D3.JS

互動式資料視覺化

Lecturer: LinJer 林哲

evin92@gmail.com

CSS-選擇器

CSS樣式規則：

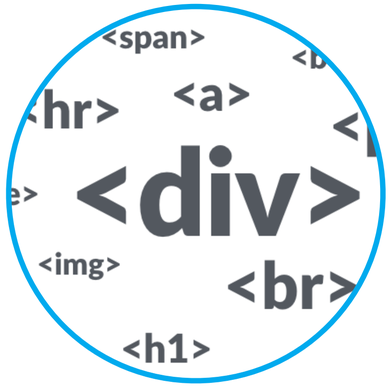
選擇器 { 屬性1: 值1, 宣告2, ... }

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

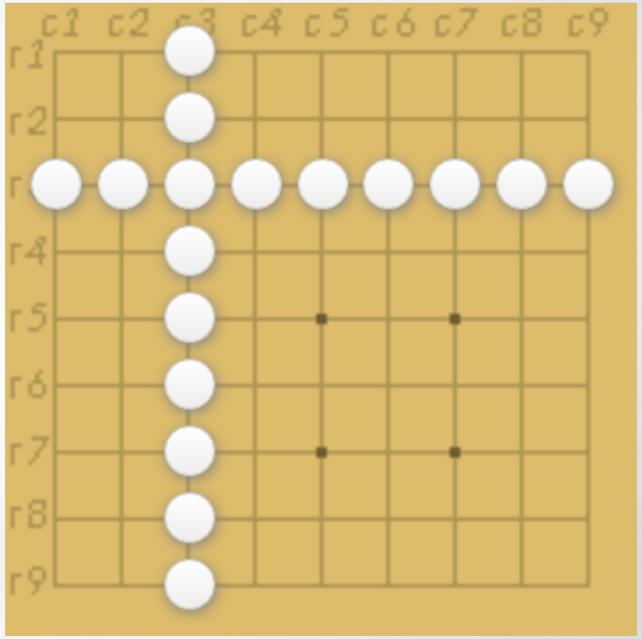
即使用上了四種選擇器:

例如: a#target.tall[align=“center”]

你還是有可能選不到你要的那些元素

棋盤問題

[目標] 選取棋盤中任一欄位



按我前往

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：

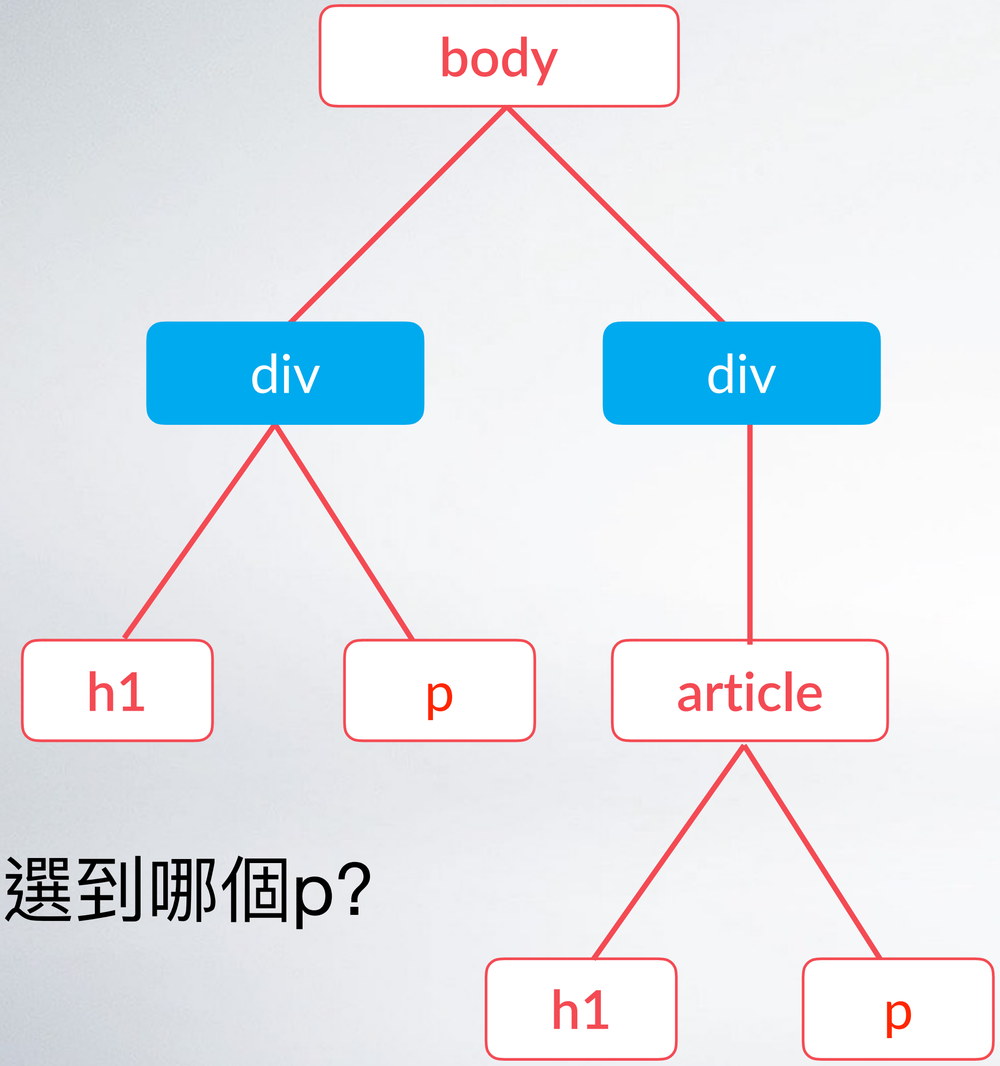
div>p { 屬性1: 值1, 宣告2, ... }

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



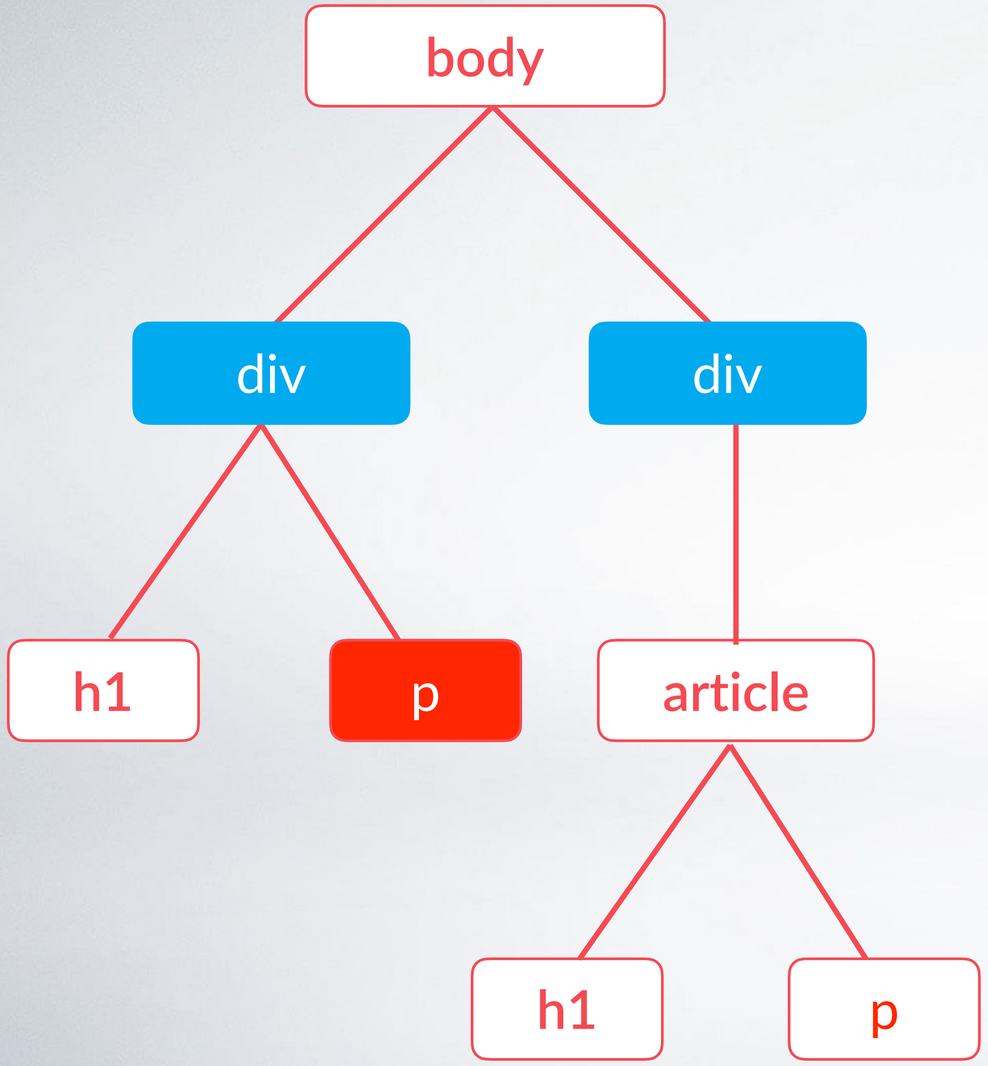
選到哪個p?

