CSS

網頁設計X視覺特效專題班

Lecturer: LinJer (林哲)

evin92@gmail.com

網頁前端-進階
CSS

第一堂：初探CSS基礎入門

第二堂：進階CSS編寫技巧

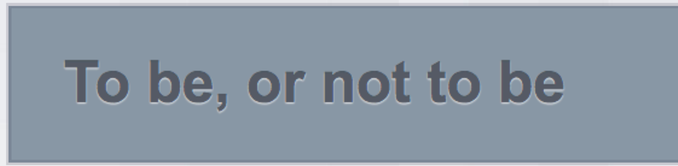
第三堂：解構CSS網頁排版

第四堂：玩轉CSS視覺特效

CSS Text Effect
仿真文字效果

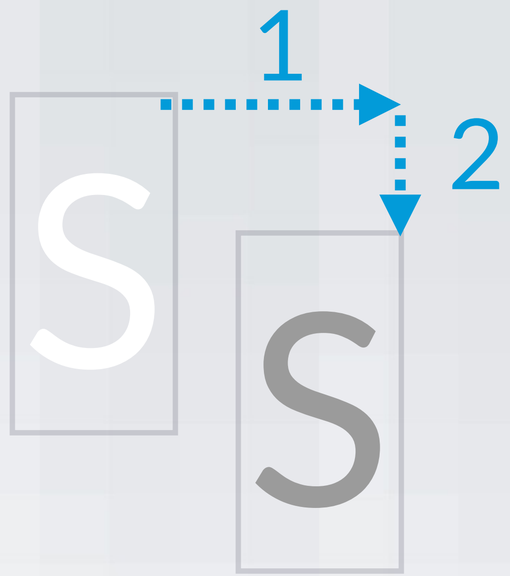
運用視覺錯覺

近看可能就破功！



只用文字陰影屬性就能完成

先來認識 text-shadow



1 2 3 4

text-shadow: 水平偏移(右為正) 垂直偏移(下為正) 模糊值 顏色值

To be, or not to be

範例: text-shadow: 6px 3px 4px gray;　前往範例

文字可以有不只一層陰影

每一層陰影都會被疊合在一起

當我們用了4層陰影:

看看怎麼做

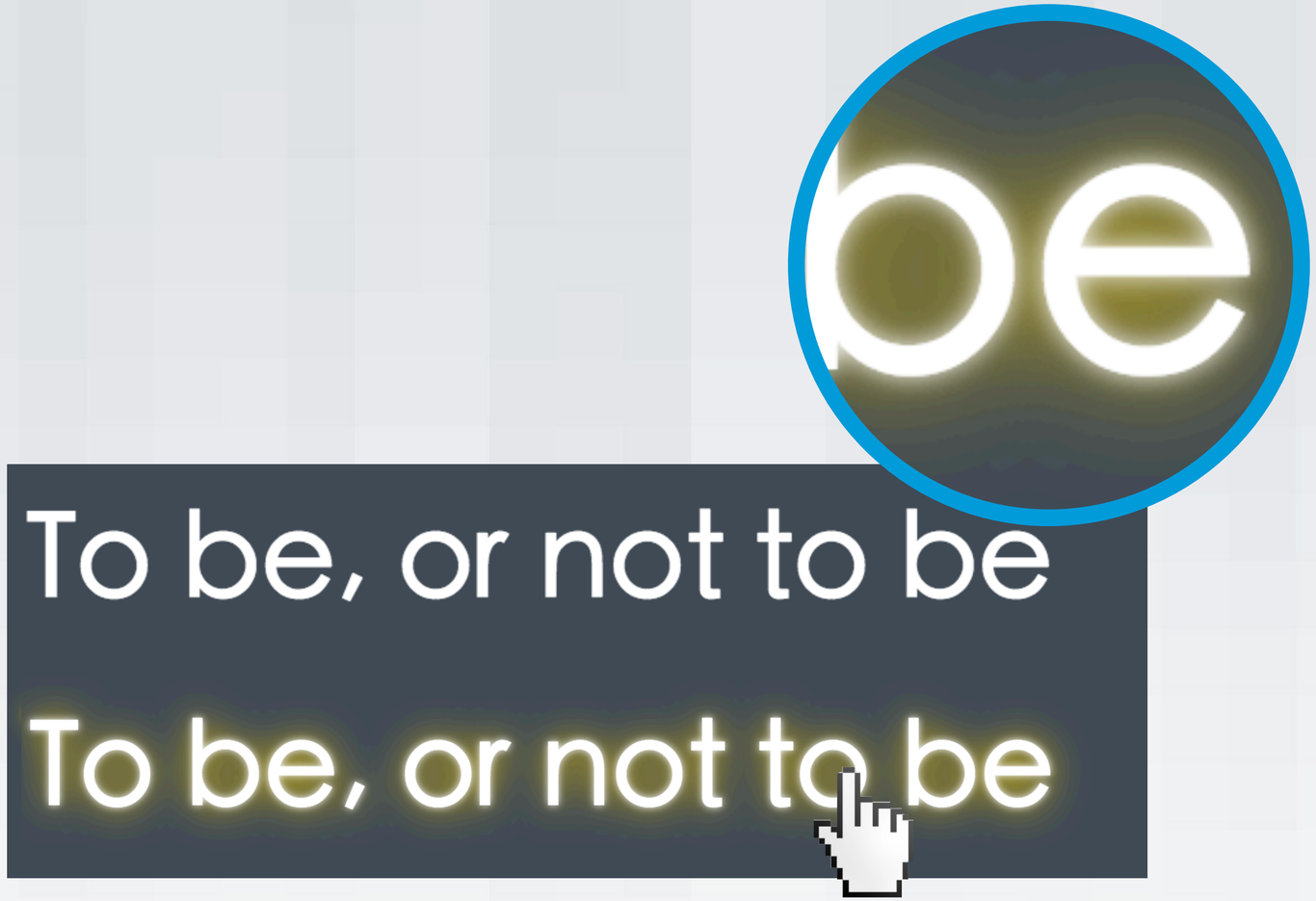
第1層　　第2層

```
text-shadow: 1px 1px 1px gold, 1px 1px 1px gold ..... ;
```

動手時間

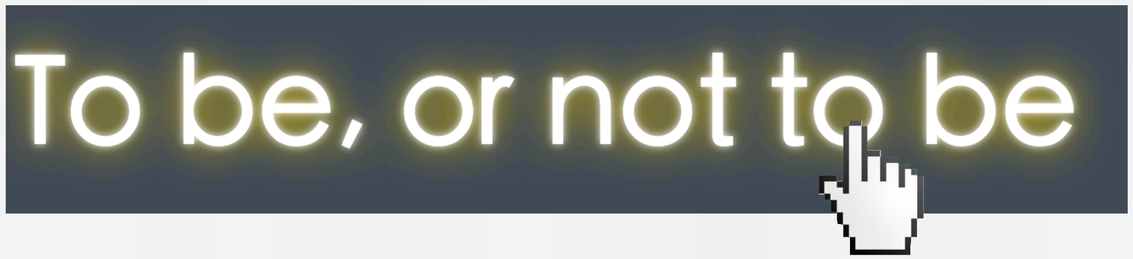
當滑鼠移過時點亮
用兩層陰影來做光暈效果的文字
[提示] 顏色為white,gold, 模糊值不同

前往練習



答對了嗎？

查看解答




```
.text.part1 h1 :hover{
  text-shadow: 0 0 .1em white, 0 0 .5em gold;
  cursor: pointer;
}
```

文字可以有不只一層陰影

每一層陰影都會被疊合在一起

當我們用了12道陰影: To be, or not to be

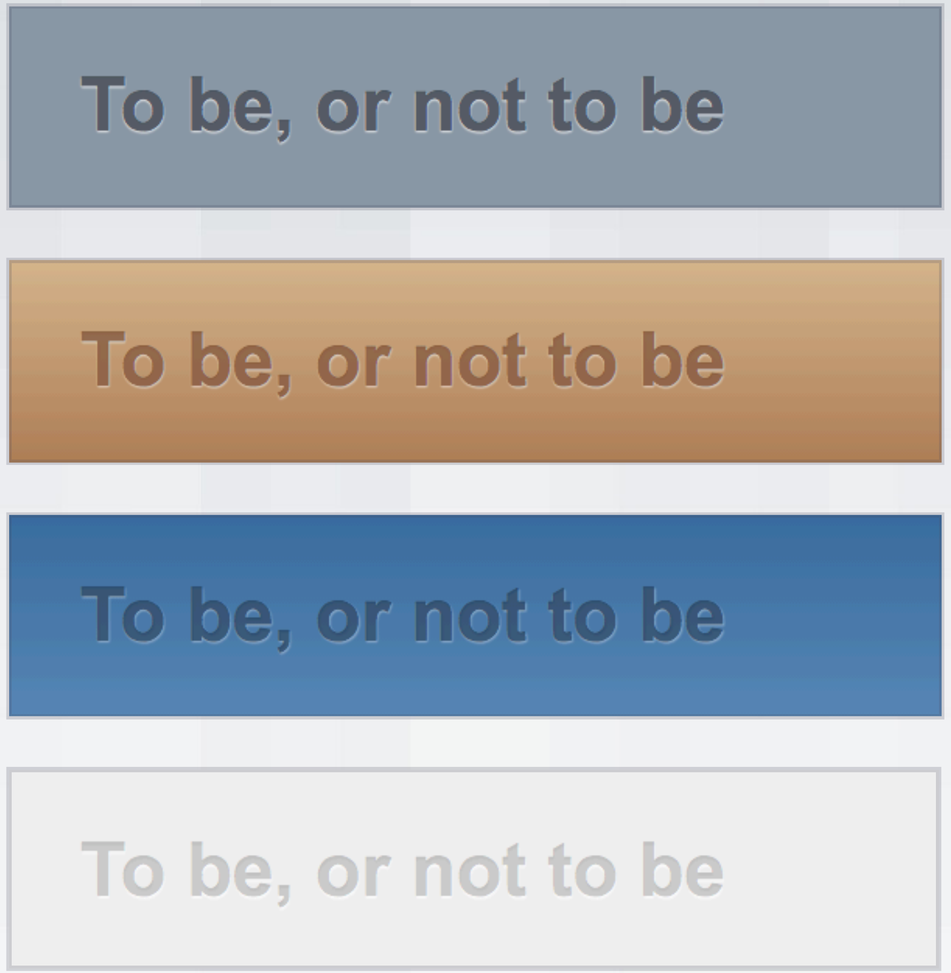
看看怎麼做

第1層　　第2層

```
text-shadow: 1px 1px 1px gold, 1px 1px 1px gold ..... ;
```

凹文字效果

想一想需要什麼條件，
讓我們覺得他是凹下？



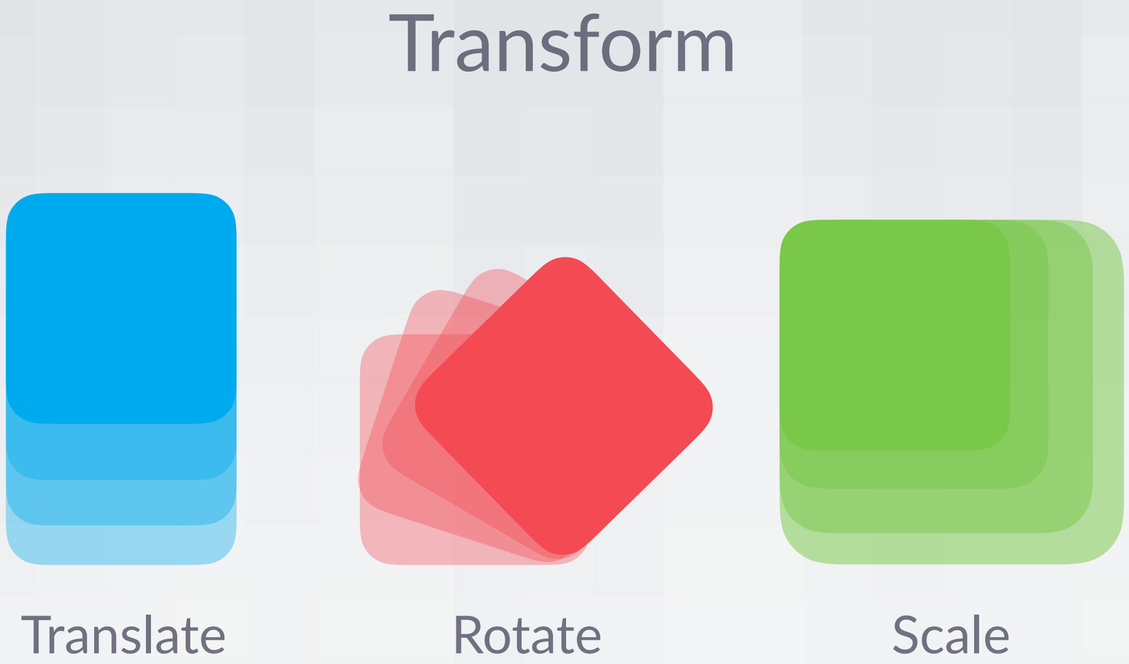
凹文字效果

1. 直覺假設光源在上方
2. 沒有凹進去的地方會被點亮

text-shadow: 0 1px 1px rgba(255, 255, 255, 0.8);

查看範例

CSS3 Animation
基礎動畫效果



CSS3 動畫兩種方式

1

Transition 轉場動畫

至多兩個狀態
(開始跟結束)

2

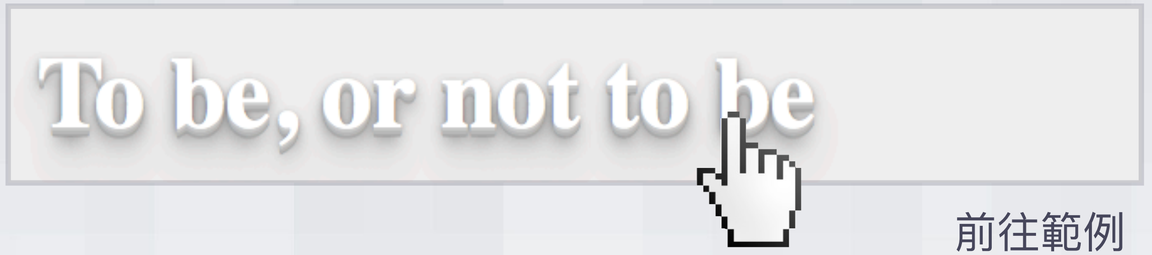
Animation 劇本動畫

多個時間點狀態

CSS3 Transition 轉場動畫

屬性	值(舉例)	說明
transition-duration	1s	時間長度(s/ ms)
transition-delay	1s	多久後開始(s / ms)
transition-property	color	針對的CSS屬性
transition-timing-function	ease(預設)	控制速度如何變化

transition: 時間長度 多久後開始 針對屬性 速度函式;



前往範例

範例1: 基本

```
.text.part1 h1: hover{
   transition: 1s;
}
```

範例2: 限定CSS屬性

```
.text.part1 h1: hover{
   color: snow;
   transition: 1s color;
}
```

範例3: 不同屬性分別設定

```
.text.part1 h1: hover{
   color: snow;
   transition: 1s color,  1s 1s text-shadow;
}
```

