

# 01. Intro to Web Programming

Ric Huang / NTUEE

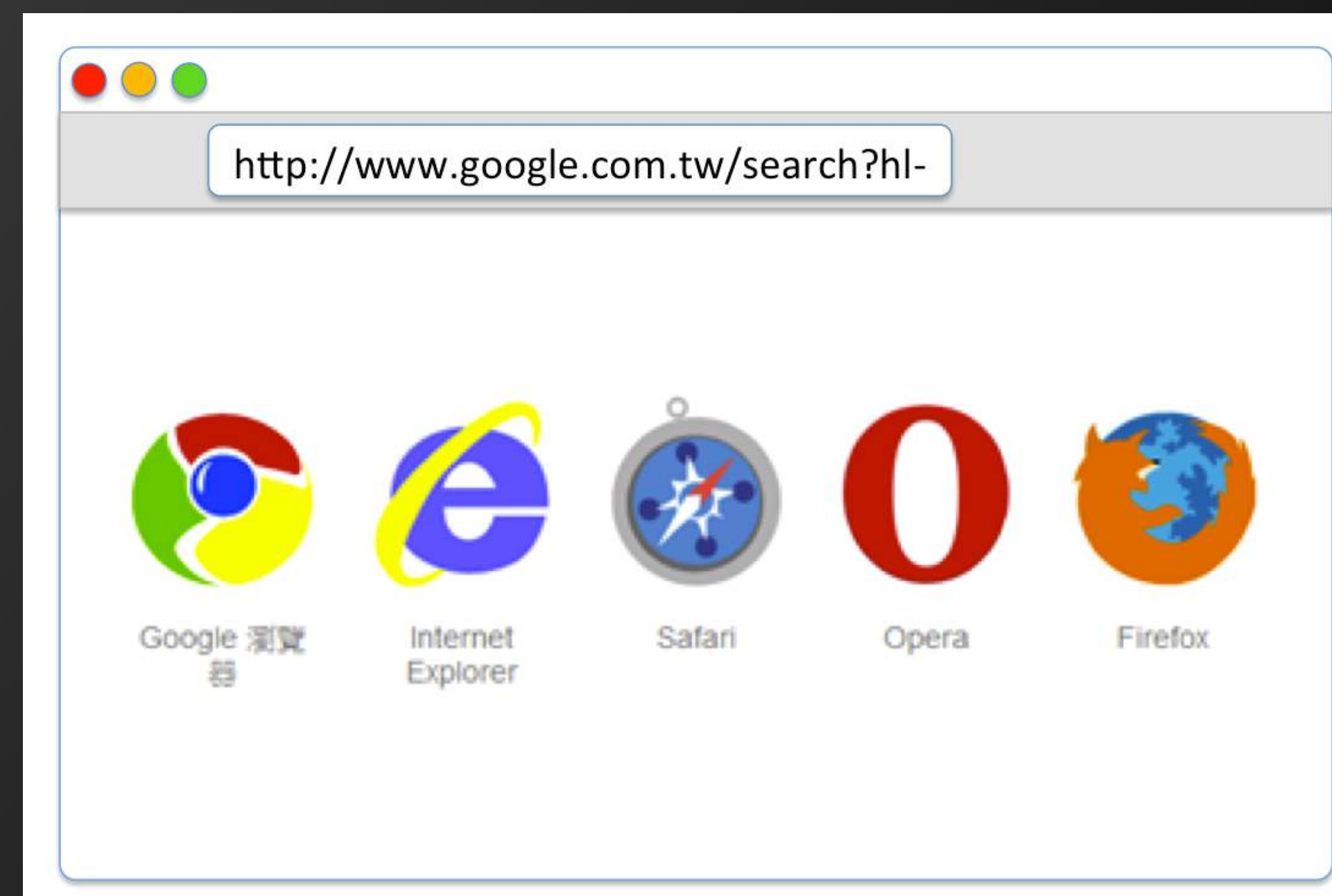
(EE 3035) Web Programming

# Objectives of this Lecture Note

- Get to know the history of web services
- Equipped with the background knowledge on web service development
- Given a webpage design, able to create a **static** webpage using HTML and CSS
- Be able to work on HW#1

# Web • Programming

- 顧名思義，就是提供各式各樣「網路服務」的程式
- 對於使用者而言，可以透過瀏覽器(Web Browser)、APP、或是特定裝置上的軟體獲得服務
- 簡化起見，本課程將focus 在透過瀏覽器所產生的網路服務





## 前端。後端

- 顧名思義，前端就是在前面利用各種介面(UI)設計、服務使用者的程式，而後端就是在後面負責各種(較為消耗資源的)運算與儲存、回傳給前端服務資料的程式
- 也許更精確的講法是：  
client-side (客戶端) vs.  
server-side (伺服器端)





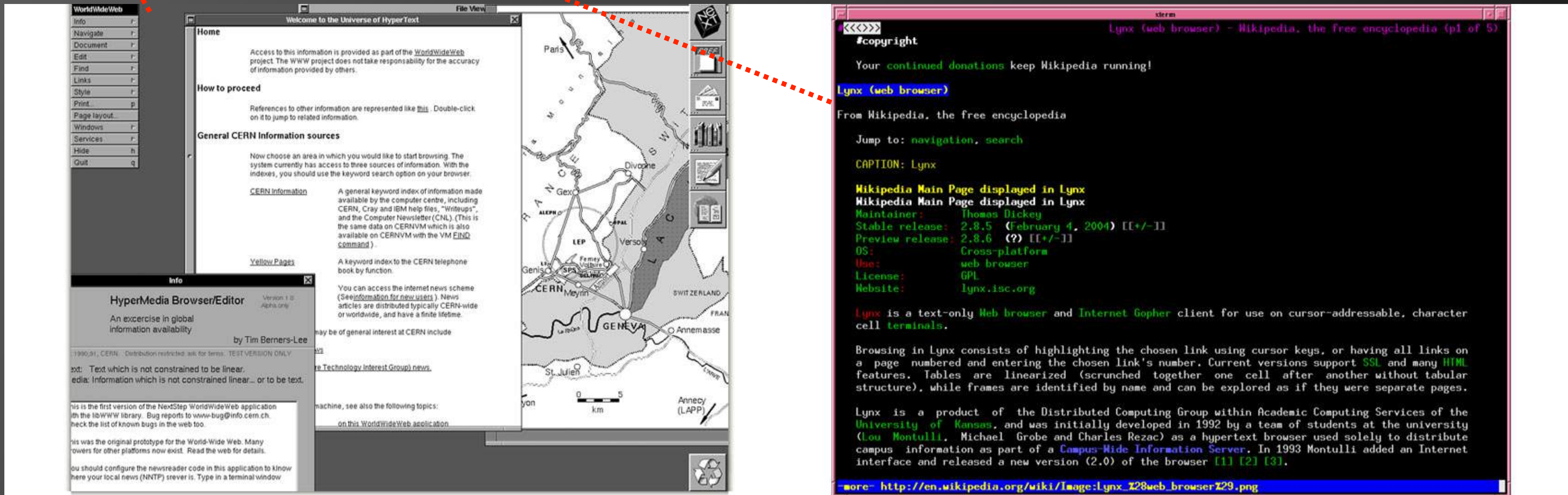
1990

2000

2010

2020

# 前端。瀏覽器 (Web Browser)



First web browser:  
WorldWideWeb (aka. Nexus),  
by Tim Berners Lee, 1990

Lynx, a text-based browser, by  
a team of students and staff at  
the university (of Kansas), 1992  
(<http://lynx.browser.org/>)



