display

# 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.

## § 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。



# CSS 自定义属性

```
无效变量

1 :root {
2     --foo: initial;
3 }
4
5 bar {
6     color: var(--foo); // ✘
7 }
8
9 foo {
10    color: var(--foo, orange); // ✔
11 }
```

# CSS 自定义属性

无效变量

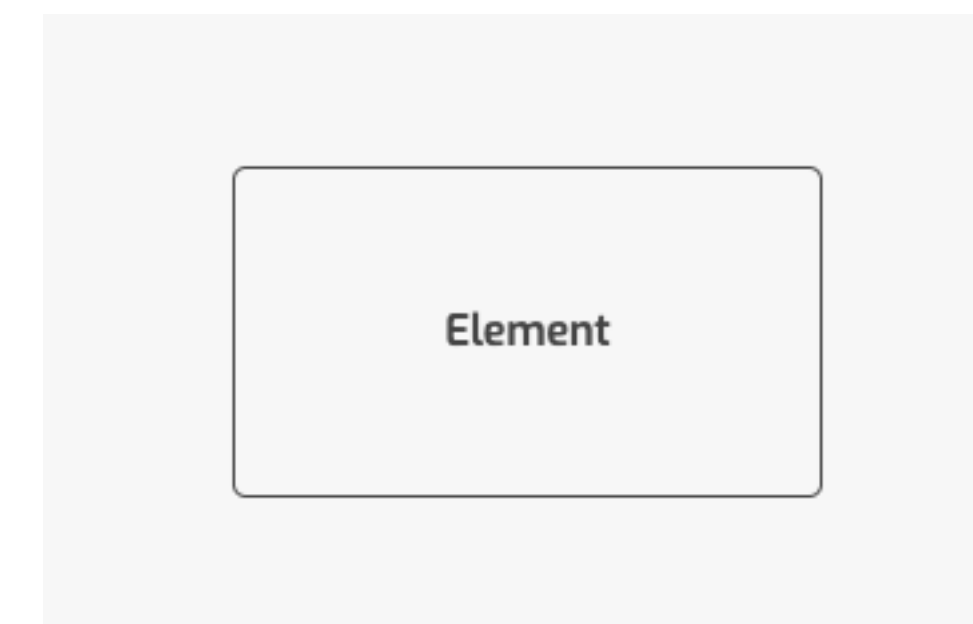
```
1 <!-- HTML -->
2 <div class="element">
3   <i>Element</i>
4 </div>
5
6 <i>Element</i>
7
8 /* CSS */
9 body { color: #fff; }
10 .element { --color: red; }
11 i {
12   --foo: initial;
13   --color: var(--foo);
14   background-color: var(--color, orange);
15 }
```



[CodePen →](#)

# CSS 自定义属性

```
有效变量
1 <!-- HTML -->
2 <section>Element</section>
3
4 /* CSS */
5 section {
6   --color: ;
7   background-color: var(--color, orange);
8 }
```



[CodePen](#) →

# CSS 自定义属性

```
有效变量 vs. 无效变量

1 :root {
2   --initial: initial; // ❌
3   --valid: ; // ✅
4 }
5
6 foo {
7   background-color: var(--initial, red); // red
8   color: var(--valid, red); // 空格
9 }
```



[CodePen →](#)

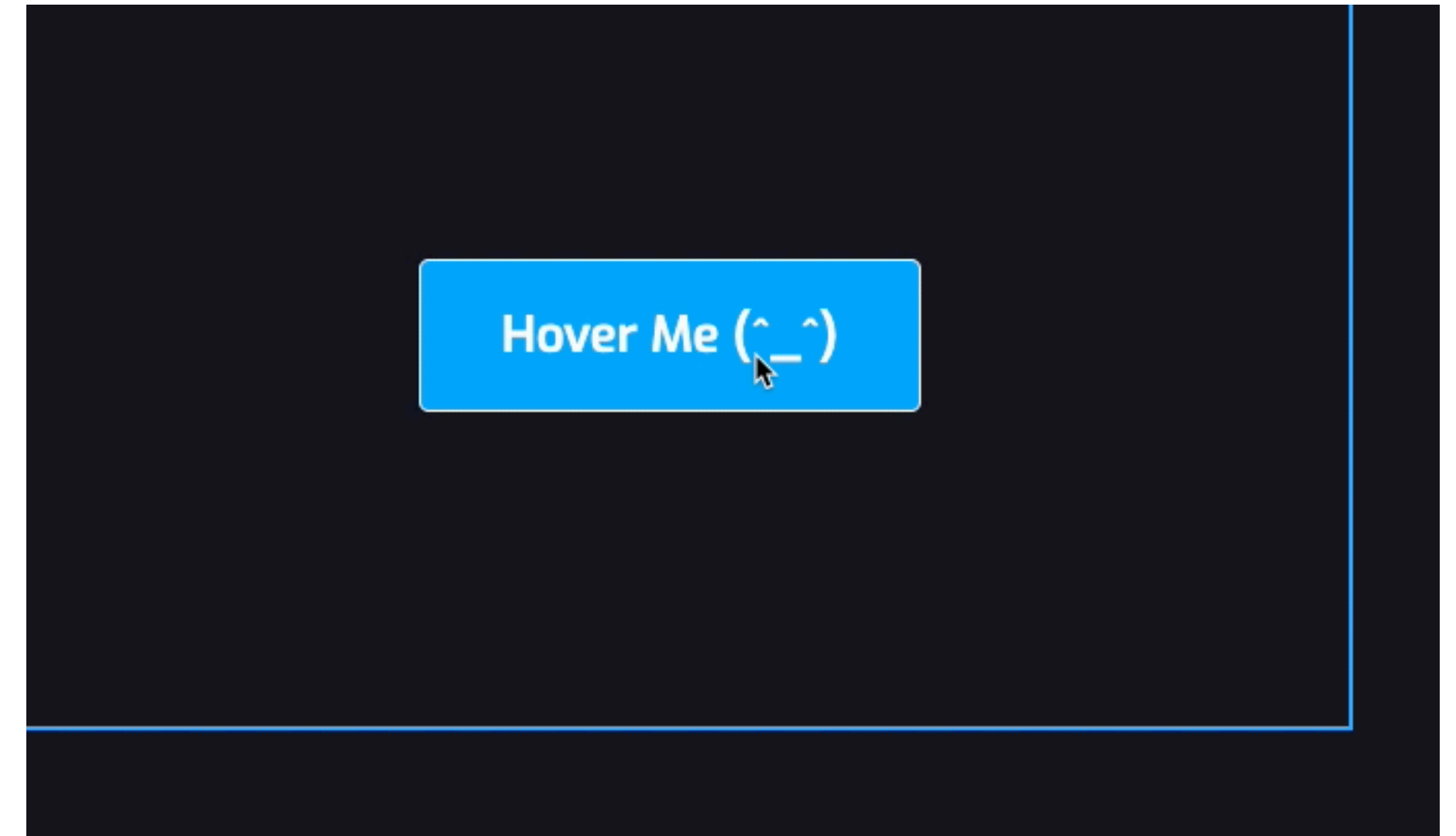
# CSS 自定义属性

```
有效变量 vs. 无效变量

1 :root {
2   --initial:; // ❌ 冒号和分号之间无空格符
3   --valid: |; // ✅ 冒号和分号之间有空格符
4 }
5
6 foo {
7   background-color: var(--initial, red); // red
8   color: var(--valid, red); // 空格
9 }
```

# CSS 自定义属性

```
33 ▾ :root {
34 ▾   --ON: initial; /* ❌ ← 等同 --ON:; */
35 ▾   --OFF: ; /* ✅ ← 注意冒号和分号之间空格 */
36 }
37
38 ▾ button {
39   --is-raised: var(--OFF);
40   --border-color: var(--is-raised, rgb(0 0 0 / 0.1));
41   --gradient: var(
42     --is-raised,
43     linear-gradient(hsl(0 0% 100% / 0.3), transparent)
44   );
45   --box-shadow: var(
46     --is-raised,
47     0 1px hsl(0 0% 100% / 0.8) inset,
48     0 0.1em 0.1em -0.1em rgb(0 0 0 / 0.2)
49   );
50   --text-shadow: var(--is-raised, 0 -1px 1px rgb(0 0 0 / 0.3));
51
52   border: 1px solid var(--border-color);
53   background: hsl(200 100% 50%) var(--gradient);
54   box-shadow: var(--box-shadow);
55   text-shadow: var(--text-shadow);
56 }
57 ▾ button:hover {
58   --is-raised: var(--ON);
59 }
```

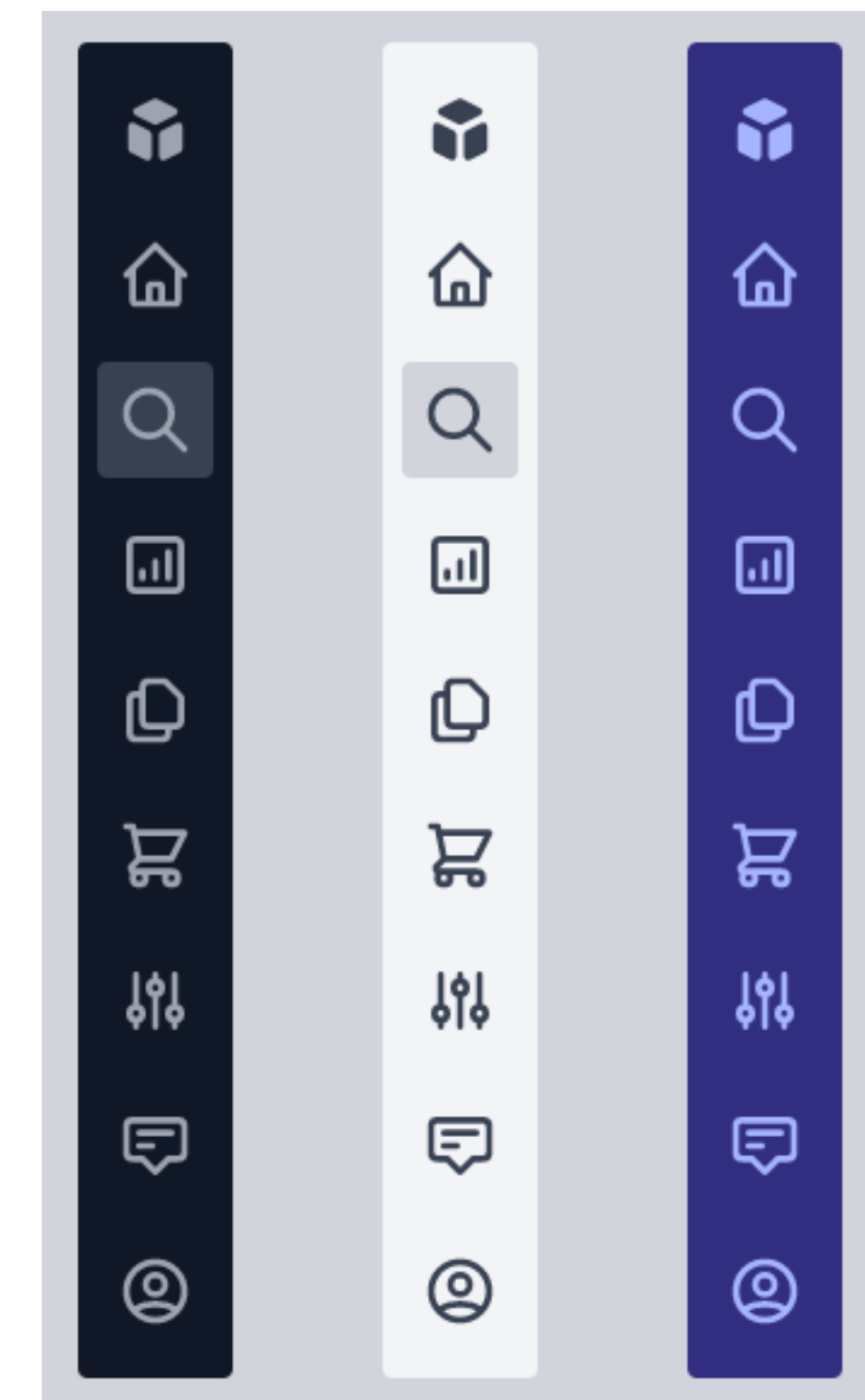


[CodePen →](#)

# CSS 自定义属性

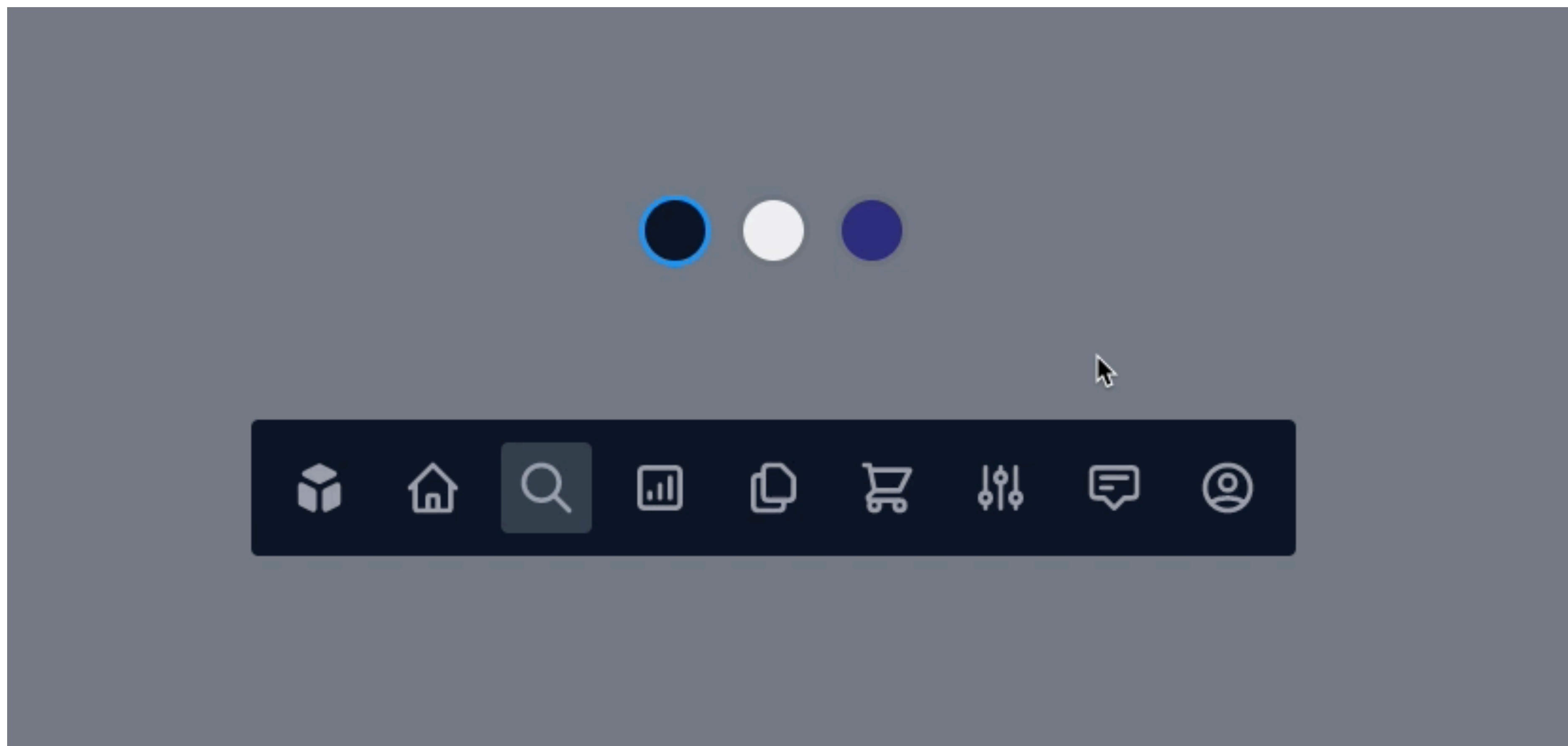
```
1 /* 设置切换开关 */
2 :root {
3   --ON: initial; // ❌ 等同 --ON:;
4   --OFF: ;      // ✅ 注意冒号和分号之间空格
5 }
6 /* 默认为Dark */
7 .dark {
8   --light: var(--OFF);
9   --dark: var(--ON);
10  --blue: var(--OFF);
11 }
12 /* 默认为Light */
13 .light {
14   --light: var(--ON);
15   --dark: var(--OFF);
16   --blue: var(--OFF);
17 }
18 /* 默认为Blue */
19 .blue {
20   --light: var(--OFF);
21   --dark: var(--OFF);
22   --blue: var(--ON);
23 }
```

```
1 .nav {
2   /* Dark */
3   --dark-color: rgb(156 163 175);
4   --dark-bgcolor: rgb(17 24 39);
5   --dark-active-bgcolor: rgb(55 65 81);
6
7   /* Light */
8   --light-color: rgb(55 65 81);
9   --light-bgcolor: rgb(243 244 246);
10  --light-active-bgcolor: rgb(209 213 219);
11
12  /* Blue */
13  --blue-color: rgb(165 180 252);
14  --blue-bgcolor: rgb(49 46 129);
15  --blue-active-bgcolor: rgb(67 56 202);
16 }
17 .nav{
18   color: var(
19     --light, var(
20       --light-color)) var(
21       --dark, var(
22         --dark-color)) var(
23         --blue, var(--blue-color)
24     );
25   background-color: var(
26     --light, var(
27       --light-bgcolor)) var(
28       --dark, var(
29         --dark-bgcolor)) var(
30         --blue, var(--blue-bgcolor)
31     );
32 }
```