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

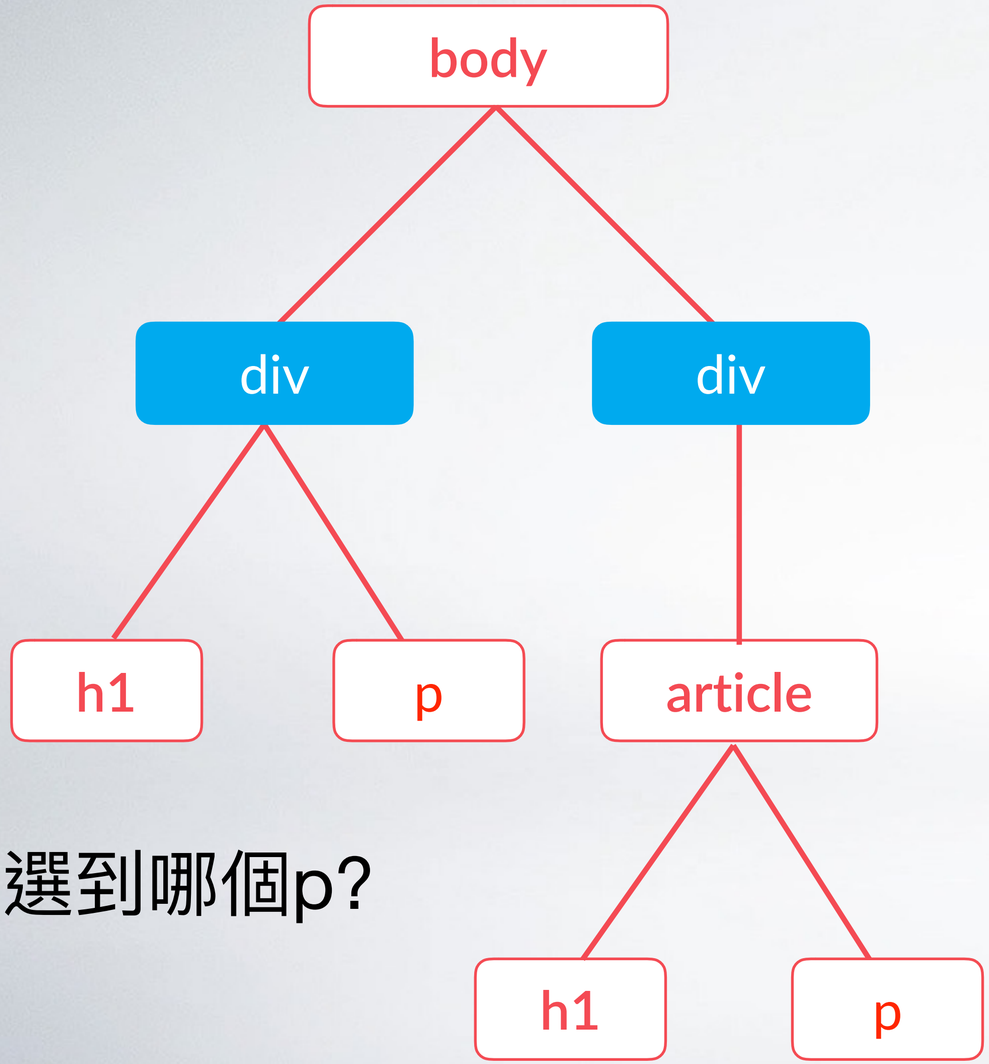
CSS樣式規則：

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



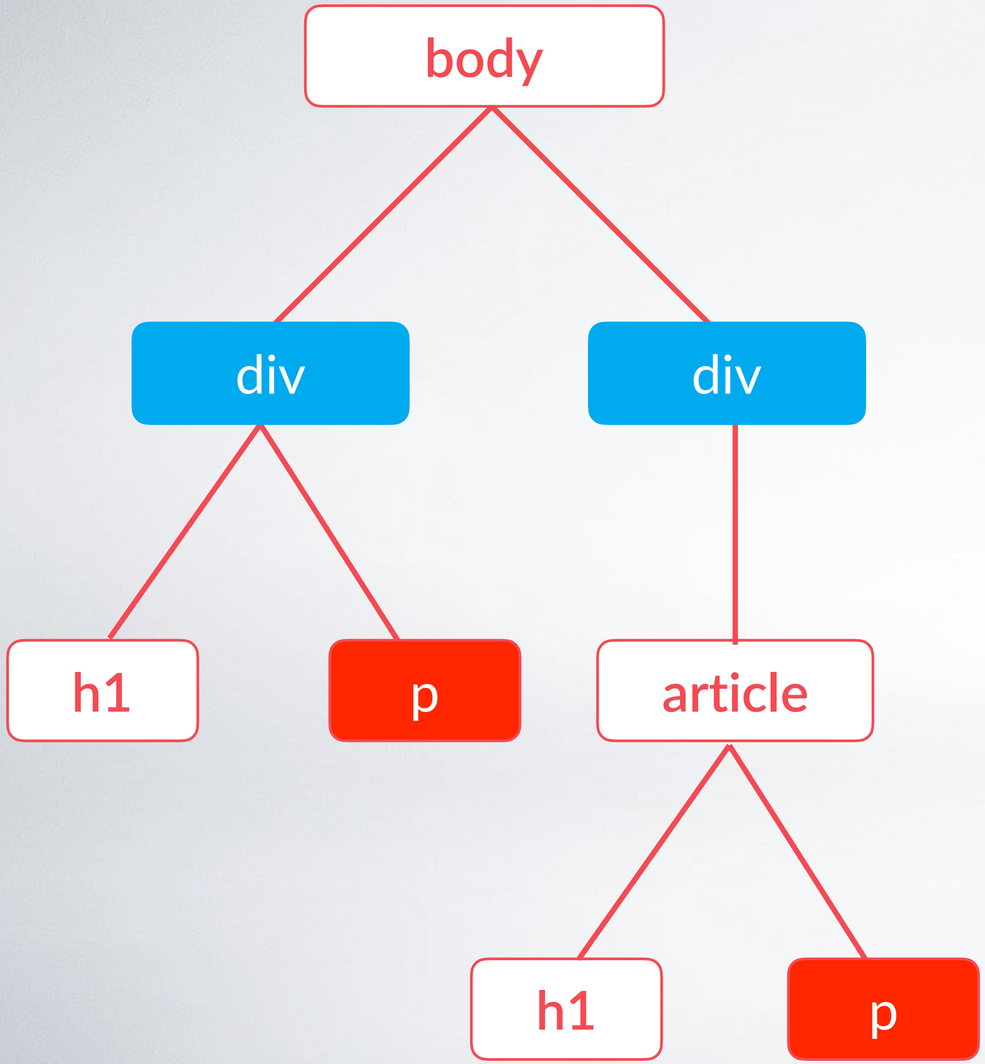
選到哪個p?