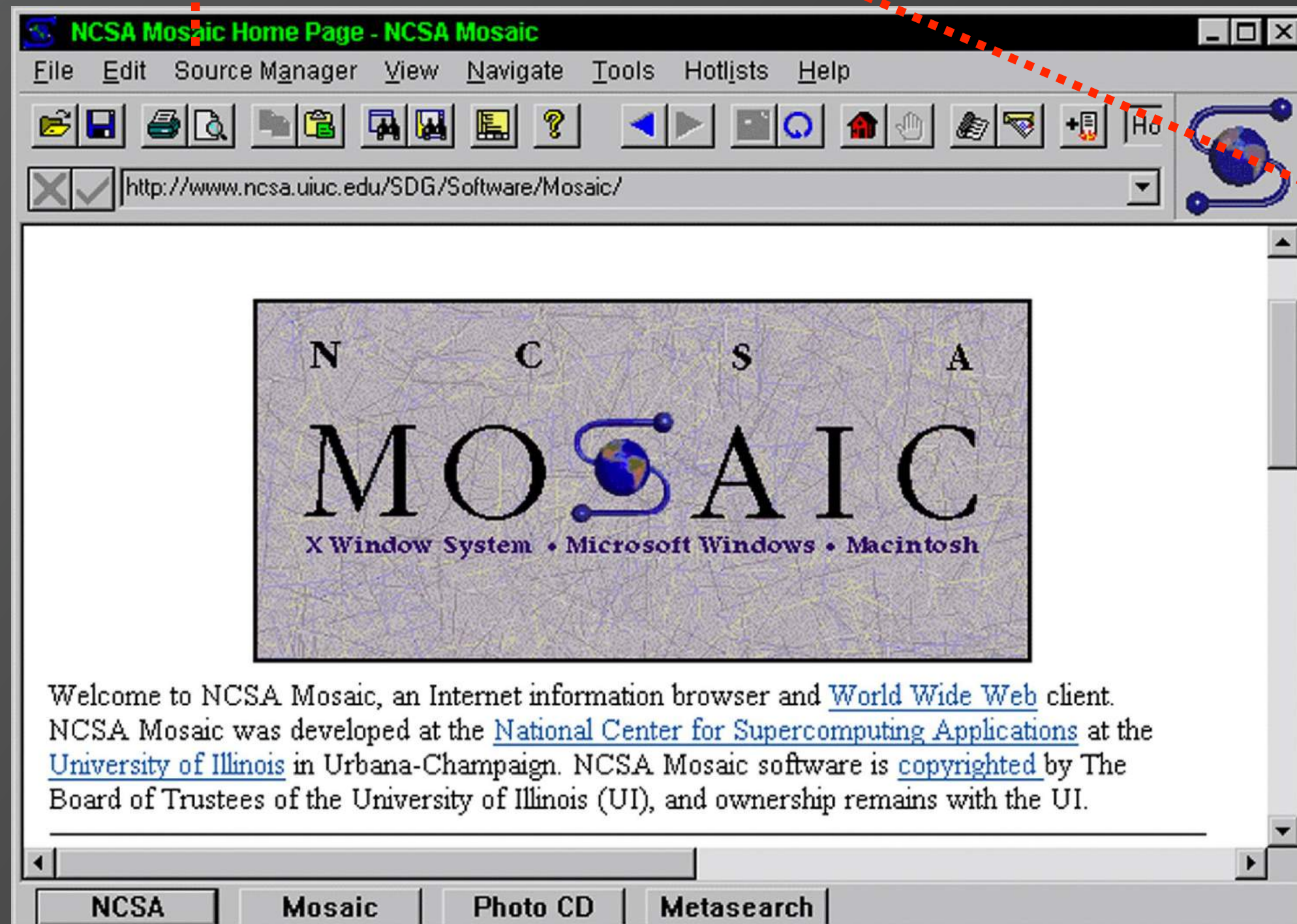
1990

2000

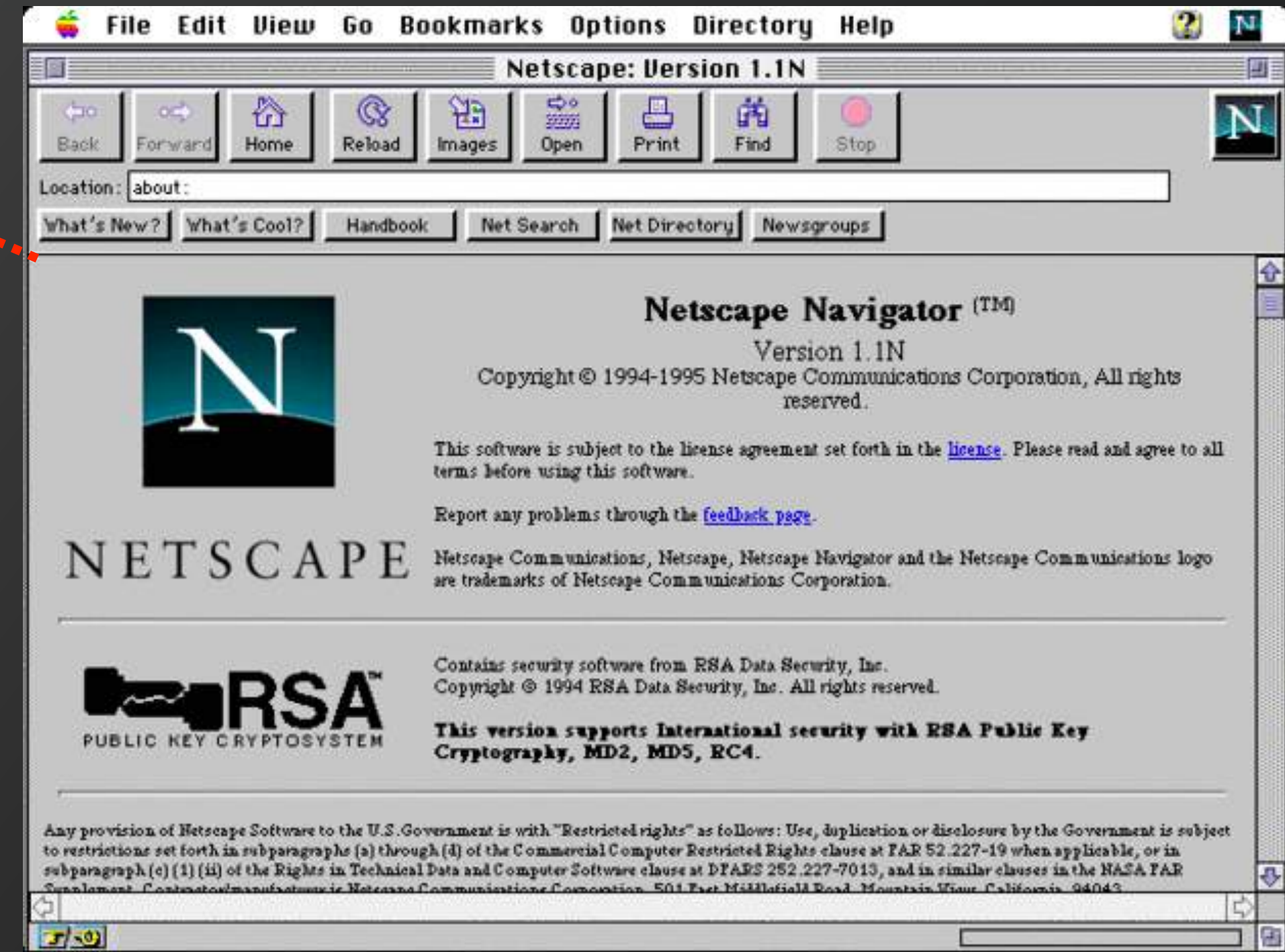
2010

2020

# 前端。瀏覽器 (Web Browser)



Mosaic, often regarded as the first graphical web browser, was developed by National Center for Supercomputing Applications (NCSA) of UIUC, in late 1992 (Led by Marc Andreessen)



Netscape, founded by Marc Andreessen, James H. Clark, and 4 others from Mosaic, was the most successful commercial web company in 90s