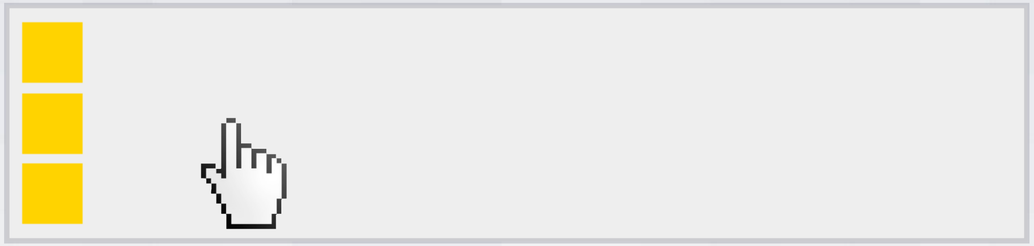
動手時間

製作一個
可互動的長條圖

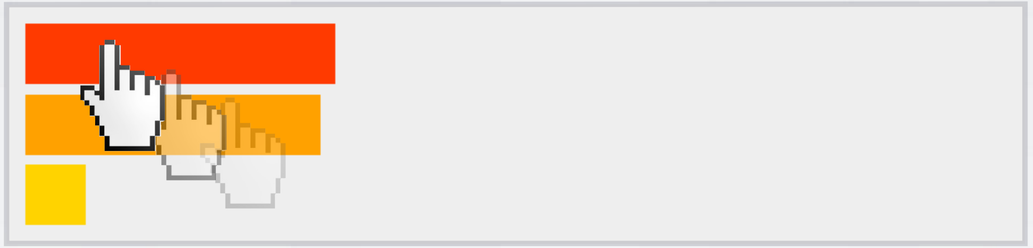
[目標] 三條變長後:

1. 長度不一
2. 顏色變成 red
3. 能平滑地復原

原型



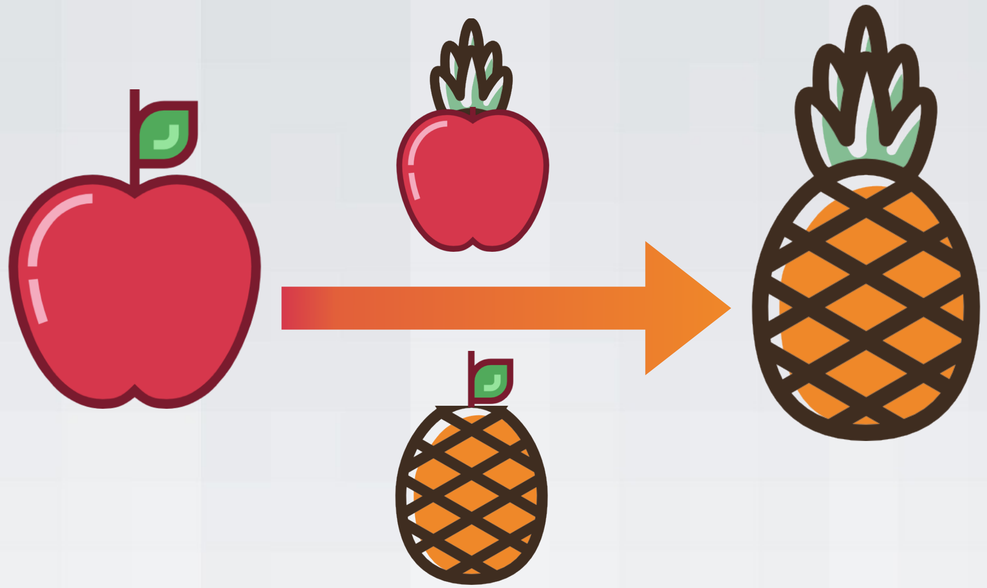
成品



[按我開始練習]

查看前頁解答

CSS3 Transition 轉場動畫



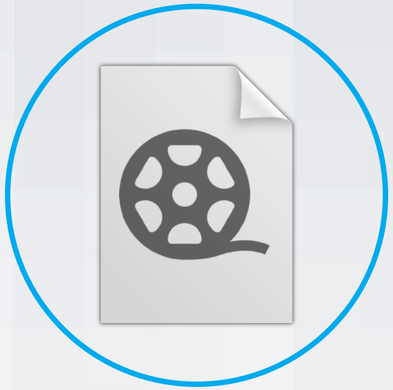
Transition 的限制：

- width: auto (不確定的值) 至 width: 100px (具體數值)
- display: none 至 display: block
- background: url(apple.jpg) 至 background: url(pineapple.jpg)

開始及結束
都是具體的值

CSS3 Animation劇本動畫3步驟

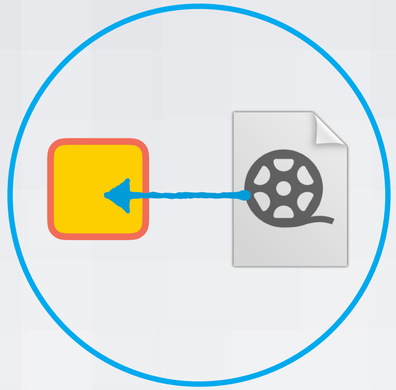
1



編輯劇本

依需求設定
某時間點的動作

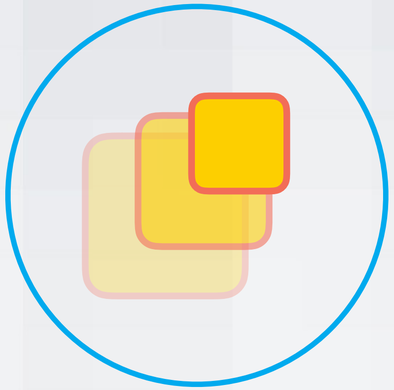
2



綁定元素

把劇本指定給某目標
並設定動畫參數

3

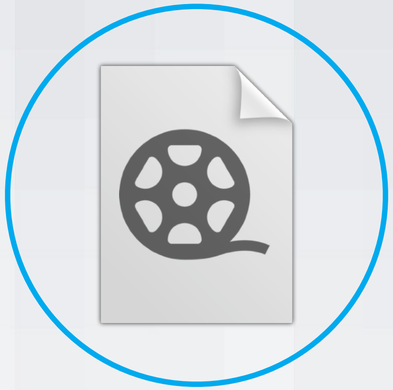


測試

預覽並修改劇本

CSS3 動畫3步驟

1



編輯劇本

依需求設定
某時間點的動作

```
@keyframes pulse {
    0% {
        transform: scale(0.9);
    }
    50% {
        transform: scale(1.5);
    }
    100% {
        transform: scale(0.9);
    }
}
```

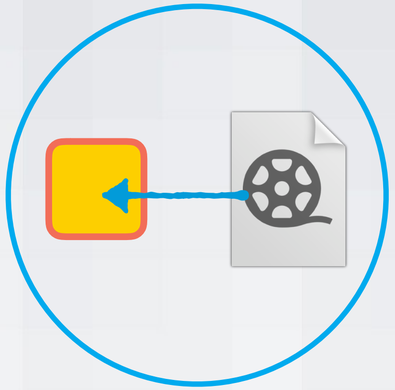
劇本名稱(完全自訂)

時間點

動作

CSS3 動畫3步驟

2



綁定元素

把劇本指定給某目標
並設定動畫參數

.box{

屬性	值(舉例)	說明
animation-name	pulse	指定劇本名稱
animation-duration	2s (預設: 0)	動畫長度(s / ms)
animation-iteration-count	infinite (預設:1)	這個動畫會重複n次
animation-delay	0s (預設)	多久後開始(可為負值)
animation-timing-function	ease (預設)	控制速度如何變化

}

animation: 時間長度 多久後開始 劇本名稱 速度函式 重複次數;

CSS3 動畫3步驟

快來試試

讓它開始跳動

.box{

屬性	值(舉例)	說明
animation-name	pulse	指定劇本名稱
animation-duration	2s (預設: 0)	動畫長度(s / ms)
animation-iteration-count	infinite (預設:1)	這個動畫會重複n次
animation-delay	0s (預設)	多久後開始(可為負值)
animation-timing-function	ease (預設)	控制速度如何變化

}

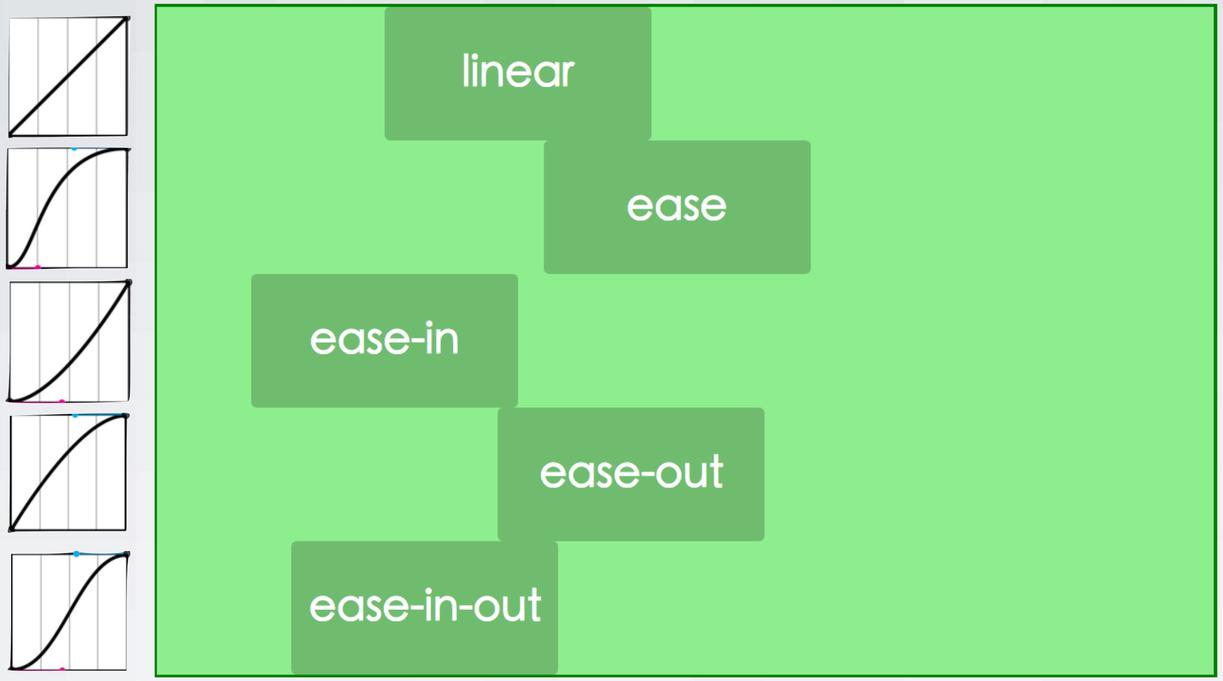
練習改成簡式

animation: 時間長度 多久後開始 劇本名稱 速度函式 重複次數;

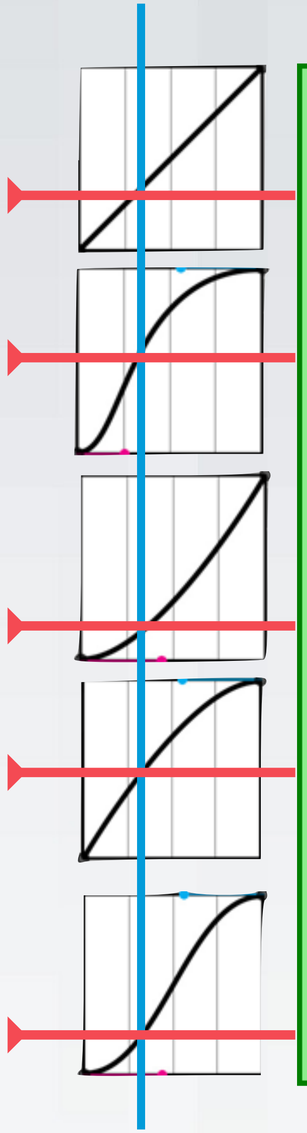
關於時間函式

timing-function

控制速度如何變化



前往範例



linear

ease

ease-in

ease-out

ease-in-out

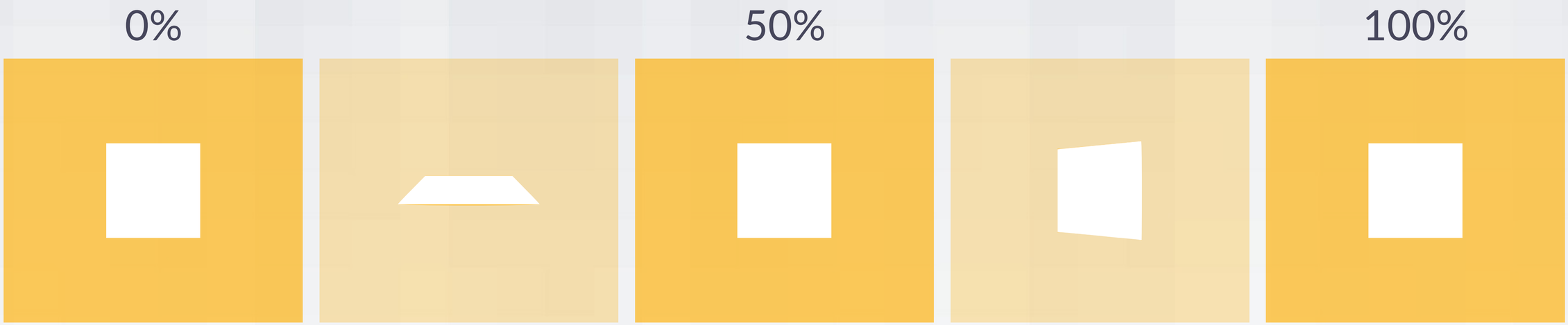
自訂速度函示

動手時間

rotate-plane

劇本編輯練習

前往練習




```
0%    transform: perspective(120px) rotateX(0deg) rotateY(0deg);
50%   transform: perspective(120px) rotateX(-180.1deg) rotateY(0deg);
100%  transform: perspective(120px) rotateX(-180deg) rotateY(-179.9deg);
```