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

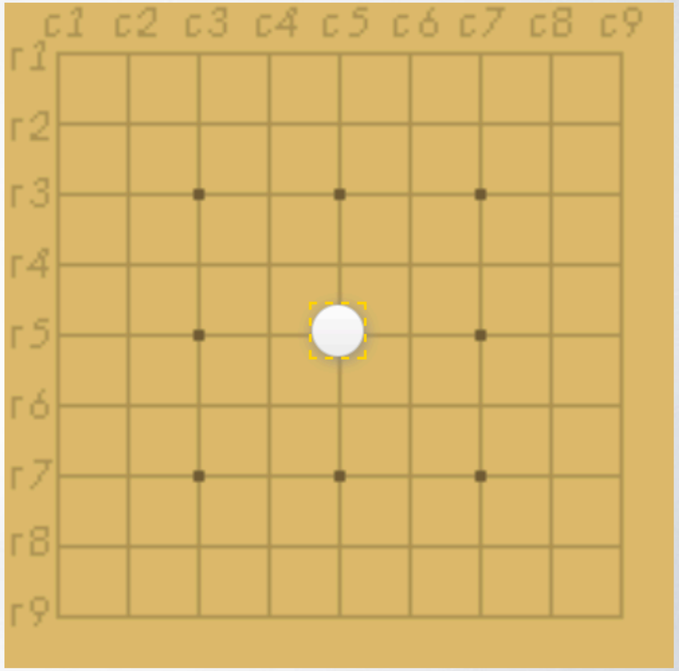
CSS樣式規則：

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

棋盤問題



按我前往

會了 孩子/子孫 模式
你就可以輕易解決問題了

[目標] 選取棋盤中(r5 , c5)欄位

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

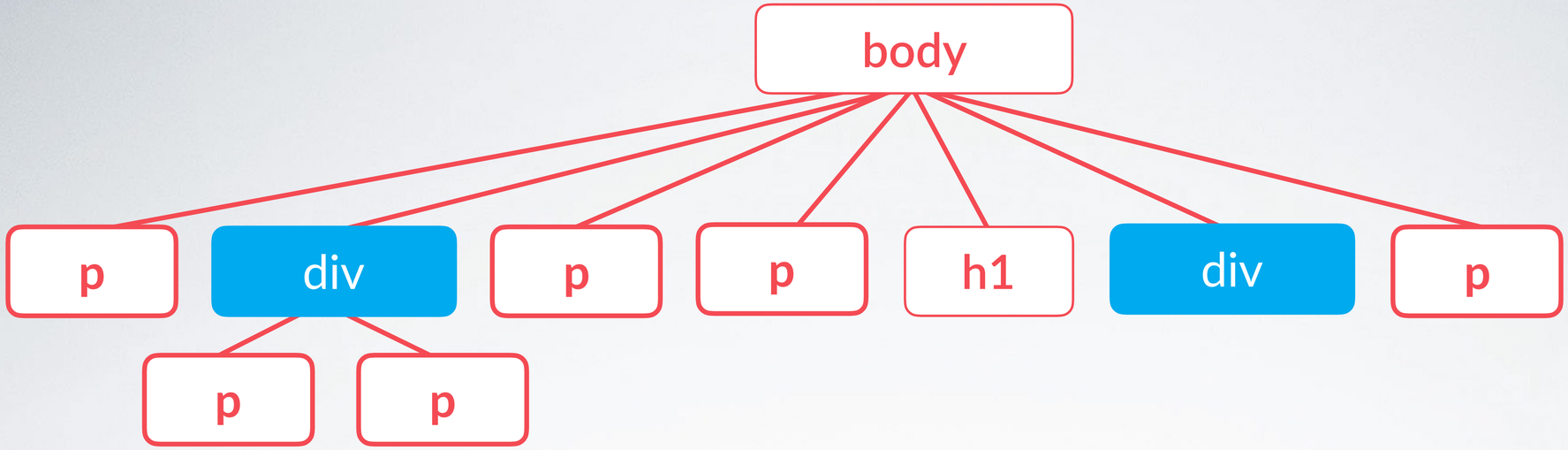
Adjacent Silbing Selector

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

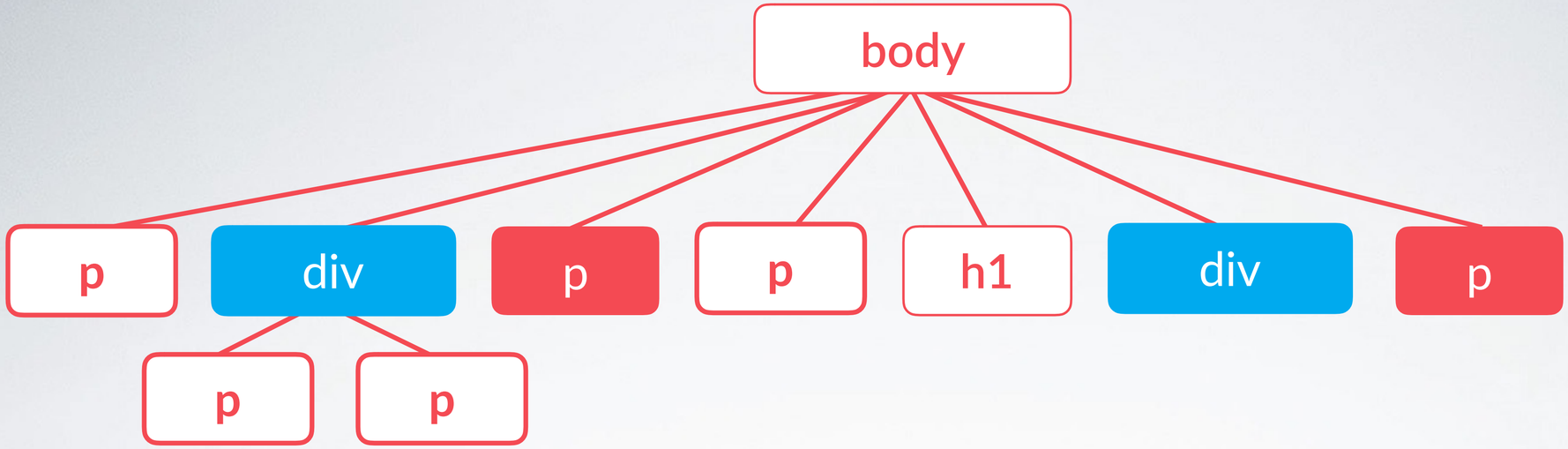
Adjacent Silbing Selector

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

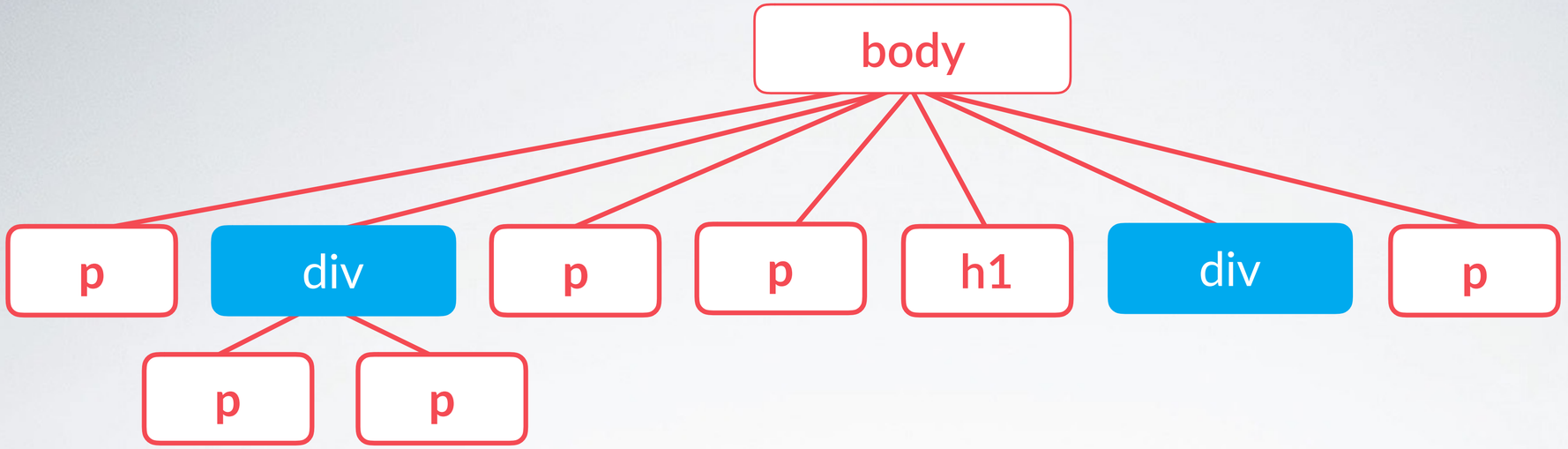
Sibling Selector

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

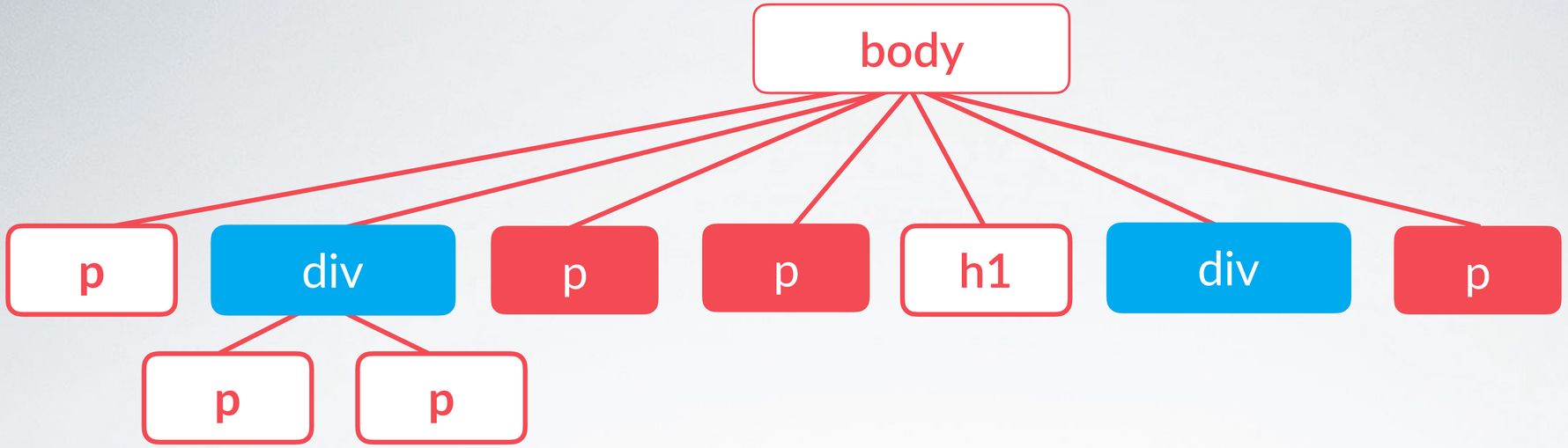
Sibling Selector

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：

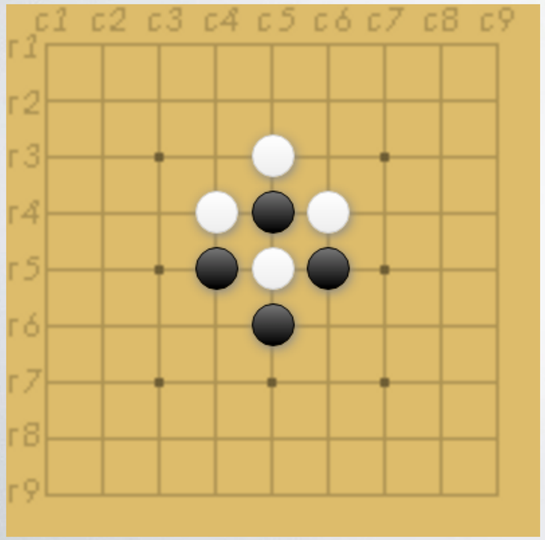
div~p { 屬性1: 值1, 宣告2, ... }

在div後為p者

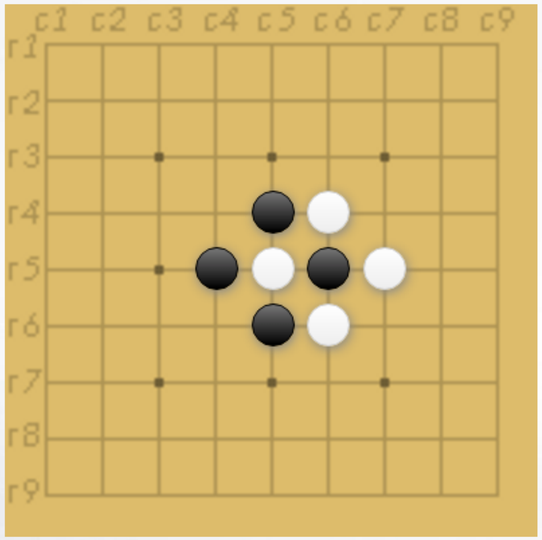
= 選擇所有p，他前面有div

會了 跟屁蟲 模式

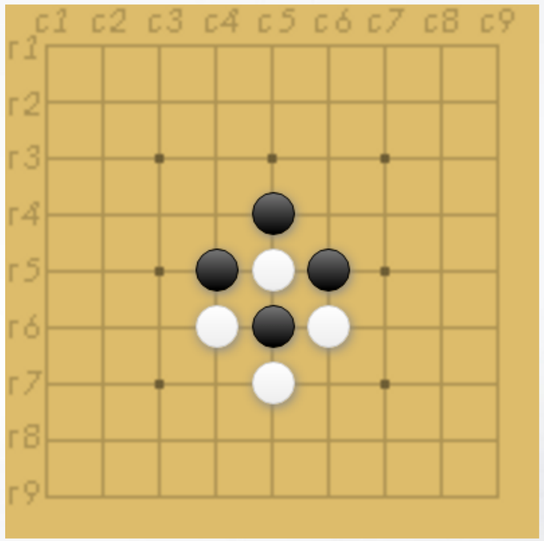
這個問題應該難不倒你？ .bk+.col{ visibility: visible; }