[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

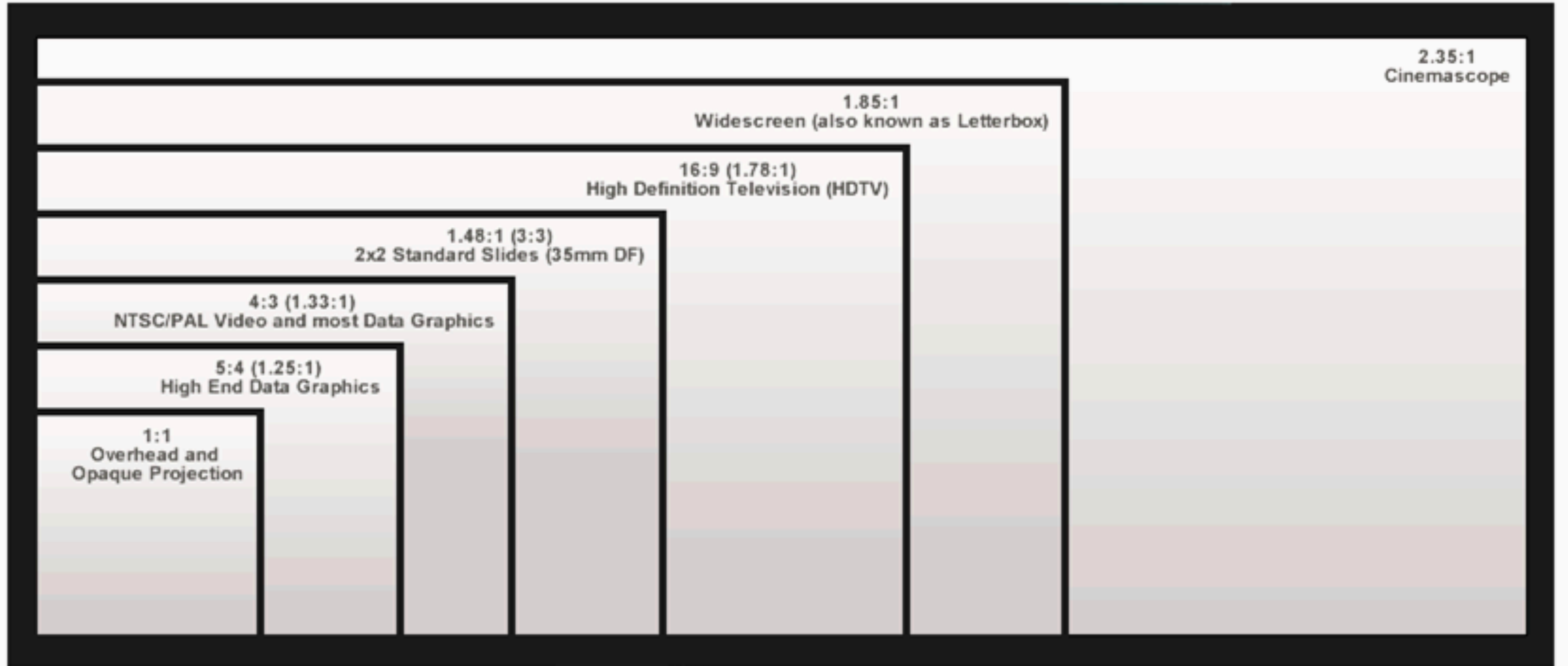
⑫ CSS比较函数

⑬ CSS内容可见性

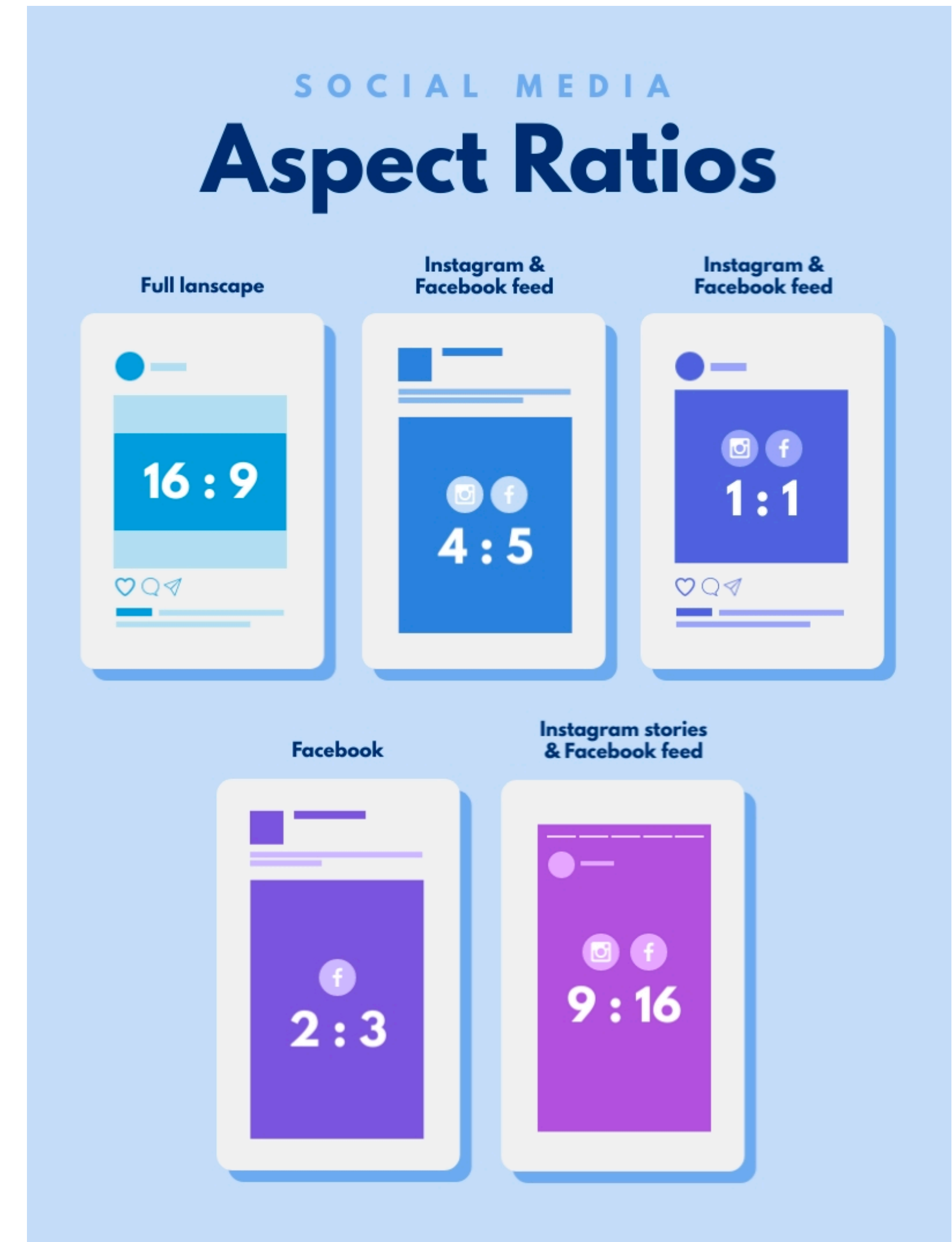
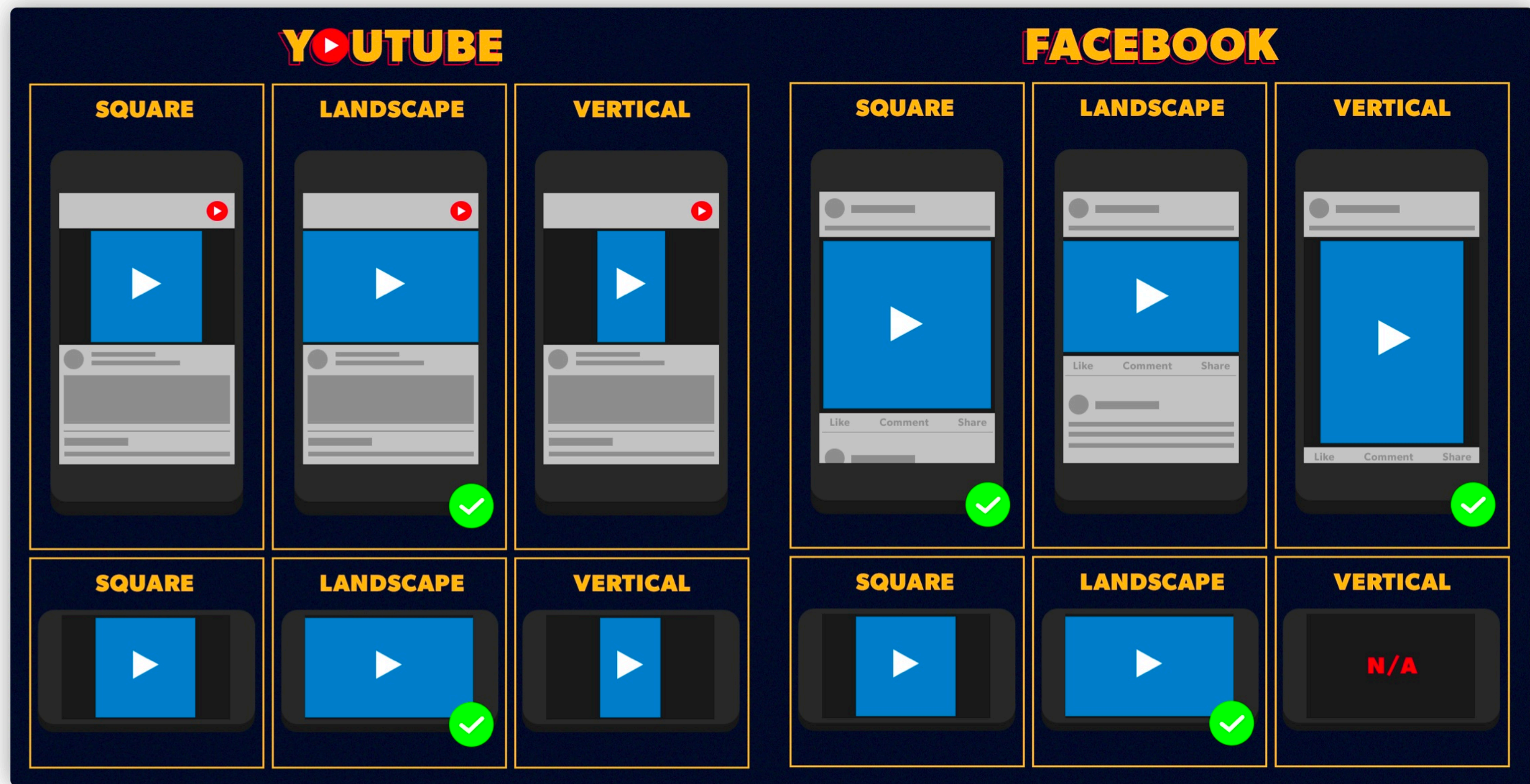
⑭ CSS外在尺寸和内在尺寸

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

1 /* padding-bottom/top */
2 .aspect-ratio-container {
3   position: relative;
4   height: 0;
5   padding-top: 56.25%;
6   width: 100%;
7 }
8 .aspect-ratio-content {
9   position: absolute;
10  top: 0;
11  left: 0;
12  width: 100%;
13 }
```

```
等比缩放: padding-top 或 padding-bottom

1 /* CSS Custom Property */
2 .aspect-ratio-container {
3   width: 100%;
4 }
5 .aspect-ratio-content {
6   --ratio: 16/9;
7   height: calc(var(--width) * 1 / (var(--
8   ratio)));
9   width: 100%;
10 }
```

```
等比缩放: aspect-ratio

1 /* CSS aspect-ratio */
2 .aspect-ratio-content {
3   width: 100%;
4   aspect-ratio: 16 / 9;
5 }
```

[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

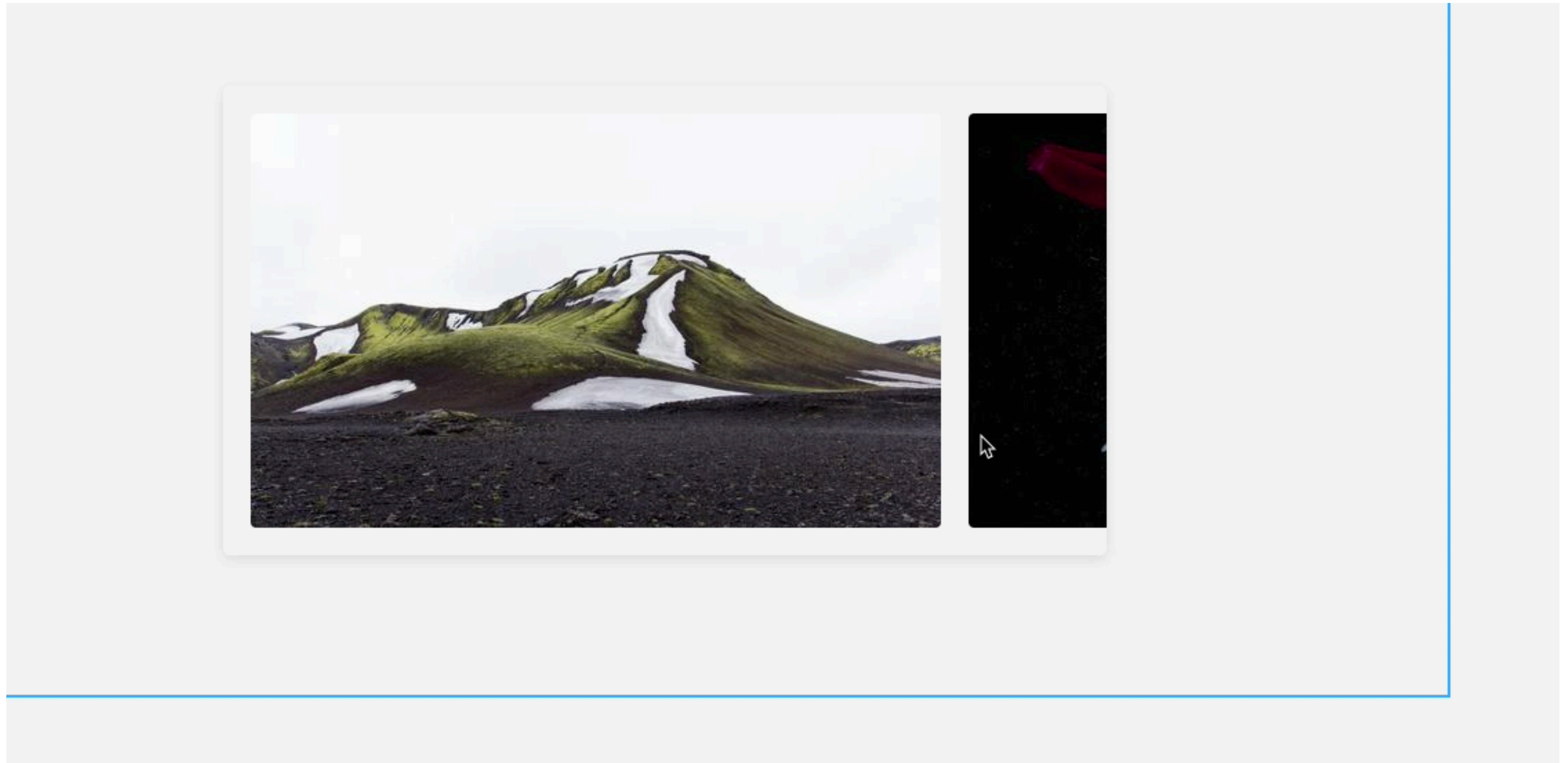
⑭ CSS外在尺寸和内在尺寸

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
1 .container {
2   scroll-behavior: smooth;
3   overflow-x: auto;
4   scroll-snap-type: x mandatory;
5   scroll-padding: 20px;
6 }
7
8 img {
9   scroll-snap-align: center;
10  scroll-snap-stop: always;
11 }
```

[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

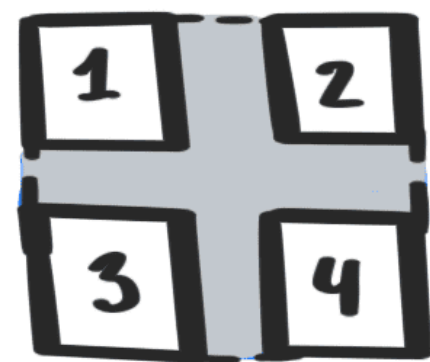
⑬ CSS内容可见性

⑭ CSS外在尺寸和内在尺寸

⑮ CSS的Display

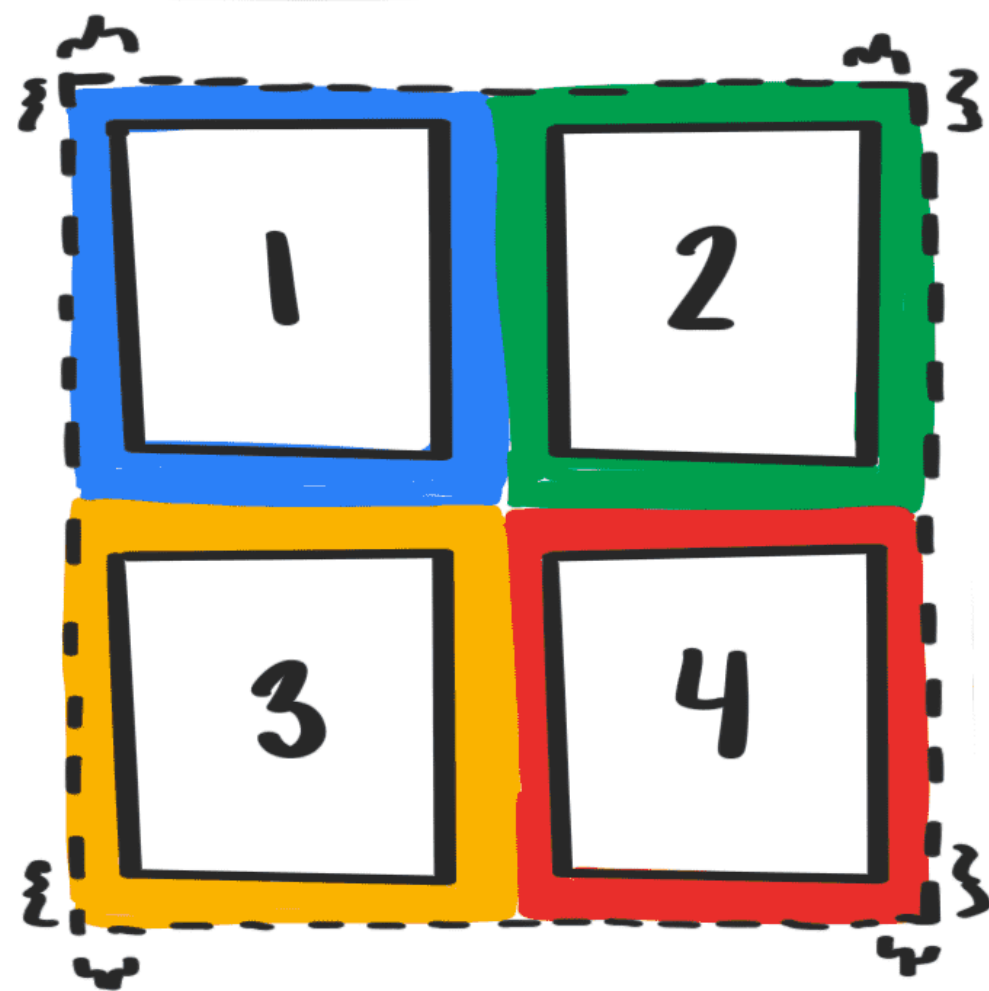
# CSS Gap

design:



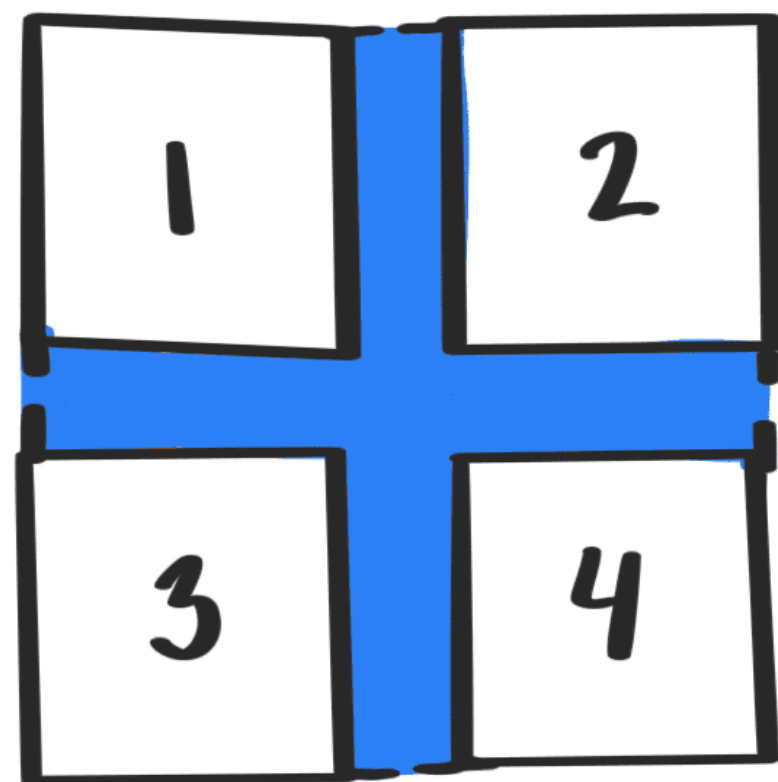
← intended spacing

margin



☺ extra spacing

gap



↑ space between ☺

```
CSS gap

1 .container {
2   display: flex;
3   gap: 10px;
4 }
5
6 .container {
7   display: grid;
8   gap: 20px
9 }
```

# 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

facebook 邮箱或手机号 密码 登录 忘记密码?