1990

2000

2010

2020

# 前端。瀏覽器 (Web Browser)



Internet Explorer (IE), by Microsoft, in 1995



Opera, by Telenor, in 1996



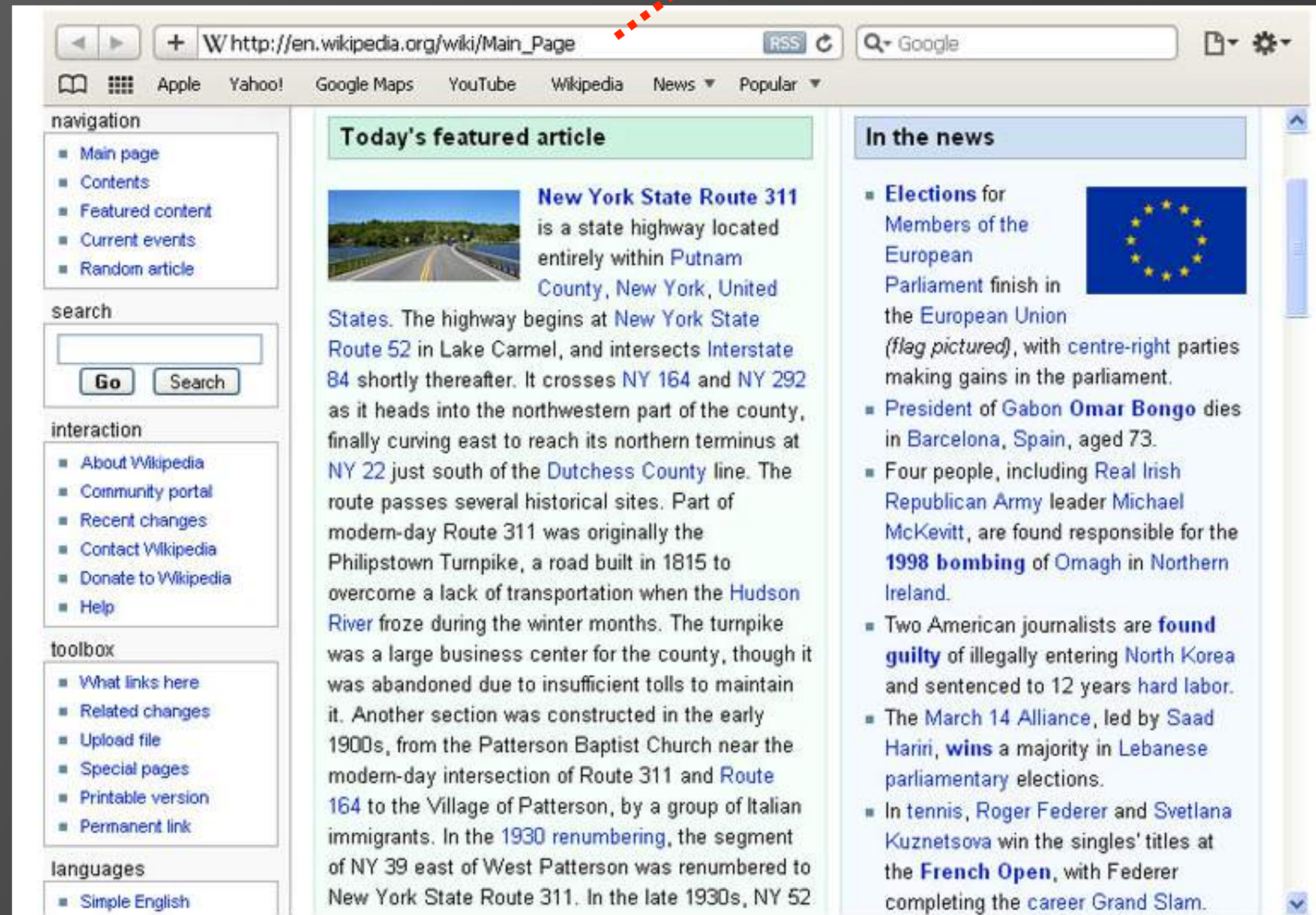
1990

2000

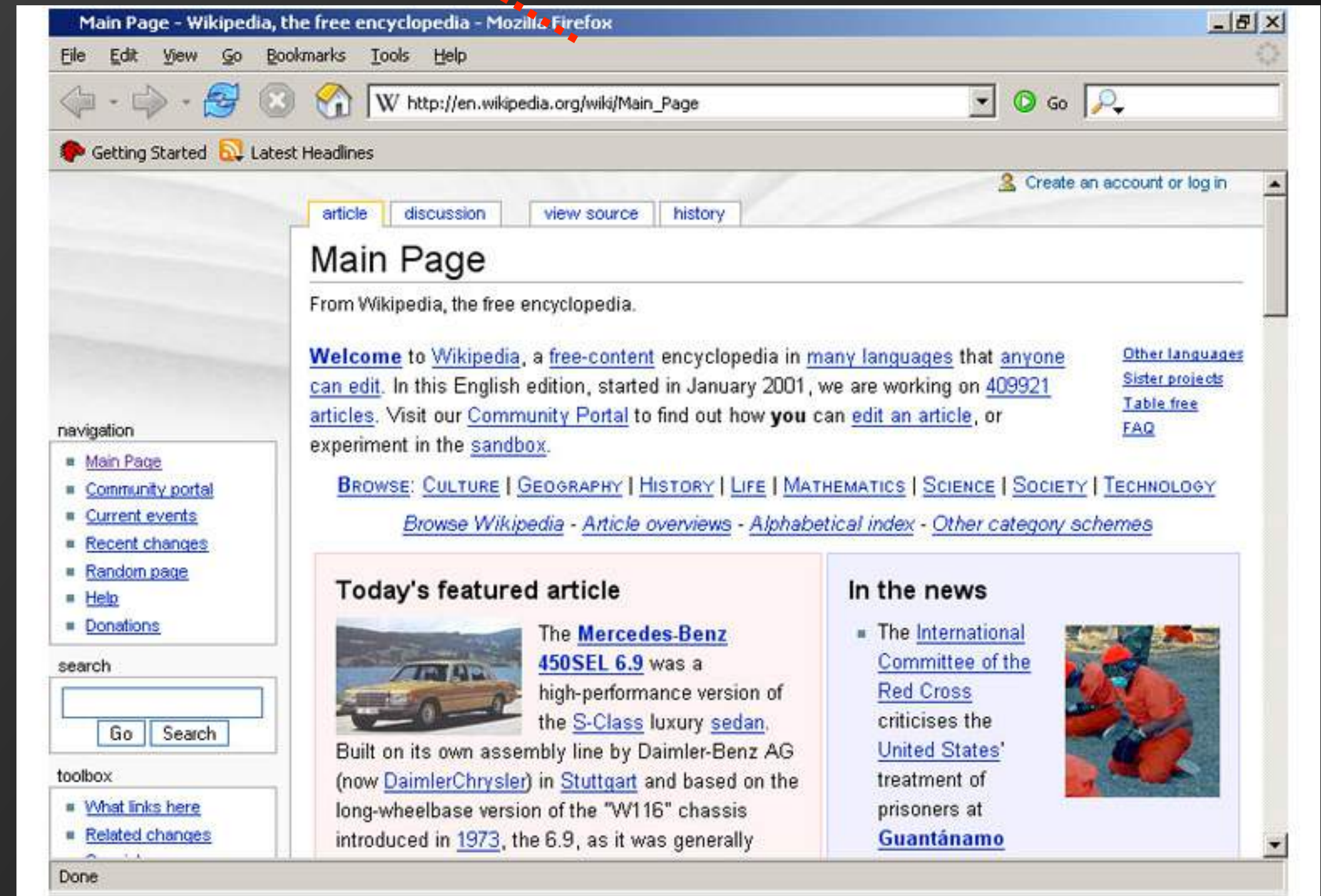
2010

2020

# 前端。瀏覽器 (Web Browser)



Safari, by Apple, in 2003



Firefox, by Mozilla, in 2004



1990

2000

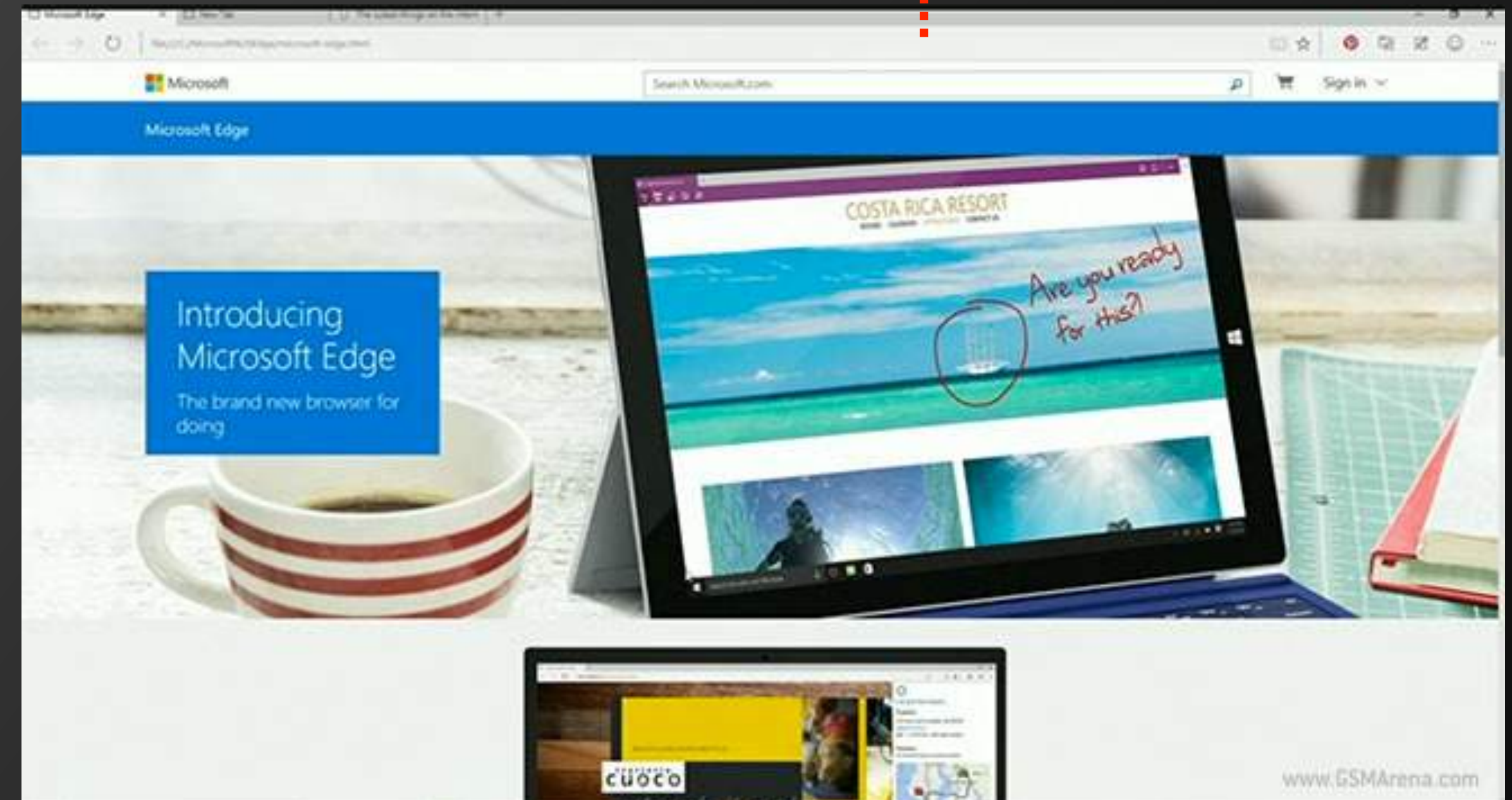
2010

2020

# 前端。瀏覽器 (Web Browser)



Chrome, by Google, in 2008



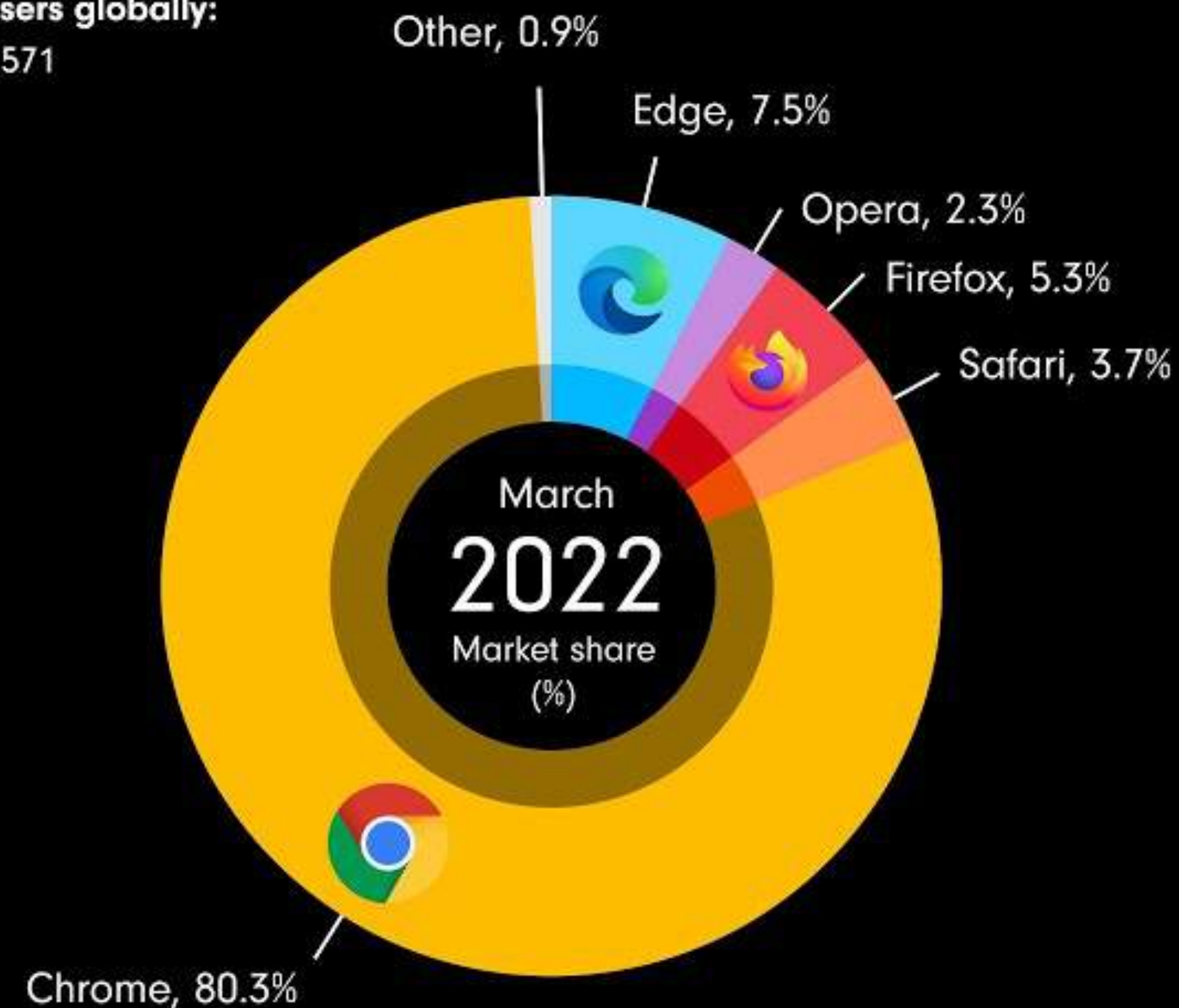
Edge, by Microsoft, in 2015



# 前端。瀏覽器 (Web Browser)

## Web browsers for the last 28 years

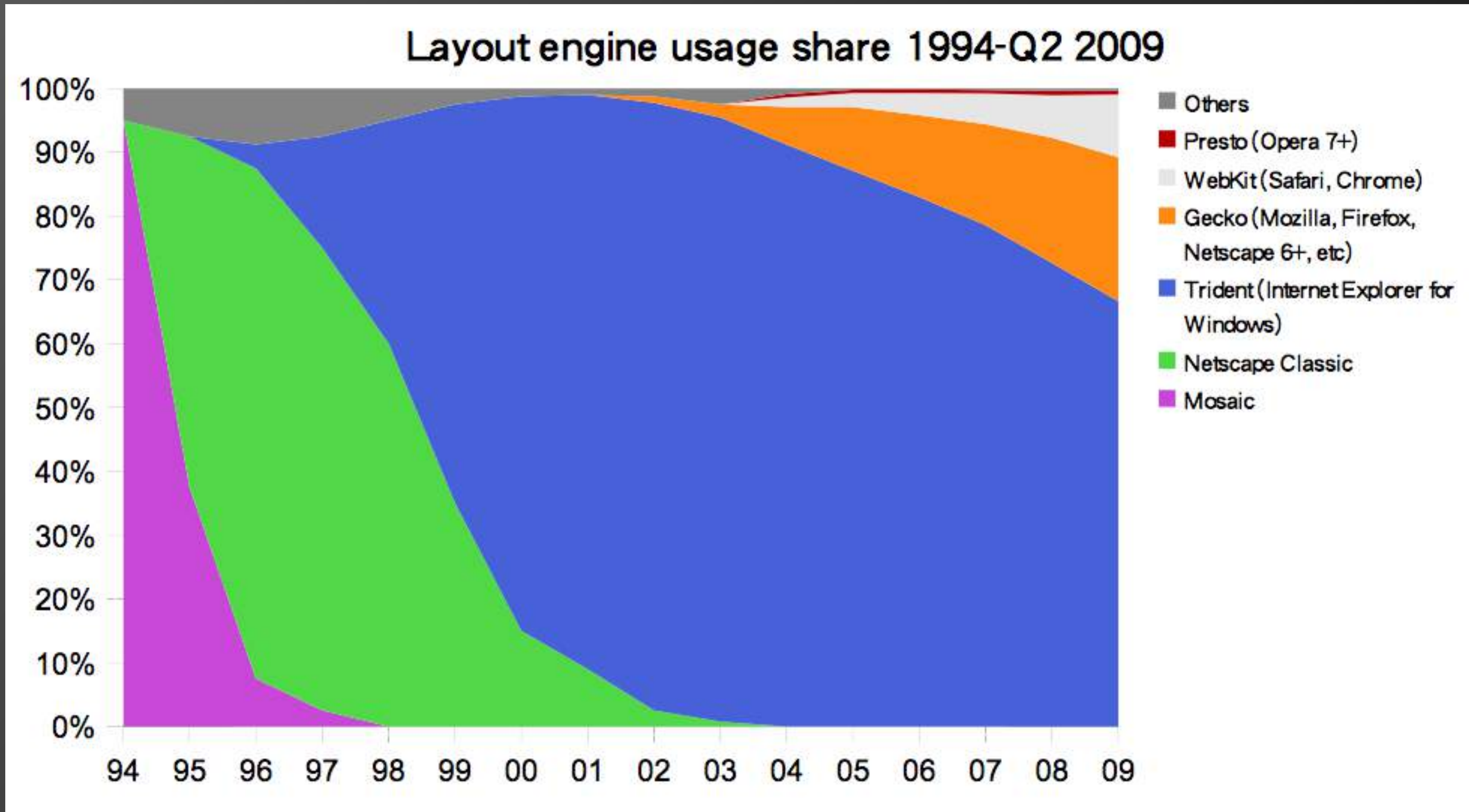
Internet users globally:  
4,878,428,571  
(61.9%)



Sources:  
W3Schools (Jul-99 to present)  
WebSideStory (Feb-99 to Jun-06)  
GVU WWW user survey (Jan-94 to Oct-98)  
EWS Web Server at UIUC (Jun-96 to Dec-98)