動手時間

關於動畫週期與延遲

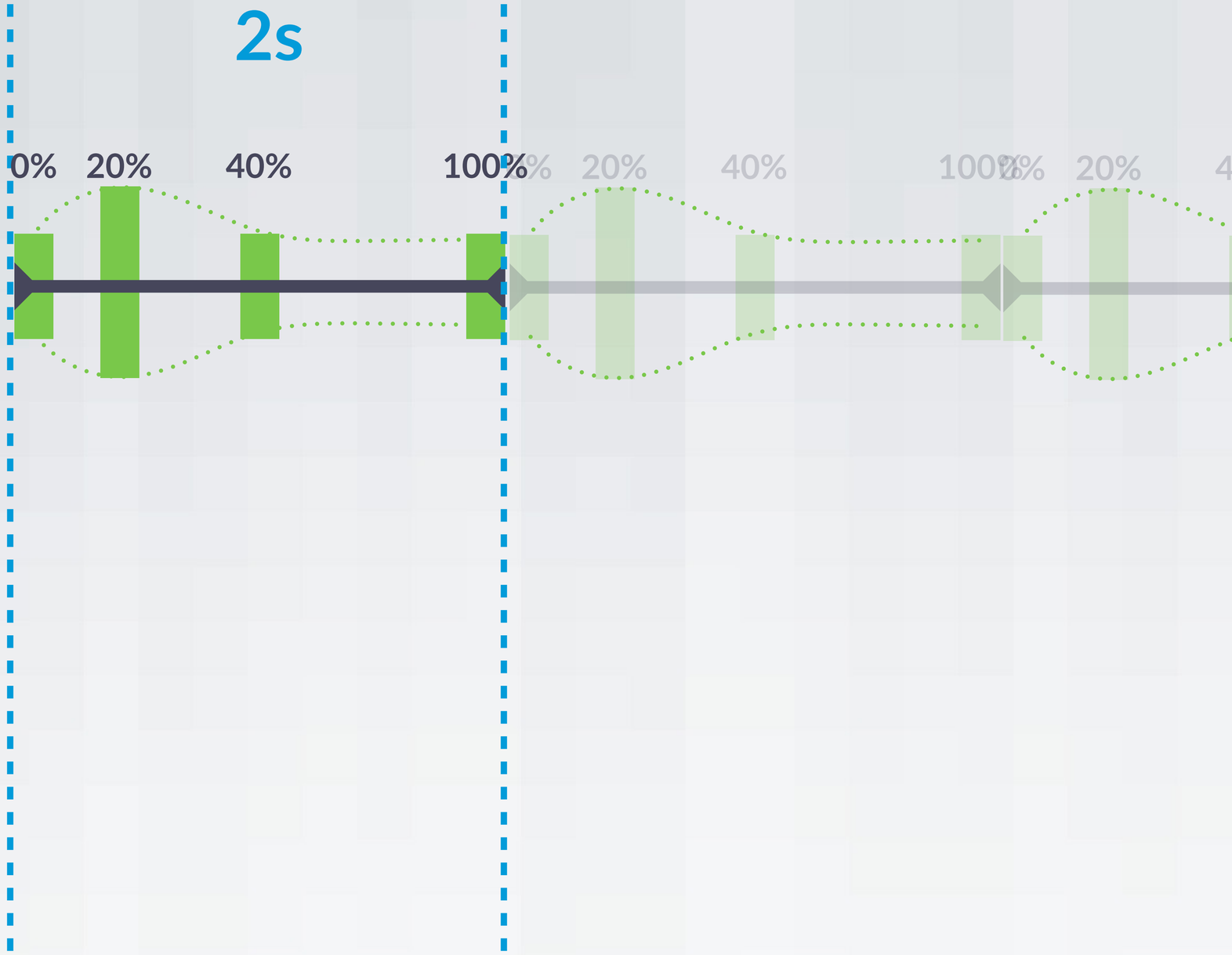
animation-delay

前往練習

先來看時間軸

關於動畫週期與延遲

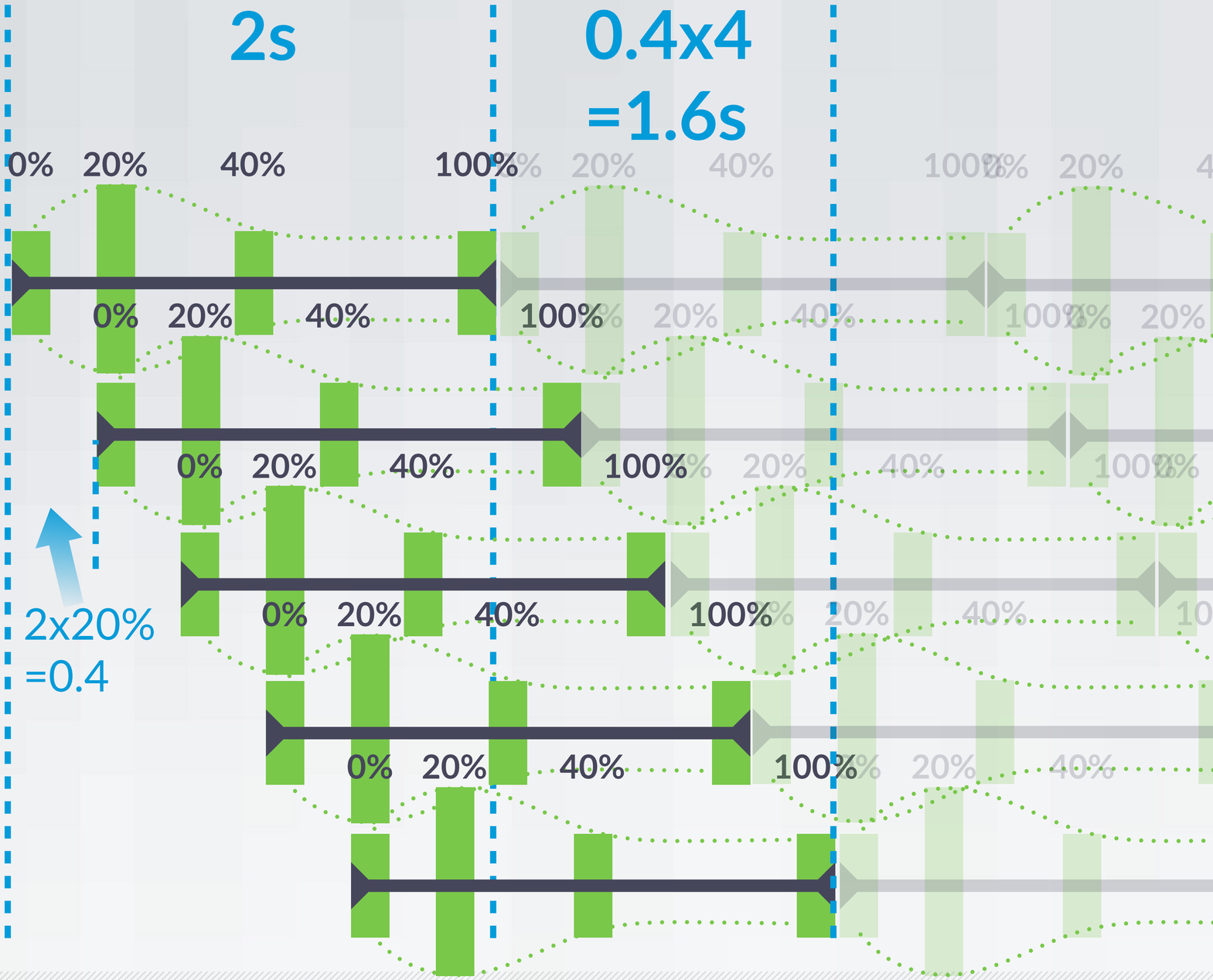
animation-delay



先來看時間軸

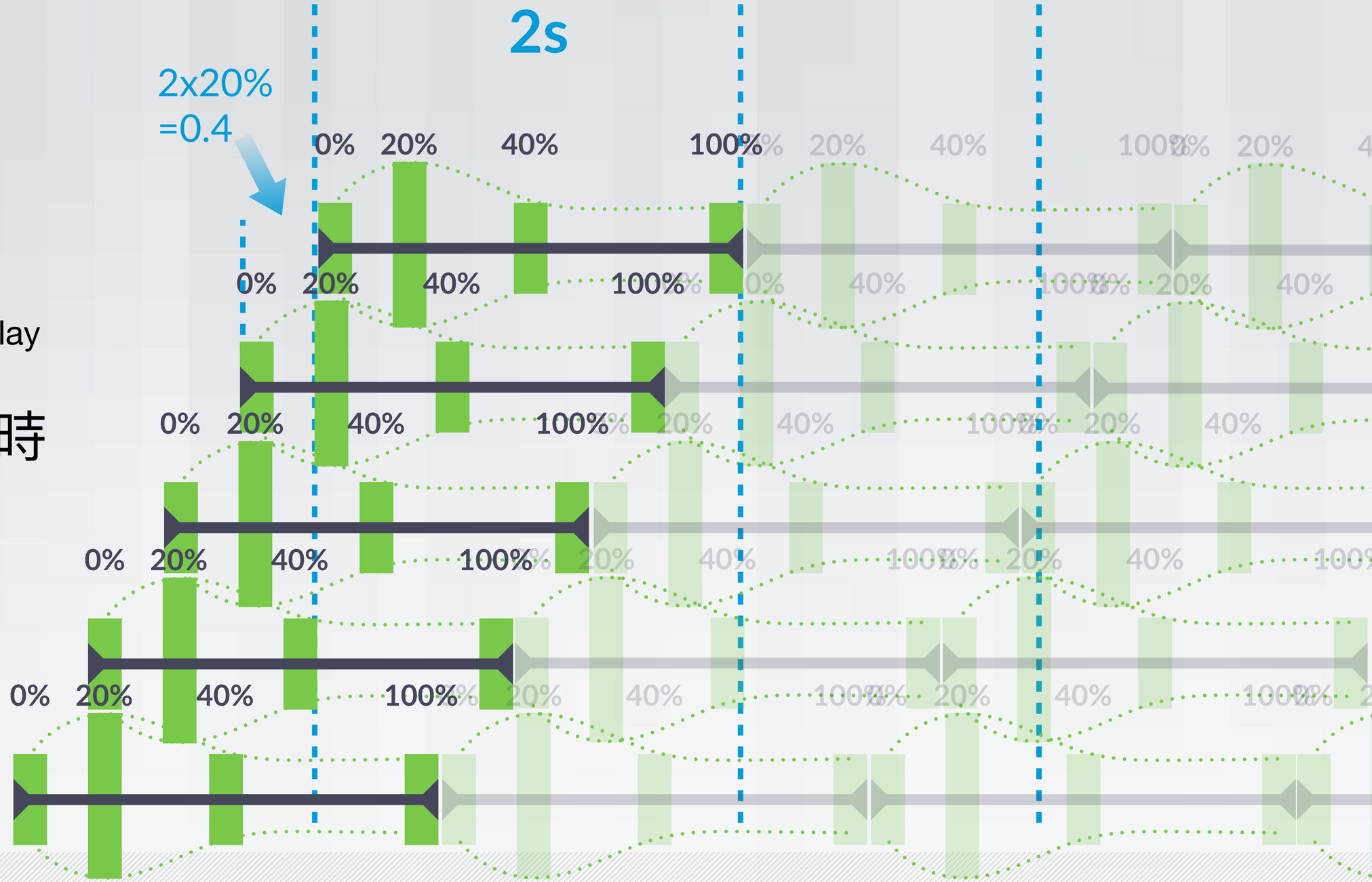
關於動畫週期與延遲

animation-delay

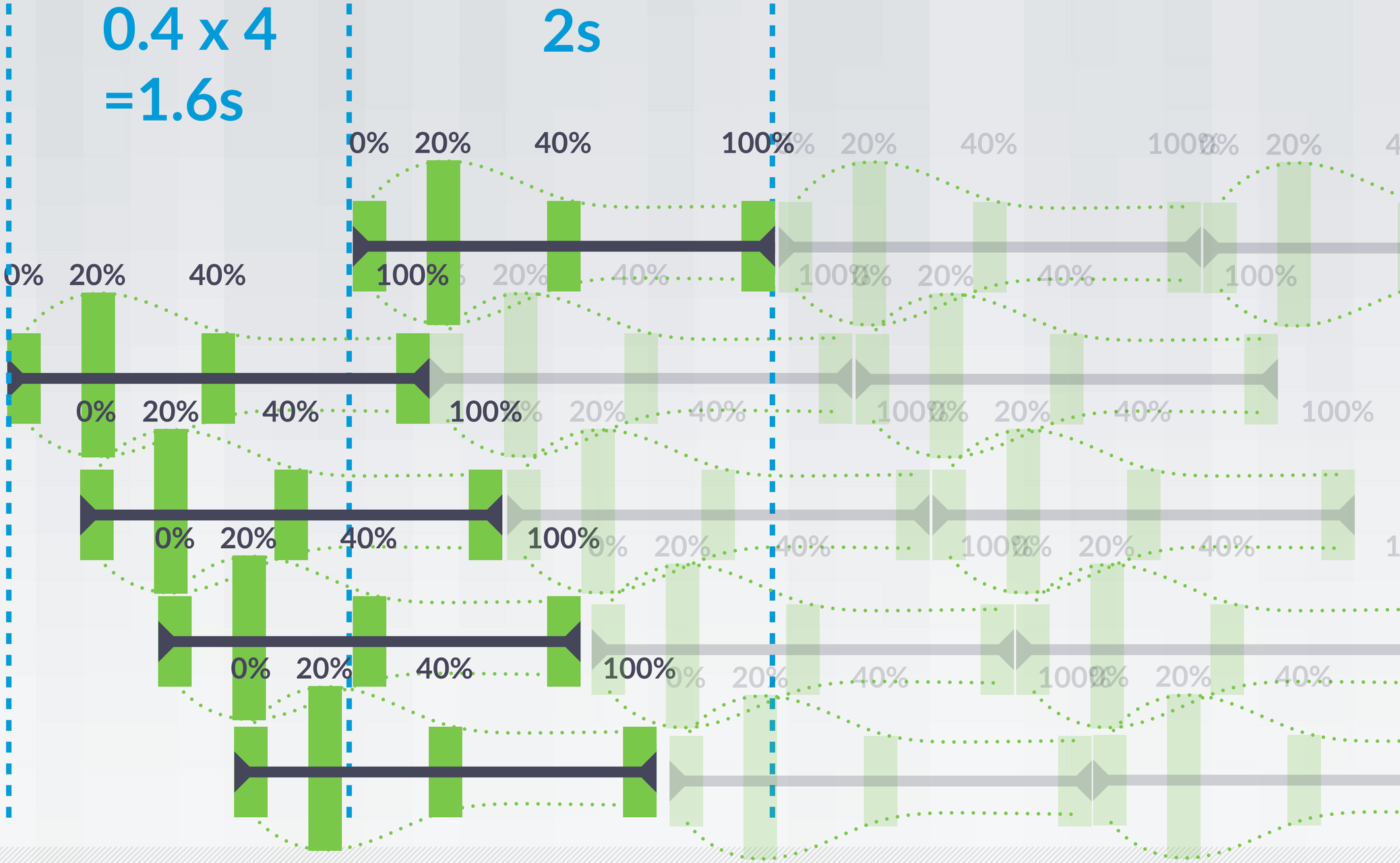


當 animation-delay

其值為負數時



這樣配置時
就是
週期與延遲
最佳組合



做對了嗎？

關於動畫週期與延遲

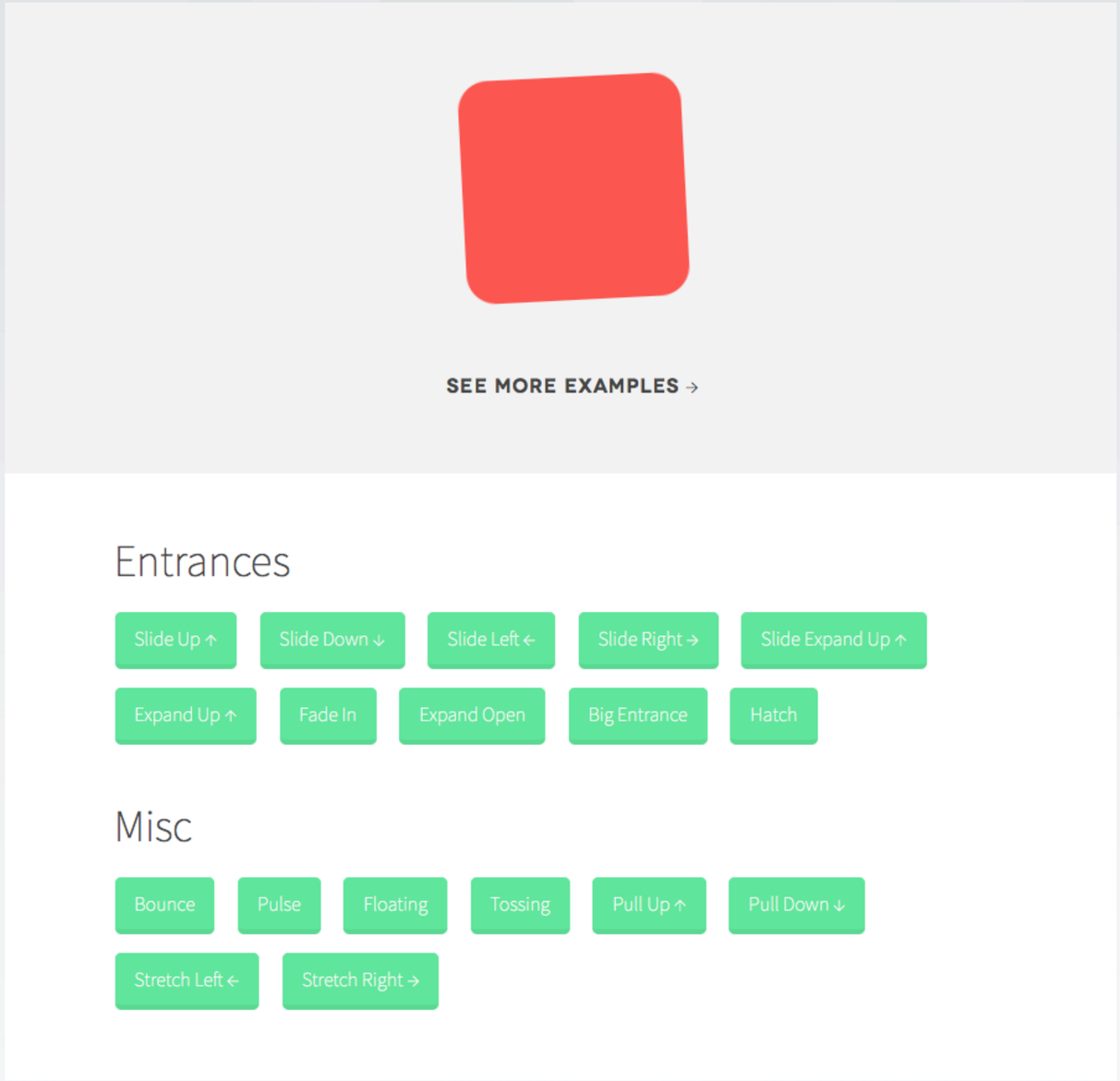
animation-delay

前往查看

誰說動畫劇本
一定要自己來

劇本1 分享:

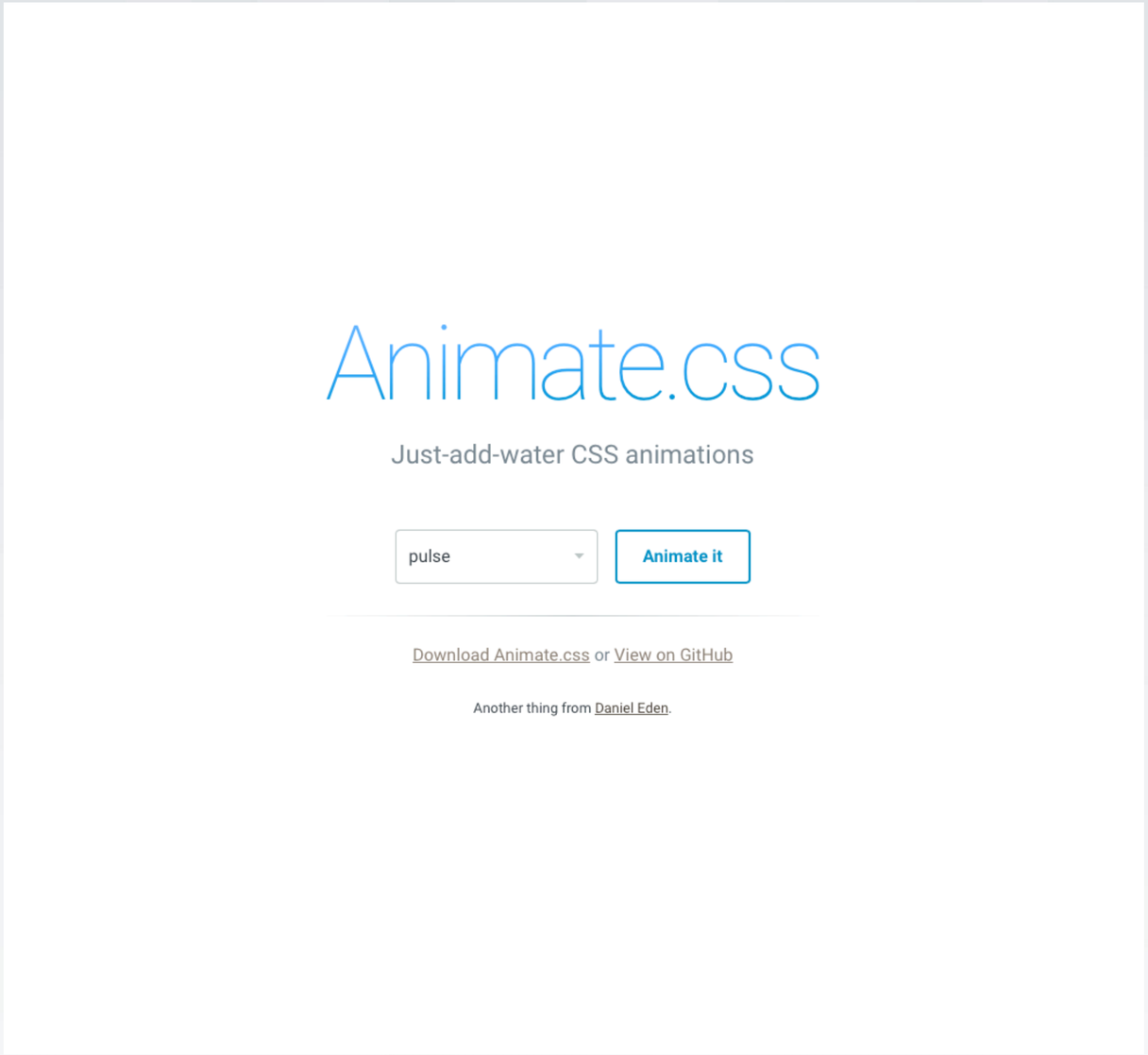
CSS Animation Cheat Sheet



誰說動畫劇本
一定要自己來

劇本2 分享:

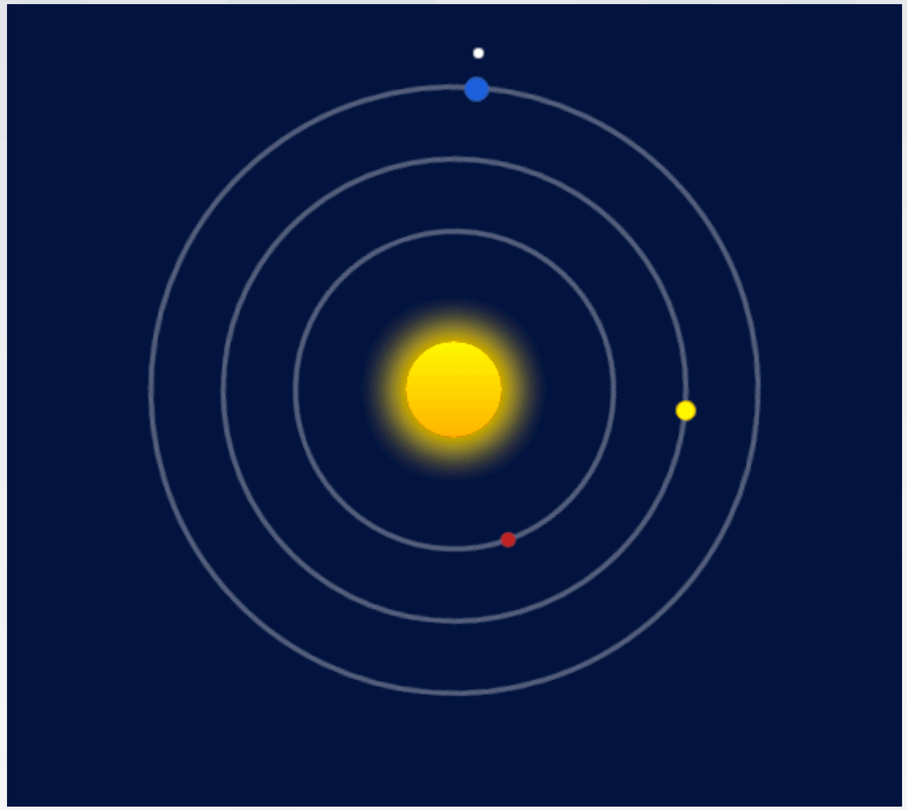
Animate.css



動手時間

太陽系軌道動畫

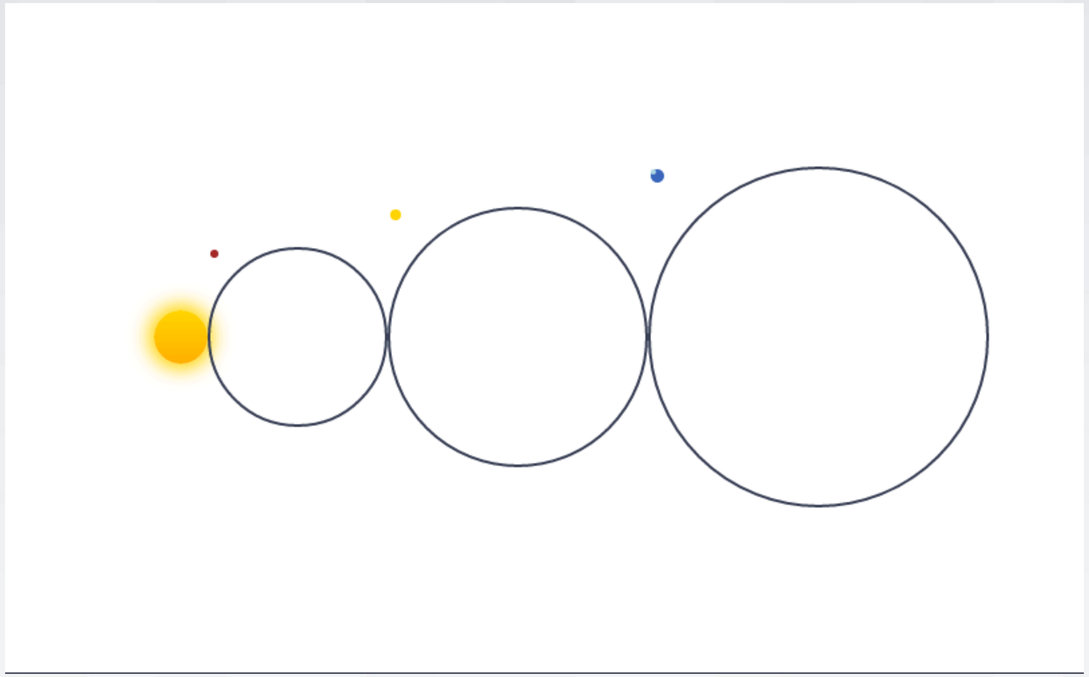
按我開始



動手時間

太陽系軌道動畫

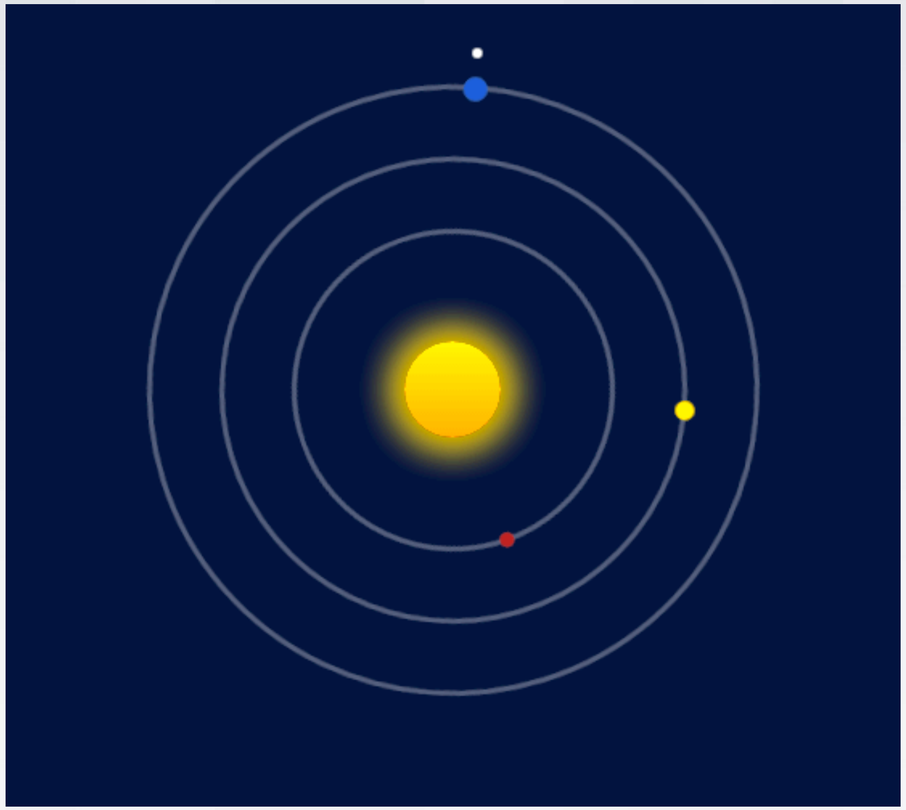
[提示]
1.用flexbox來做水平垂直置中
2.疊合所有物件(用position:absolute)
3.微調各行星位置
4.編輯劇本
5.綁定元素跟劇本



做對了嗎

太陽系軌道動畫

查看答案



CSS-選擇器

CSS樣式規則：

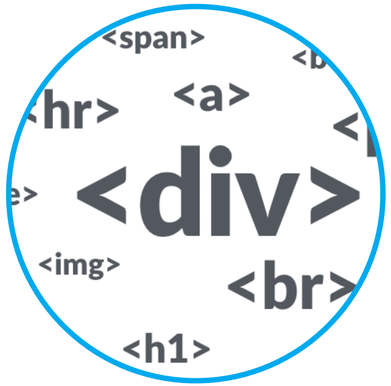
選擇器 { 屬性1: 值1, 宣告2, ... }

選擇器 - 分4類

CSS樣式規則：

選擇器 { 屬性1: 值1, 宣告2, ... }

1



標籤 選擇器

2

class=“選我”>

類別(Class) 選擇器

3

id=“選我”>

ID 選擇器

4

屬性=“內容”>

屬性 選擇器

選擇器只有四種不夠用？
其實你還可以這麼做

大家還記得: div, p { 屬性1: 值1, 宣告2, ... } ???