联系你我，分享生活，尽在 Facebook

## 注册

快速又简便。

姓  名

手机号或邮箱

创建密码

出生日期 1994 12月 25

性别  女  男  自定义

点击注册，即表示你同意接受我们的条款、数据使用政策和 Cookie 政策。你可能会收到我们的短信通知，并可以随时退订。

[注册](#)

为名人、团体或企业创建主页。

中文(简体) English (US) Español Français (France) العربية Português (Brasil) Italiano 한국어 Deutsch हिन्दी 日本語 +

注册 登录 Messenger Facebook Lite Watch 用户 公共主页 主页类别 地点 游戏 位置 Marketplace 小组 Instagram 本地 筹款活动 服务 公司简介 创建广告 创建主页 开发者 招聘信息 隐私权政策 Cookie Ad Choices 条款 帮助中心

Facebook © 2019

中文 (LTR)

facebook البريد الإلكتروني أو الهاتف كلمة السر تسجيل الدخول هل نسيت الحساب?

## افتح حسابًا في فيس بوك

يتميز بالسرعة والسهولة.

اسم العائلة الاسم الأول

رقم الهاتف المحمول أو البريد الإلكتروني

كلمة السر الجديدة

تاريخ الميلاد 1994 ديسمبر 25

الجنس  أنثى  ذكر  مخصص

بالنقر على "إنشاء حساب في فيس بوك"، فإنك توافق على الشروط وسياسة البيانات وسياسة ملفات تعريف الارتباط. ويمكن أن تتلقى إشعارات عبر رسائل SMS من فيس بوك، ويمكنك إلغاء تلقي هذه الإشعارات في أي وقت.

[إنشاء حساب في فيس بوك](#)

إنشاء صفحة لفرق موسيقية أو مشاهير أو أنشطة تجارية.

+ 日本語 हिन्दी Deutsch 한국어 Italiano Português (Brasil) Français (France) Español English (US) 中文(简体) العربية

إنشاء حساب في فيس بوك تسجيل الدخول Marketplace المجموعات Instagram اختيارات الإعلانات مساعدة Watch Facebook Lite Messenger محلية حملات جمع التبرعات الخدمات حول فيس بوك إنشاء إعلان إنشاء صفحة المطورون الوظائف الخصوصية ملفات تعريف الارتباط

Facebook © 2019

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-rl

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. баганадсан барилга бүхий баршгүй олон тооны тэнгэр барилга байгууламж, тоолж дурсгалт газрууд, 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

writing-mode: horizontal-tb

writing-mode: horizontal-tb

文档流

文档流



# CSS 逻辑属性

Left to Right (LTR)

 UX Design

Make it easier to search and filter

Right to Left (RTL)

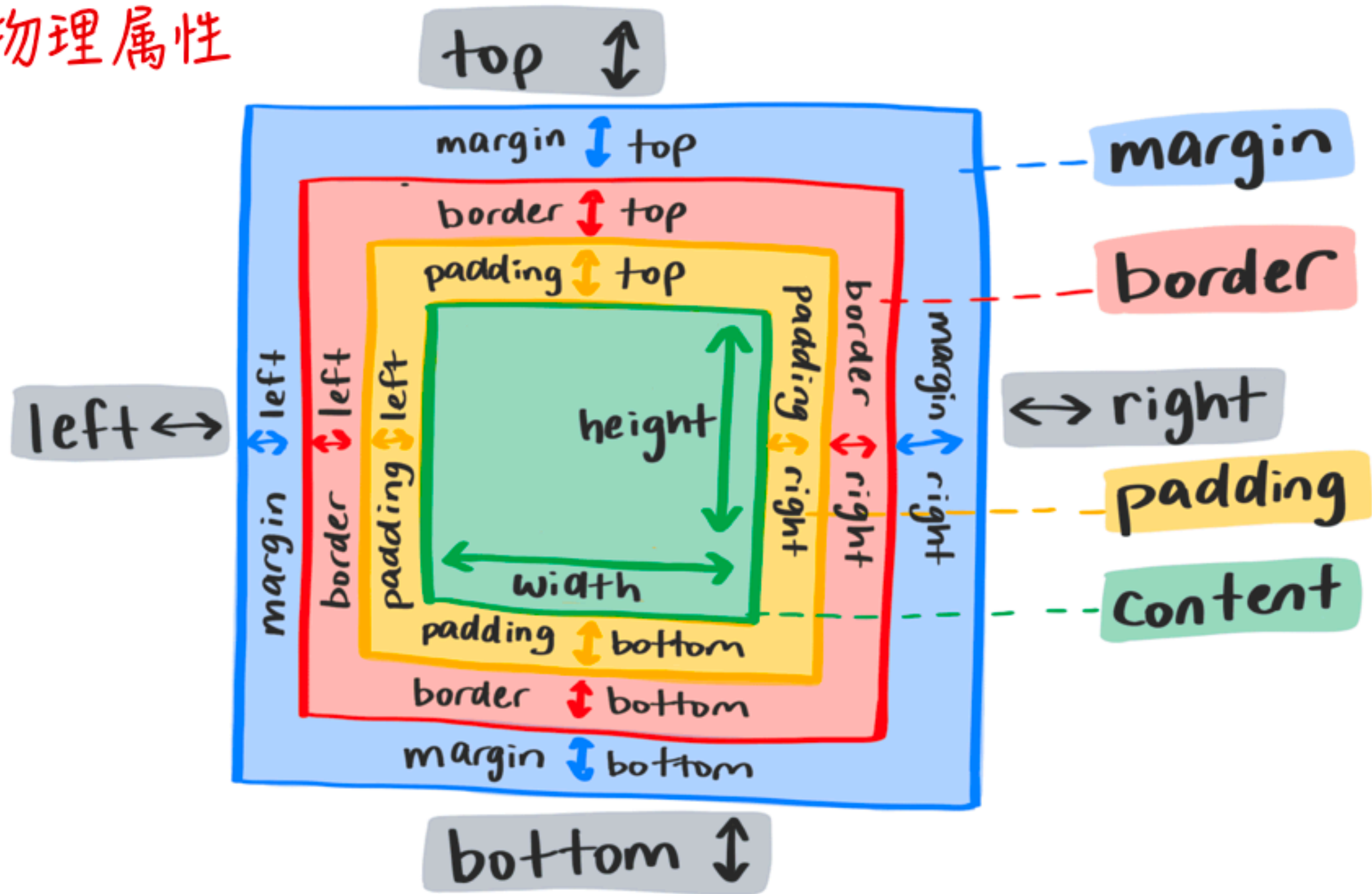
تجربة المستخدم

تسهيل الوصول للبحث والتصنيف

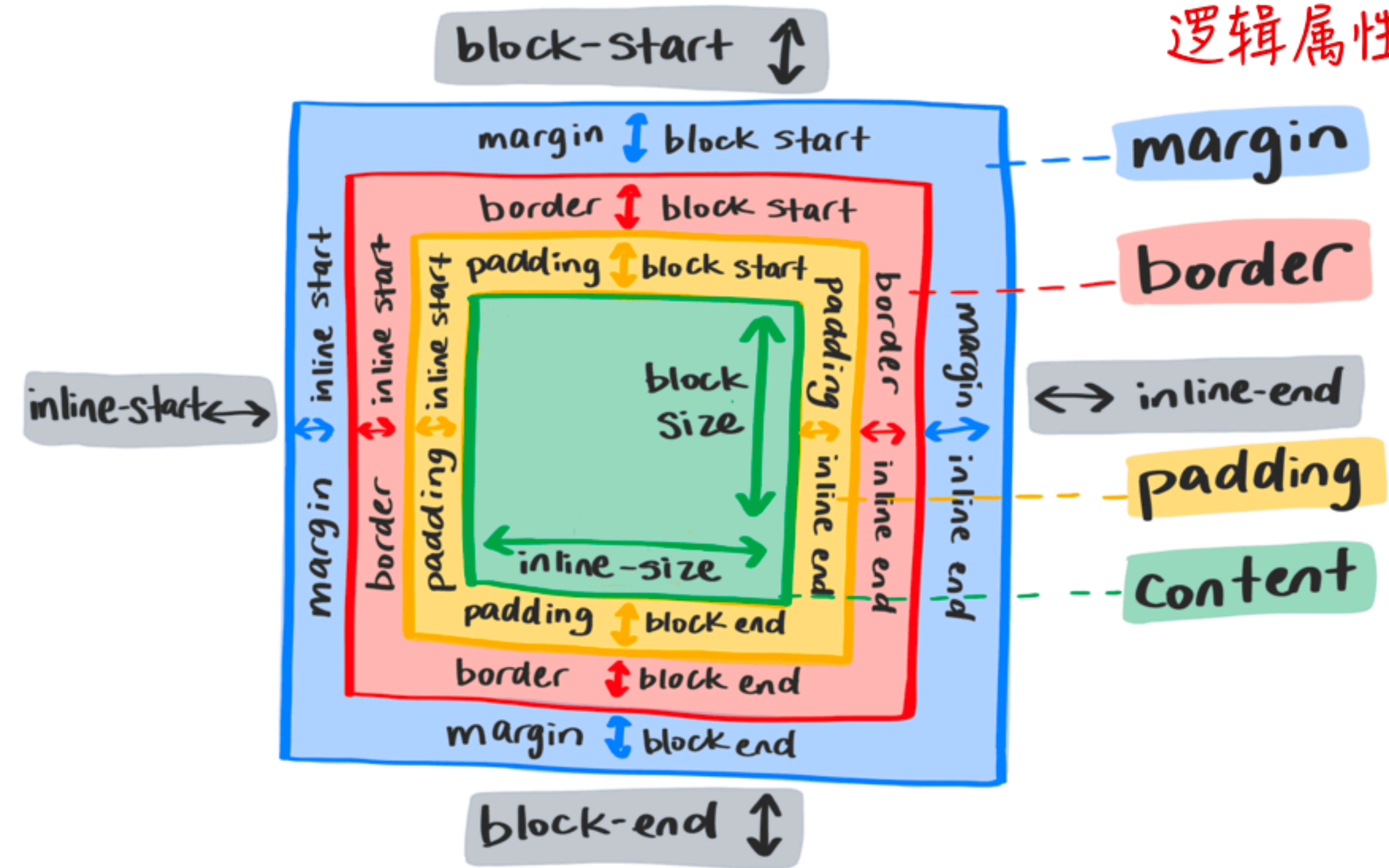
```
1 .avatar {  
2   margin-right: 1rem;  
3 }  
4  
5 html[dir="rtl"] .avatar {  
6   margin-right: 0;  
7   margin-left: 1rem;  
8 }
```

# CSS 逻辑属性

物理属性



逻辑属性



[CodePen](#) →

# CSS 逻辑属性



```
margin-inline-end: 1rem
```

[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**

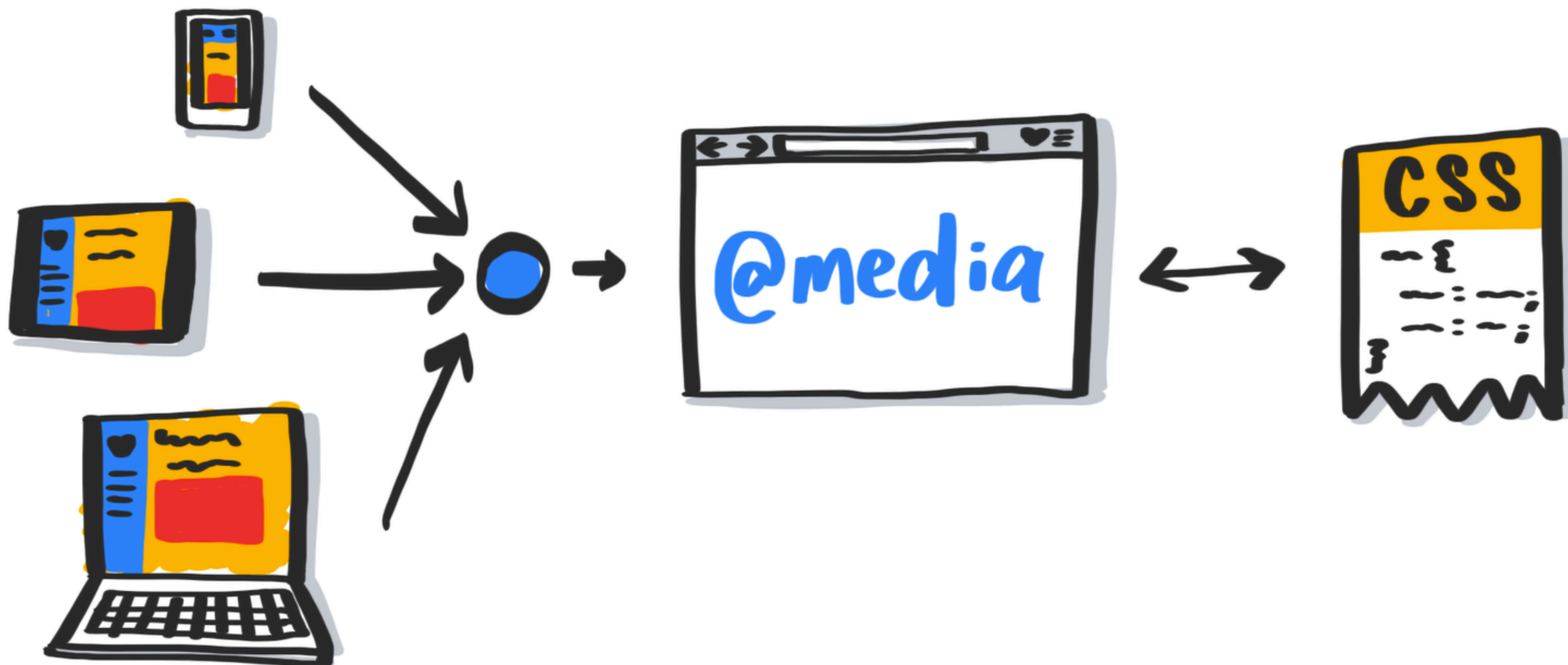
~~⑫ CSS比较函数~~

~~⑬ CSS内容可见性~~

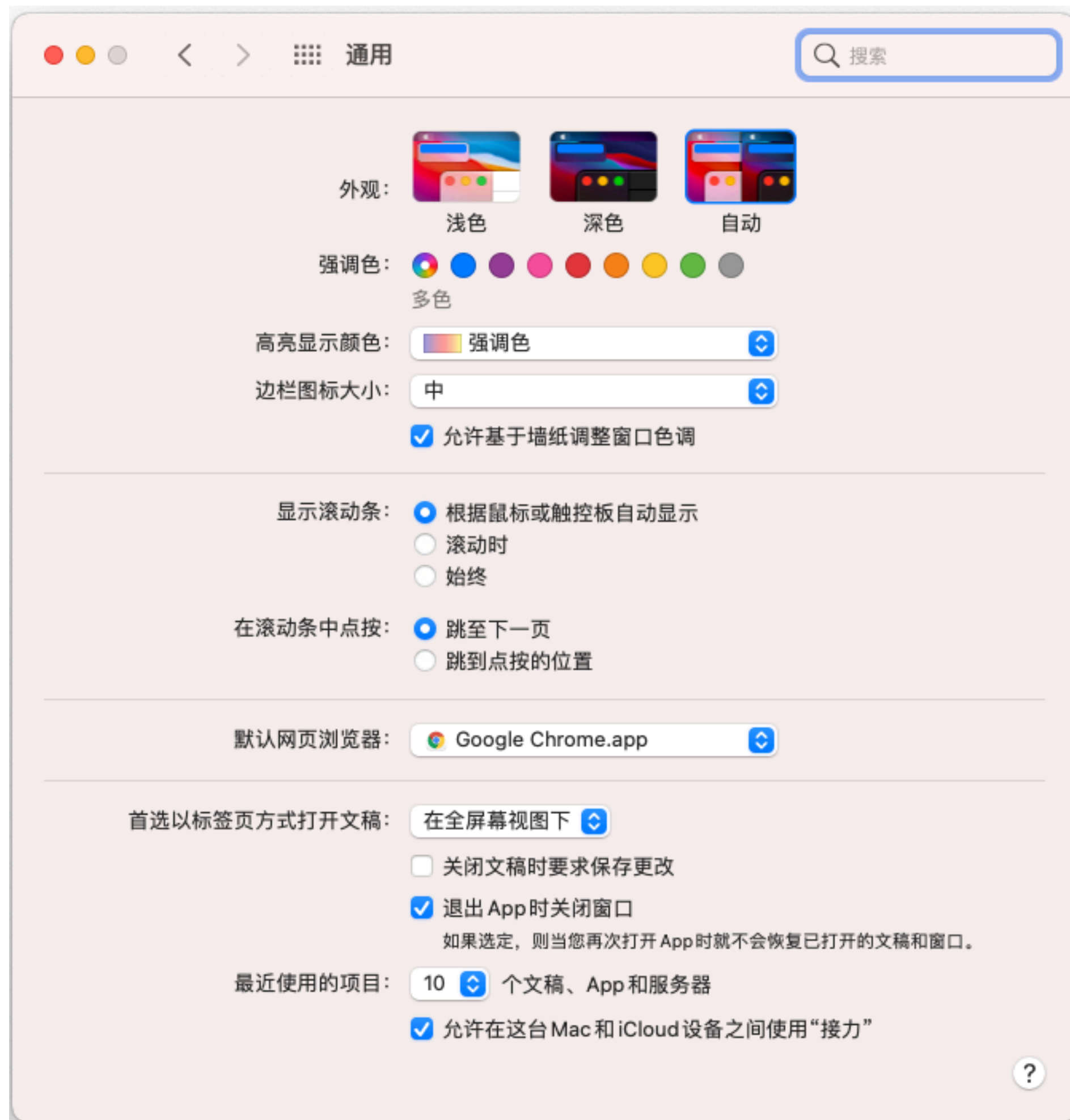
~~⑭ CSS外在尺寸和内在尺寸~~

~~⑮ CSS的Display~~

# CSS 媒体查询



# CSS 媒体查询



```
1 // dark & light mode
2 :root {
3   /* Light theme */
4   --c-text: #333;
5   --c-background: #fff;
6 }
7
8 body {
9   color: var(--c-text);
10  background-color: var(--c-background);
11 }
12
13 @media (prefers-color-scheme: dark) {
14   :root {
15     /* Dark theme */
16     --c-text: #fff;
17     --c-background: #333;
18   }
19 }
```