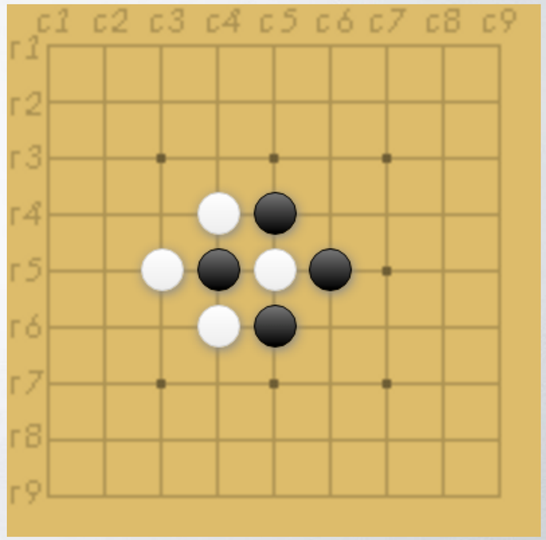
A



B



C

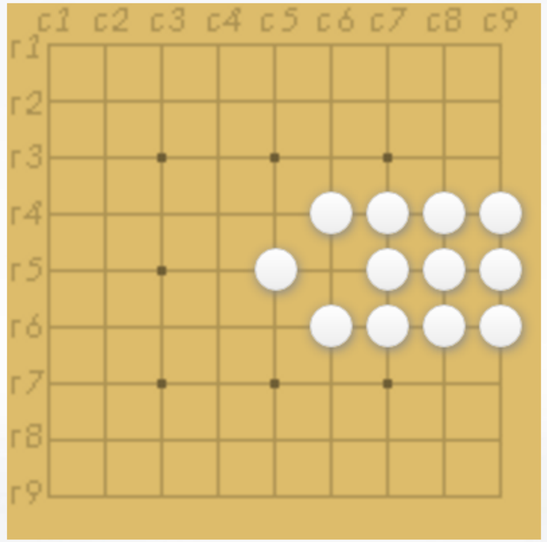


D

按我測試

怎麼用一條選擇器就選出來？

[提示] 剛剛學過了



按我練習

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

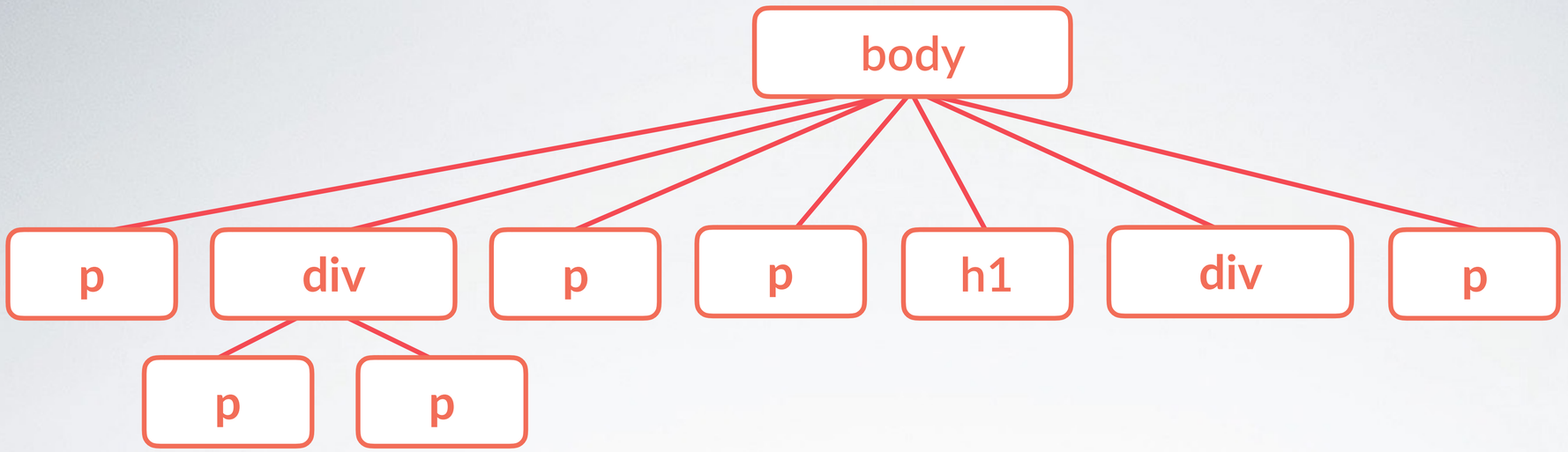
選擇器再進化 - 全選模式 Universal Selector

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



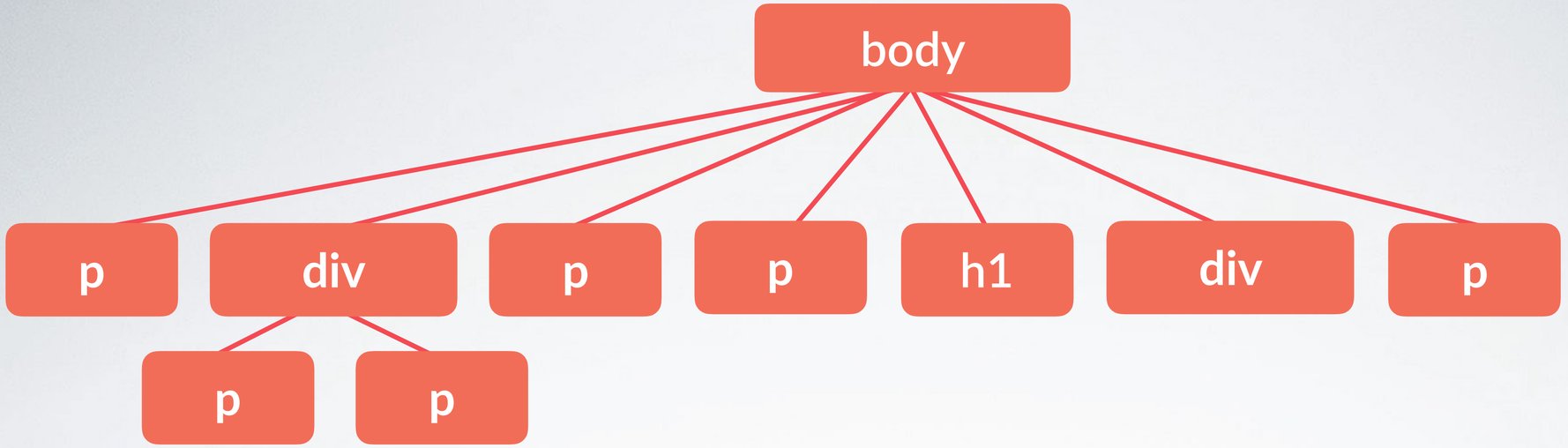
全選模式
Universal Selector

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



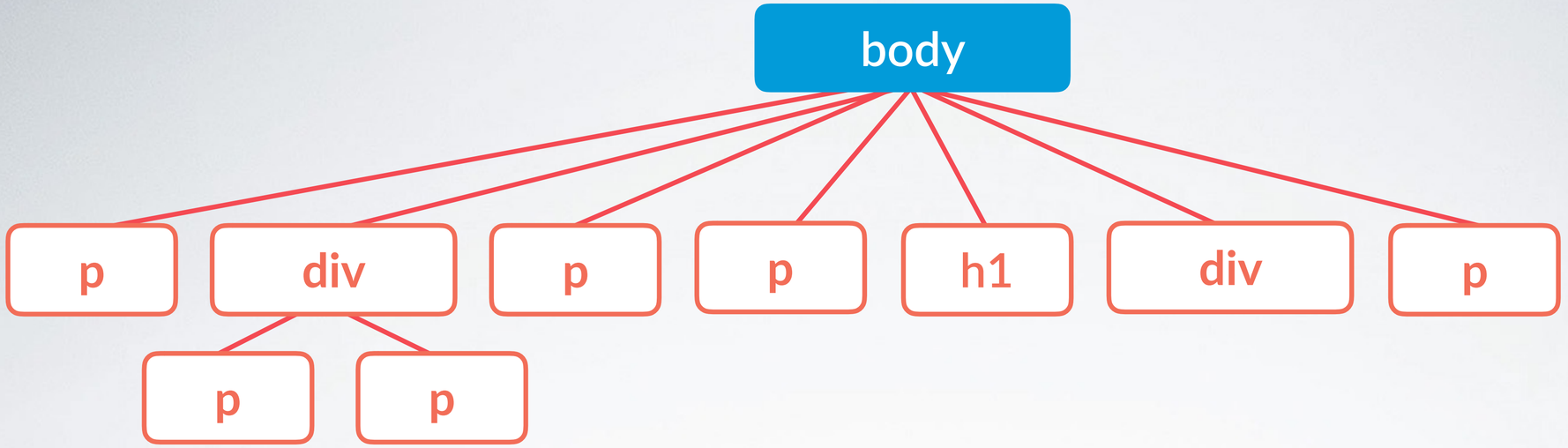
全選模式
Universal Selector

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