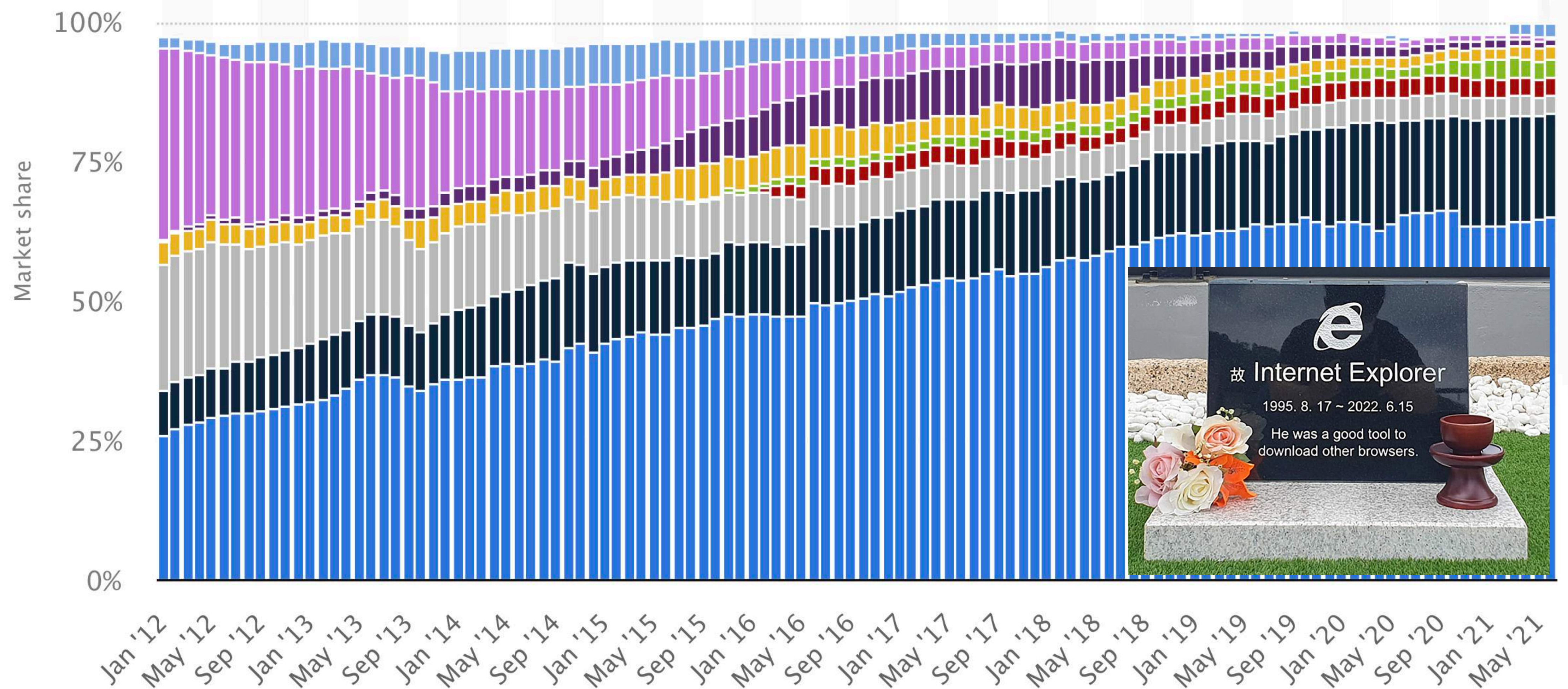


# 前端。瀏覽器 (Web Browser)





# 前端。瀏覽器 (Web Browser)





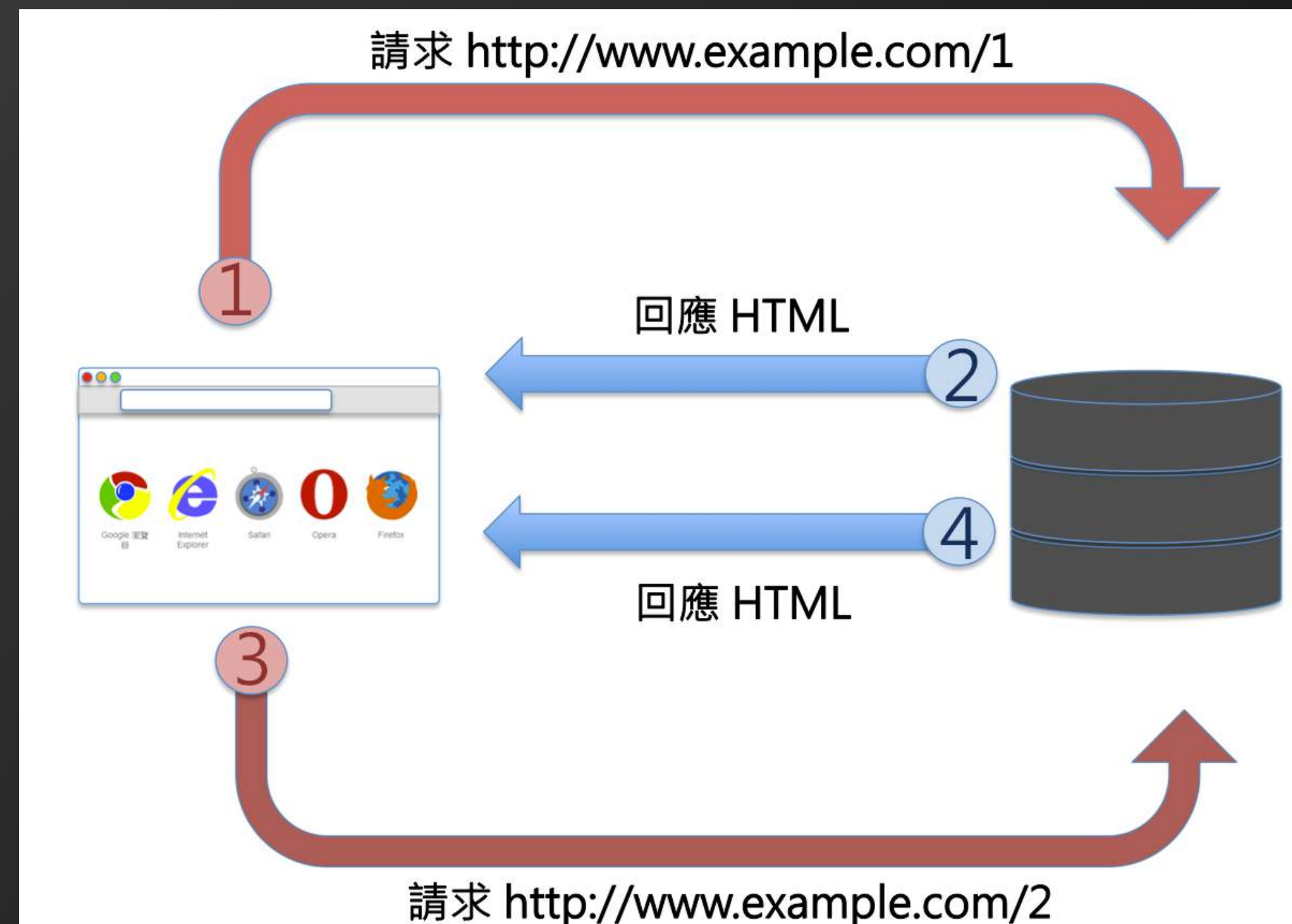
這二十幾年來網頁有許多的改變：<https://archive.org/>

- Web 1.0 (90's ~ early 00's)
  - 文字 + 圖片 + 超連結 (HyperText Markup Language, HTML)
  - 各種小花招 (跑馬燈、計數器、奇怪的游標...)
  - 開始有搜尋功能
- Web 2.0 (early 00's ~ now)
  - wikipedia.org、各種論壇/部落格
  - 各種需要會員登入的網站
  - e-commerce
  - Social network



## 網頁。靜態網頁

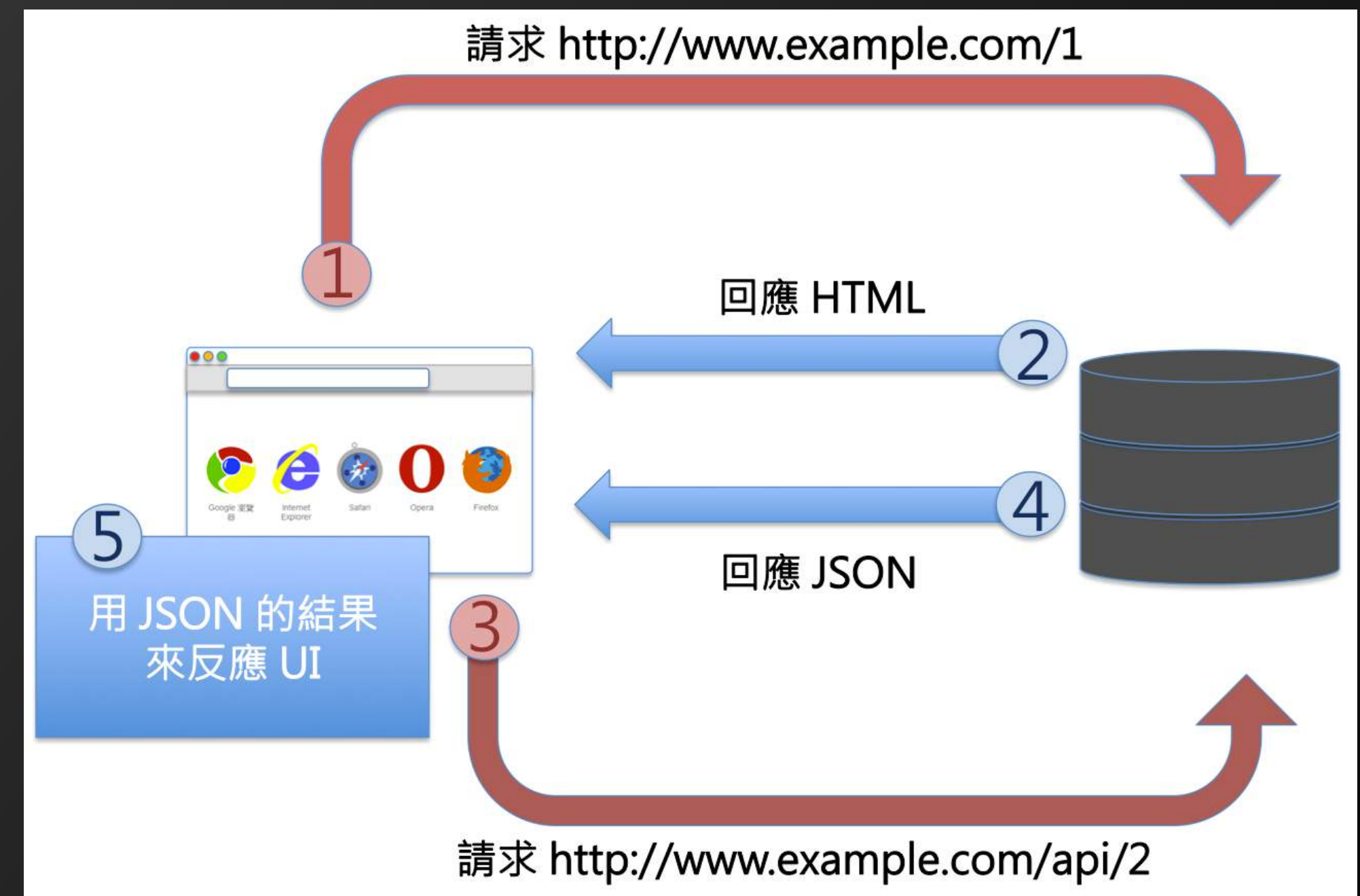
- 瀏覽器打開某網址，而從該網址的 web server 回傳 HTML 檔，show 在瀏覽器上，網頁的內容不會隨著使用著的瀏覽行為而有所改變
- 如果按下超連結或是提交表單，是跳到別的網址，網頁內容會重新下載
- 換頁時會有「白畫面」