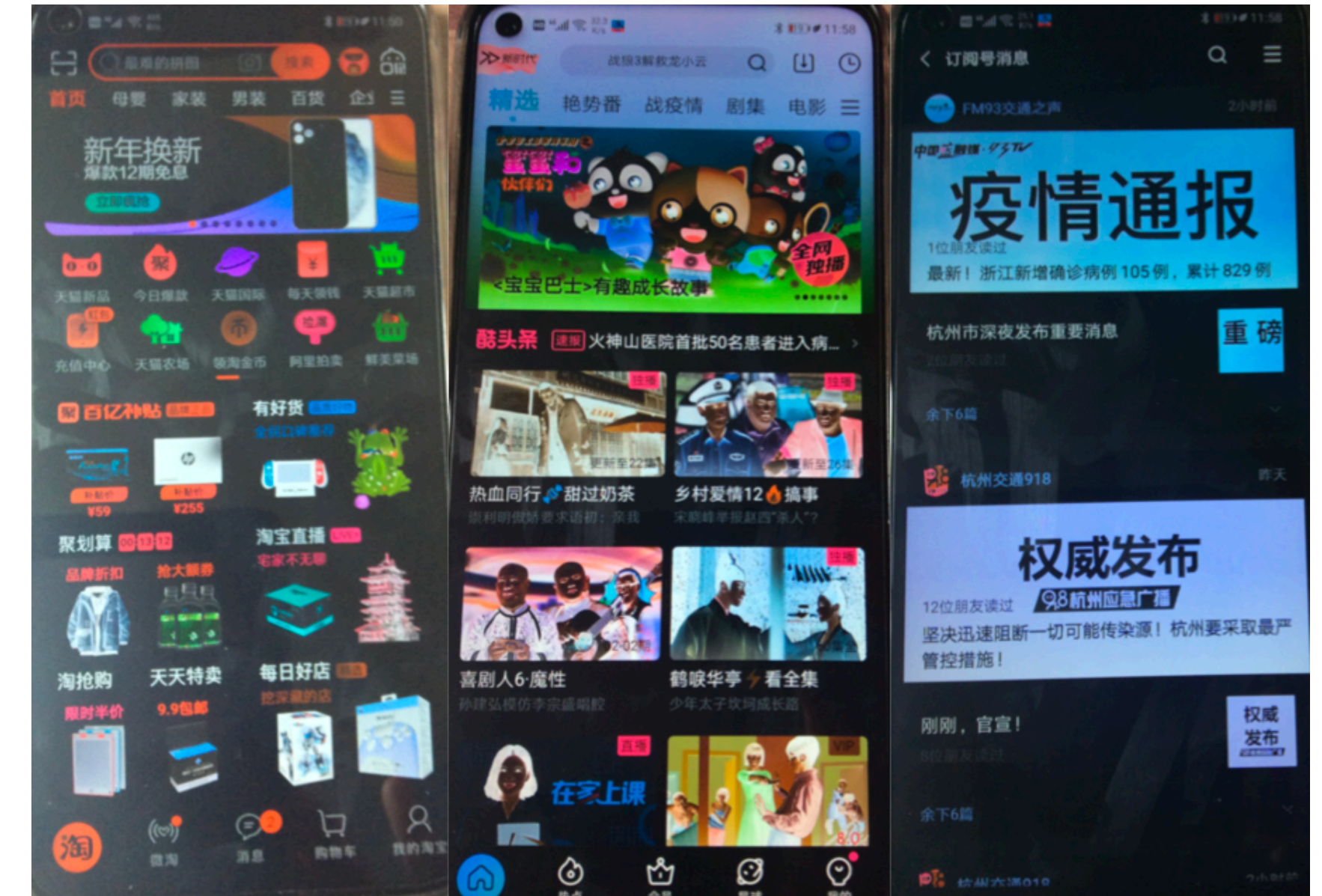
[CodePen →](#)

# CSS 媒体查询

設定

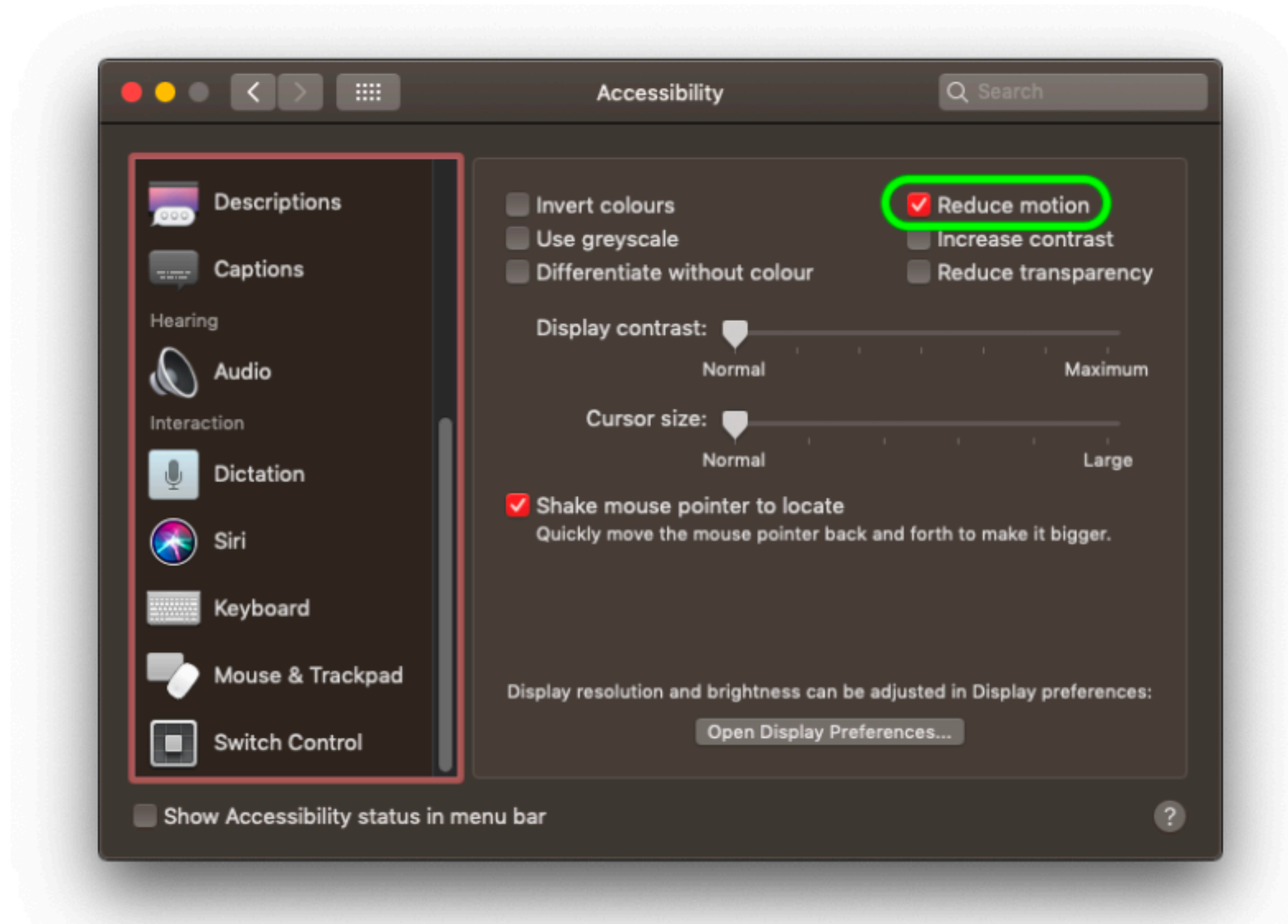
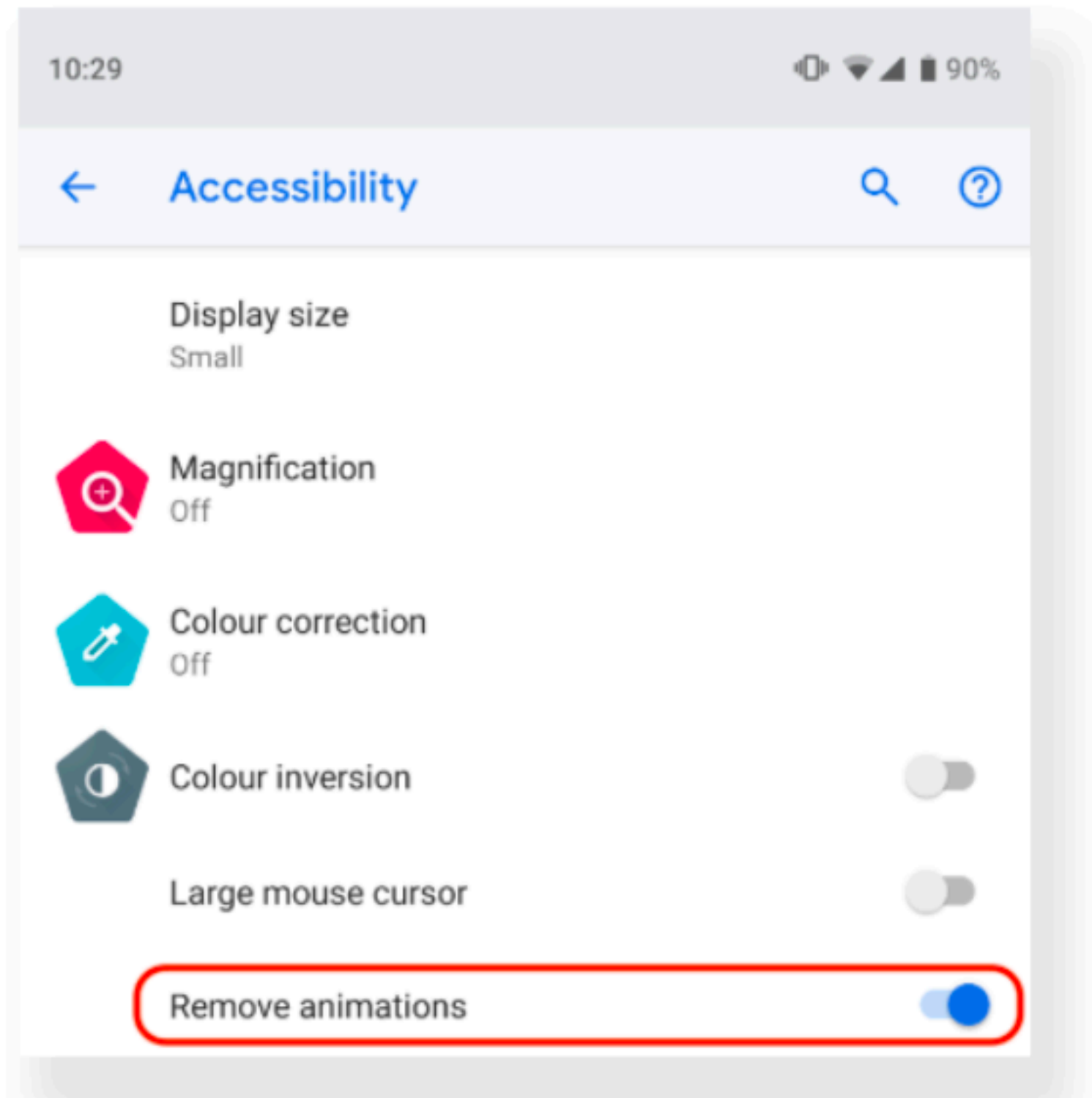
搜尋設定項目

- 勿擾模式
- 聲音與震動
- 顯示與亮度** 1
- 鎖定螢幕、桌面與桌布
- 系統升級
- 指紋與密碼
- 安全與隱私
- 電池



```
1 @media (inverted-colors) {  
2   img {  
3     filter: invert(1);  
4   }  
5   * {  
6     box-shadow: none !important; text-shadow:  
7     none !important;  
8   }  
}
```

# CSS 媒体查询



```
1 .animation {  
2   animation: vibrate 0.3s linear infinite both;  
3 }  
4  
5 @media (prefers-reduced-motion: reduce) {  
6   .animation { animation: none; }  
7 }
```

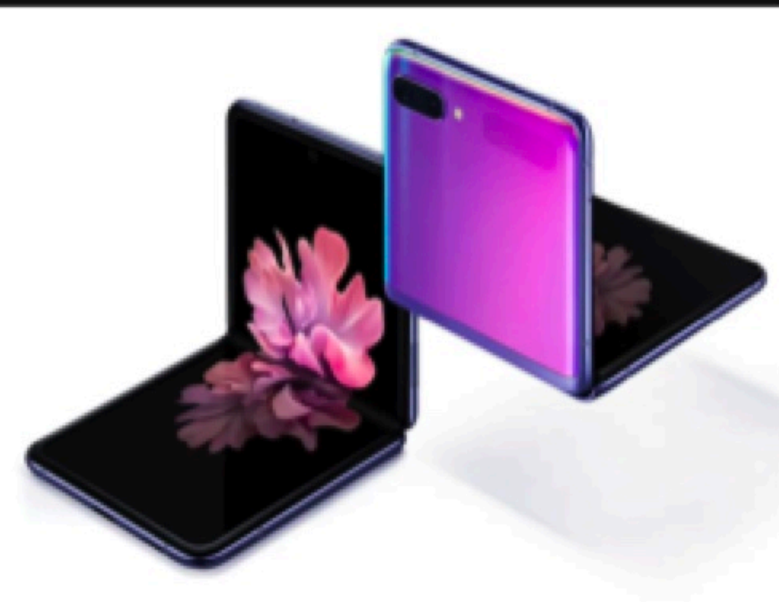
[CodePen →](#)