選擇器再進化 - 孩子模式
Child Selector

CSS樣式規則：

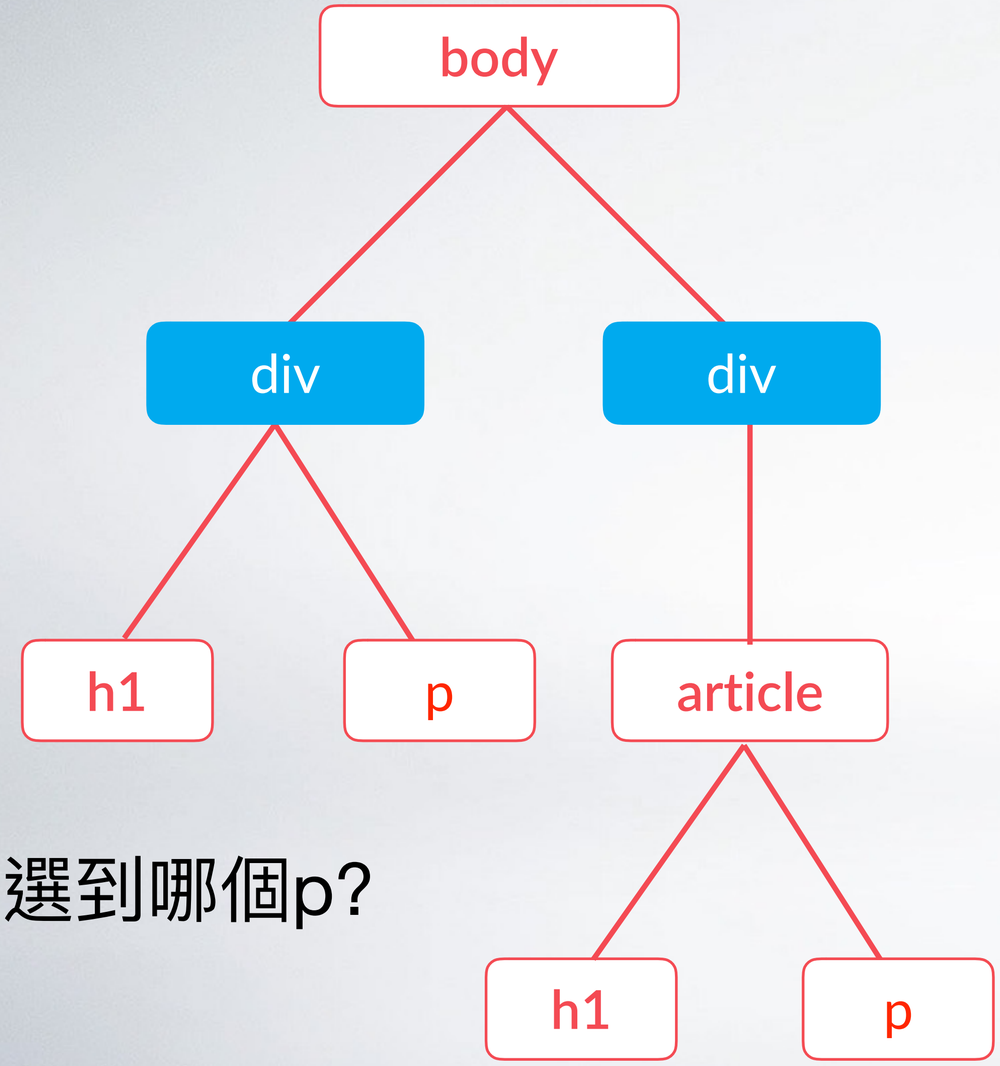
div>p { 屬性1: 值1, 宣告2, ... }

div的孩子中為p者

= 選擇所有p，他在div下一層

div, p { 屬性1: 值1, 宣告2, ... } -> 送作堆

孩子模式
Child Selector



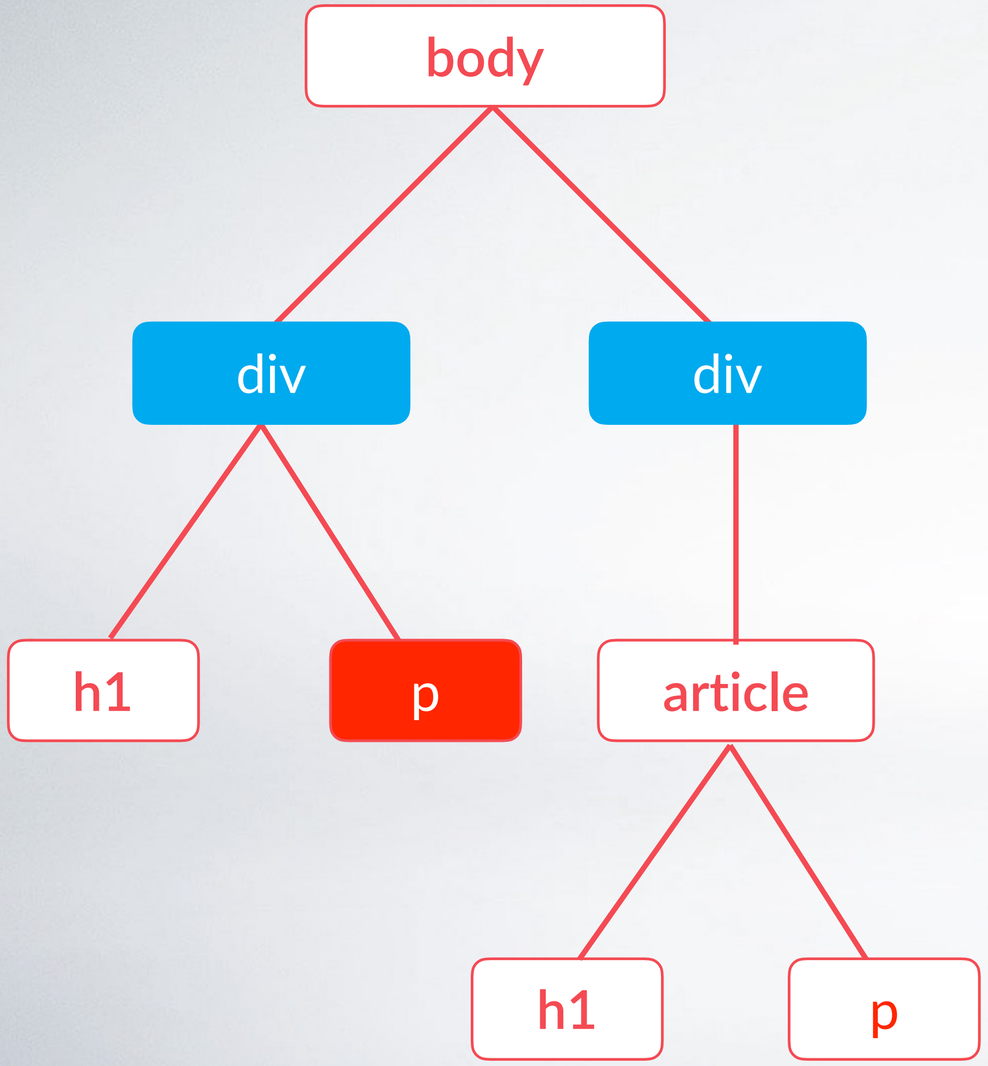
選到哪個p?

CSS樣式規則：

div>p { 屬性1: 值1, 宣告2, ... }

div的孩子中為p者

= 選擇所有p，他在div下一層



孩子模式
Child Selector

CSS樣式規則：

div>p { 屬性1: 值1, 宣告2, ... }

div的孩子中為p者

= 選擇所有p，他在div下一層

div, p { 屬性1: 值1, 宣告2, ... } -> 送作堆

div>p { 屬性1: 值1, 宣告2, ... } ->孩子模式

選擇器再進化 - 子孫模式

Descendant Selector

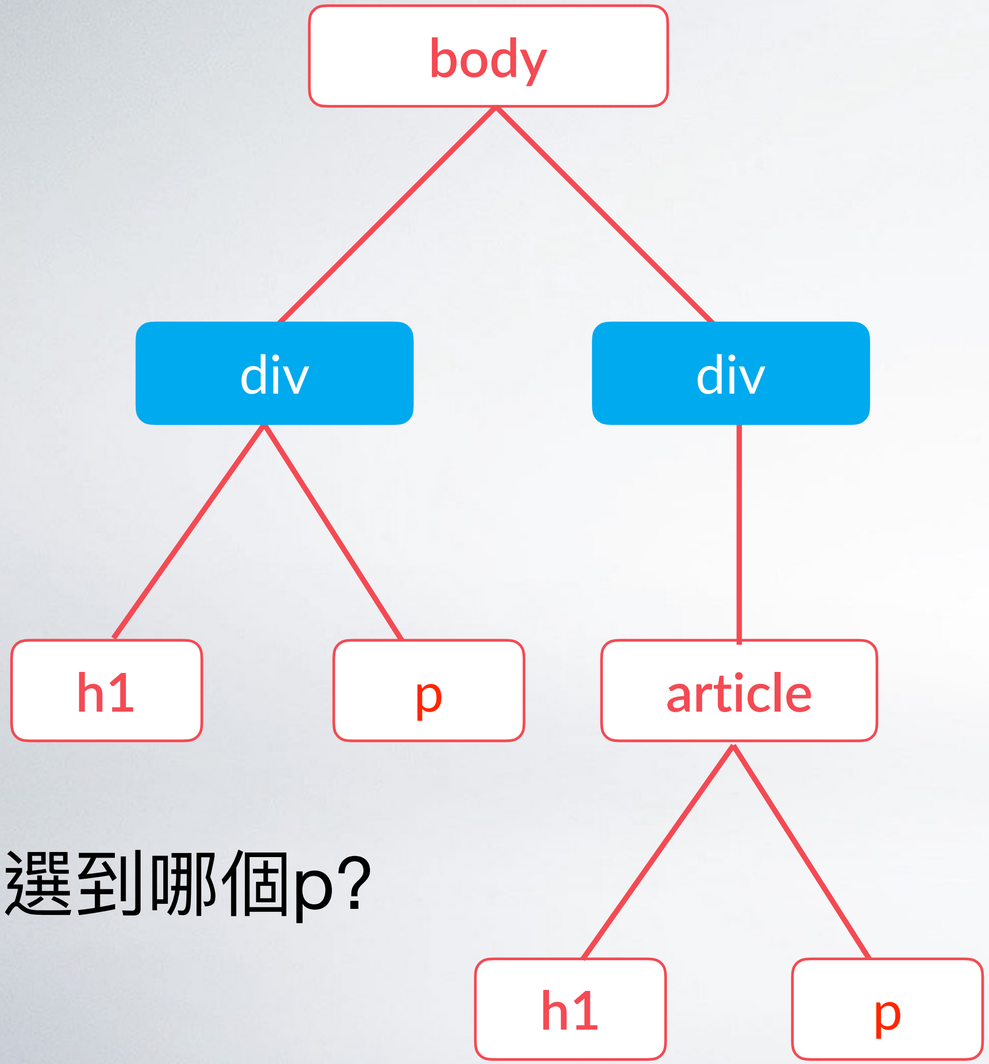
CSS樣式規則：

div p { 屬性1: 值1, 宣告2, ... }

div的子孫中為p者

= 選擇所有p，他在div底下

子孫模式
Descendant Selector



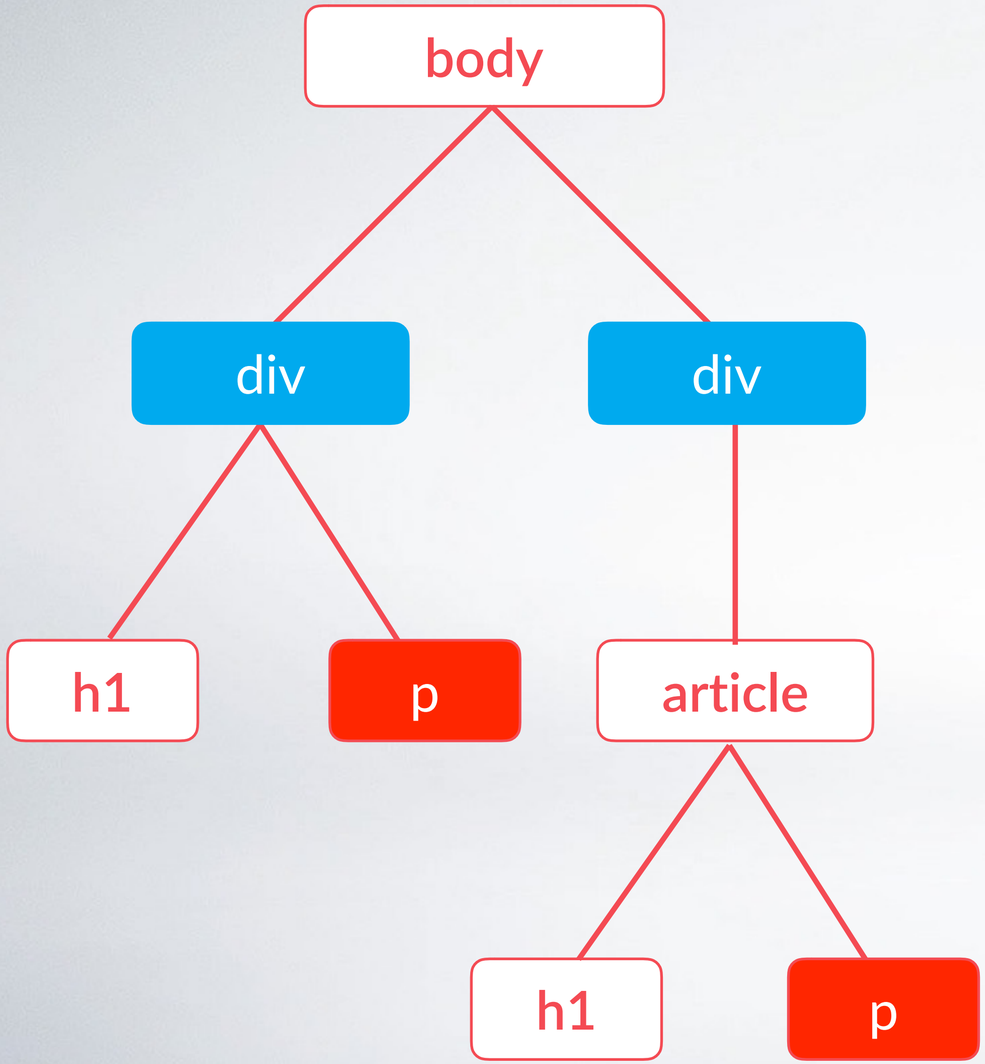
選到哪個p?