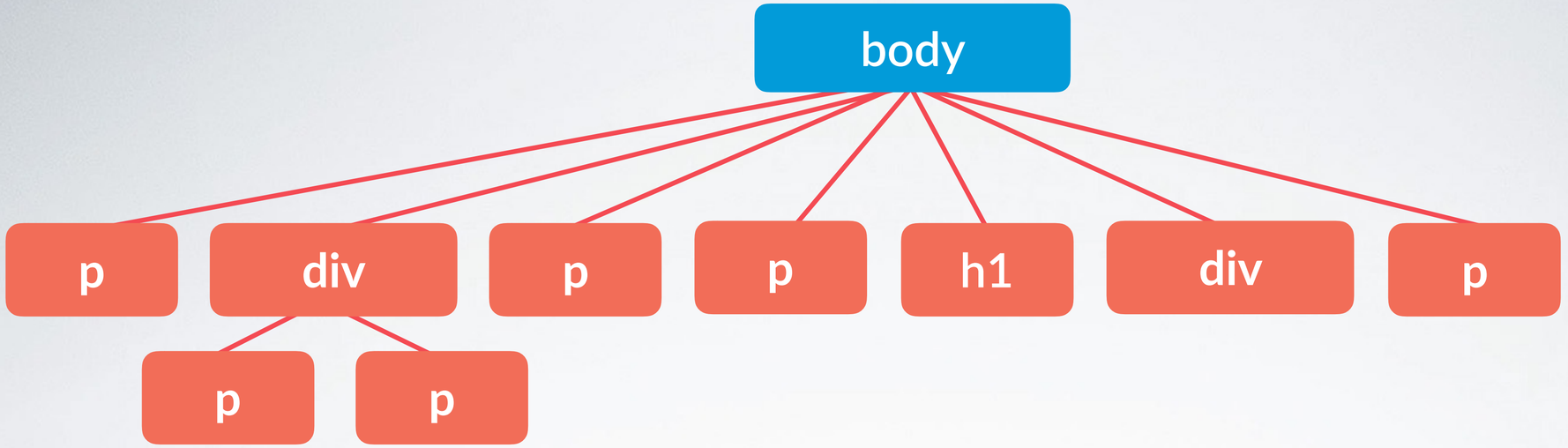
全選模式
Universal Selector

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



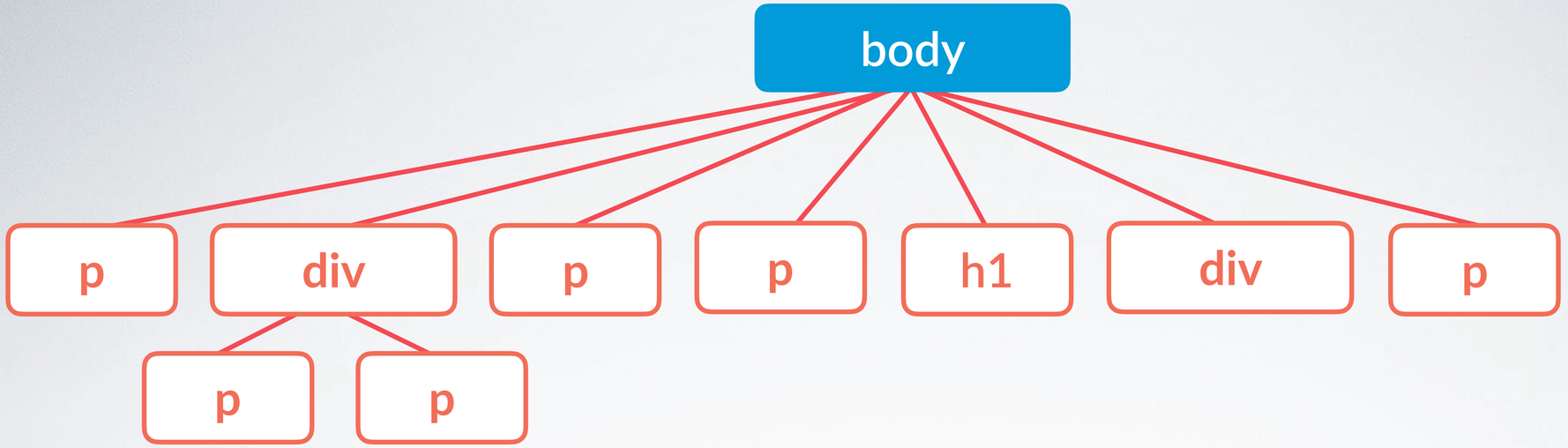
全選模式
Universal Selector

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



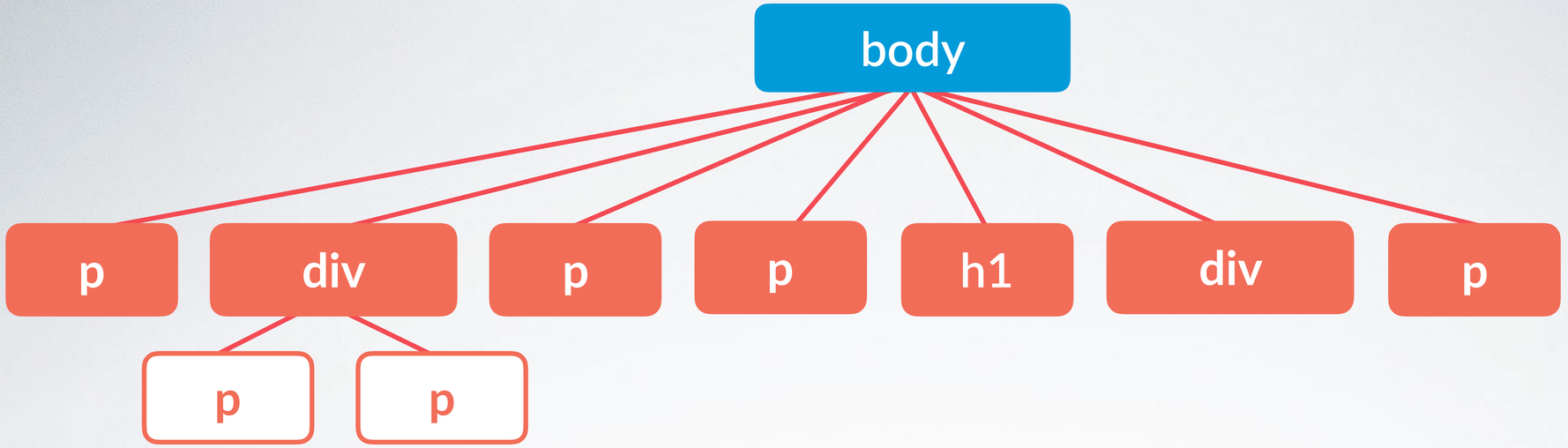
全選模式
Universal Selector

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

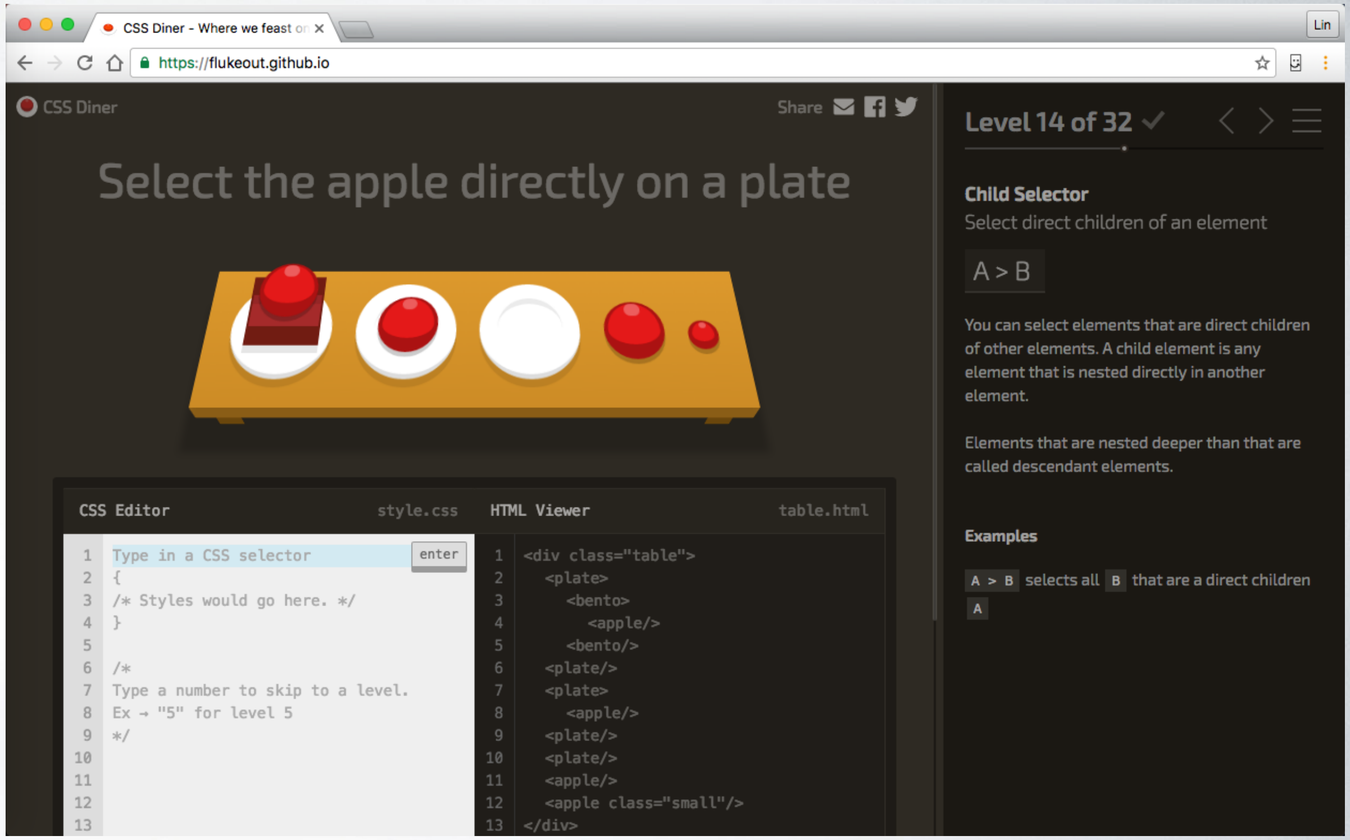
Another kid

Third kid

遊戲時間

CSS-Dinner

[注意] 玩到第14題即可！

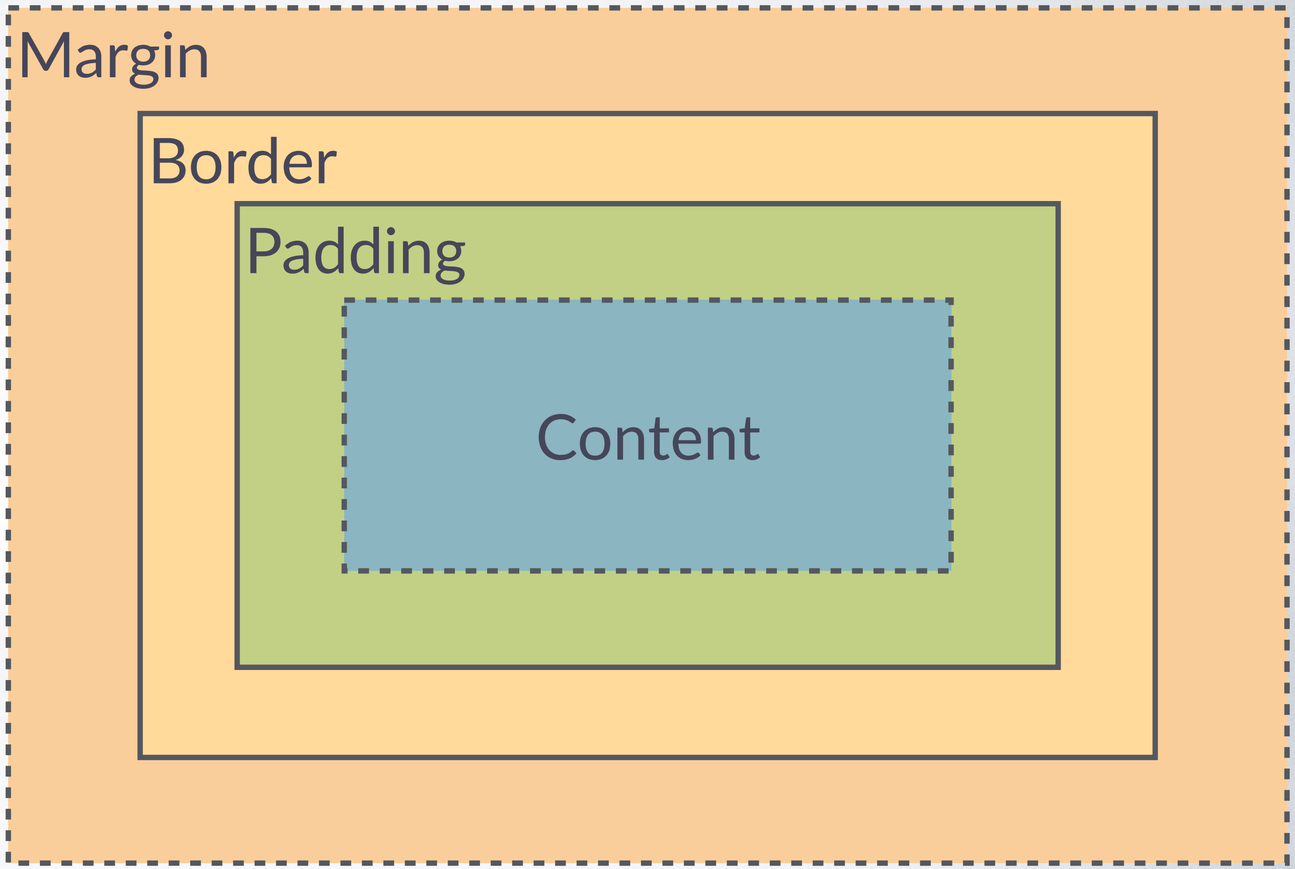


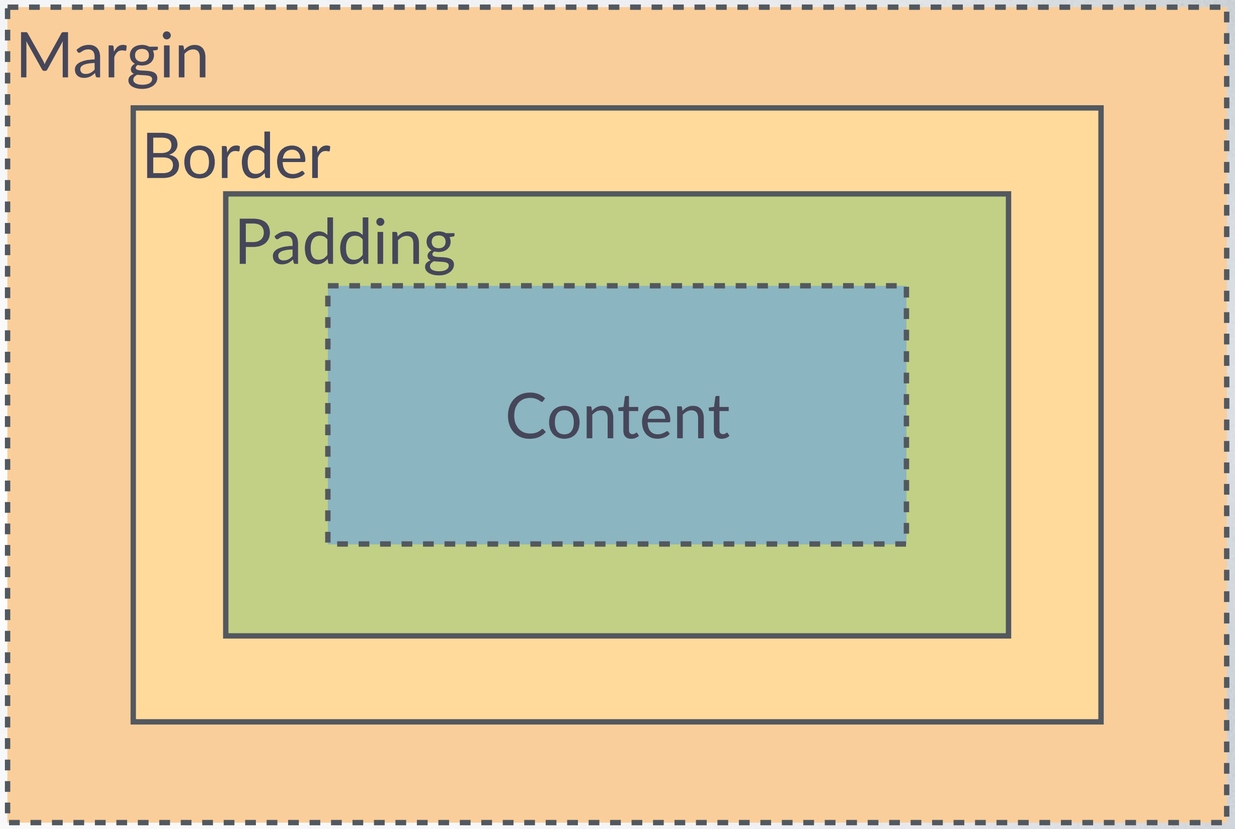
[按我前往]