# CSS 媒体查询



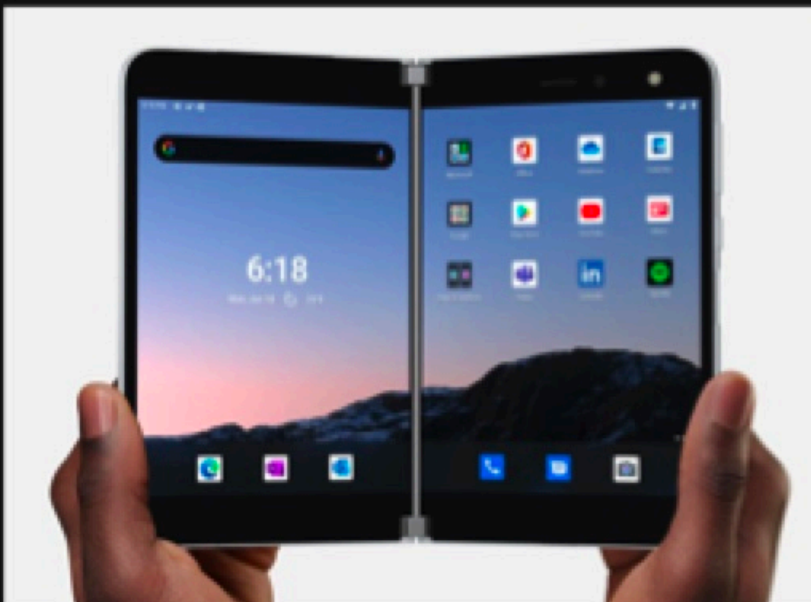
Samsung Galaxy Z Fold 2



Samsung Galaxy Z Flip



Huawei Mate Xs



Microsoft Surface Duo



Moto Razr 2019



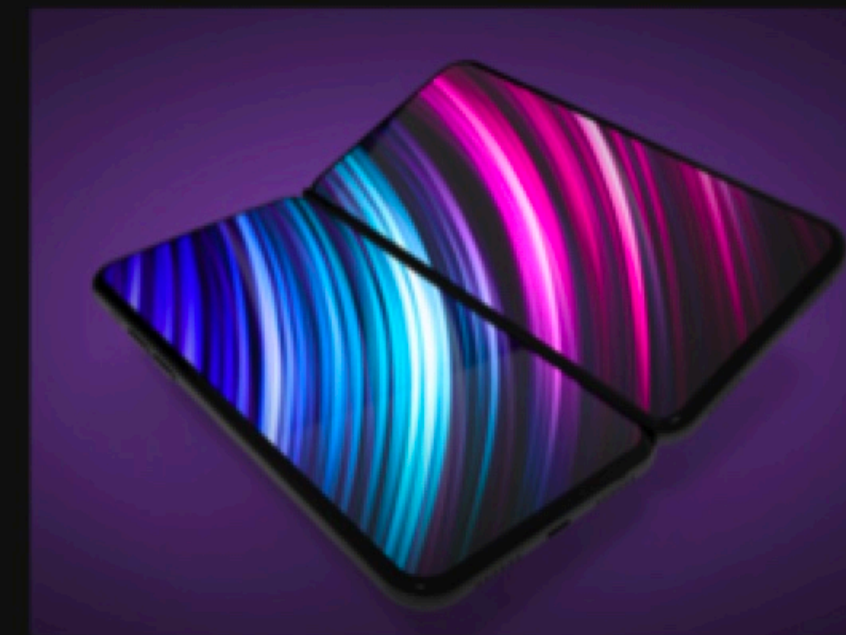
Royole Flexpai



LG G8X ThinkQ



Xiaomi Dual Flex



Apple's Plans For Foldable iPhone



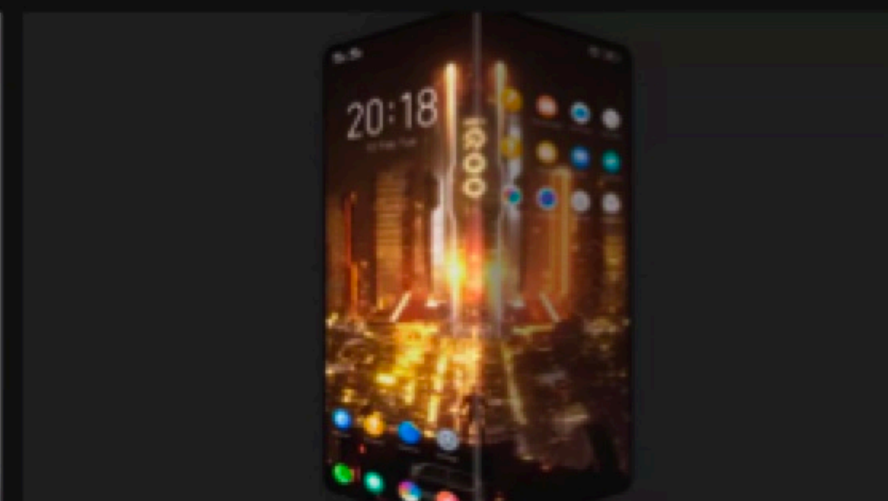
OPPO Foldable Smartphone



TCL

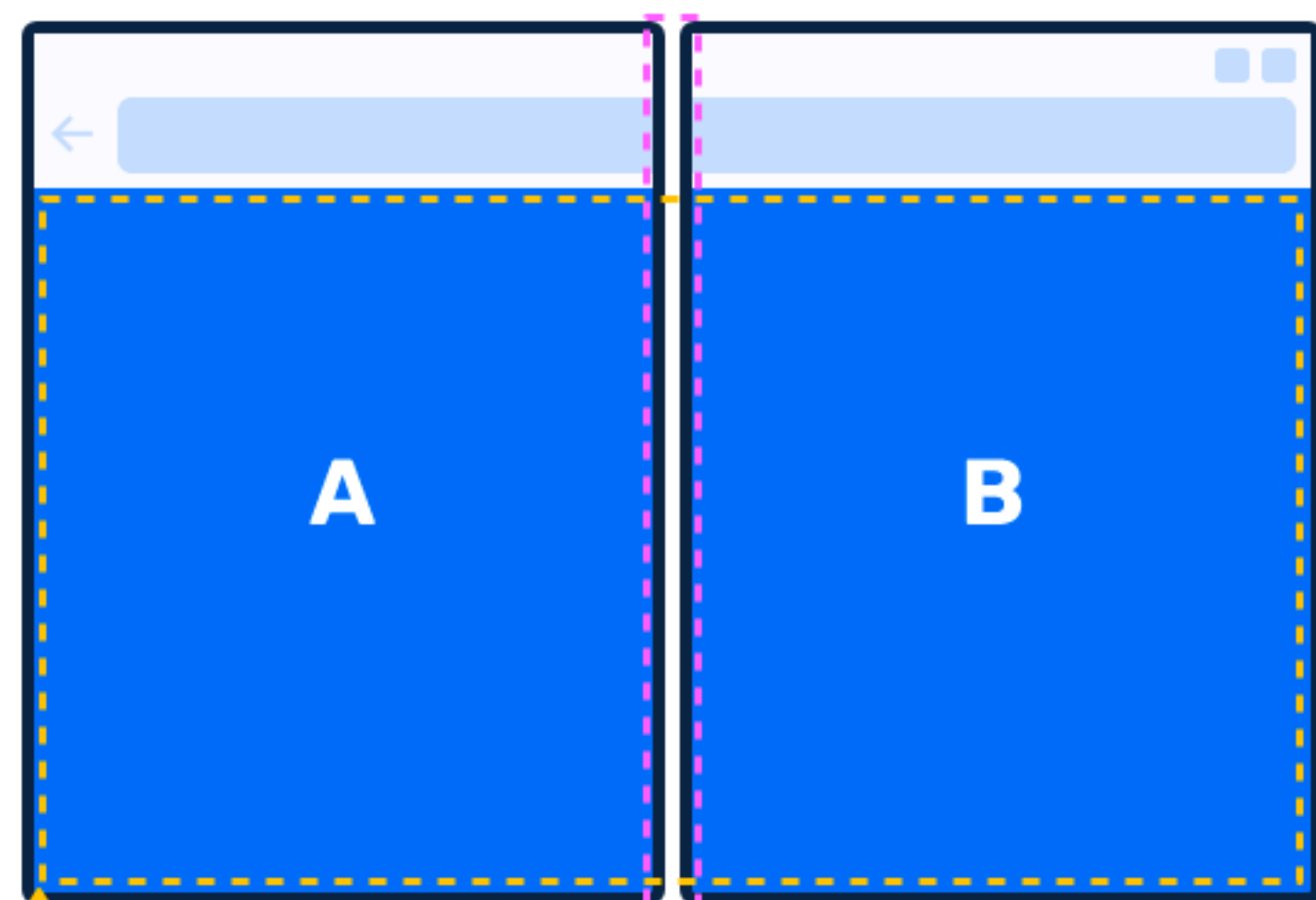


Sony Xperia F



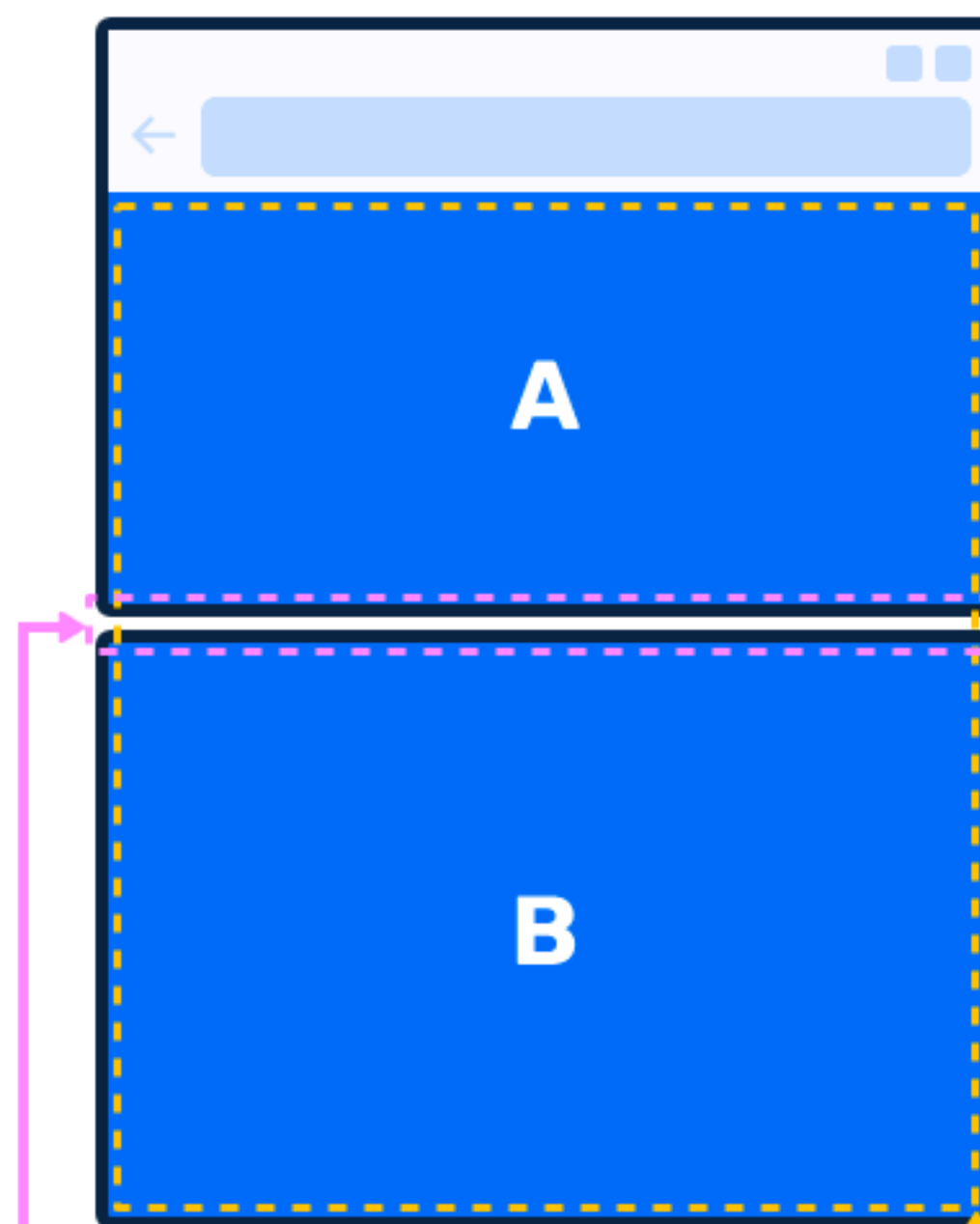
Vivo IQOO

# CSS 媒体查询



orientation:landscape

spanning: single-fold-vertical

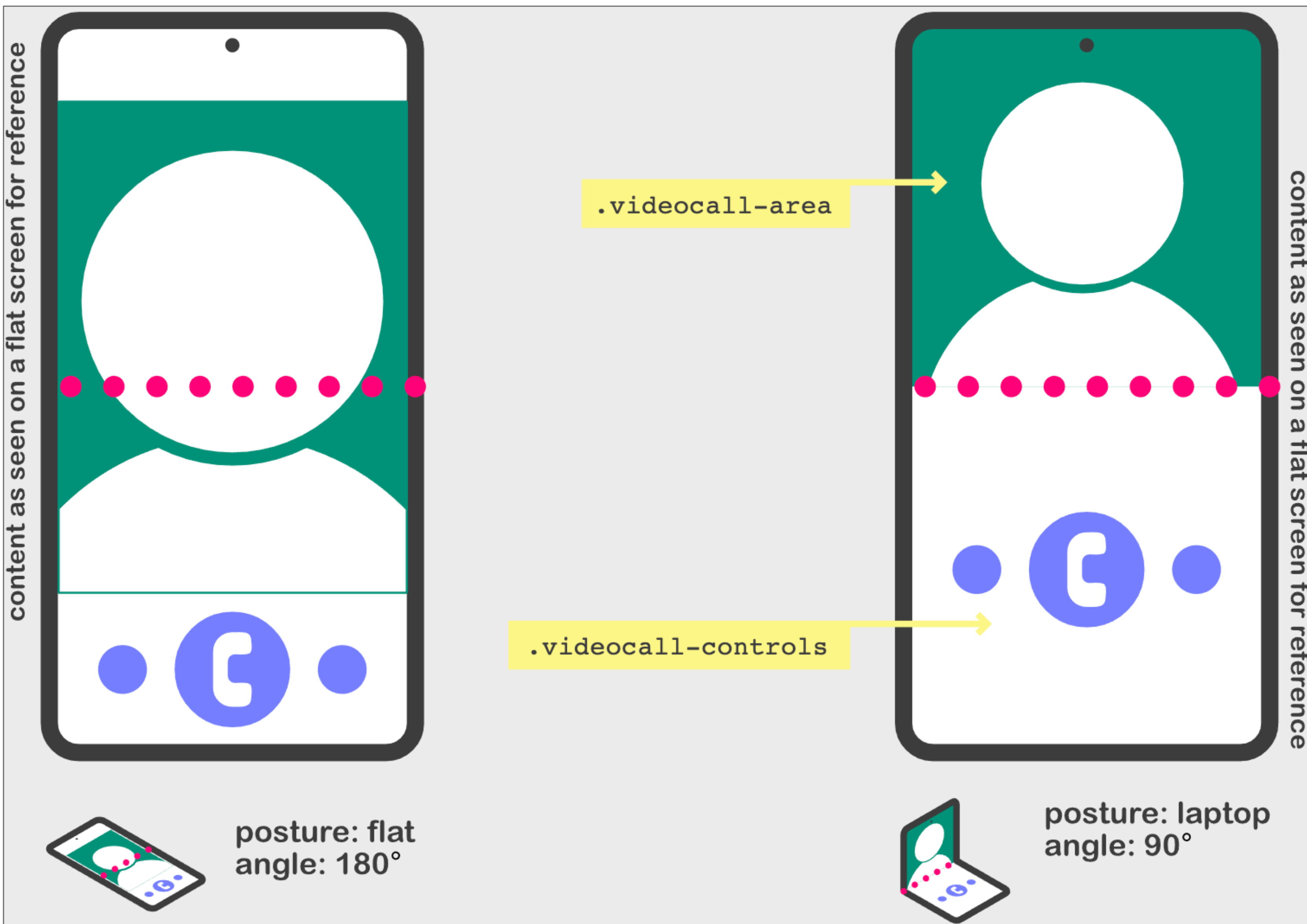


spanning: single-fold-horizontal

orientation:portrait

```
1 @media (spanning: single-fold-vertical) {  
2     // css  
3  
4 }
```

# 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~~

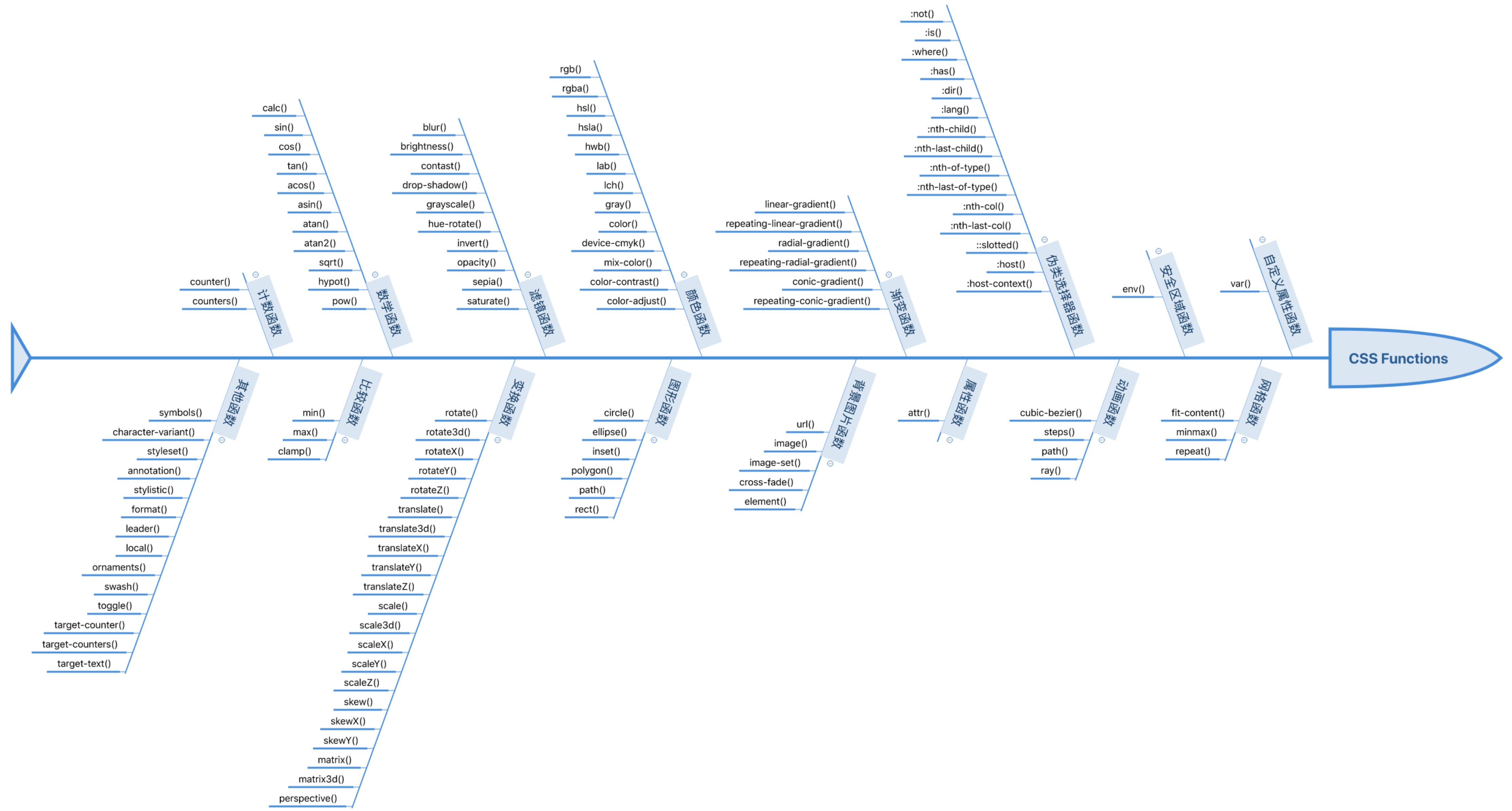
**⑫ CSS比较函数**

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()

CodePen - min() Function

width: 500px

$\min(50vw, 500px)$

viewport width = 1200  $\rightarrow$  50vw = 600px

$\min(50vw, 500px) \rightarrow \min(600px, 500px)$

```
<div class="element"></div>
```

border-radius: 4vh;  
background-color: #2196f3;  
width: min(50vw, 500px);

500x98

CodePen - min() Function

width: 50vw

$\min(50vw, 500px)$

viewport width = 760px  $\rightarrow$  50vw = 380px

$\min(50vw, 500px) \rightarrow \min(380px, 500px)$

```
<div class="element"></div>
```

min-height: 50vh;  
border-radius: 4vh;  
background-color: #2196f3;  
width: min(50vw, 500px);

380x98

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

[CodePen](#)  $\rightarrow$

# CSS 比较函数 min()/max()/clamp()

CodePen - A Pen by Airen

https://cdpn.io/airen/debug/oNjdGyv/mWAoNLqKgqjr

1200px x 287px

width: 50vw

$\max(50vw, 500px)$

viewport width = 1200  $\rightarrow$  50vw = 600px

$\max(50vw, 500px) \rightarrow \max(600px, 500px)$

```
<!DOCTYPE html>
<html lang="en">
  <head>
  </head>
  <body translate="no">
    <div class="element"></div>
  </body>
  <scrollbar orient="horizontal" clickthrough="always" root="true" active="true" curpos="0" disabled="true" maxpos="0" pageincrement="832" increment="35"></scrollbar>
  <scrollbar orient="vertical" clickthrough="always" root="true" curpos="0" active="true" disabled="true" maxpos="0" pageincrement="157" increment="50"></scrollbar>
  <scrollcorner></scrollcorner>
  <div class="moz-custom-content-container" role="presentation">
  </div>
</html>
```

元素 { 内联: 30 }  
:element { 内联: 30 }  
min-height: 20vh;  
border-radius: 2vh;  
background-color: #2196f3;

margin: 0 0 0 0  
border: 0 0 0 0  
padding: 0 0 0 0  
600x57.4

600x57.4 static

CodePen - A Pen by Airen

https://cdpn.io/airen/debug/oNjdGyv/mWAoNLqKgqjr

760px x 287px

width: 500px

$\max(50vw, 500px)$

viewport width = 760px  $\rightarrow$  50vw = 380px

$\max(50vw, 500px) \rightarrow \max(380px, 500px)$

```
<!DOCTYPE html>
<html lang="en">
  <head>
  </head>
  <body translate="no">
    <div class="element"></div>
  </body>
  <scrollbar orient="horizontal" clickthrough="always" root="true" active="true" curpos="0" disabled="true" maxpos="0" pageincrement="832" increment="35"></scrollbar>
  <scrollbar orient="vertical" clickthrough="always" root="true" curpos="0" active="true" disabled="true" maxpos="0" pageincrement="157" increment="50"></scrollbar>
  <scrollcorner></scrollcorner>
  <div class="moz-custom-content-container" role="presentation">
  </div>
</html>
```