CSS樣式規則：

div p { 屬性1: 值1, 宣告2, ... }

div的子孫中為p者

= 選擇所有p，他在div底下



子孫模式
Descendant Selector

CSS樣式規則：

div p { 屬性1: 值1, 宣告2, ... }

div的子孫中為p者

= 選擇所有p，他在div底下

div, p { 屬性1: 值1, 宣告2, ... } -> 送作堆

div>p { 屬性1: 值1, 宣告2, ... } ->孩子模式

div p { 屬性1: 值1, 宣告2, ... } ->子孫模式

選擇器再進化 - 跟屁蟲模式

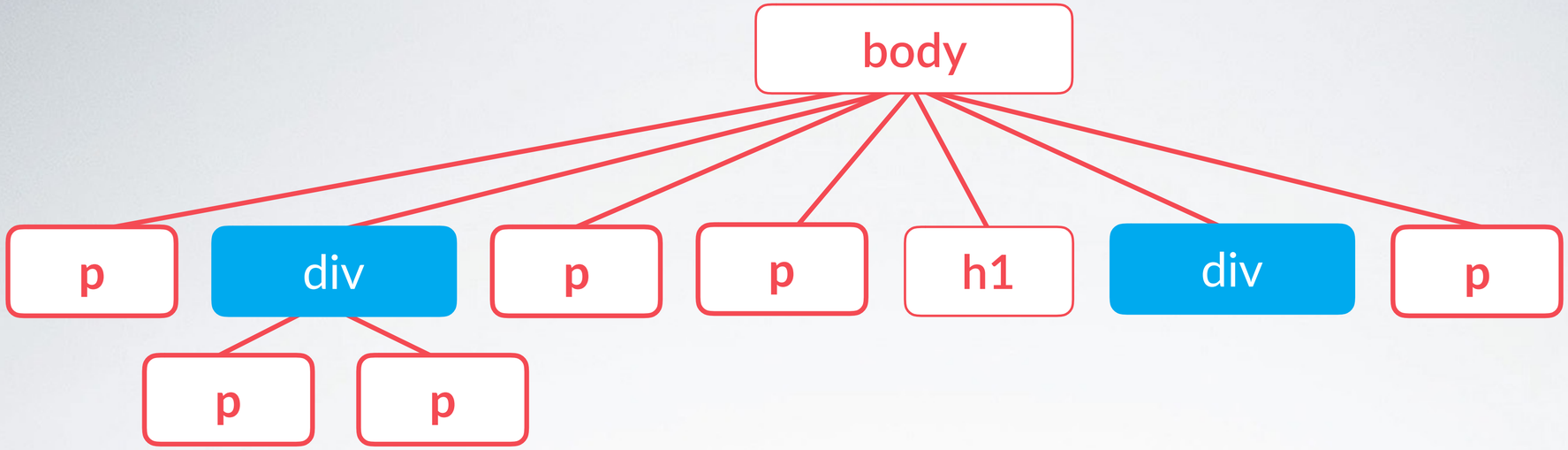
Adjacent Silbing Selector

CSS樣式規則：

div+p { 屬性1: 值1, 宣告2, ... }

緊接在div後為p者

= 選擇所有p，他前一個是div



跟屁蟲模式

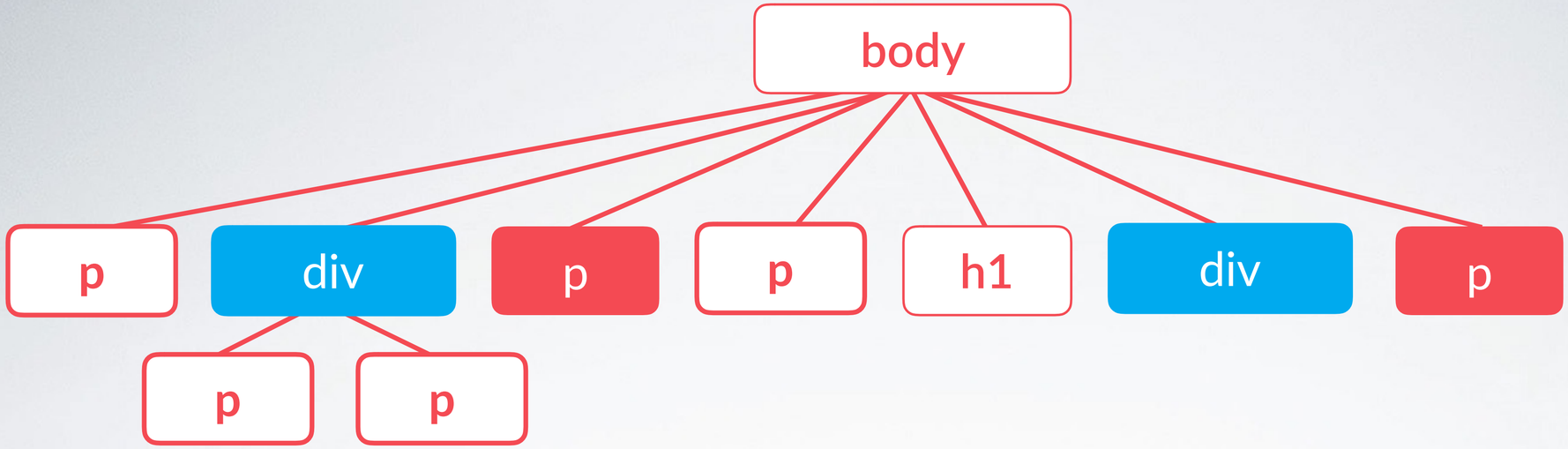
Adjacent Silbing Selector

CSS樣式規則：

div+p { 屬性1: 值1, 宣告2, ... }

緊接在div後為p者

= 選擇所有p，他前一個是div



跟屁蟲模式

Adjacent Silbing Selector

CSS樣式規則：

div+p { 屬性1: 值1, 宣告2, ... }

緊接在div後為p者

= 選擇所有p，他前一個是div

div, p { 屬性1: 值1, 宣告2, ... } -> 送作堆

div>p { 屬性1: 值1, 宣告2, ... } ->孩子模式

div p { 屬性1: 值1, 宣告2, ... } ->子孫模式

div+p { 屬性1: 值1, 宣告2, ... } ->跟屁蟲模式

選擇器再進化 - 蟲蟲模式

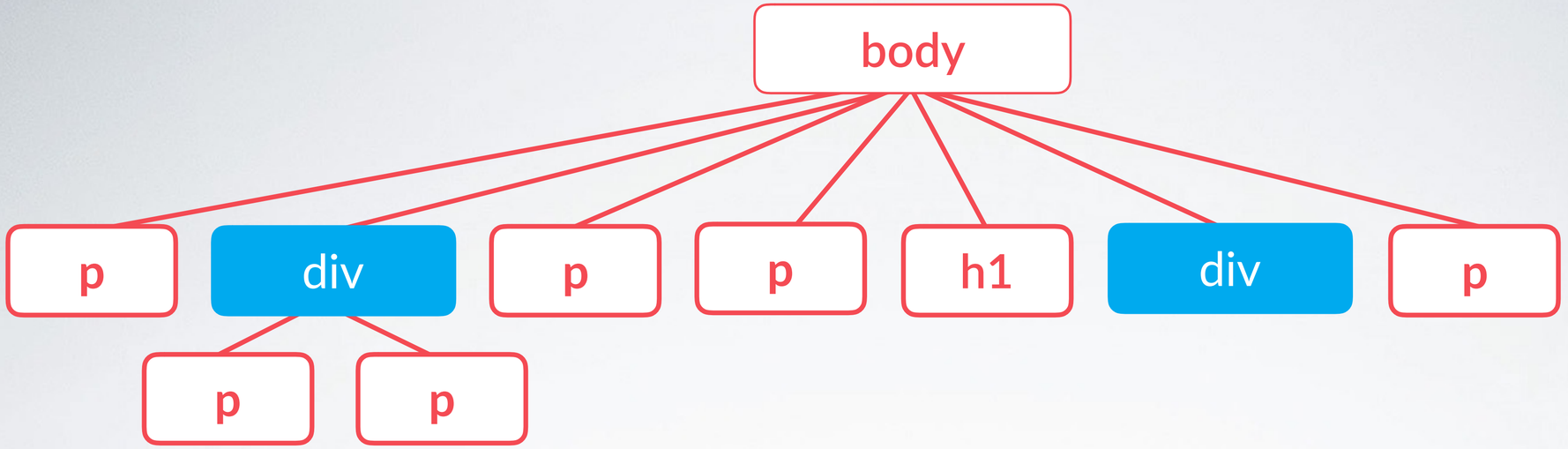
Sibling Selector

CSS樣式規則：

div~p { 屬性1: 值1, 宣告2, ... }

在div後為p者

= 選擇所有p，他前面有div



蟲蟲模式

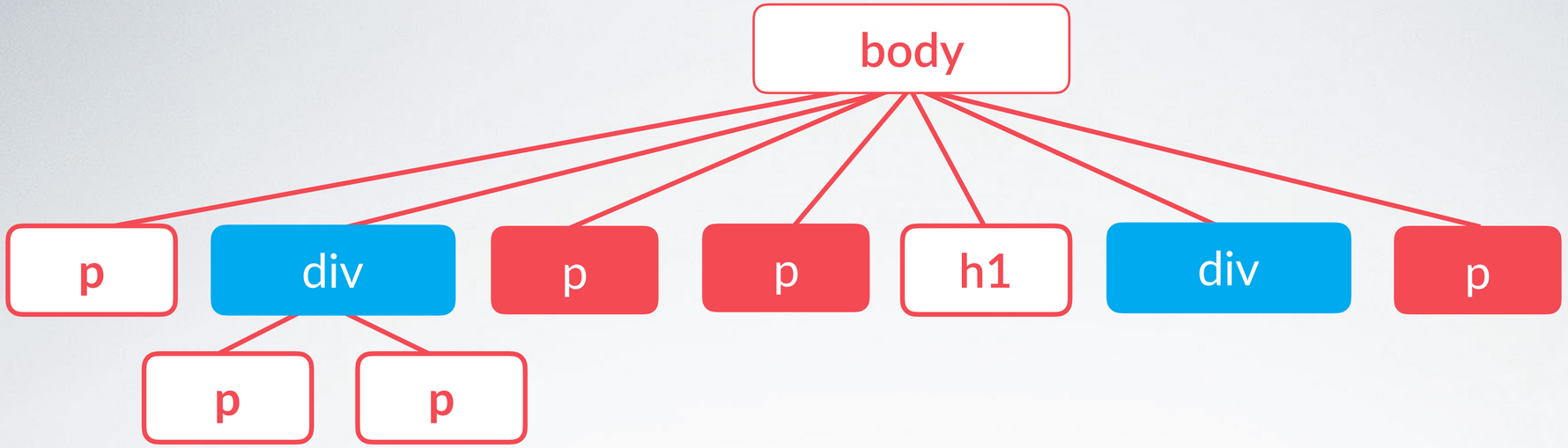
Sibling Selector

CSS樣式規則：

div~p { 屬性1: 值1, 宣告2, ... }

在div後為p者

= 選擇所有p，他前面有div



蟲蟲模式

Sibling Selector

CSS樣式規則：

div~p { 屬性1: 值1, 宣告2, ... }

在div後為p者

= 選擇所有p，他前面有div

div, p { 屬性1: 值1, 宣告2, ... } -> 送作堆

div>p { 屬性1: 值1, 宣告2, ... } ->孩子模式

div p { 屬性1: 值1, 宣告2, ... } ->子孫模式

div+p { 屬性1: 值1, 宣告2, ... } ->跟屁蟲模式

div~p { 屬性1: 值1, 宣告2, ... } ->蟲蟲模式

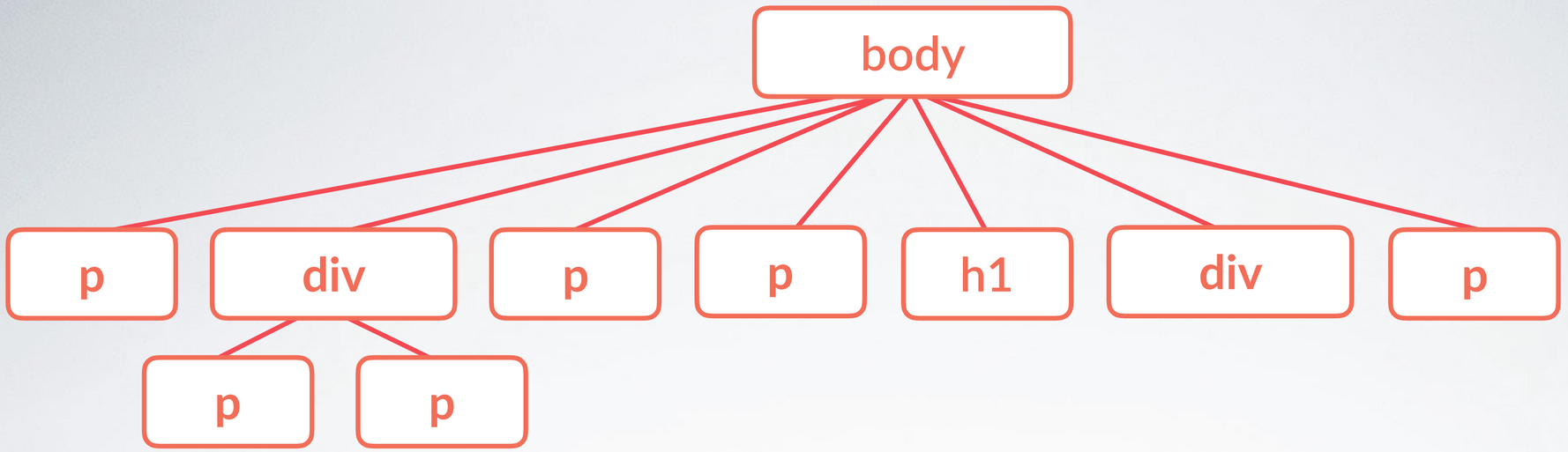
選擇器再進化 - 全選模式 Universal Selector

CSS樣式規則：

*{ 屬性1: 值1, 宣告2, ... }

*代表：任意元素

選擇 任意元素



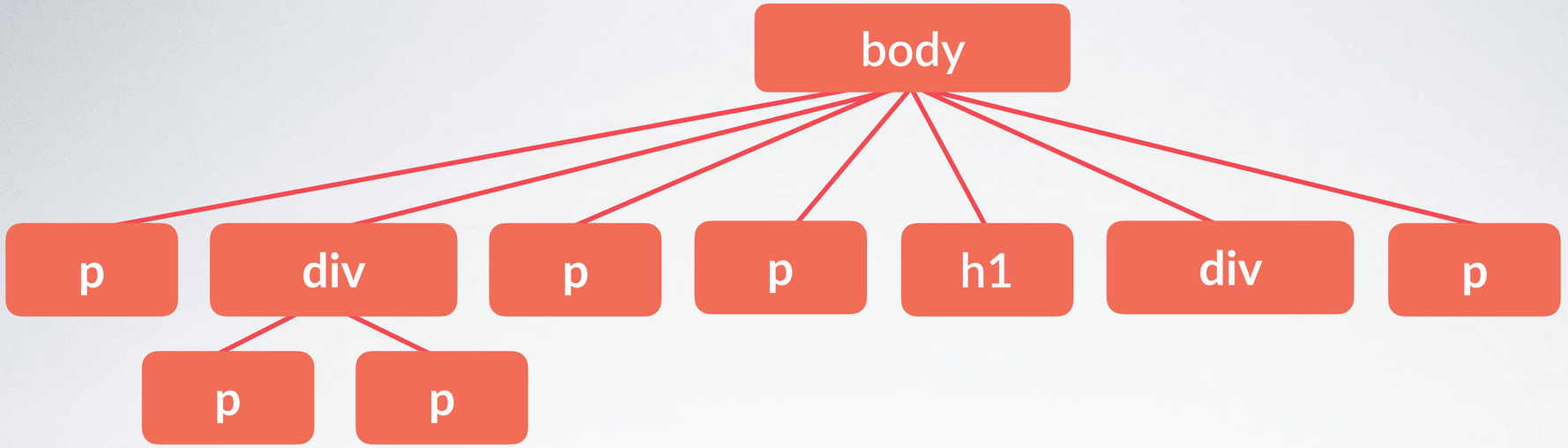
全選模式
Universal Selector

CSS樣式規則：

*{ 屬性1: 值1, 宣告2, ... }

*代表任意元素

選擇 任意元素



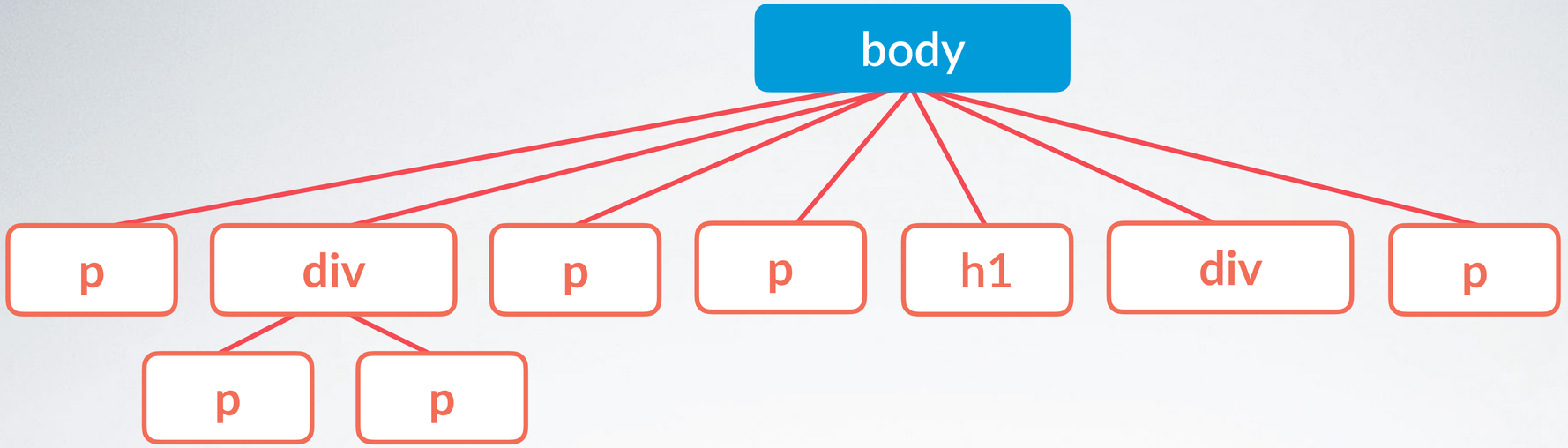
全選模式
Universal Selector

CSS樣式規則：

***{ 屬性1: 值1, 宣告2, … }**

***代表任意元素**

選擇 任意元素



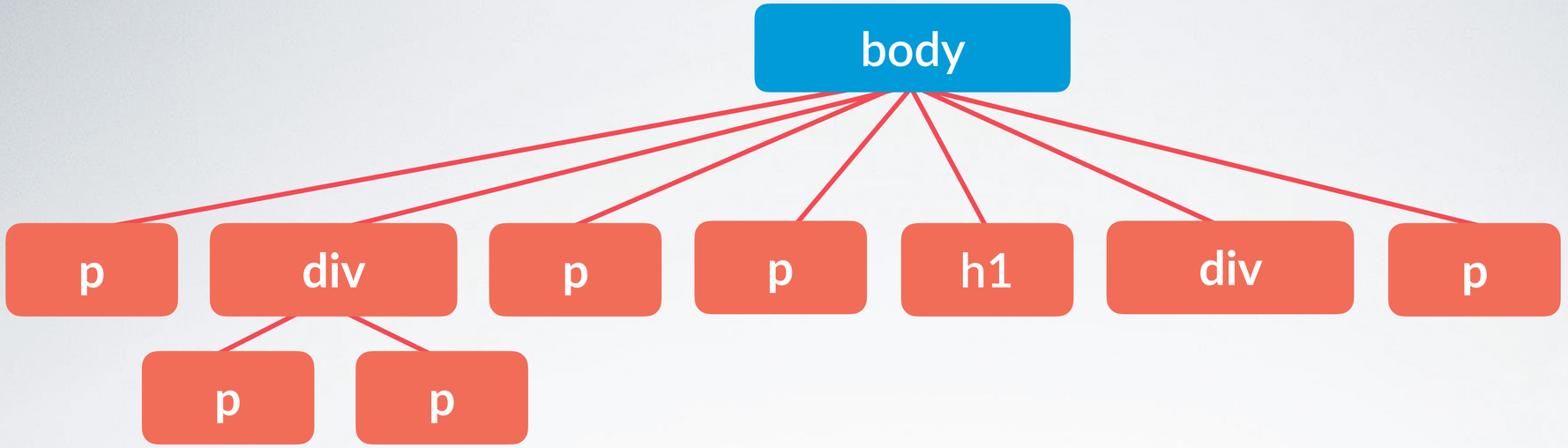
全選模式
Universal Selector

CSS樣式規則：

body *{ 屬性1: 值1, 宣告2, ... }

*代表任意元素

選擇 body 的子孫們



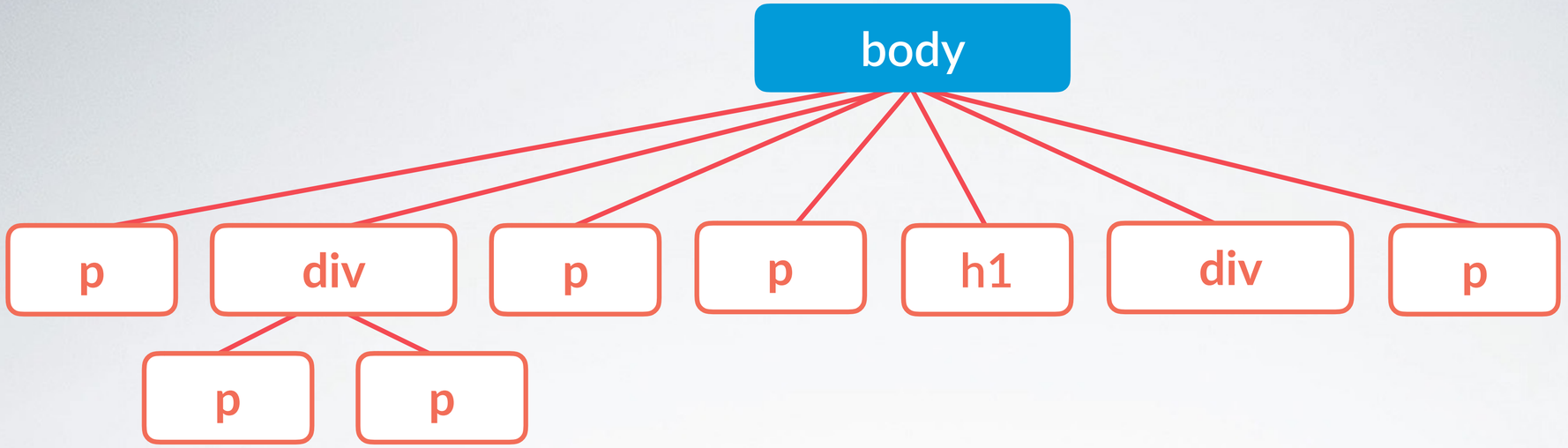
全選模式
Universal Selector

CSS樣式規則：

body *{ 屬性1: 值1, 宣告2, ... }