## 網頁。動態網頁

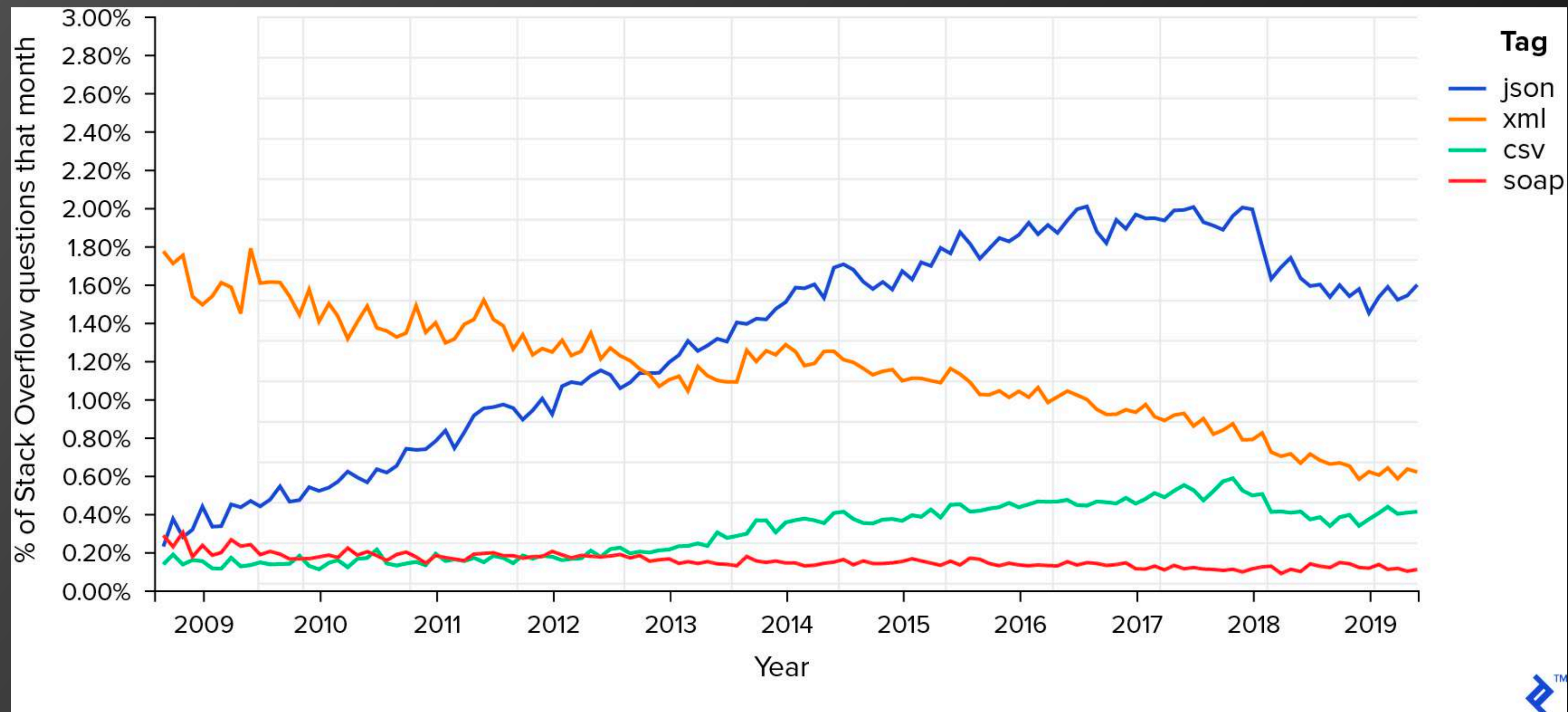
- 網頁內嵌會聽取(listen)某些事件(events, 如：滑鼠點擊、移動，鍵盤輸入，登入...等)綁定的腳本語言(script language, 如：JavaScript)，當事件發生時，瀏覽器會根據腳本的描述去伺服器(i.e. web server)與資料庫抓取資料，回傳到瀏覽器，而更動「部分」的網頁內容
- 計數器：登入時人數加一
- 相簿：按下按鈕可以切換相片
- 搜尋框：輸入文字可以搜尋資料
- 會員登入：根據會員資料顯示不同內容
- 社群網路：朋友互動的即時顯示





# Ajax: Asynchronous JavaScript and XML

- Ajax 透過**非同步**的方式跟後台拿資料，在這中間使用者還是可以進行其他操作，不會被打斷
- 雖然叫 Ajax, 但現代的網頁都用 JSON





# Ajax: Asynchronous JavaScript and XML

- XML vs. JSON

```
<?xml version="1.0" encoding="iso-8859-1" ?>
- <department>
-   <employee>
      <name>John Doe</name>
      <job>Software Analyst</job>
      <salary>2000</salary>
    </employee>
-   <employee>
      <name>Jane Fletcher</name>
      <job>Designer</job>
      <salary>2500</salary>
    </employee>
  </department>
```

```
{
  "arguments" : { "number" : 10 },
  "url" : "http://localhost:8080/restty-tester/collection",
  "method" : "POST",
  "header" : {
    "Content-Type" : "application/json"
  },
  "body" : [
    {
      "id" : 0,
      "name" : "name 0",
      "description" : "description 0"
    },
    {
      "id" : 1,
      "name" : "name 1",
      "description" : "description 1"
    }
  ],
  "output" : "json"
}
```

- 前後端的分工逐漸明確

- 前端：JavaScript 來負責顯示畫面 (client-side rendering)
- 後端：as an API server, 負責透過定義好的介面來產生資料
- 框架：使用Ajax，並定義一些 MVX (如：MVC) 的架構



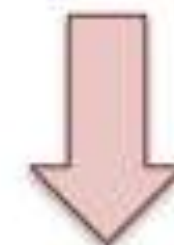
## 前端。Client-side Rendering 的演進

- 從 2010 年開始，風向一直在變

 BACKBONE.JS



 ANGULARJS  
by Google



 React



# React.js ◦ 元件式開發 + 狀態決定 UI

## Component-based implementation with states

- 以前：某個 UI 呈現出來的 bug 可能要數個操作步驟來重現，debug 過程可能會讓你很崩潰
- With React.js components/states：

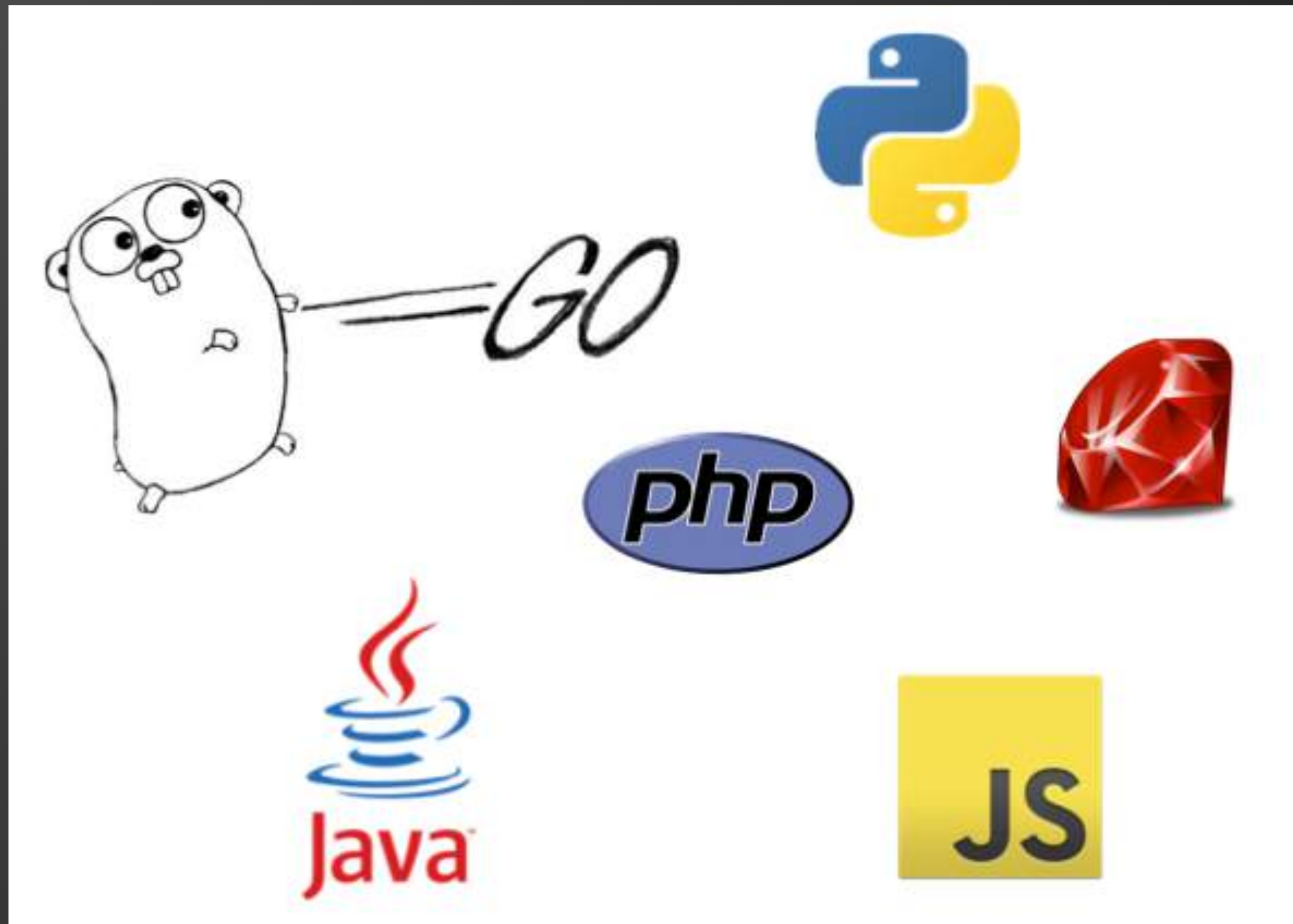
```
let AppUI = AppComponent(state);
```

- ▶ 讓資料儲存在單一地方 (i.e. single source of truth)，所有的改變會立即反應在 component 上面
- ▶ 除了 debug 較為輕鬆之外，頁面更新的速度也可以變快 (more to cover later)