Box Model

所有的HTML元素可以被視為一方塊(Box)，方塊內由四個子框組成。

從內到外分別是：
內容(Content)、**空白區(Padding)**、**邊線(Border)**、**邊境(Margin)**



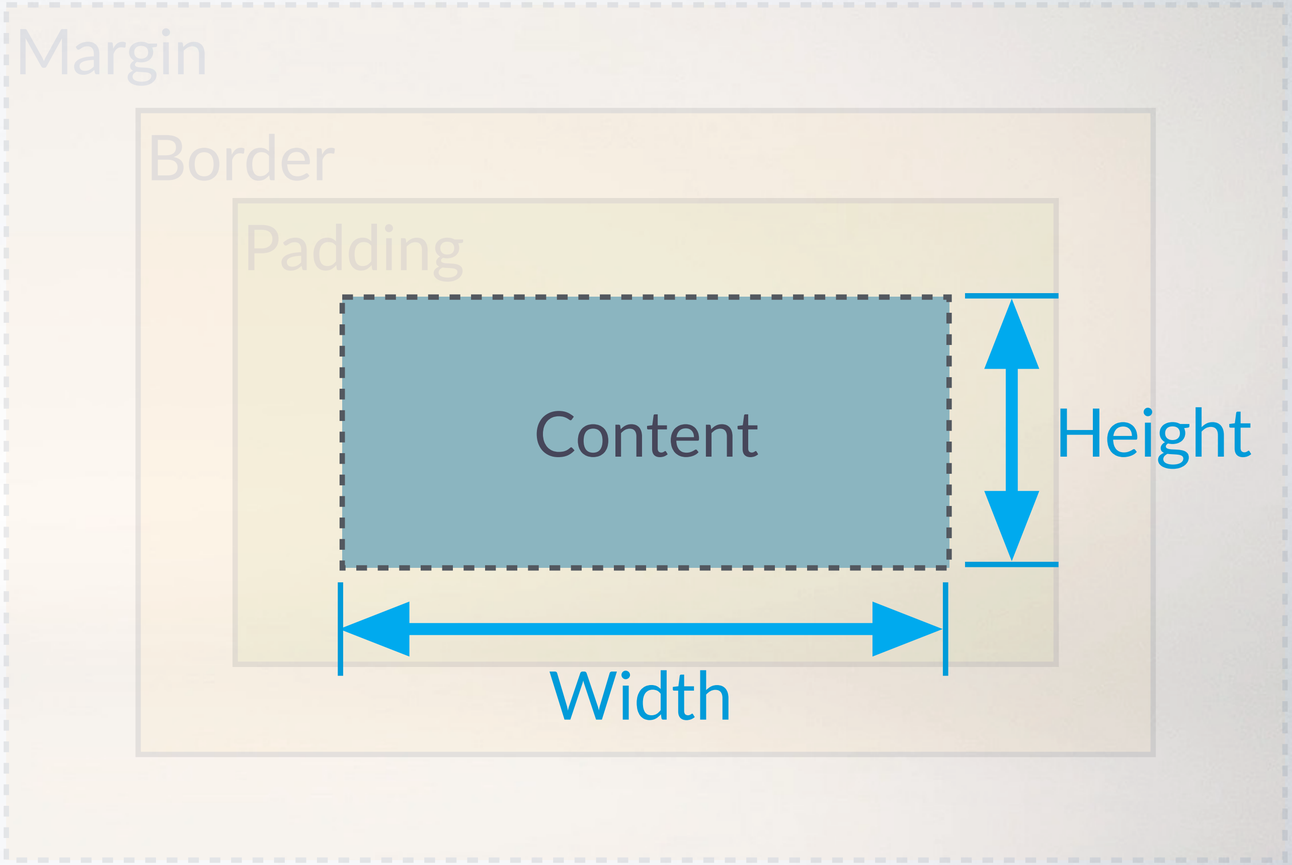

Margin
Border
Padding
Content

內容(Content) - 城堡

文字或圖片出現的方框

- **width**
- **height**

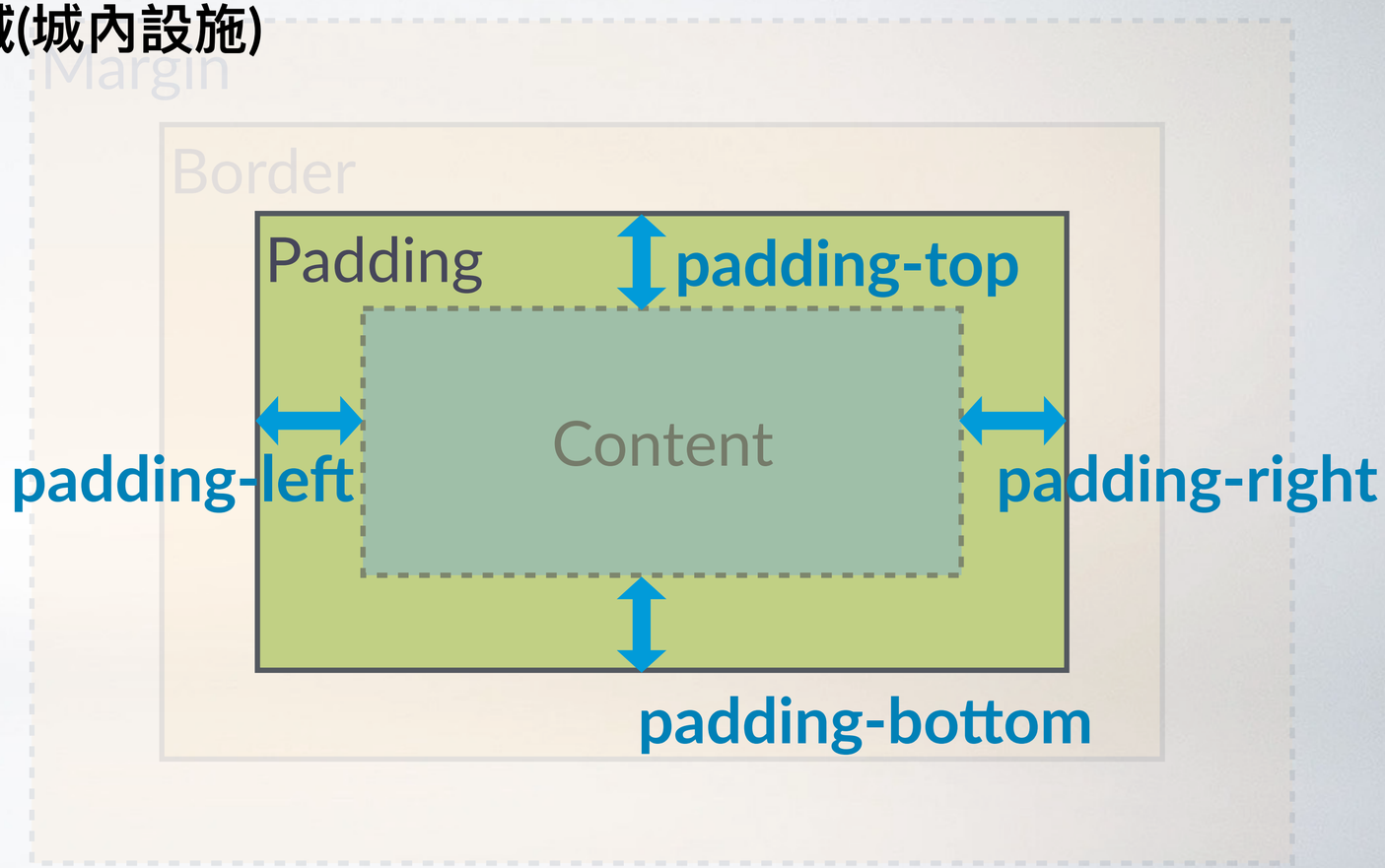
讓content寬度定為400px:
width : 400px;



空白區(Padding) - 城堡至城牆區域(城內設施)

Content外圍的空白方框，
背景受background影響

- padding-top
- padding-bottom
- padding-right
- padding-left
- padding



一”條”成型(簡寫)

空白區(Padding) - 城堡至城牆區域(城內設施)

(1)padding: 上下左右;

(2)padding: 上下 左右;

(3)padding: 上 左右 下;

(4)padding: 上 右 下 左;

[貼貼看]

```
div{
  background: pink;
  width: 5em;

  /* padding: 1em; */
  /* padding: 2em 1em; */
  /* padding: 2em 1em 5em; */
  /* padding: 4em 1em 3em 9em; */
}
```

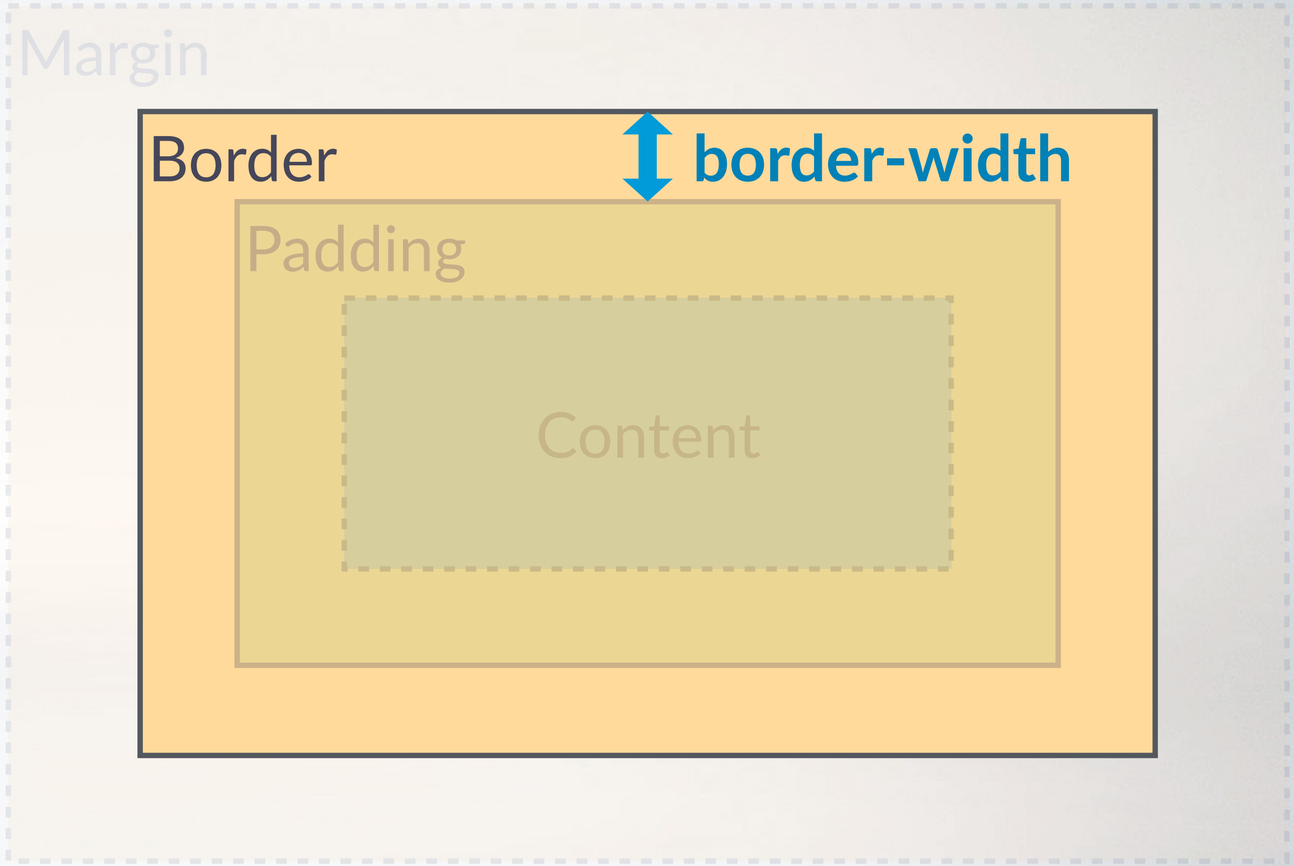
```
<div> Content, period. </div>
<div> Content, period. </div>
<div> Content, period. </div>
<div> Content, period. </div>
```

邊線(Border) - 城牆

Padding外圍的方框(線條)

- border-width
- border-style (必填:預設none)
- border-color

- border: width style color



—”條”成型(簡寫)

邊線(Border) - 城牆

- **border: width style color**

- **border-top: width style color**
- **border-right: width style color**
- **border-bottom: width style color**
- **border-left: width style color**

[按我連結]

```
p{width: 200px;}

p.dotted {border-style: dotted}
p.dashed {border-style: dashed}
p.solid {border-style: solid}
p.double {border-style: double}
p.groove {border-style: groove}
p.ridge {border-style: ridge}
p.inset {border-style: inset}
p.outset {border-style: outset}
```

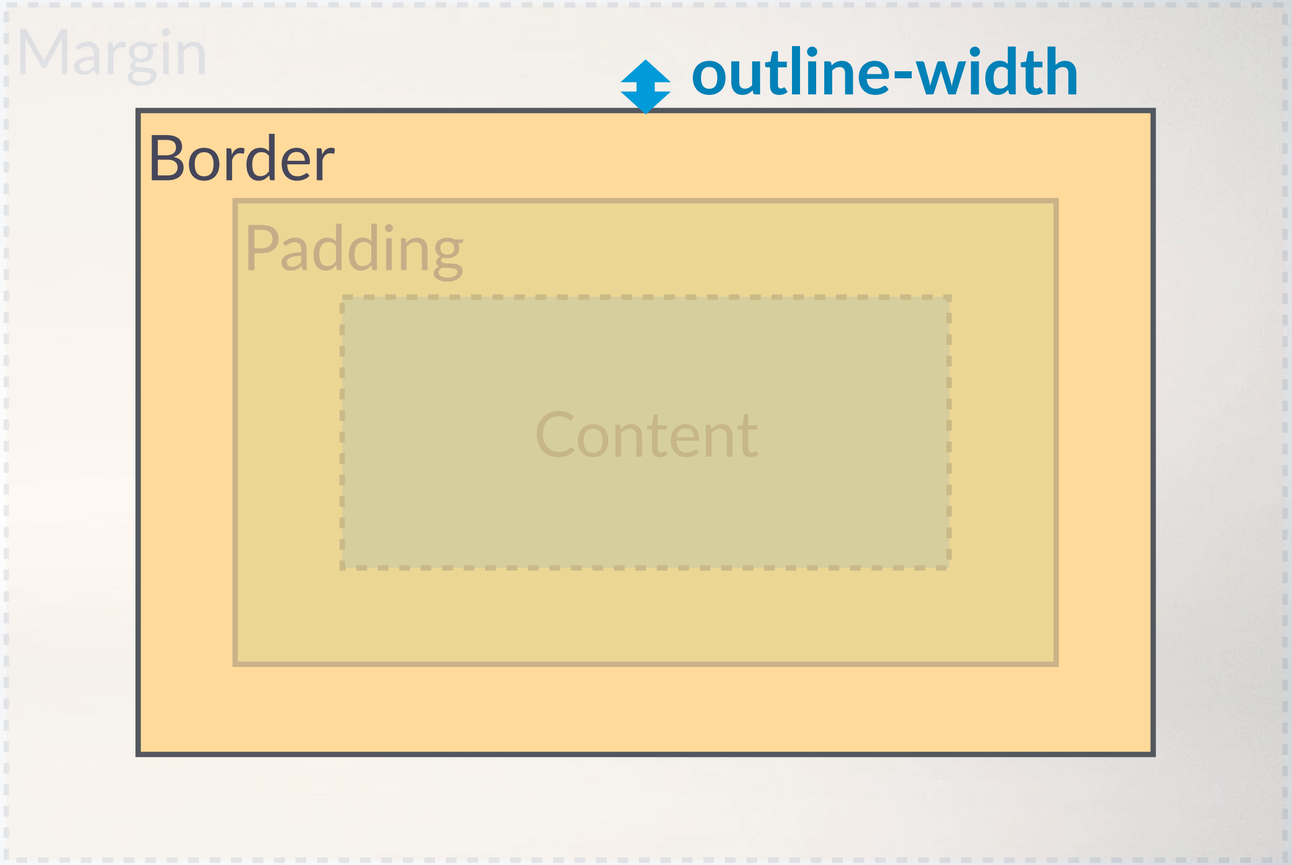
```
<p class="dotted">A dotted border</p>
<p class="dashed">A dashed border</p>
<p class="solid">A solid border</p>
<p class="double">A double border</p>
<p class="groove">A groove border</p>
<p class="ridge">A ridge border</p>
<p class="inset">An inset border</p>
<p class="outset">An outset border</p>
```