元素 { 内联: 30 }  
:element { 内联: 30 }  
min-height: 20vh;  
border-radius: 2vh;  
background-color: #2196f3;

margin: 0 0 0 0  
border: 0 0 0 0  
padding: 0 0 0 0  
500x57.4

500x57.4 static

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

[CodePen](#)  $\rightarrow$

# CSS 比较函数 min()/max()/clamp()

width: 500px

`clamp(100px, 50vw, 500px)`

viewport width = 1200  $\rightarrow$  50vw = 600px

`clamp(100px, 50vw, 500px) \rightarrow clamp(100px, 600px, 500px)`

MIN=100px VAL=50vw MAX=500px  $\rightarrow$  VAL > MAX VAL > MIN

width: 50vw

`clamp(100px, 50vw, 500px)`

viewport width = 760  $\rightarrow$  50vw = 380px

`clamp(100px, 50vw, 500px) \rightarrow clamp(100px, 380px, 500px)`

MIN < VAL < MAX

width: 100px

`clamp(100px, 50vw, 500px)`

viewport width = 170  $\rightarrow$  50vw = 85px

`clamp(100px, 50vw, 500px) \rightarrow clamp(100px, 85px, 500px)`

MIN < VAL < MAX

- 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) }`

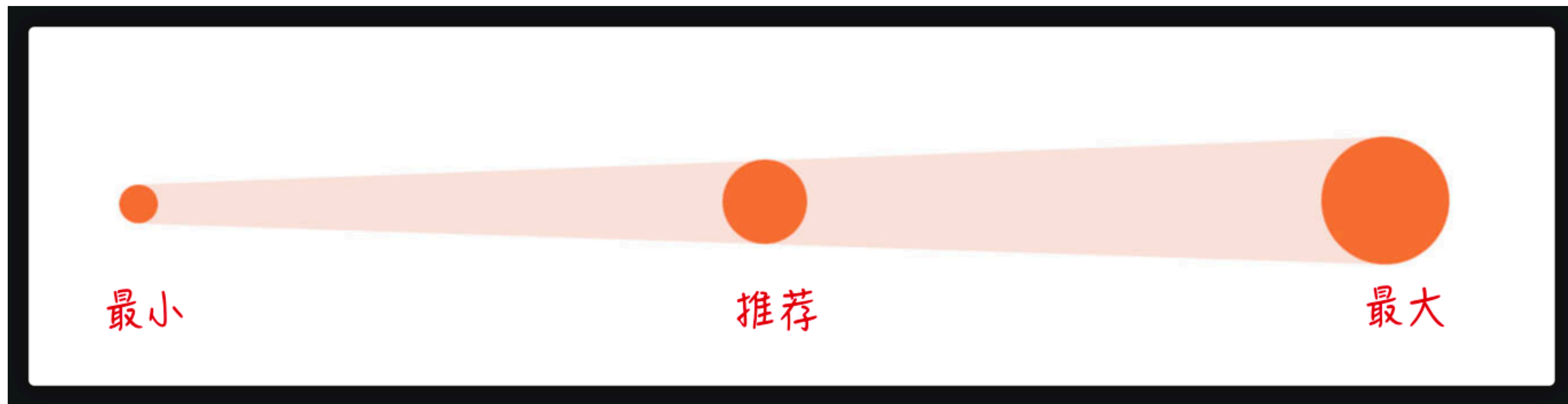
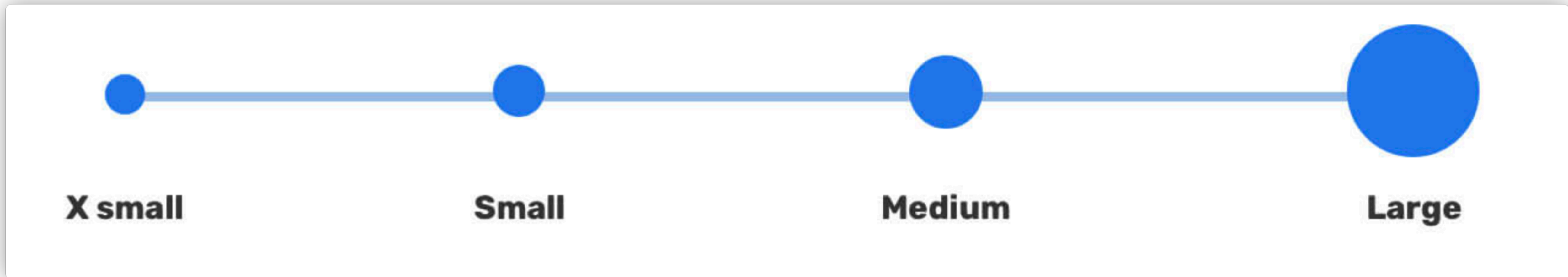
[CodePen](#)  $\rightarrow$



# CSS 比较函数 min()/max()/clamp()

```
1 .element {
2   width: clamp(100px, 50vw, 500px);
3
4 // 50vw相当于视窗宽度的一半,
5 // 如果视窗宽度是760px的话, 那么50vw相当等于380px
6   width: clamp(100px, 380px, 500px);
7
8 // 用min()和max()描述
9 // clamp(MIN, VAL, MAX) = max(MIN, min(VAL,
   Max))
10  width: max(100px, min(380px, 500px))
11
12 // min(380px, 500px)返回的值是380px
13  width: max(100px, 380px)
14
15 // max(100px, 380px)返回的值是380px
16  width: 380px;
17 }
```

# CSS 比较函数 min()/max()/clamp()



# CSS 比较函数 min()/max()/clamp()

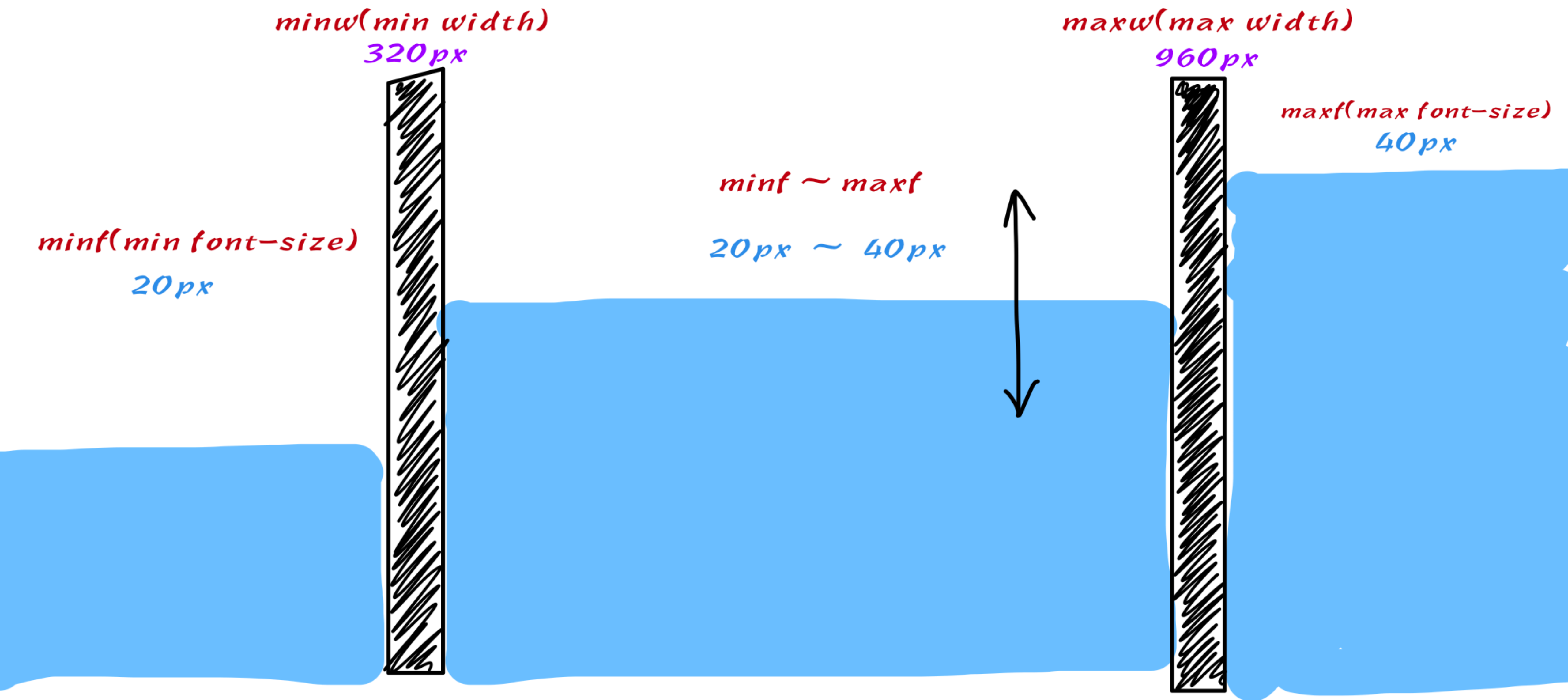
最小 16px ——— CSS Comparison Functions

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()

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

```
1 font-size:
2   calc(
3     [minf]px + ([maxf] - [minf]) * ( (100vw -
4     [minw]px) / ([maxw] - [minw]) )
5 );
```

```
1 // minf: 20px (min font-size)
2 // maxf: 40px (max font-size)
3 // current vw: 100vw
4 // minw: 320px (min viewport's width)
5 // maxw: 960px (max viewport's width)
6 body {
7   font-size:
8     calc(20px + (40 - 20) * ( (100vw -
9     320px) / (960 - 320) ));
10 }
11 @media only screen and (max-width: 320px) {
12   body {
13     font-size: 20px;
14   }
15 }
16 @media only screen and (max-width: 960px) {
17   body {
18     font-size: 40px;
19   }
20 }
```

[CodePen →](#)