## 後端語言更是百花齊放～

- 可以跑在伺服器（電腦）的程式語言都可以用



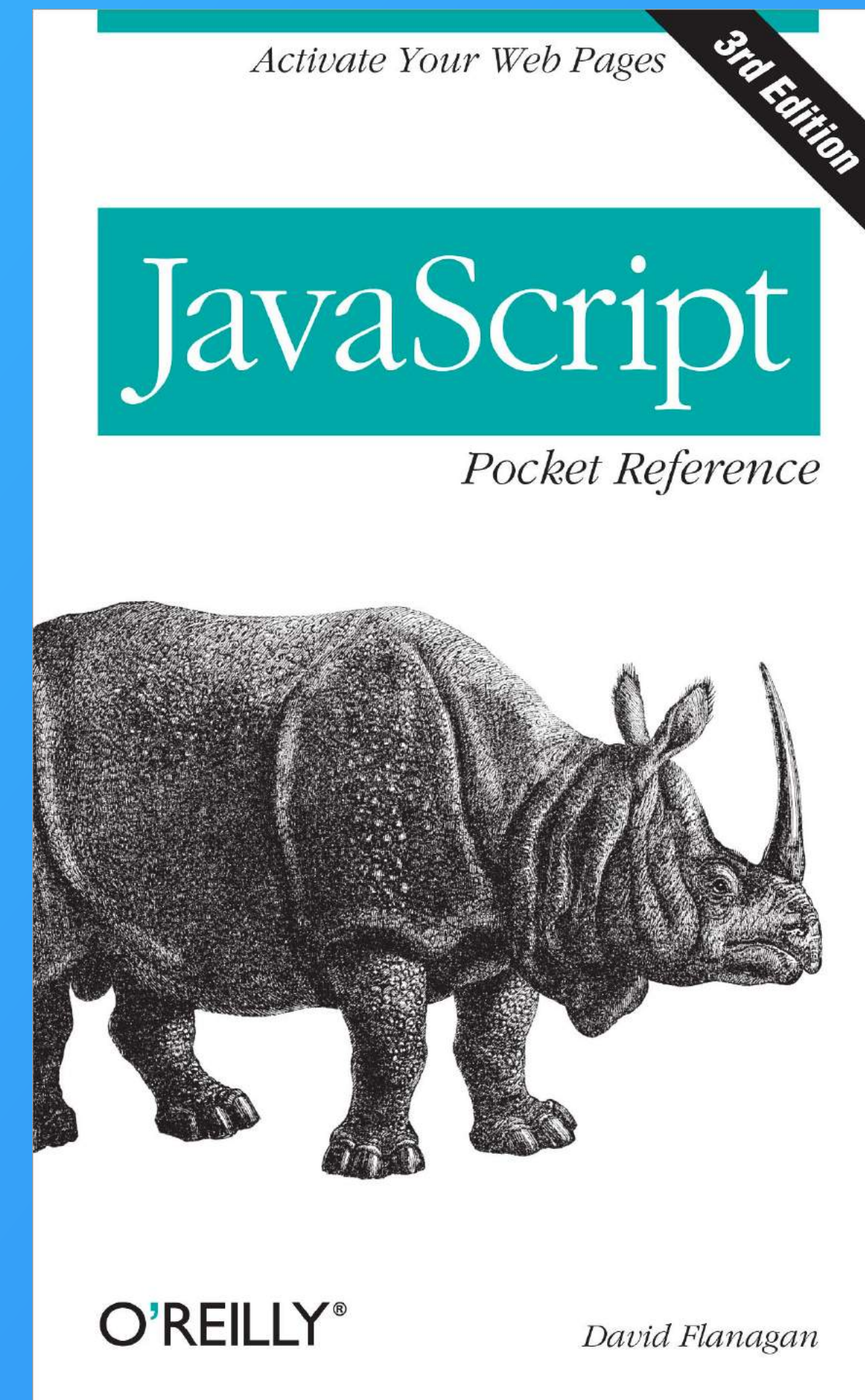
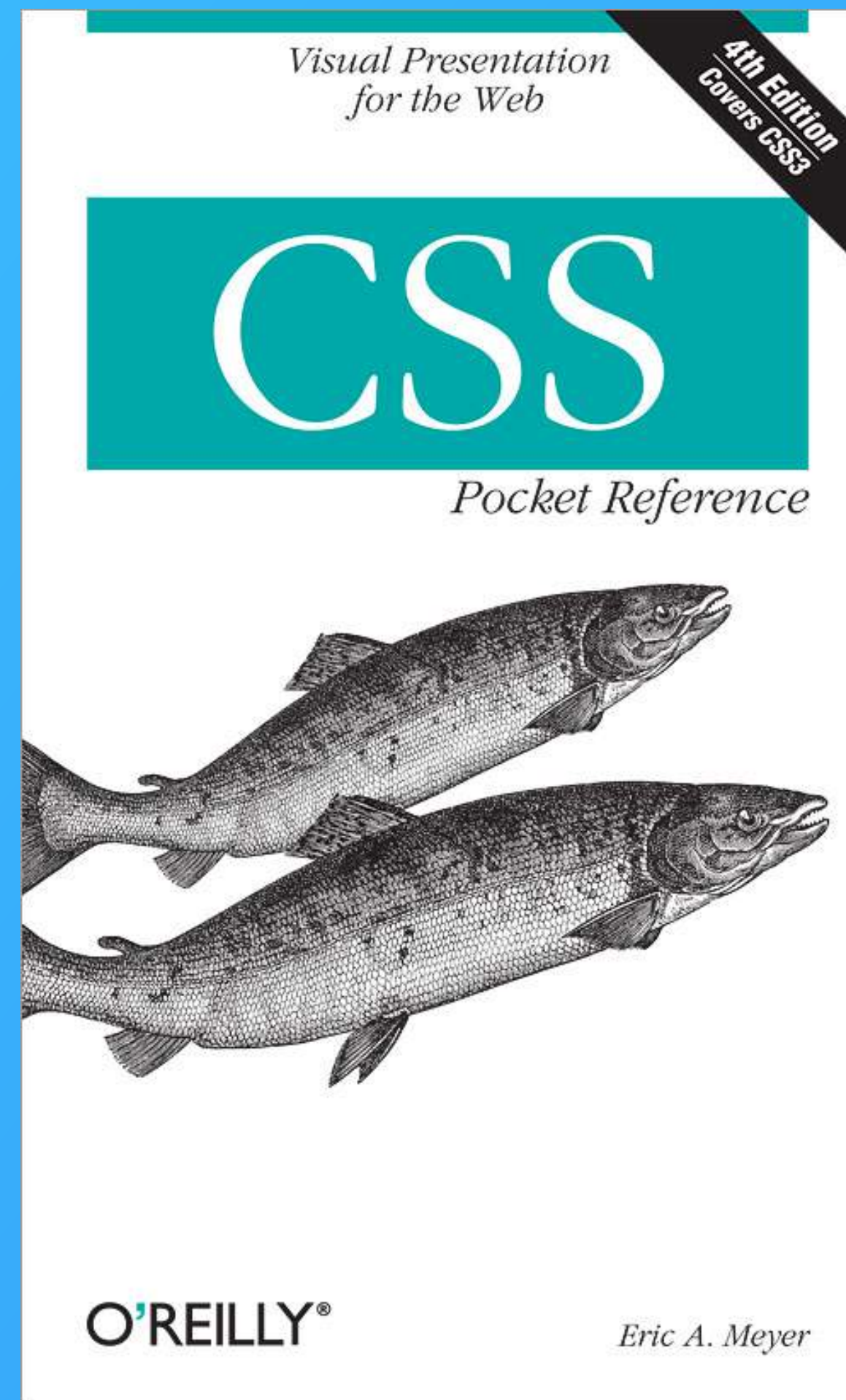
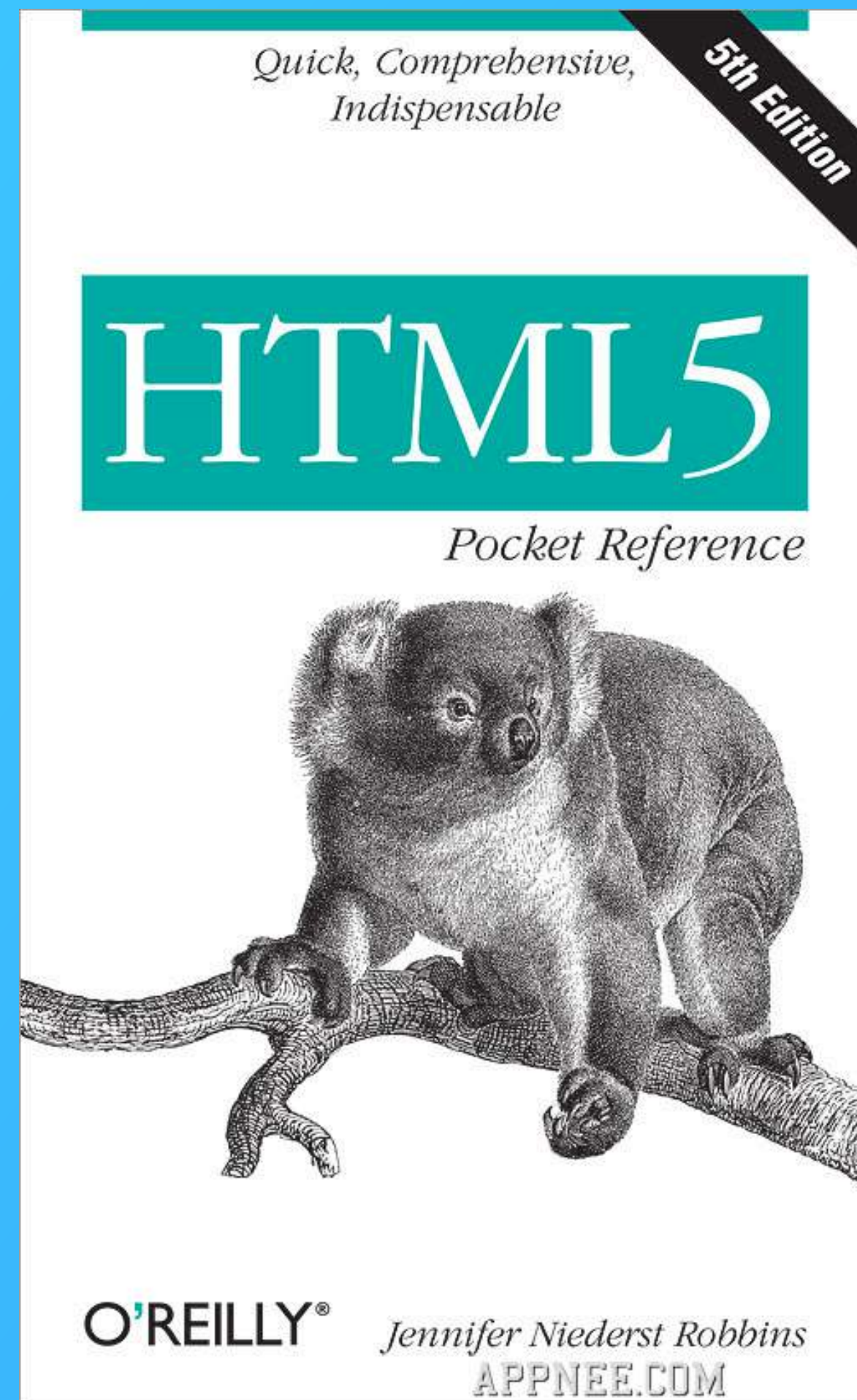


That's all for the history  
and background.