邊線的輪廓線(Outline) -
城牆上的雜草

border外圍的方框
(不增加額外空間，只佔用Margin區域)

- **outline-color**
- **outline-style (必填:預設none)**
- **outline-width**

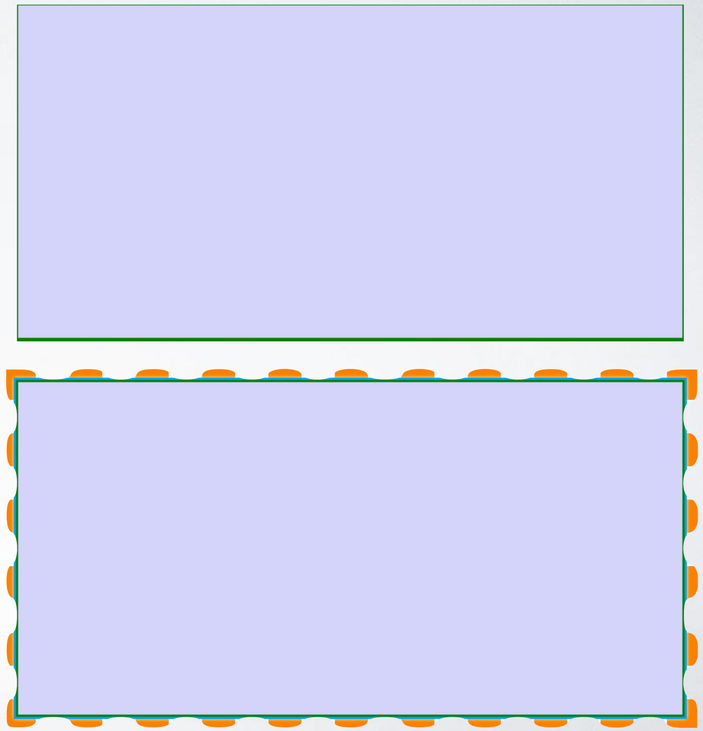
- **outline: color style width**



邊線的輪廓線(Outline) -

城牆上的雜草

border外圍的方框
(不增加額外空間，只佔用Margin區域)



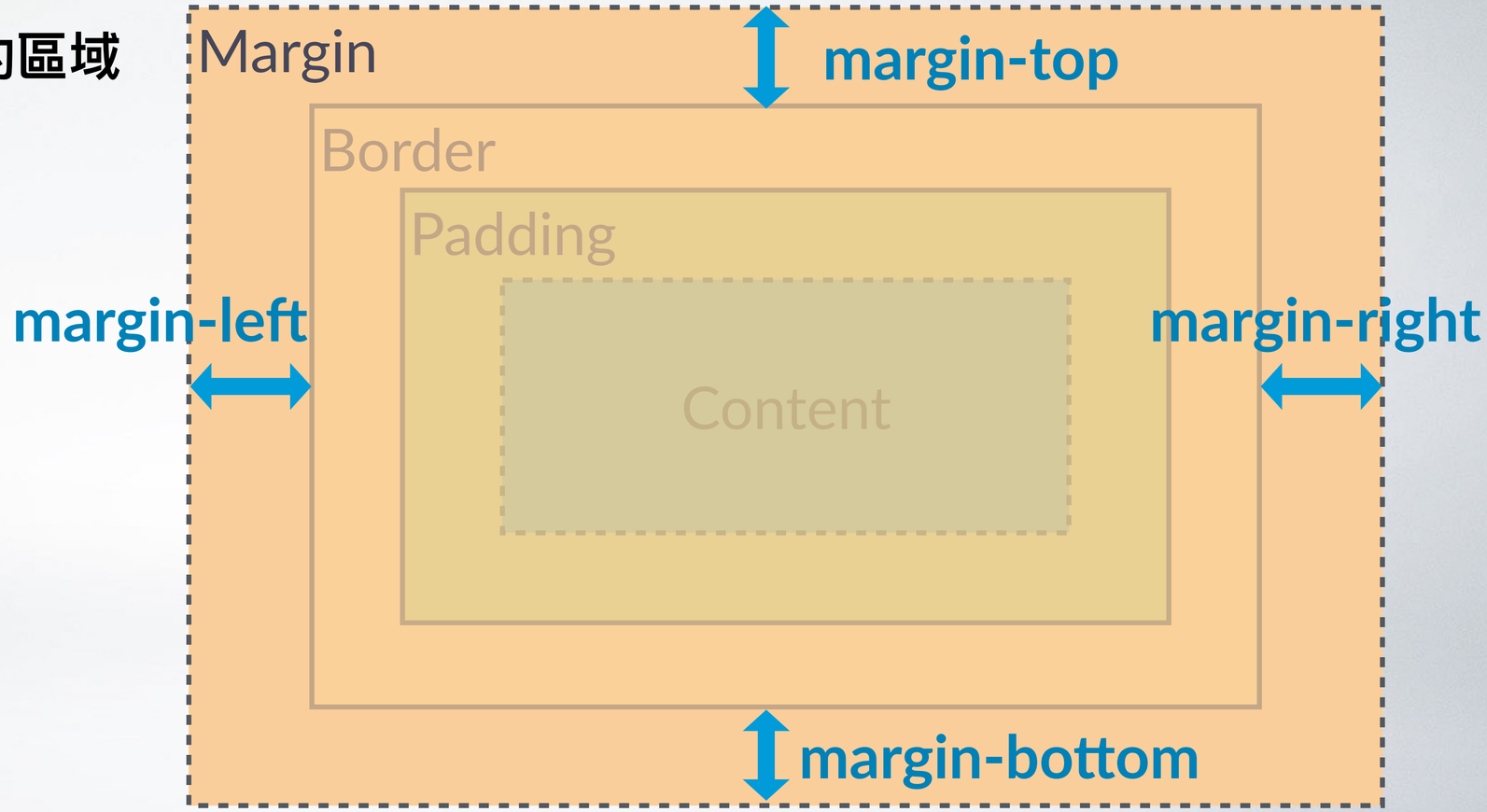
按我連結

```
<div style="background-color:#D4D4F8;width:300px;height:150px;border:2px solid green;"></div>
    <br>
<div style="background-color:#D4D4F8;width:300px;height:150px;border:2px solid green;outline:#FA8000 dashed 5px;"></div>
```

邊境(Margin) - 城牆至國界的區域

Border外圍的空白區域，
背景必為透明

- margin-top
- margin-bottom
- margin-right
- margin-left
- margin



一”條”成型(簡寫)

邊境(Margin) - 城牆至國界的區域

(1)margin: 上下左右;

(2)margin: 上下 左右;

(3)margin: 上 左右 下;

(4)margin: 上 右 下 左;

[試試看]

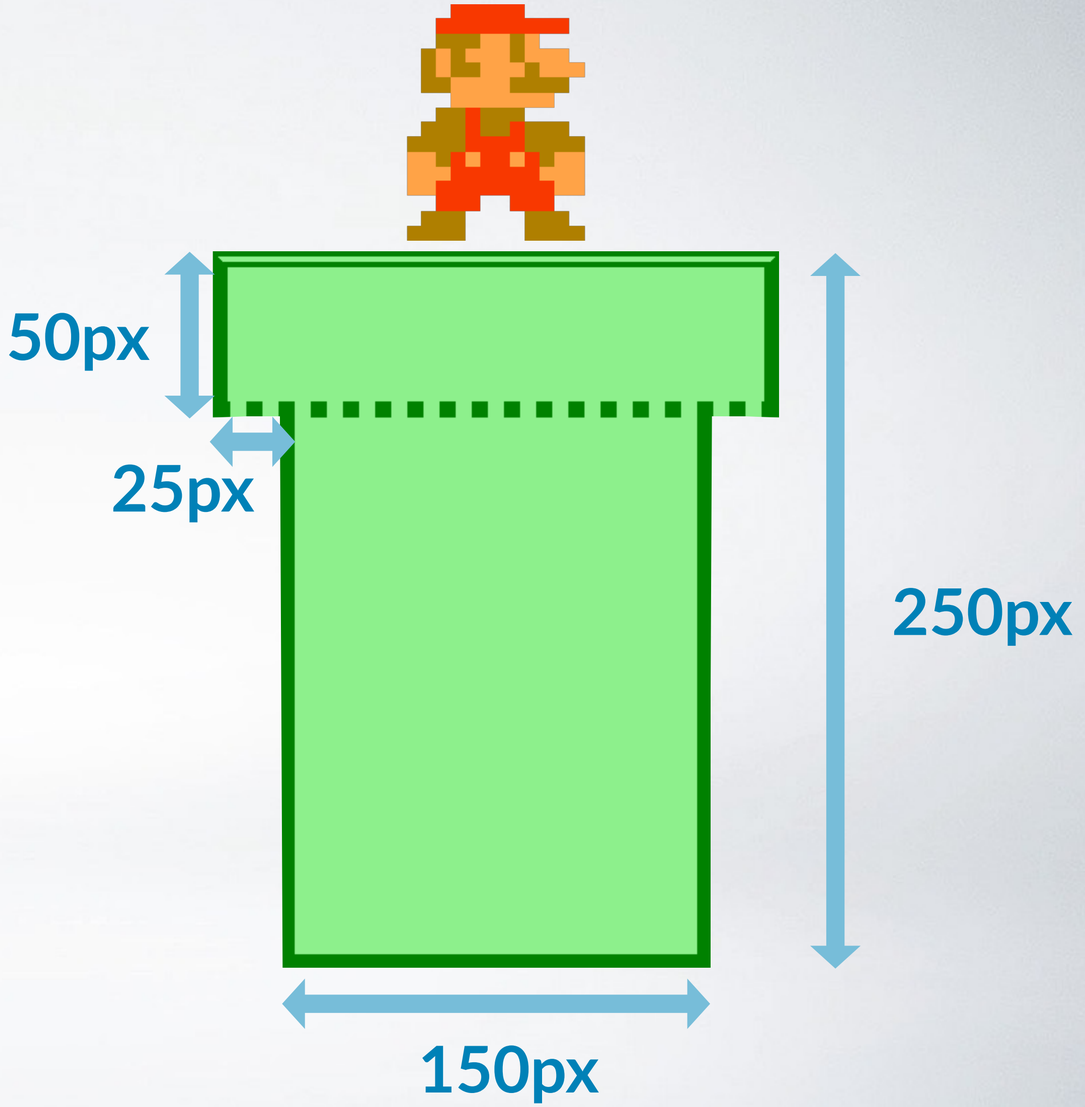
```
div{
  background: pink;
  width: 5em;

  /* margin: 1em; */
  /* margin: 2em 1em; */
  /* margin: 2em 1em 5em; */
  /* margin: 4em 1em 3em 9em; */
}
```

```
<div> Content, period. </div>
<div> Content, period. </div>
<div> Content, period. </div>
<div> Content, period. </div>
```