# CSS 比较函数 min()/max()/clamp()

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

[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~~

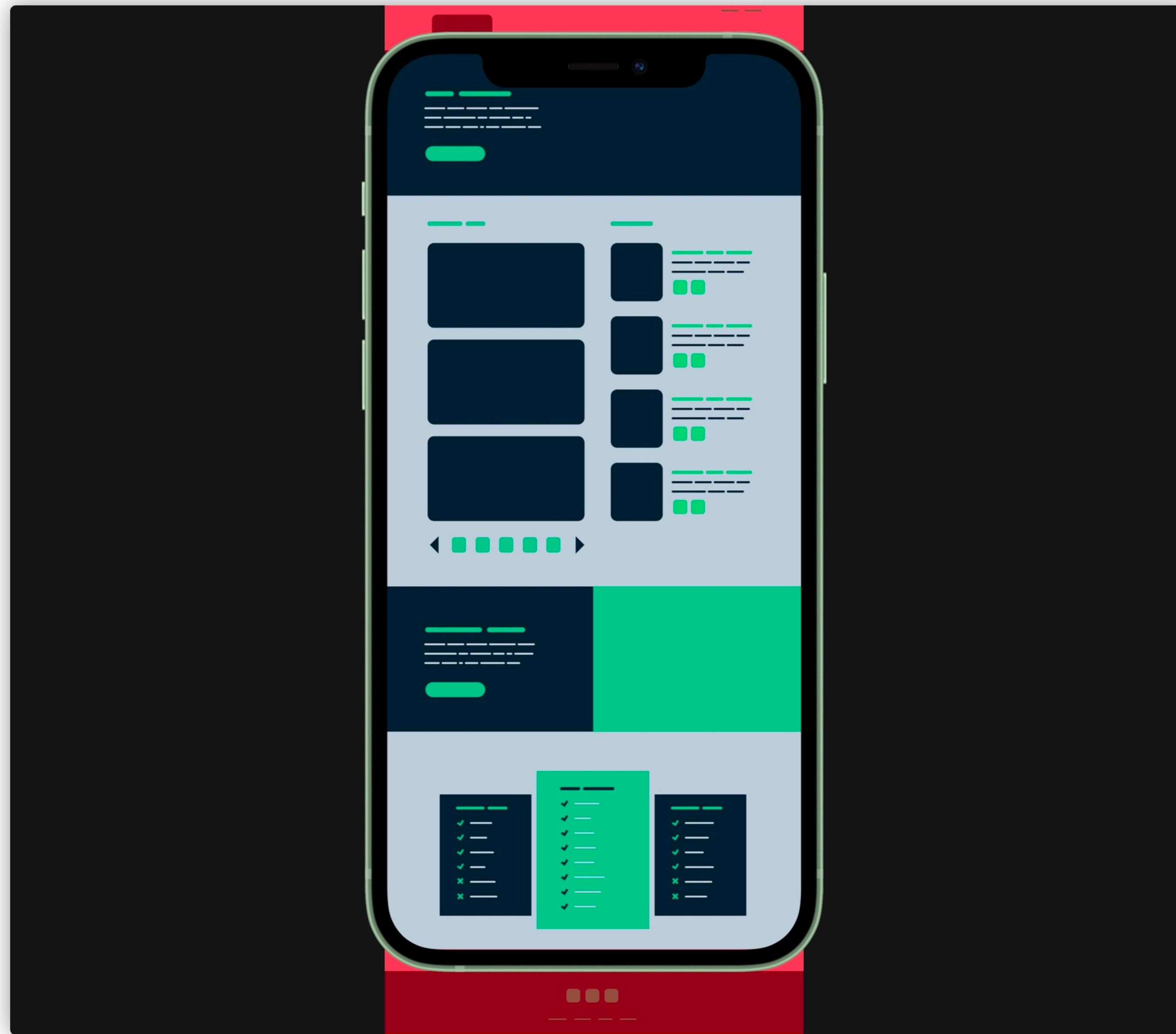
~~⑫ CSS比较函数~~

**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

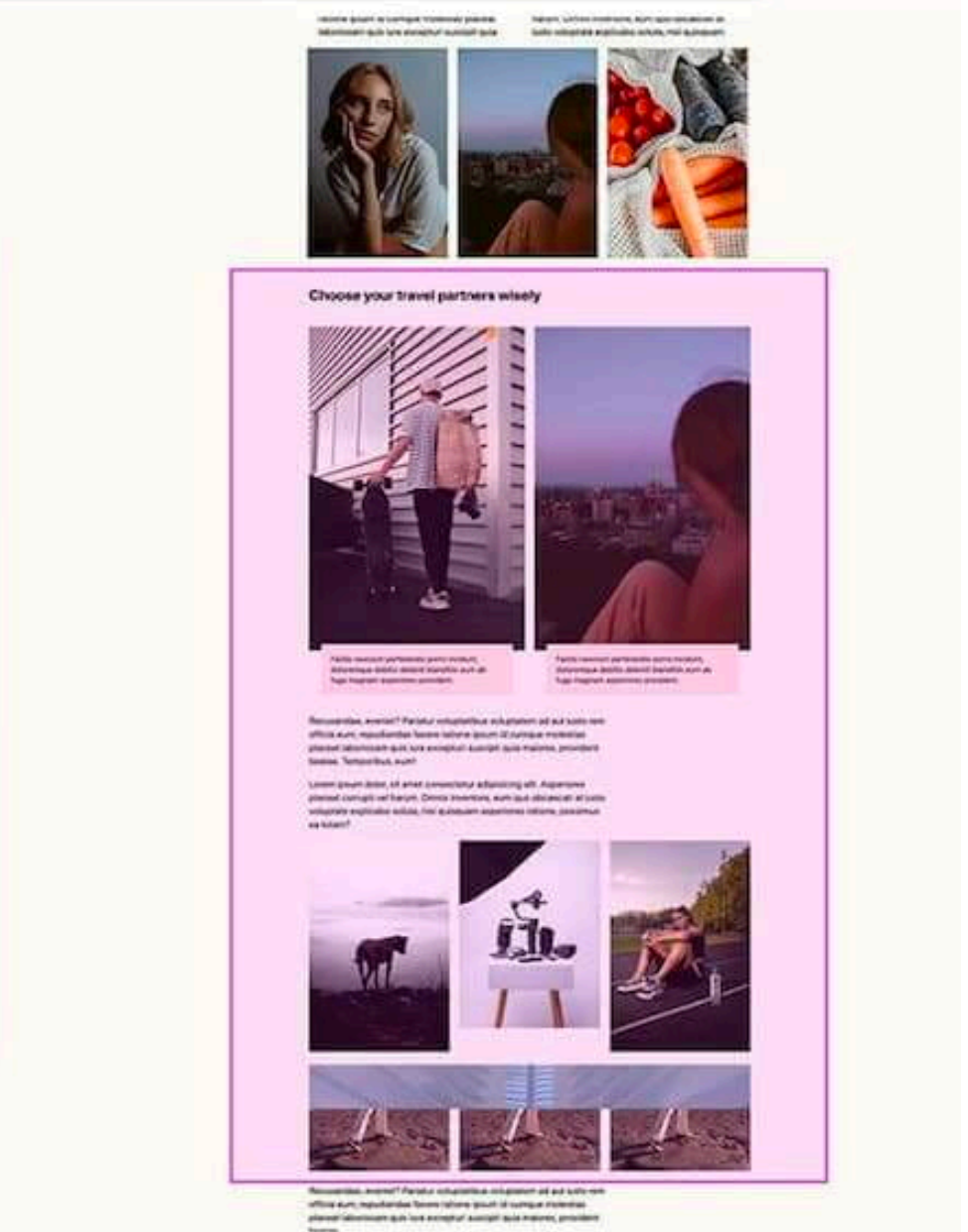
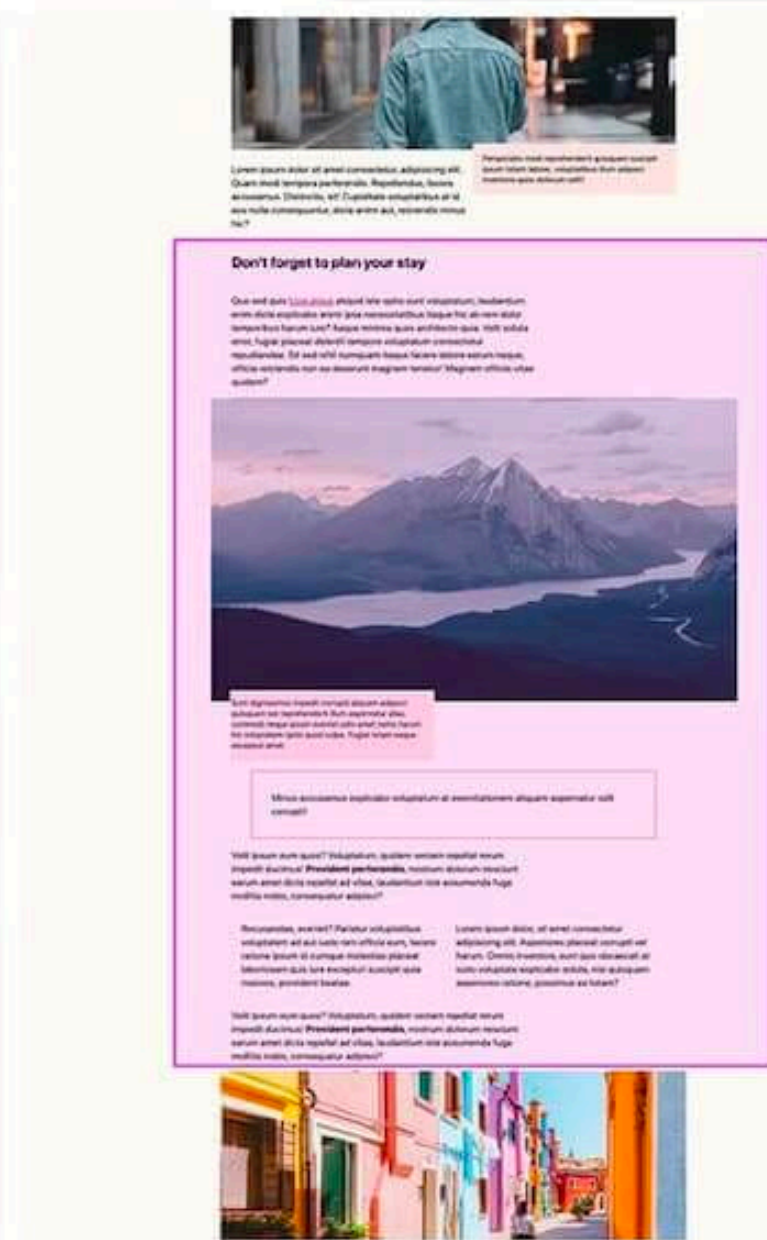
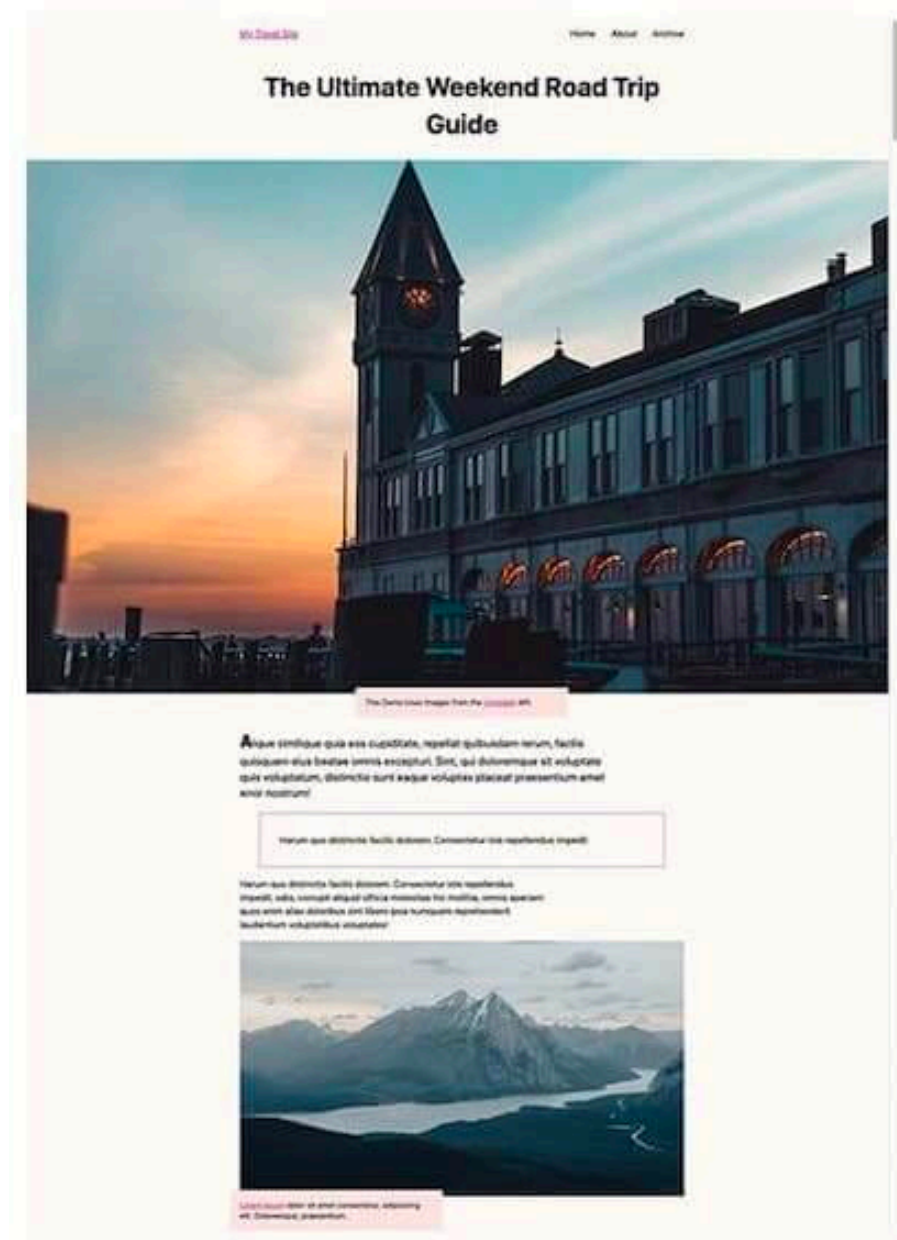




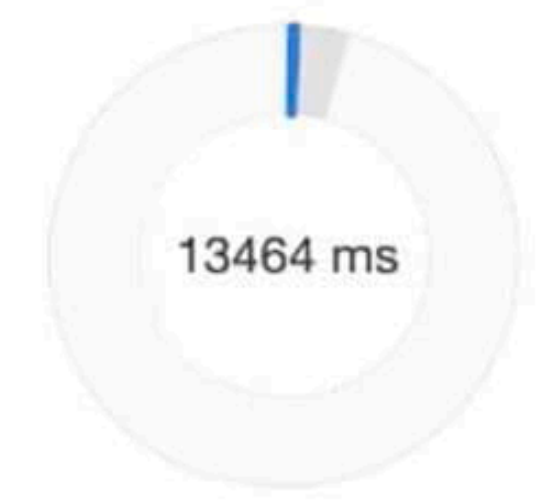
# CSS 内容可见性 content-visibility

baseline

chunks with content-visibility: auto



23 ms	■ Loading
22 ms	■ Scripting
232 ms	■ Rendering
55 ms	■ Painting
320 ms	■ System
1637 ms	□ Idle
<b>2288 ms</b>	<b>Total</b>



23 ms	■ Loading
25 ms	■ Scripting
30 ms	■ Rendering
34 ms	■ Painting
476 ms	■ System
12877 ms	□ Idle
<b>13464 ms</b>	<b>Total</b>

# 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~~

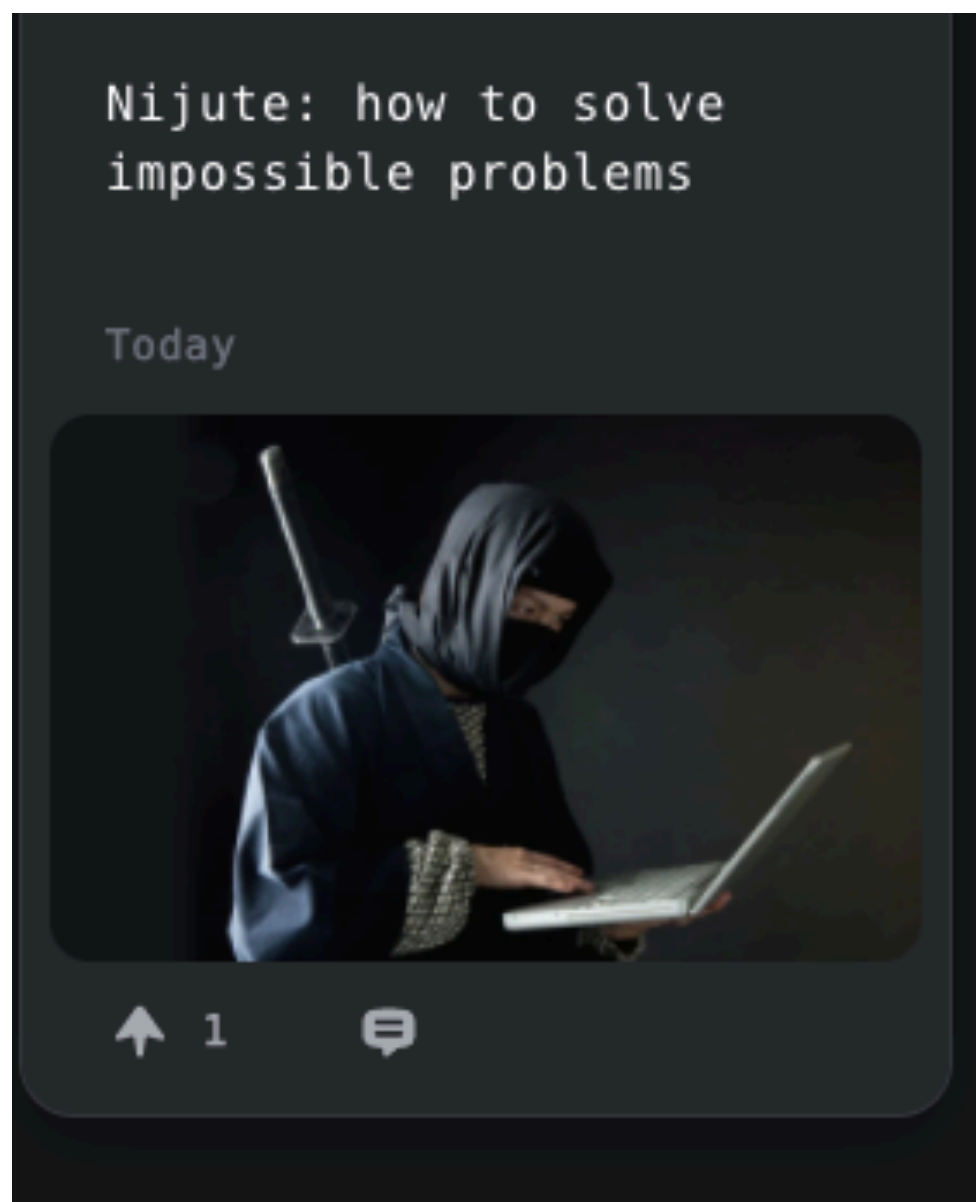
~~⑫ CSS比较函数~~

~~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">  
  <!-->  
  ▶  == $0  
</div>
```



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



```
element.style {  
  -webkit-user-select: none;  
  margin: auto;  
  background-color: hsl(0, 0%, 90%);  
  transition: background-color 300ms;  
  width: 400px;  
  height: auto;  
}  
  
img[Attributes Style] {  
  width: 800px;  
  aspect-ratio: auto 800 / 533;  
  height: 533px;  
}
```

# CSS 外在尺寸 vs. 内在尺寸

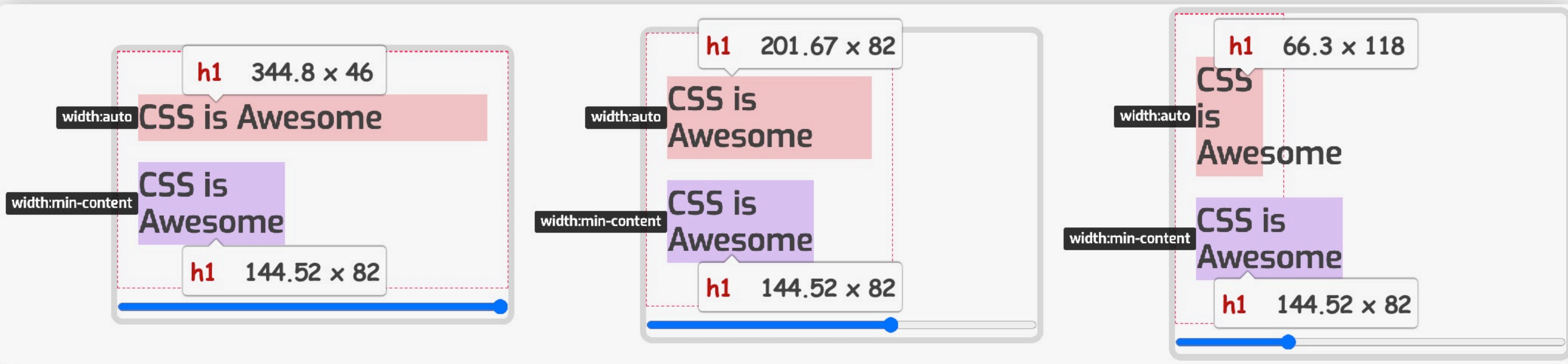
由元素内部内容来决定:

① min-content

② max-content

③ fit-content

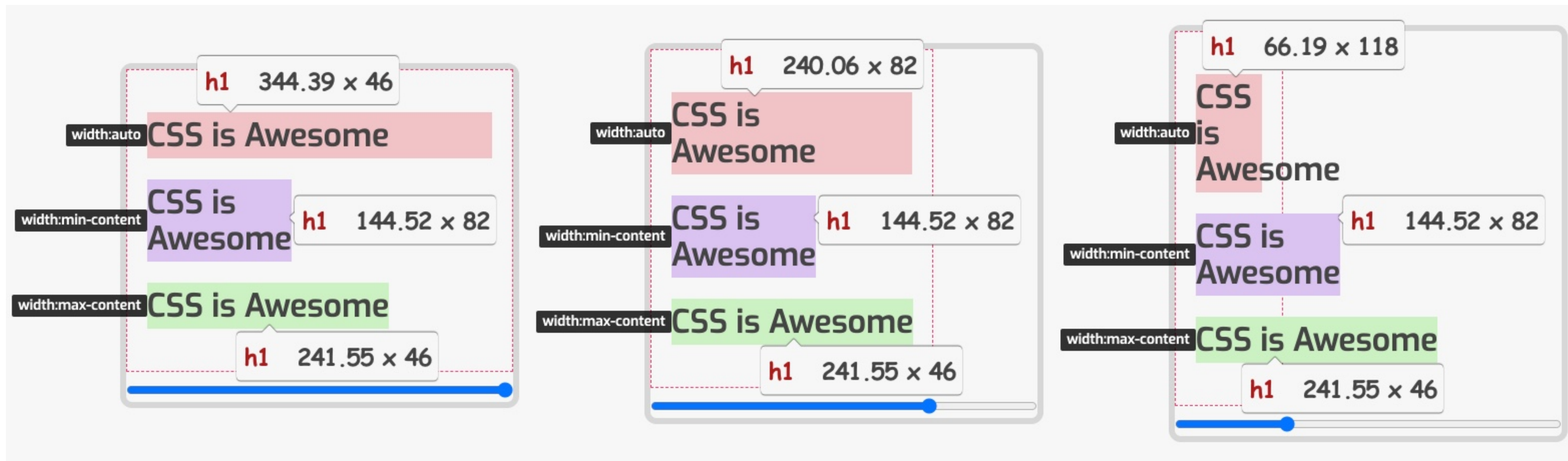
# CSS 外在尺寸 vs. 内在尺寸



```
1 <h1>CSS is Awesome</h1>
2
3 h1 {
4   width: auto; // 外在尺寸
5 }
6
7 h1 {
8   width: min-content; // 内在尺寸
9 }
```

[CodePen →](#)

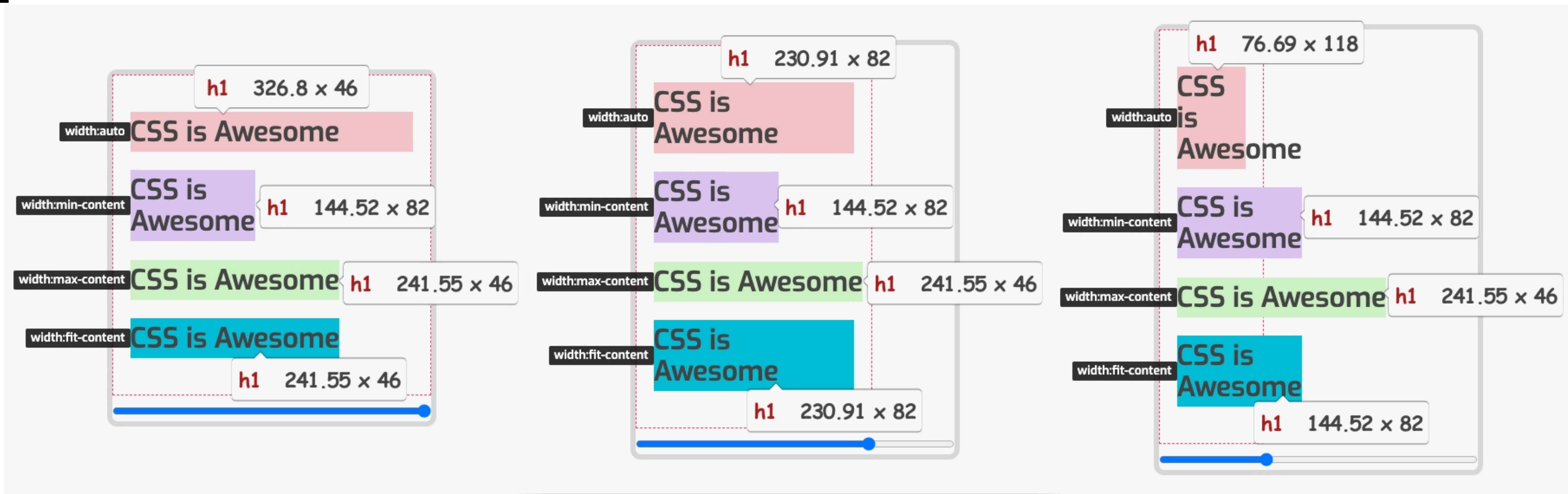
# CSS 外在尺寸 vs. 内在尺寸



```
1 // 外在尺寸
2 h1 { width: auto; }
3
4 // 内在尺寸
5 h1 { width: min-content; }
6
7 h1 { width: max-content; }
```

[CodePen →](#)

# CSS 外在尺寸 vs. 内在尺寸