*代表任意元素

選擇 body 的子孫們



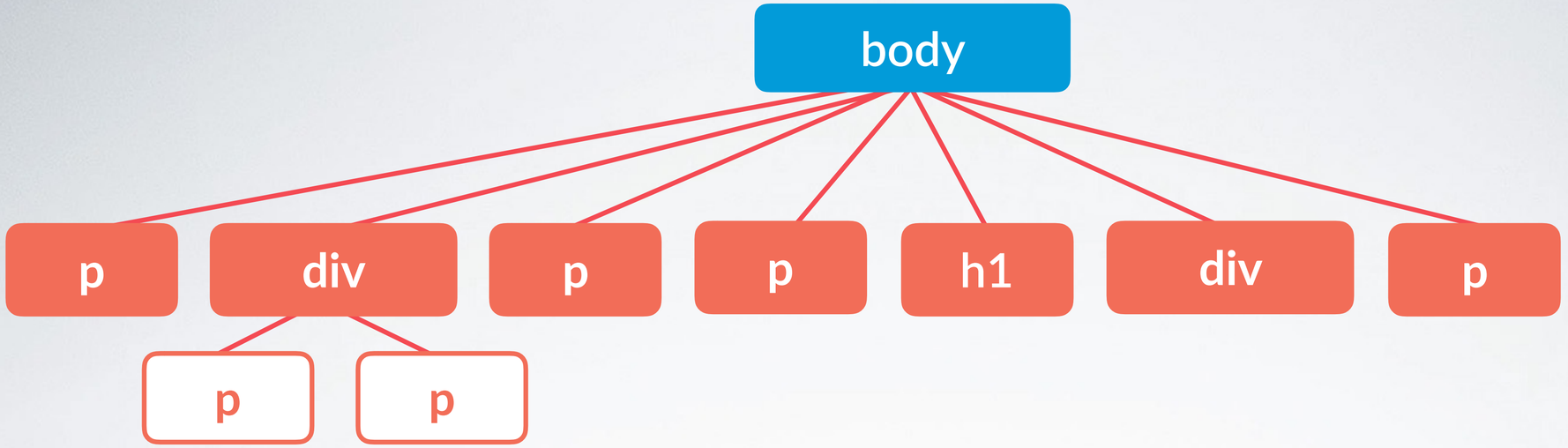
全選模式
Universal Selector

CSS樣式規則：

body>*{ 屬性1: 值1, 宣告2, ... }

*代表任意元素

選擇 body 的孩子們



全選模式
Universal Selector

CSS樣式規則：

body>*{ 屬性1: 值1, 宣告2, ... }

*代表任意元素

選擇 body 的孩子們

全選模式
Universal Selector

CSS樣式規則：

.root *{ 屬性1: 值1, 宣告2, ... }

*代表任意元素

選擇 .root 的子孫們

按我前往

HTML

```
<div class="root">
  Parent div has
  <div>Child div 1</div> and
  <div>Child div 2, which has some
    <div>Young kids</div>
    <p>Another <span>kid</span>
</p>
    <span>Third kid</span>
  </div>
</div>
```

CSS

```
div{ padding: 3px; }
.root *{
  border: 1px solid green;
}
```

Parent div has

Child div 1

and

Child div 2, which has some

Young kids

Another kid

Third kid

依狀態來套用「假的」Class

Ex. 滑鼠移過去，狀態改變

虛擬類別選擇器

Pseudo-classes Selector

CSS樣式規則：

元素: 虛擬類別 { 屬性1: 值1, 宣告2, ... }

Ex. :hover, :active, :focus, :checked etc.

依狀態來套用「假的」Class

Ex. 滑鼠移過去，狀態改變

虛擬類別選擇器

Pseudo-classes Selector

1. **N孩後，選 __** :nth-child(3)
2. **歸類後，選 __** :nth-of-type(3)
3. **選 __ 裡，空的** :empty
4. **選 __ 裡，非的** :not(p)

N孩後，選__

老大 :first-child

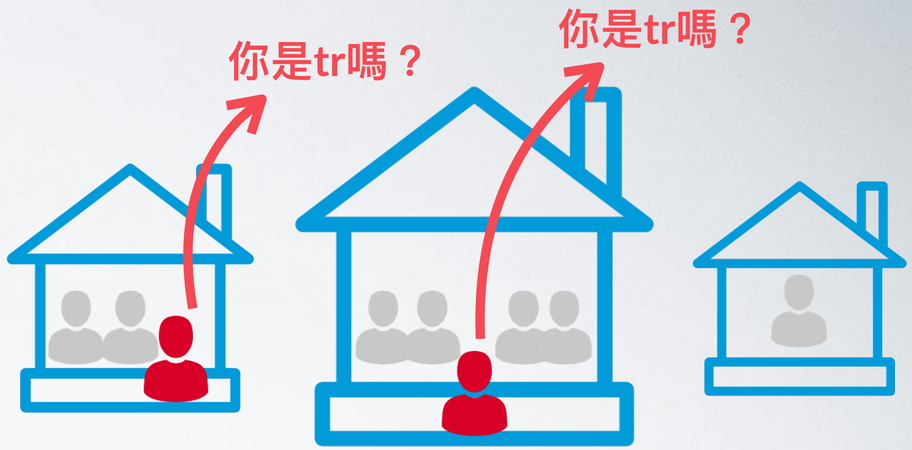
老么 :last-child

獨子 :only-child

老N :nth-child(數字)

倒著數老N :nth-last-child(數字)

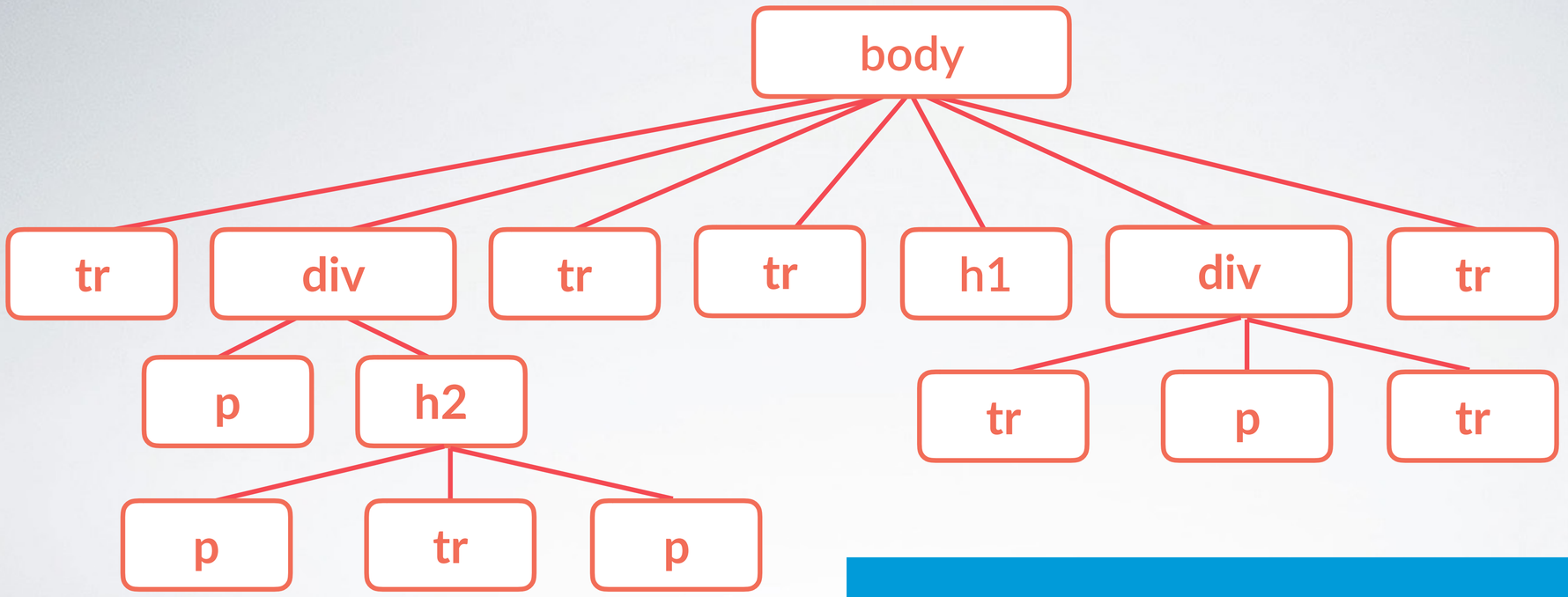
第3個出列！！



CSS樣式規則：

tr: nth-child(3) { 屬性1: 值1, 宣告2, ... }

選擇每個標籤裡的第3個孩子，而且是tr

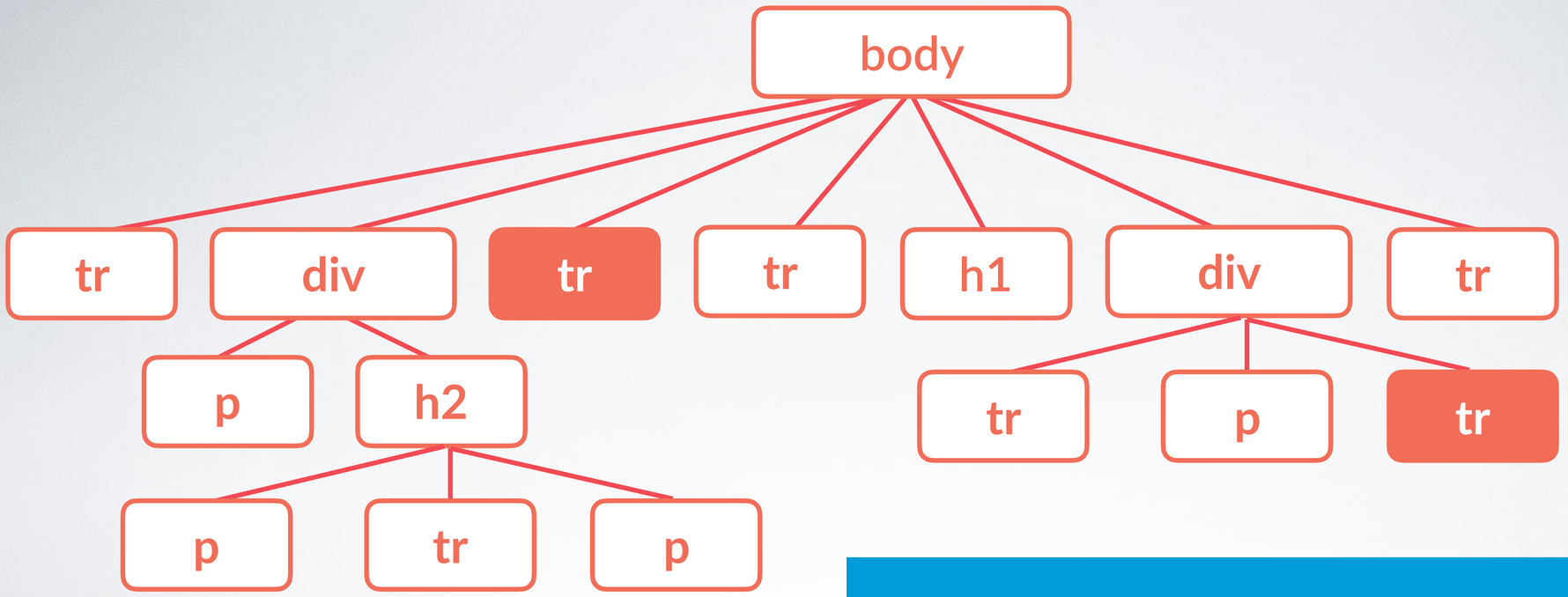


N孩後，選__

老大 :first-child
老么 :last-child
獨子 :only-child
老N :nth-child(數字)
倒著數老N :nth-last-child(數字)

CSS樣式規則：

tr: nth-child(3) { 屬性1: 值1, 宣告2, ... }
選擇每個標籤裡的第3個孩子，而且是tr



N孩後，選__

老大 :first-child
老么 :last-child
獨子 :only-child
老N :nth-child(數字)
倒著數老N :nth-last-child(數字)

CSS樣式規則：

tr: nth-child(3) { 屬性1: 值1, 宣告2, ... }
選擇每個標籤裡的第3個孩子，而且是tr

進階用法

:nth-child(an+b)

CSS樣式規則：

tr: nth-child(an+b) { 屬性1: 值1, 宣告2, ... }

選擇第an+b個tr的所有子元素

CSS

```
tr{
  background: #ccc;
}
tr:nth-child(2n+1) {
  background: lightgreen;
}
```

Output

食物名稱	熱量(kcal)
雞腿飯	700
炒麵	400
炒米粉	400
牛肉麵	470
起司三明治	200

Q :nth-last-child(2n+1) ?