動手時間

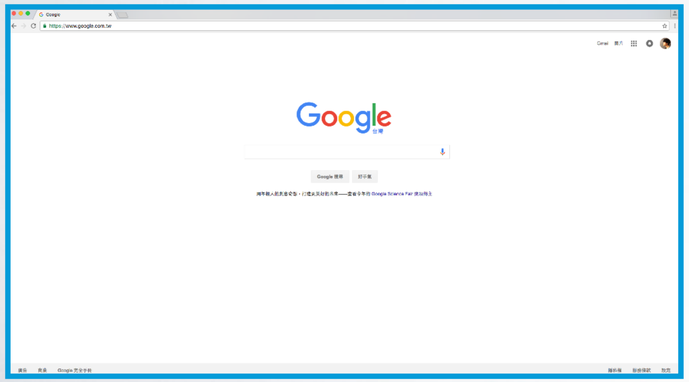
做出如右的效果

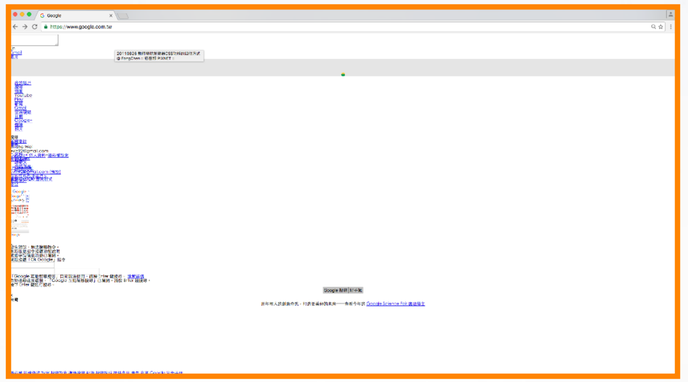


你答對了嗎？

```
<div style="background-color:lightgreen;width:200px;height:50px;border:6px solid green;border-top-style:double;border-bottom-style:dotted;”></div>
<div  style="background-color:lightgreen;width:150px;height:200px;border:6px solid green;border-top-style:none;margin-left:25px"></div>
```

網頁構成要素





1. 送出搜尋文字
2. 接收後端資料
3. 呈現搜尋結果

CSS

外觀

HTML

框架 / 內容

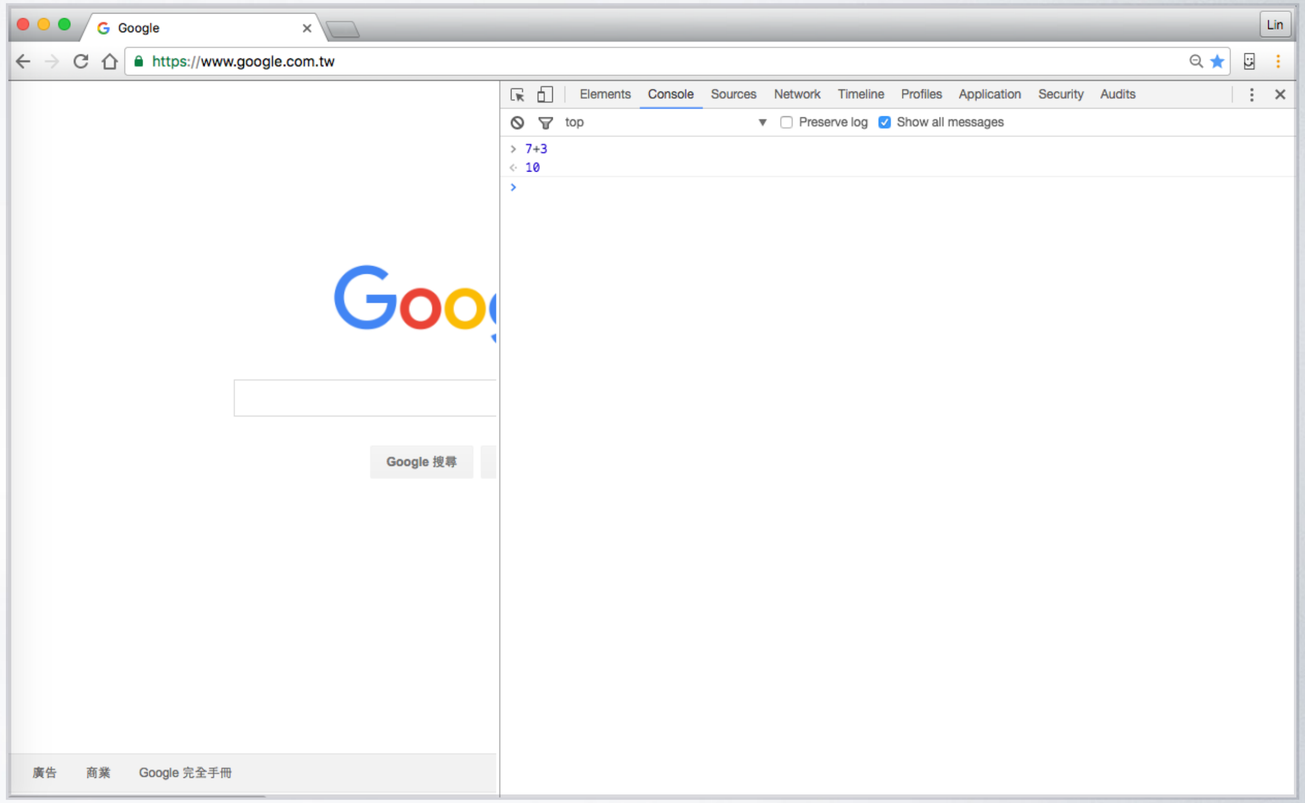
JavaScript

互動

JavaScript

- 原名LiveScript，但Java當時很紅(1995)，因此改名
- 網頁瀏覽器上的腳本語言(scripting language)
- 主要用來向HTML頁面提供互動行為

簡單的 JS 練習場
Chrome Console



網頁前端-基礎
JavaScript

LJW

JS 資料型態(type)

基本	複合
布林 (Boolean)	陣列 (Array)
數值 (Number)	函式 (Function)
字串 (String)	物件 (Object)

數值

基本算數

加法: 7+3
減法: 7-3
乘法: 7*3
除法: 7/3 = 2.33...
取餘數: 7%3 7 = 2*3+1

[餘數] 除不盡，剩下的數

打破先乘除後加減規則-
括號( ) 內先做:

7-3+3*2 = 10

(7-3+3)*2 = 14

7-(3+3)*2 = -5

7-(3+3*2) = -2

數值

數學方法

功能	寫法	結果
四捨五入	Math.round(7/3)	2
無條件進位(天花板)	Math.ceil(7/3)	3
直接捨去(地板)	Math.floor(7/3)	2
絕對值	Math.abs(-7)	7
隨機數(產生0至1之間的數)	Math.random()	0~0.99 任一數

0≤ Math.random() < 1

動腦時間

如何使用Math.random()
產生一個介於範圍
N~M之間的整數

1

2

動腦提示

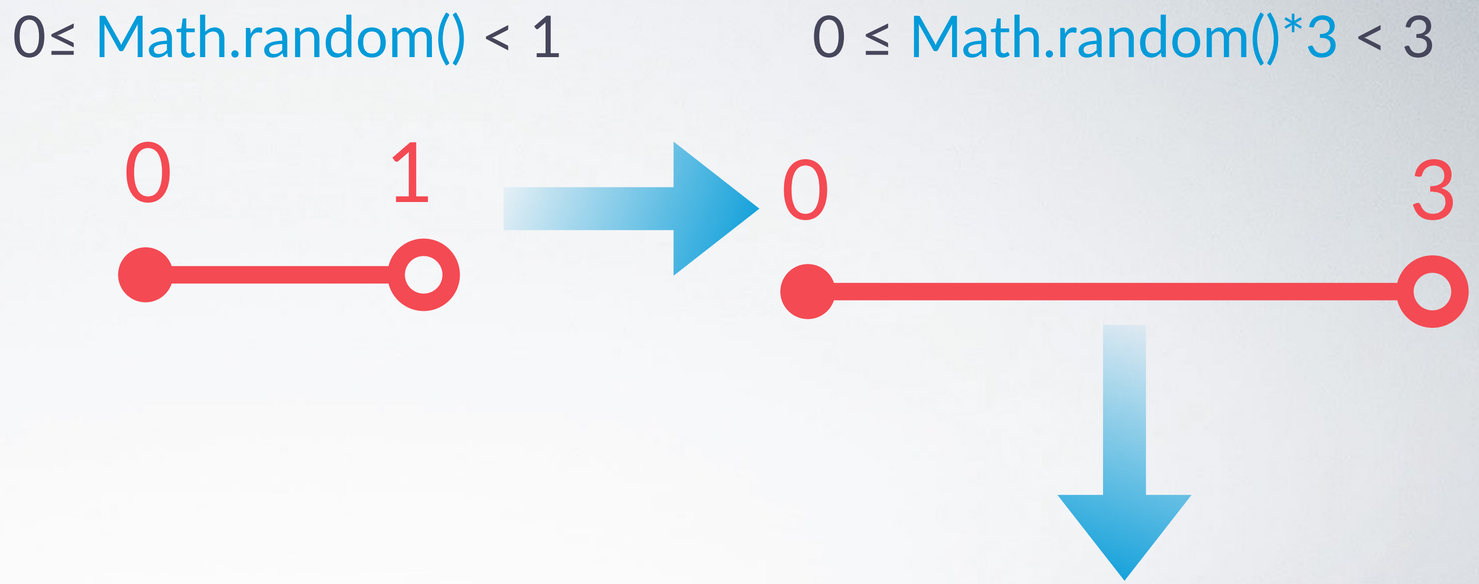
先看看如何產生1至3之間的數?

0≤ Math.random() < 1

0 ≤ Math.random()*? < 3

動腦提示

先看看如何產生1至3之間的數?



[再問] 用Math.ceil() 會發生什麼事？

布林 (對or錯兩值)

只有兩種情況之一：true, false

x 是一個「變數」(可自行命名)

順序	輸入	測試	結果
1	x = true	x	?
2		!x	?
3	x = !x	x	?
4		!x	?

! : 反轉 (對變成錯，錯變成對)

布林 (對or錯兩值)

只有兩種情況之一：true, false

合法變數命名:

1. 第一個字只能是英文或底線(_)
2. 第二個後可為英文、數字、底線、錢字號($)
3. 不可以使用資料型態命名(ex. number)

順序	輸入	測試	結果
1	x = true	x	true
2		!x	false
3	x = !x	x	false
4		!x	true

哪些才是
合法的變數名稱？

合法變數命名:

1. **第一個字只能是英文或底線(_)**
2. **第二個後可為英文、數字、底線、錢字號($)**
3. **不可以使用資料型態命名(ex. number)**

順序	輸入	合法嗎	解釋
1	my_Var1		
2	my Var2		
3	my@Var3		
4	4myVar		
5	my$Var5		
6	_myVar6		

哪些才是
合法的變數名稱？

合法變數命名:

1. **第一個字只能是英文或底線(_)**
2. **第二個後可為英文、數字、底線、錢字號($)**
3. **不可以使用資料型態命名(ex. number)**

順序	輸入	合法嗎	解釋
1	my_Var1	V	無話可說
2	my Var2		空白非法字元
3	my@Var3		@非法字元
4	4myVar		數字不能當首字
5	my$Var5	V	$除了首字，皆可
6	_myVar6	V	_都可

字串(多個字元)

'JS' + '好好玩'

"JS" + "好好玩"

x = "JS好好玩"

0	1	2	3	4
J	S	好	好	玩

字串操作

功能	寫法	結果
字串長度	x.length	5
取字元	x[0]	"J"
取子字串(位置, [長度])	x.substring(2,3)	"好好玩"
	x.substring(2)	"好好玩"
索引位置(字串)	x.indexOf('好好')	2
	x.indexOf('學')	-1
後方索引位置(字串)	x.lastIndexOf('好好')	3