```
1 // 外在尺寸
2 h1 { width: auto; }
3
4 // 内在尺寸
5 h1 { width: min-content; }
6 h1 { width: max-content; }
7 h1 { width: fit-content; }
8
```

[CodePen →](#)

# CSS 外在尺寸 vs. 内在尺寸

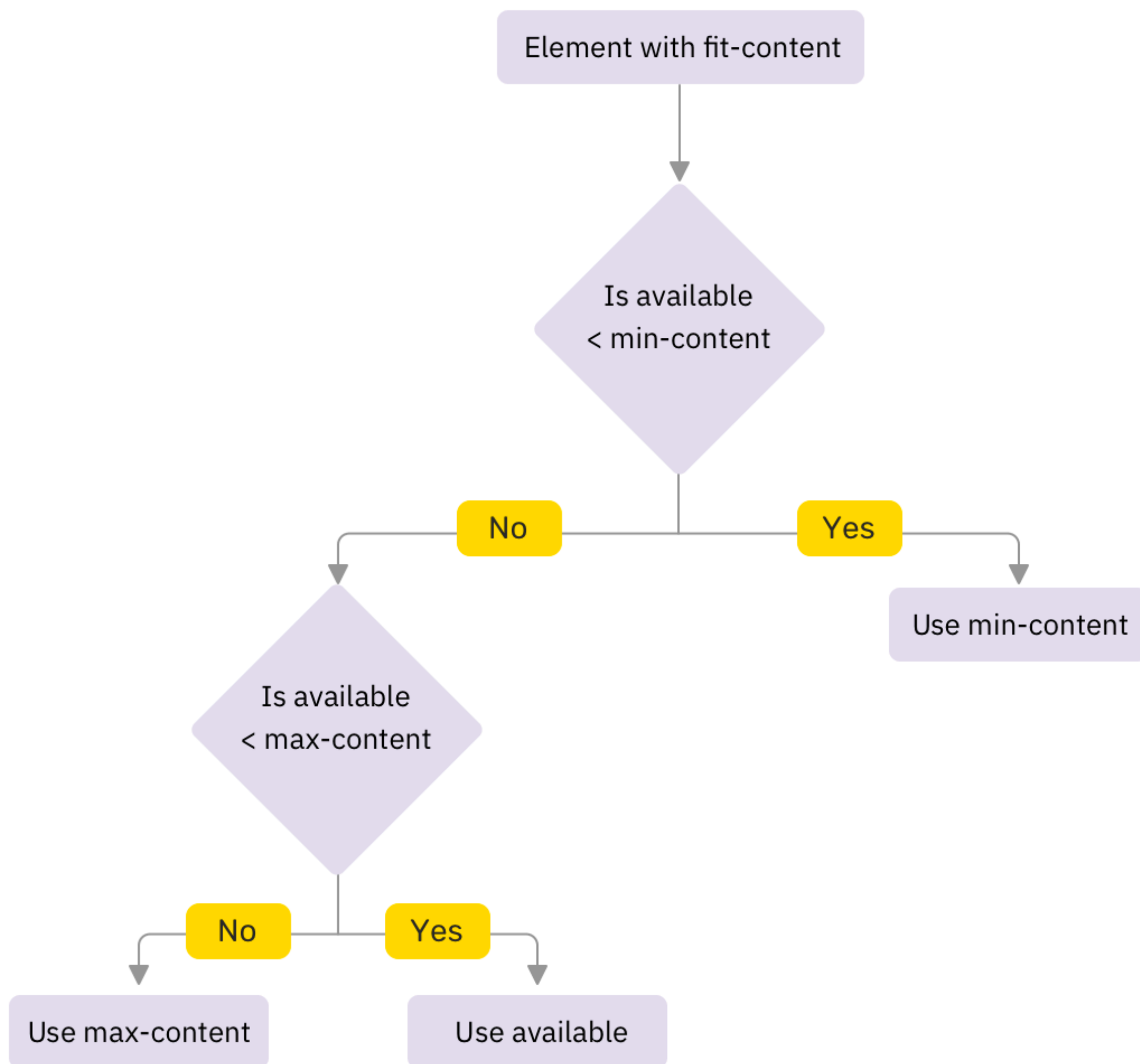
```
1 h1 { width: fit-content; }  
2  
3 // 等同于  
4 h1 {  
5   width: auto;  
6   min-width: min-content;  
7   max-width: max-content;  
8 }  
9
```

[CodePen](#) →



# CSS 外在尺寸 vs. 内在尺寸

**Available:** stands for the available space in the viewport



## 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~~

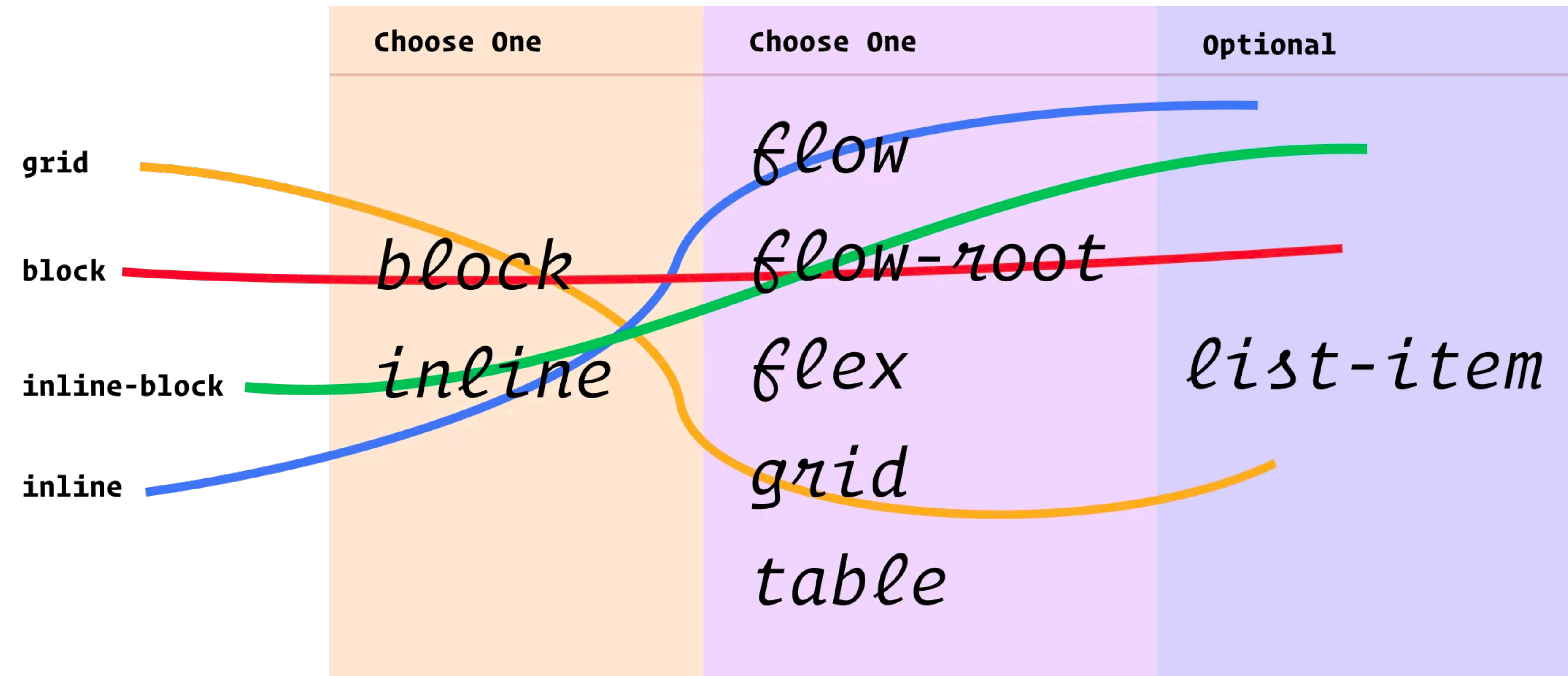
~~⑫ CSS比较函数~~

~~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

```
1 <!-- HTML -->
2 <div class="outer">
3   I'm some content
4   <div class="inner">I'm some inner
   content</div>
5 </div>
6
7 /* CSS */
8 .outer {
9   border: 2px solid lightcoral;
10  background-color: lightpink;
11  padding: 20px;
12 }
13
14 .inner {
15  background-color: #ffdb3a;
16  padding: 20px;
17 }
```

I'm some content

I'm some inner content

# 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)

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)

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

```
1 <!-- HTML -->
2 <div class="outer">
3   I'm some content
4   <div class="inner">I'm some inner content</div>
5 </div>
```

```
1 <!-- HTML -->
2
3   I'm some content
4   <div class="inner">I'm some inner content</div>
5
```

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
    
  </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
    
  </div>
</header>

▼ <header style="--display:grid;"> grid
  ▼ <a href="https://www.imgcook.com" target="_blank"> grid item
    
  </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
    
  </div>
</header>
```

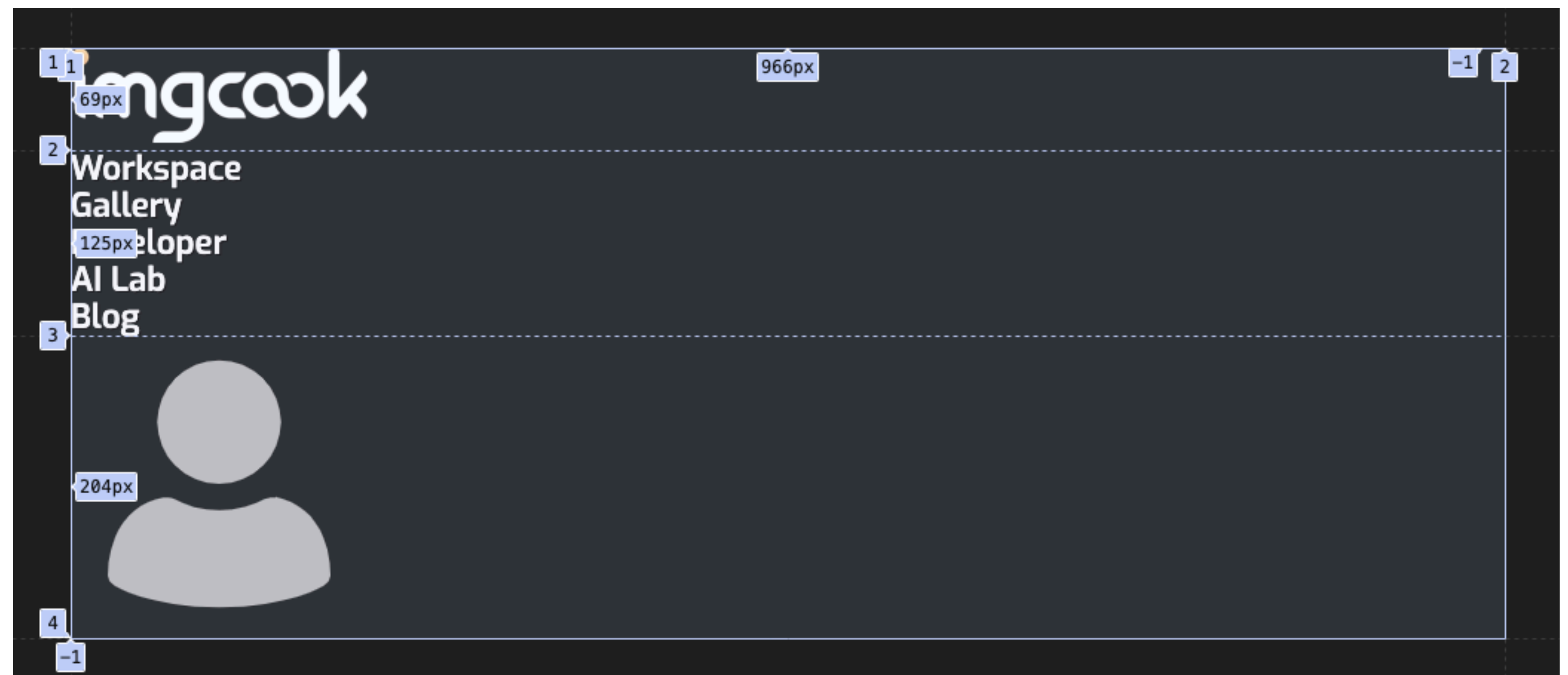
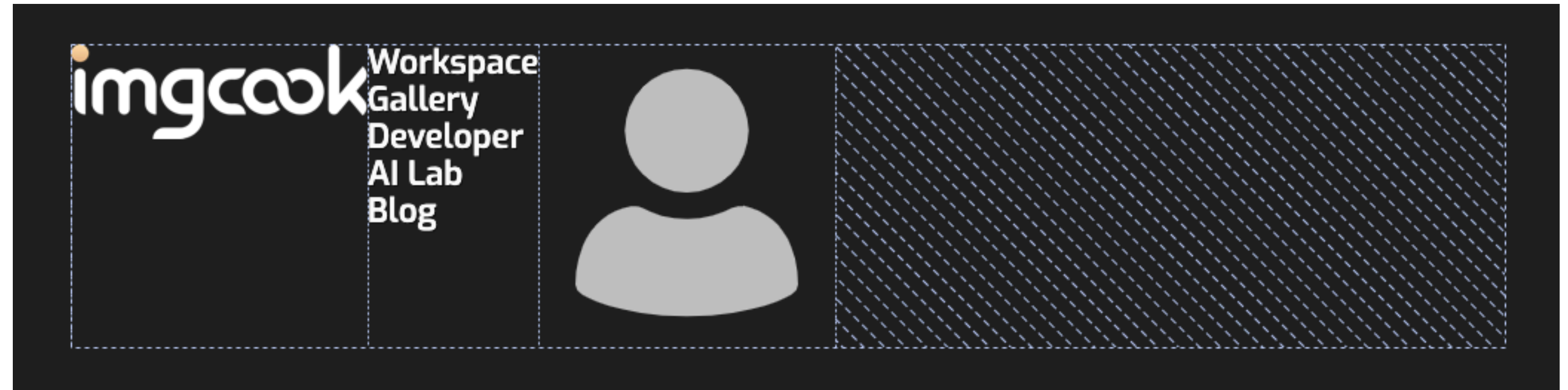
```
1 header {
2     display:
3     flex |
4     inline-flex |
5     grid |
6     inline-grid
7 }
```

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
    
  </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
    
  </div>
</header>
```

```
▼ <header style="--display:grid;"> grid
  ▼ <a href="https://www.imgcook.com" target="_blank"> grid item
    
  </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
    
  </div>
</header>
```



# CSS 的display

```
▼ <header> flex
  ▼ <a href="https://www.imgcook.com" target="_blank"> flex item
    
  </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
    
  </div>
</header>
```

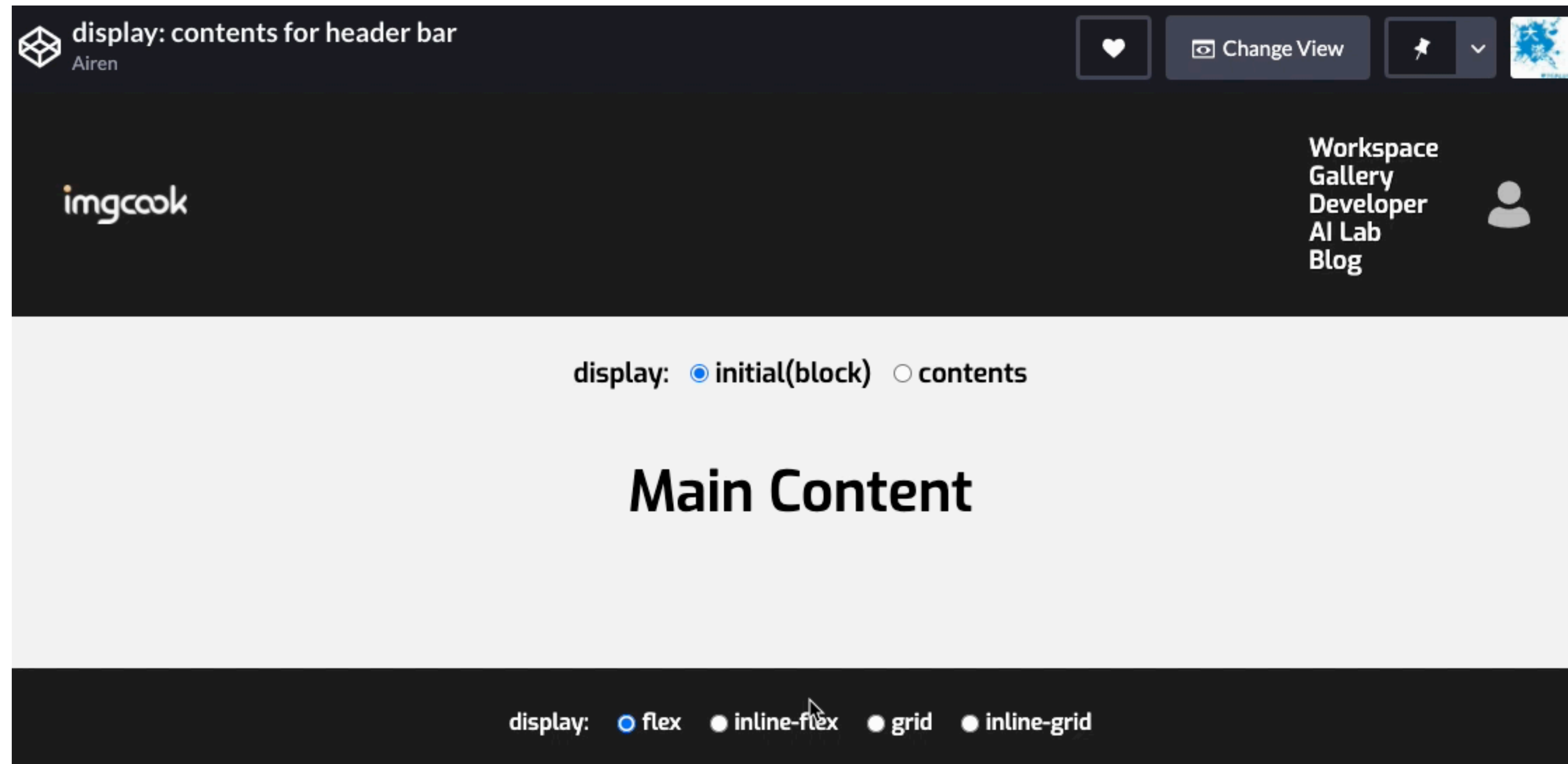
```
▼ <header style="--display:grid;"> grid
  ▼ <a href="https://www.imgcook.com" target="_blank"> grid item
    
  </a>
  ▼ <ul> grid item
    ▼ <li>
      <a href="https://www.imgcook.com/workspace" target="_blank">
        Workspace</a>
      </li>
    ><li>...</li>
    ><li>...</li>
    ><li>...</li>
    ><li>...</li>
  </ul>
  ▼ <div class="login"> grid item
    
  </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
    
  </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
    
  </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
    
  </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
    
  </div>
</header>
```

CodePen →

# CSS 的 display



The screenshot shows a CodePen workspace with a dark theme. At the top left, the title is "display: contents for header bar" by user "Airen". The workspace contains a single element with the text "imgcook". Below the element, there are two sets of radio button controls for the "display" property. The first set has "initial(block)" selected and "contents" unselected. The second set has "flex" selected, "inline-flex" unselected, "grid" unselected, and "inline-grid" unselected. The text "Main Content" is centered in the workspace area. On the right side of the workspace, there is a sidebar with the text "Workspace Gallery Developer AI Lab Blog" and a user profile icon.

[CodePen →](#)

待续...



**THANK YOU**