歸類後，選__

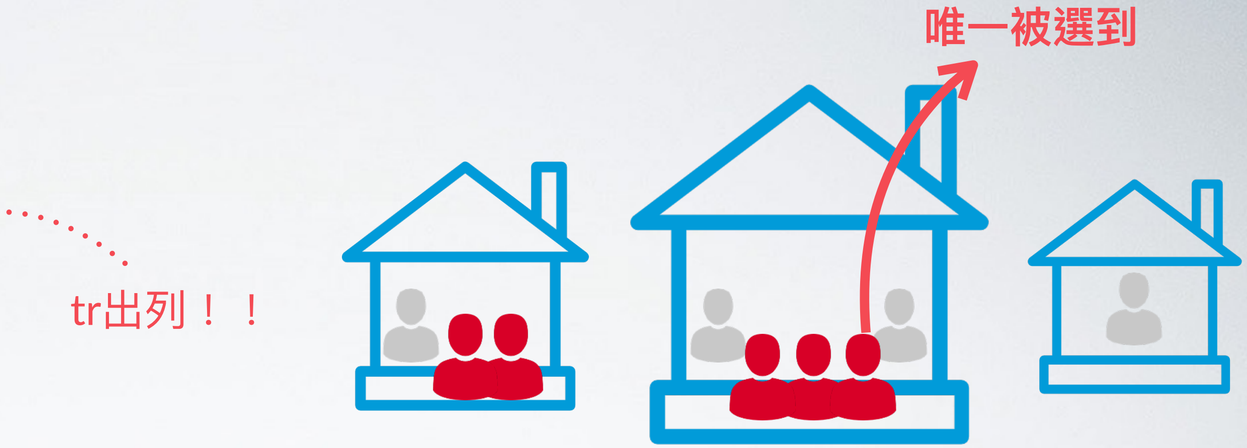
老大 :first-of-type

老么 :last-of-type

獨子 :only-of-type

老N :nth-of-type (數字)

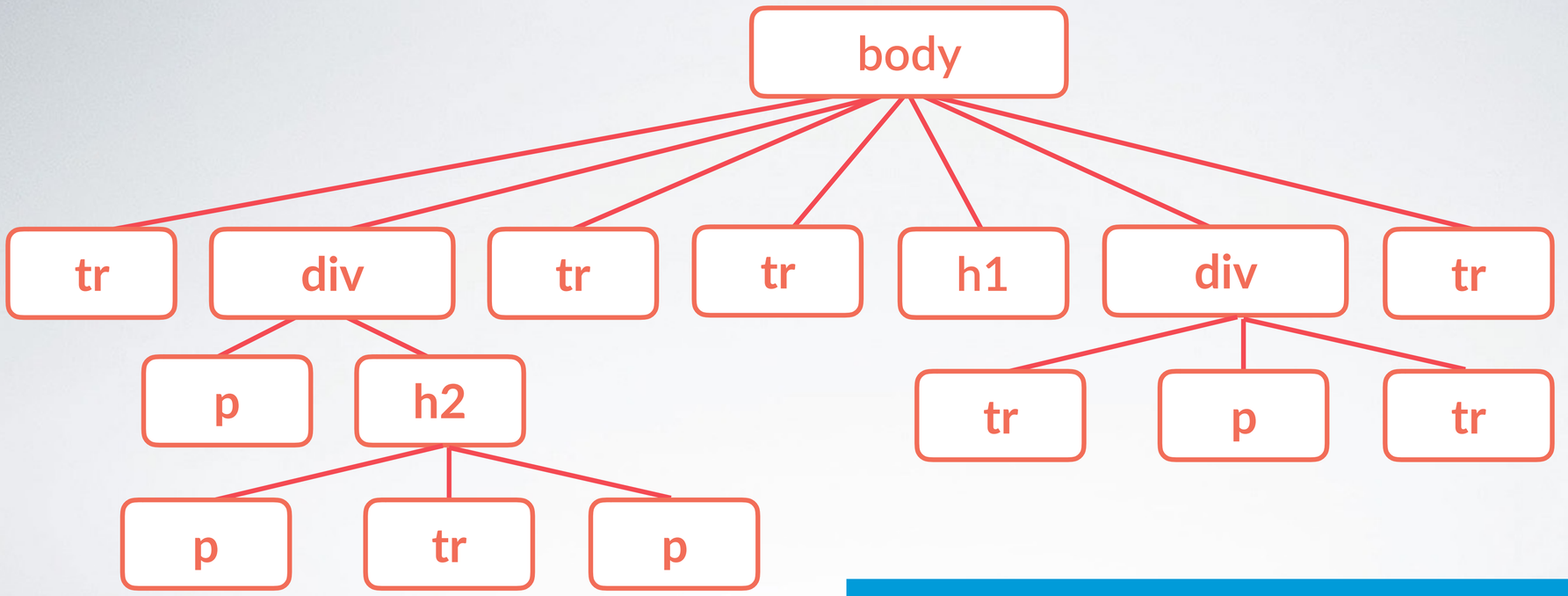
倒著數老N :nth-last-of-type(數字)



CSS樣式規則：

tr: nth-of-type(3) { 屬性1: 值1, 宣告2, ... }

選擇每個標籤裡的tr，第3個



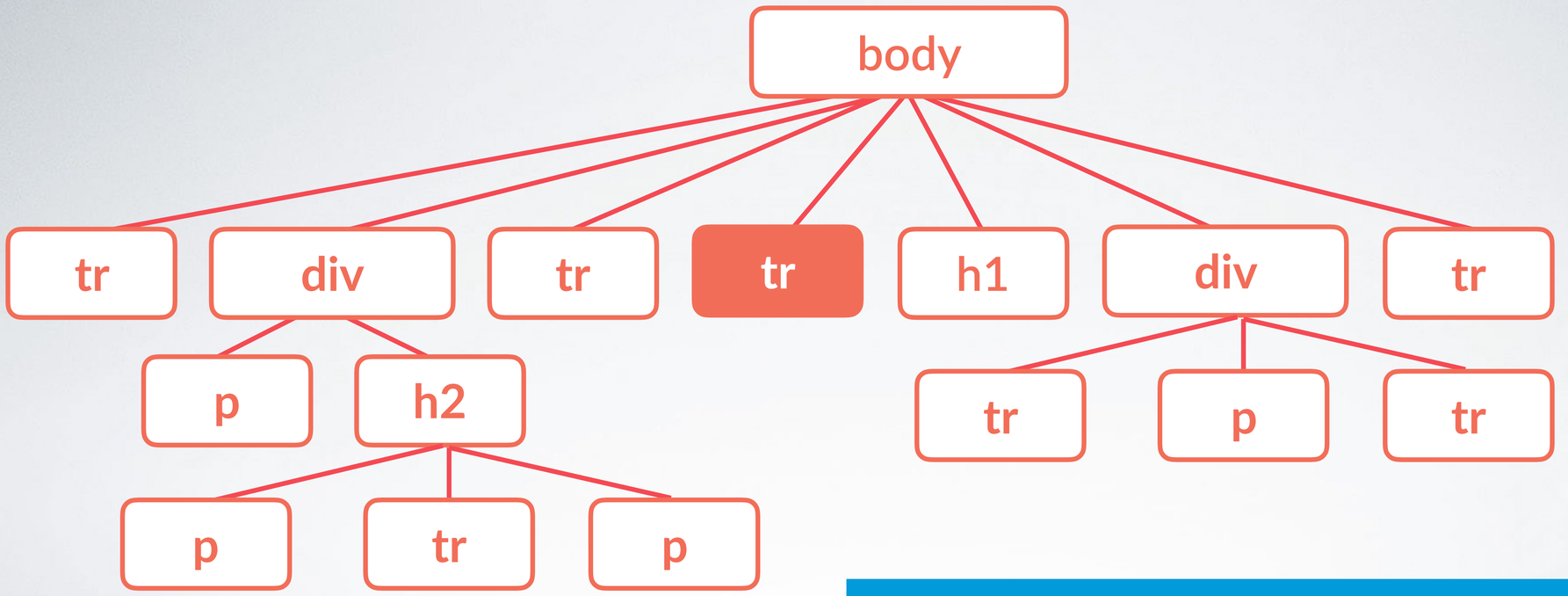
歸類後，選__

老大 :first-of-type
老么 :last-of-type
獨子 :only-of-type
老N :nth-of-type (數字)
倒著數老N :nth-last-of-type(數字)

CSS樣式規則：

tr: nth-of-type(3) { 屬性1: 值1, 宣告2, ... }

選擇每個標籤裡的tr，第3個



歸類後，選__

老大 :first-of-type
老么 :last-of-type
獨子 :only-of-type
老N :nth-of-type (數字)
倒著數老N :nth-last-of-type(數字)

CSS樣式規則：

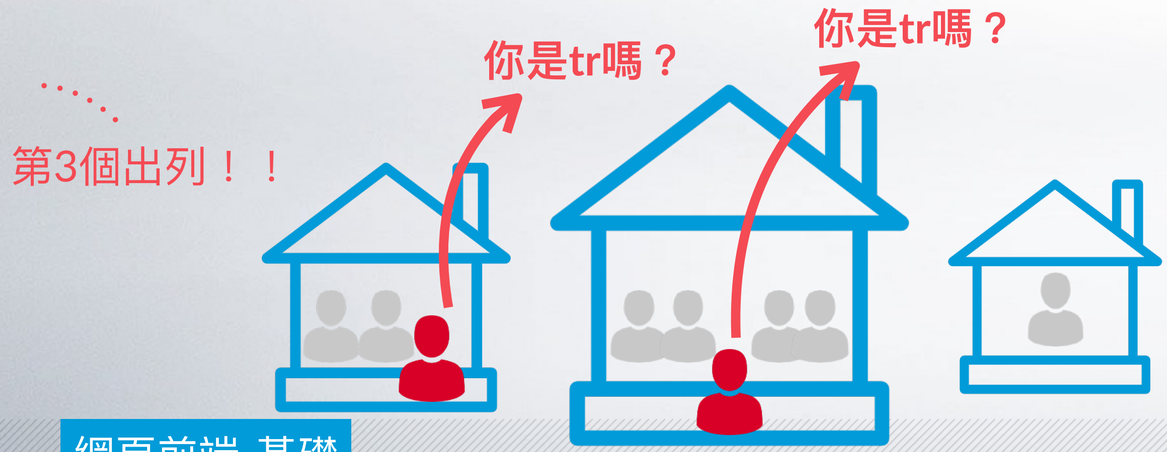
tr: nth-of-type(3) { 屬性1: 值1, 宣告2, ... }

選擇每個標籤裡的tr，第3個

哪裡不一樣： nth-child V.S. nth-of-type

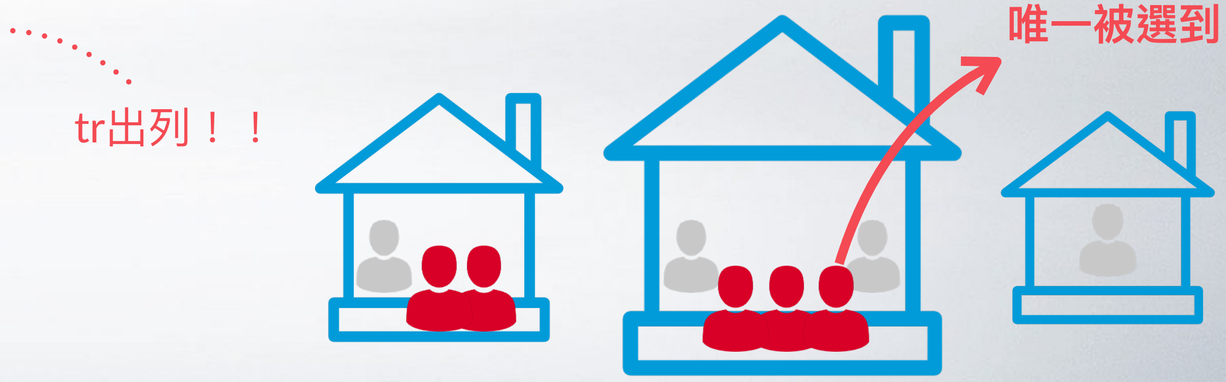
N孩後，選__：

tr: nth-child(3) { 屬性1: 值1, 宣告2, ... }
選擇每個標籤裡的第3個孩子，而且是tr



歸類後，選__：

tr: nth-of-type(3) { 屬性1: 值1, 宣告2, ... }
選擇每個標籤裡的tr，第3個



選___裡，空的

膝下無子 :empty

CSS樣式規則：

div: empty { 屬性1: 值1, 宣告2, ... }

選擇div內，沒有內容的子元素

```html
<div>
  <p>我是第1個p</p>
  <p>我是第2個p</p>
  <p></p>
  <p>我是第4個p</p>
  <p>我是第5個p</p>
</div>
```

```css
div{background: #ccc;}

p:empty{
  background: lightgreen;
  width:100px;
  height:20px;
}
```

我是第1個p

我是第2個p

我是第4個p

我是第5個p

CSS樣式規則：

div: empty { 屬性1: 值1, 宣告2, ... }

選擇div內，沒有內容的子元素

選__裡，非的

反轉選擇 :not(選擇器)

CSS樣式規則：

: not(p) { 屬性1: 值1, 宣告2, ... }

選擇不是p的元素

div: not(p) { 屬性1: 值1, 宣告2, ... }

選擇div裡，不是p的元素

```html
<div>
  <p>我是第1個p</p>
  <p>我是第2個p</p>
  <p>我是第3個p</p>
  <div>我是第1個div</div>
  <a href="#">我是第1個a</a>
</div>
```

```css
div{background: #ccc;}
p {
    color: #000000;
}
:not(p) {
    color: #f00;
}
```

我是第1個p

我是第2個p

我是第3個p

我是第1個div

我是第1個a

CSS樣式規則：

: not(p) { 屬性1: 值1, 宣告2, ... }

選擇不是p的元素

div: not(p) { 屬性1: 值1, 宣告2, ... }

選擇div裡，不是p的元素

依狀態來套用「假的」Class

Ex. 滑鼠移過去，狀態改變

虛擬類別選擇器

Pseudo-classes Selector

1. **N孩後，選 __** :nth-child(3)
2. **歸類後，選 __** :nth-of-type(3)
3. **選 __ 裡，空的** :empty
4. **選 __ 裡，非的** :not(p)