接下來，我們要學習如何用  
HTML & CSS 來製作一個  
「靜態網頁」



# HTML/CSS/~~JavaScript~~ Basics

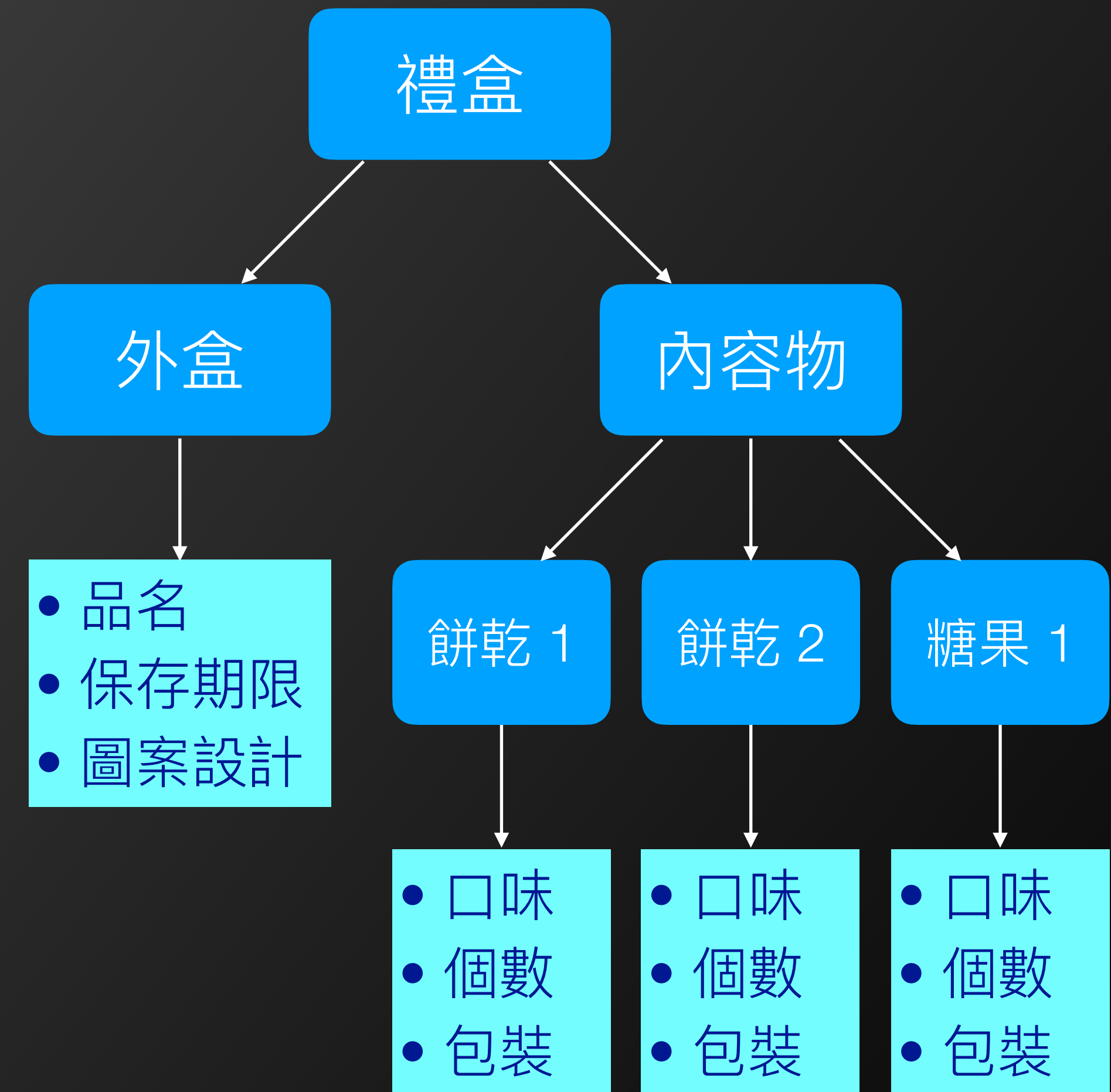




雖然接下來我會講 HTML & CSS  
但我期待各位同學這部分  
either 已經有一定的基礎  
or 在課後盡快找時間自學起來



# 網頁。禮盒 (類比)





# 網頁。禮盒 (類比)

todos

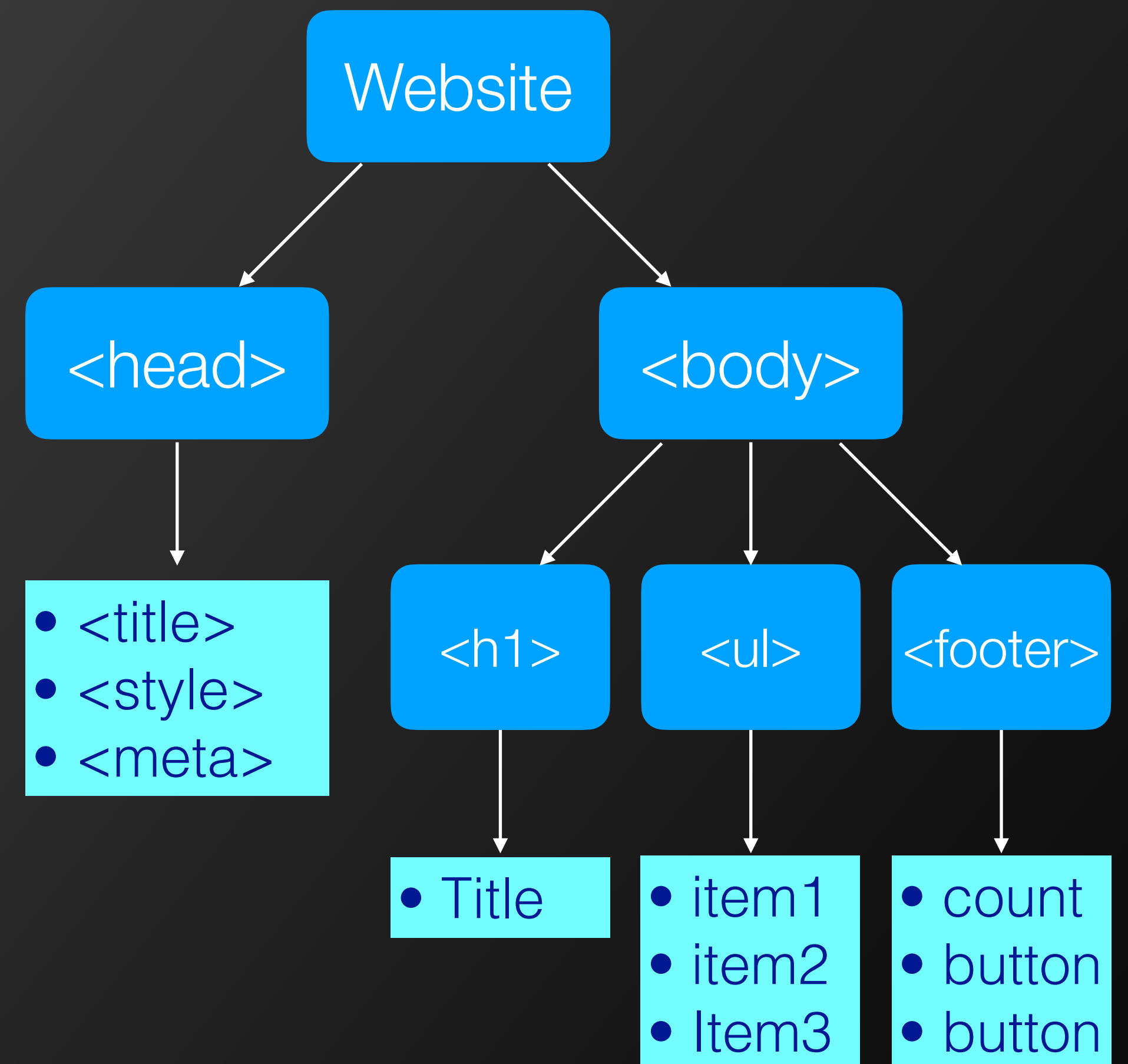
What needs to be done?

- ☐ This is the first TODO.
- ☒ This is the second TODO.
- ☐ This is the third TODO. ⓧ

2 left

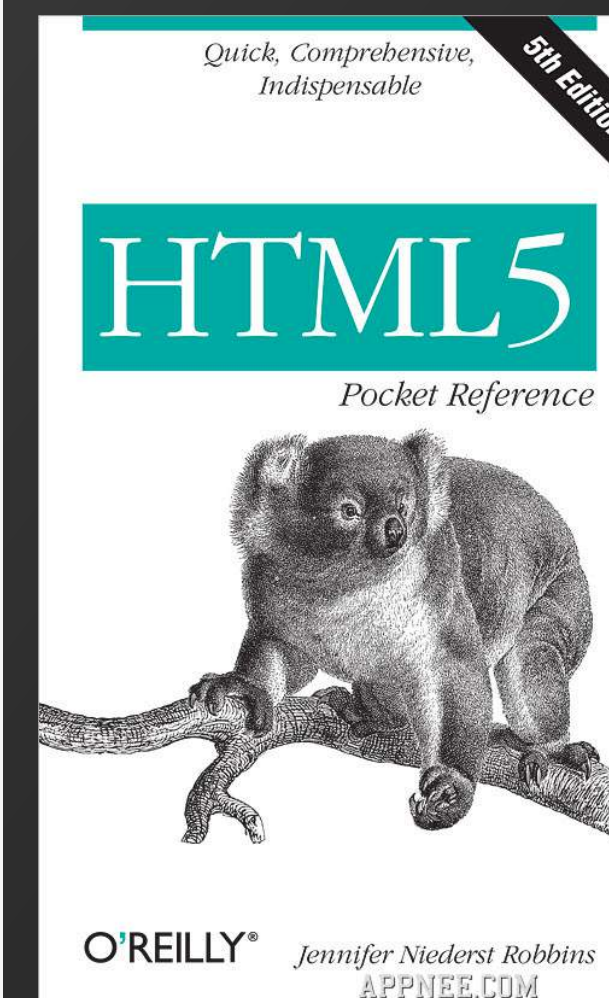
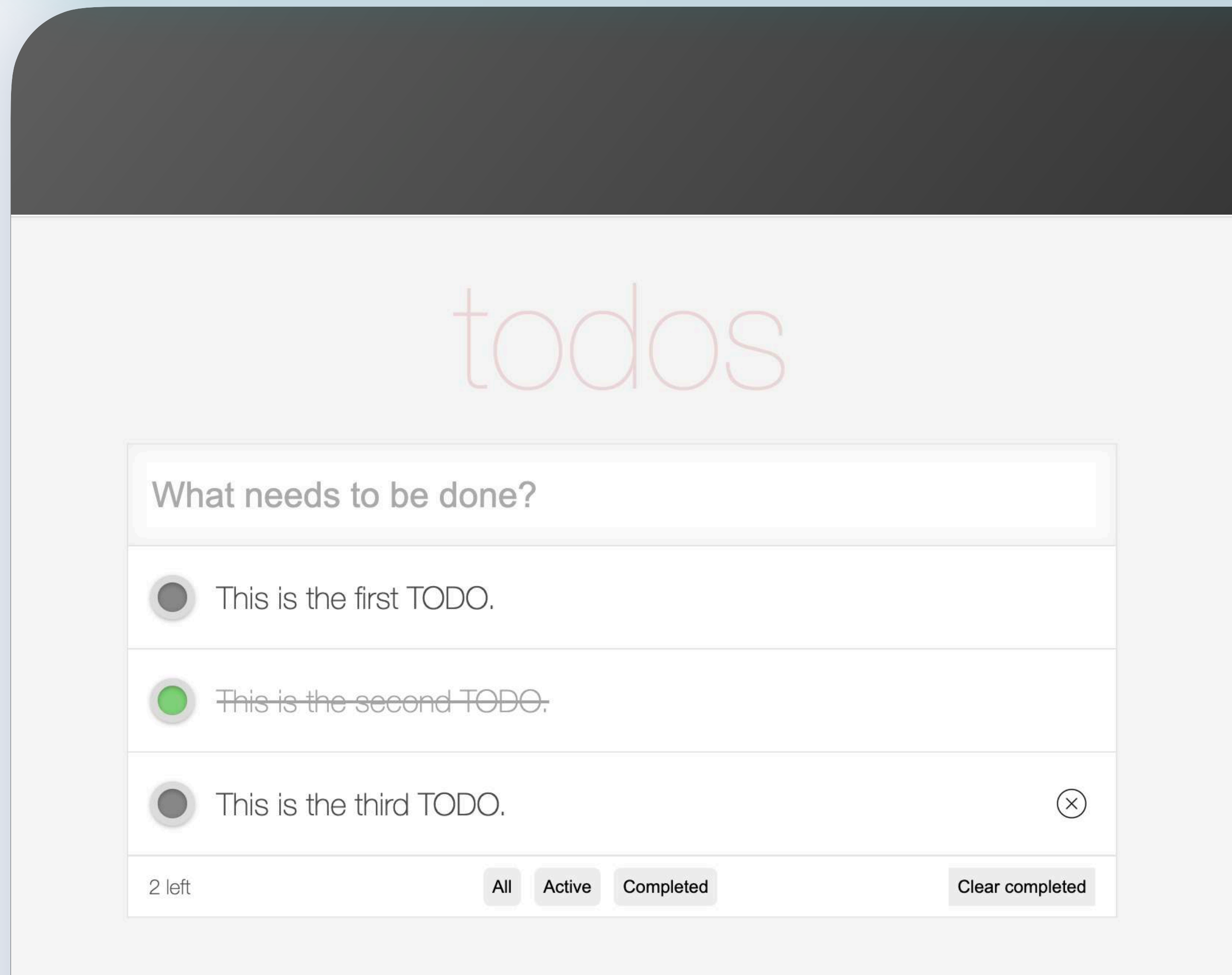
All Active Completed

Clear completed

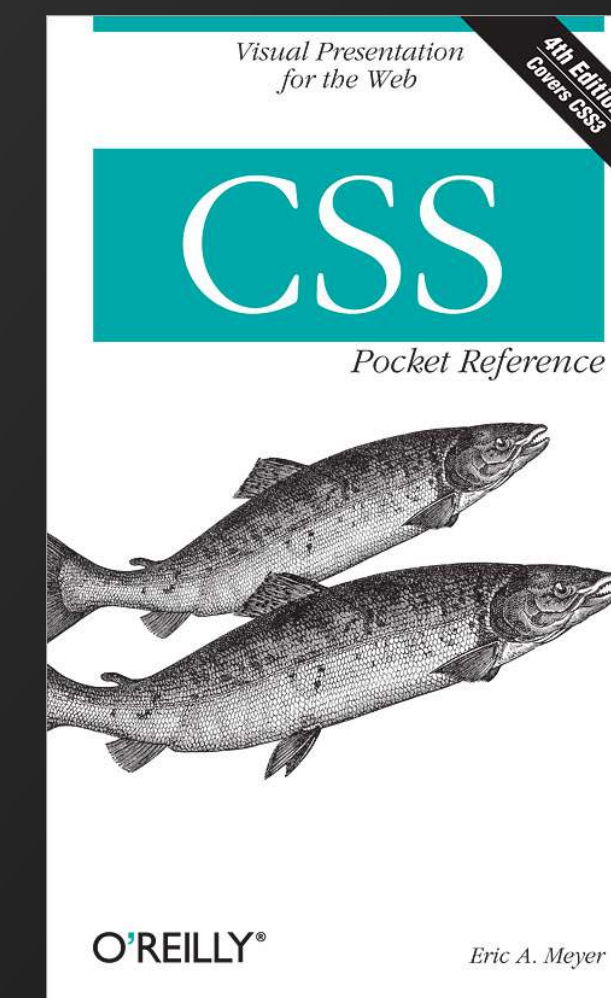




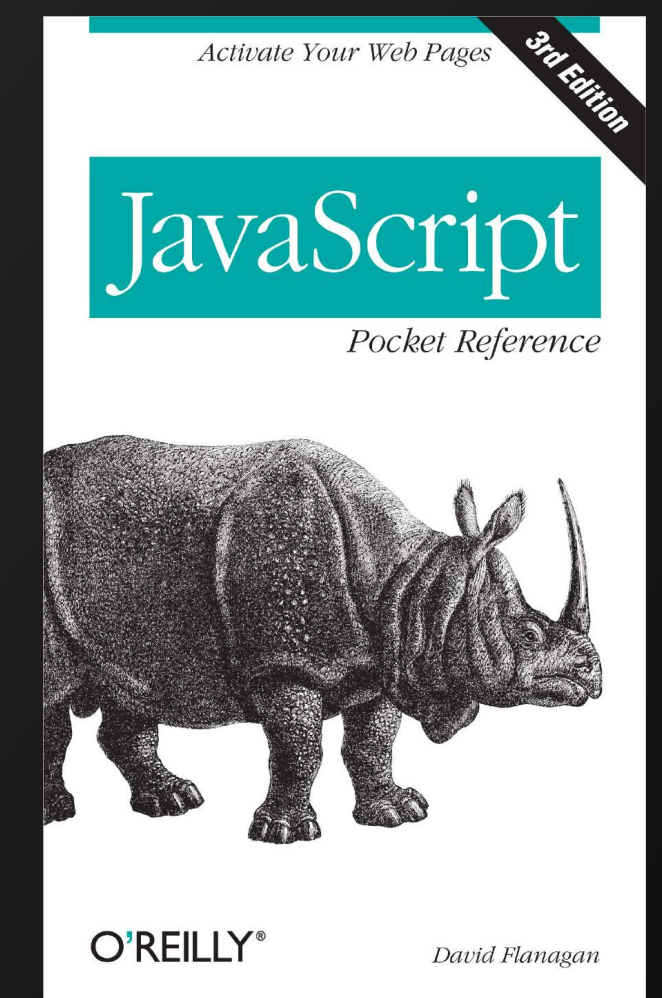
# 網頁。禮盒 (類比)



網頁框架、  
內容



風格外觀  
排版

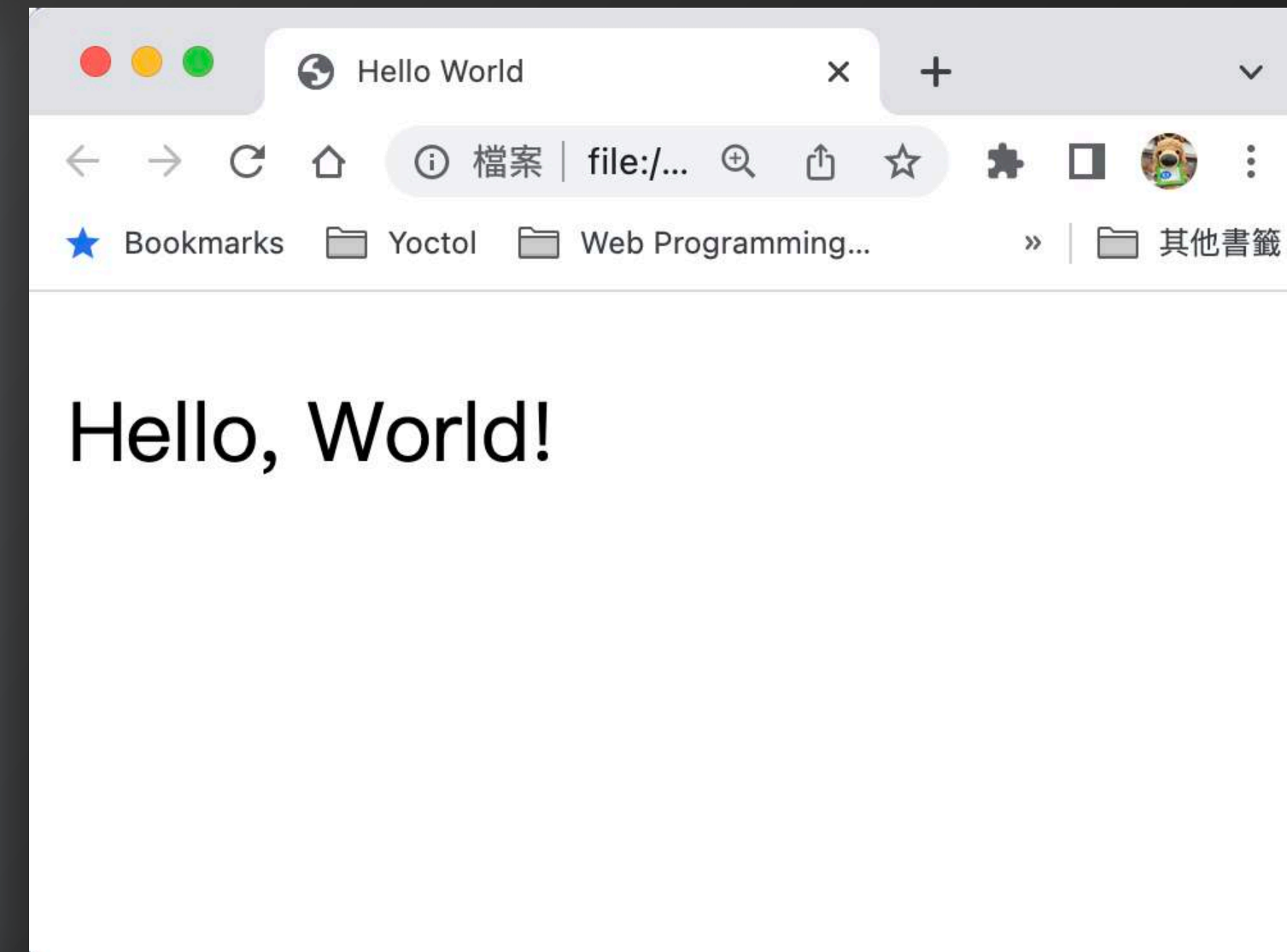


動態行為  
功能

# HTML (HyperText Markup Language)

- A "Hello World" example

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <title>Hello World</title>
  </head>
  <body>
    <p>Hello, World!</p>
  </body>
</html>
```



- Try...
  - 在 "Hello World" 前後或是中間加入空白或是換行
  - 試試看 <div>, <span> 等其他 text elements



# HTML Document Structure

## **<html> — Main root**

### **<head> — Document metadata (optional)**

- **<link>**: to specify the external resource, especially link to style sheets (CSS)
- **<style>**: style information for a document
- **<title>**: title shown in a browser's title bar
- **<meta>**: other meta data

### **<body> — Sectioning root**

- Various tags to define the document layout



# HTML — Tag-based Syntax

## 【 Tags 】

- For most of the tags...

```
<tag attribute1="value" attribute2="value">
```

```
Some text to display </tag>
```

- Some exceptions (no closing tag)

```

```

- Can use ">" to close if no text is displayed

```
<input placeholder="Enter text here..." />
```

## 【 字串(value) 】

- 可以用單引號，也可以用雙引號（避免混淆就好）

## 【 Comments 】

- are in <!-- ... -->



# HTML Elements

## Content sectioning

To organize the document content into logical pieces.

- `<h1>...<h6>`
- `<header>`, `<footer>`
- `<article>`, `<address>`, `<nav>`, `<section>`

## Text content

To organize blocks or sections of content in `<body>...</body>`

- `<p>`, `<div>`
- `<ol>`, `<ul>`: ordered, unordered list
  - `<li>`: list items
- `<dl>`: description list
  - `<dt>`: a term in `<dl>`
  - `<dd>`: details for the term `<dt>`
- `<figure>`, `<figcaption>`, `<blockquote>`
- `<hr>`



# HTML Elements

## Inline text semantics

- `<a href="...">`: hyper link
- `<span>`
- `<em>`, `<i>`, `<mark>`, `<q>`, `<s>`, `<strong>`
- `<cite>`, `<code>`, `<time>`
- `<br>`

## Image and multimedia

- ``
- `<audio>`, `<video>`, `<track>`
- `<map>`, `<area>`: to define image map

## Scripting

- `<script type="text/javascript" src="someSource.js">`
- `<noscript>`: in case a script type is unsupported
- `<canvas>`: for canvas or WebGL APIs



# HTML Elements

Table content	<ul style="list-style-type: none"><li>• <code>&lt;table&gt;</code></li><li>• <code>&lt;caption&gt;</code></li><li>• <code>&lt;th&gt;</code>, <code>&lt;tr&gt;</code>, <code>&lt;td&gt;</code>: header, row, data cell</li></ul>
Interactive elements	<ul style="list-style-type: none"><li>• <code>&lt;dialog&gt;</code></li><li>• <code>&lt;menu&gt;</code>, <code>&lt;menuitem&gt;</code></li></ul>
Embedded content	<ul style="list-style-type: none"><li>• <code>&lt;applet&gt;</code></li></ul>



# HTML Elements

## Forms

- `<form>`
- `<label for="...">`
- `<input type="..." value="...">`
- `<output name="...">`
- `<button name="...">`
- `<select>`, `<option>`
- `<fieldset>`, `<datalist>`, `<optgroup>`
- `<meter>`, `<progress>`

# In-Class Practice: HTML 排版

todos

What needs to be done?

- ☐ This is the first TODO.
- ☒ ~~This is the second TODO.~~
- ☐ This is the third TODO. ⓧ

2 left   All Active Completed Clear completed

要如何切版呢？



# In-Class Practice: HTML 排版 (1)

`<div id="root" class="todo-app__root">`

`<header class="todo-app__header">`

todos

`<section class="todo-app__main">`

What needs to be done?

☐ This is the first TODO.

☒ This is the second TODO.

☐ This is the third TODO.



2 left

All

Active

Completed

Clear completed

`<footer class="todo-app__footer" id="todo-footer">`

# In-Class Practice: HTML 排版 (2)

<div id="root" class="todo-app\_\_root">

<header class="todo-app\_\_header">

<h1 class="todo-app\_\_title">

todos

<section class="todo-app\_\_main">

What needs to be done?

☐ This is the first TODO.

☒ ~~This is the second TODO.~~

☐ This is the third TODO. ⓧ

2 left

AllActiveCompleted

Clear completed

<footer class="todo-app\_\_footer" id="todo-footer">



# In-Class Practice: HTML 排版 (3)

`<div id="root" class="todo-app__root">`  
`<header class="todo-app__header">`

`<input id="todo-input" class="todo-app__input" placeholder="What needs to be done?" />`

`<section class="todo-app__main">`

`<ul id="todo-list" class="todo-app__list">`  
`<li class="todo-app__item">`

`<footer class="todo-app__footer" id="todo-footer">`

## In-Class Practice: HTML 排版 (4)

```
<header class="todo-app__header">
```

What needs to be done?

● This is the first TODO.

☒ This is the second TODO.

```
<ul class="todo-app__view-buttons">  
  <li class="todo-app__clean">This is the third TODO.</li>  
</ul>
```

```
<ul class="todo-app__view-buttons">
```

```
<div class="todo-  
app__clean">
```

⊗

2 left

All Active Completed

Clear completed



比想像中複雜，是吧？

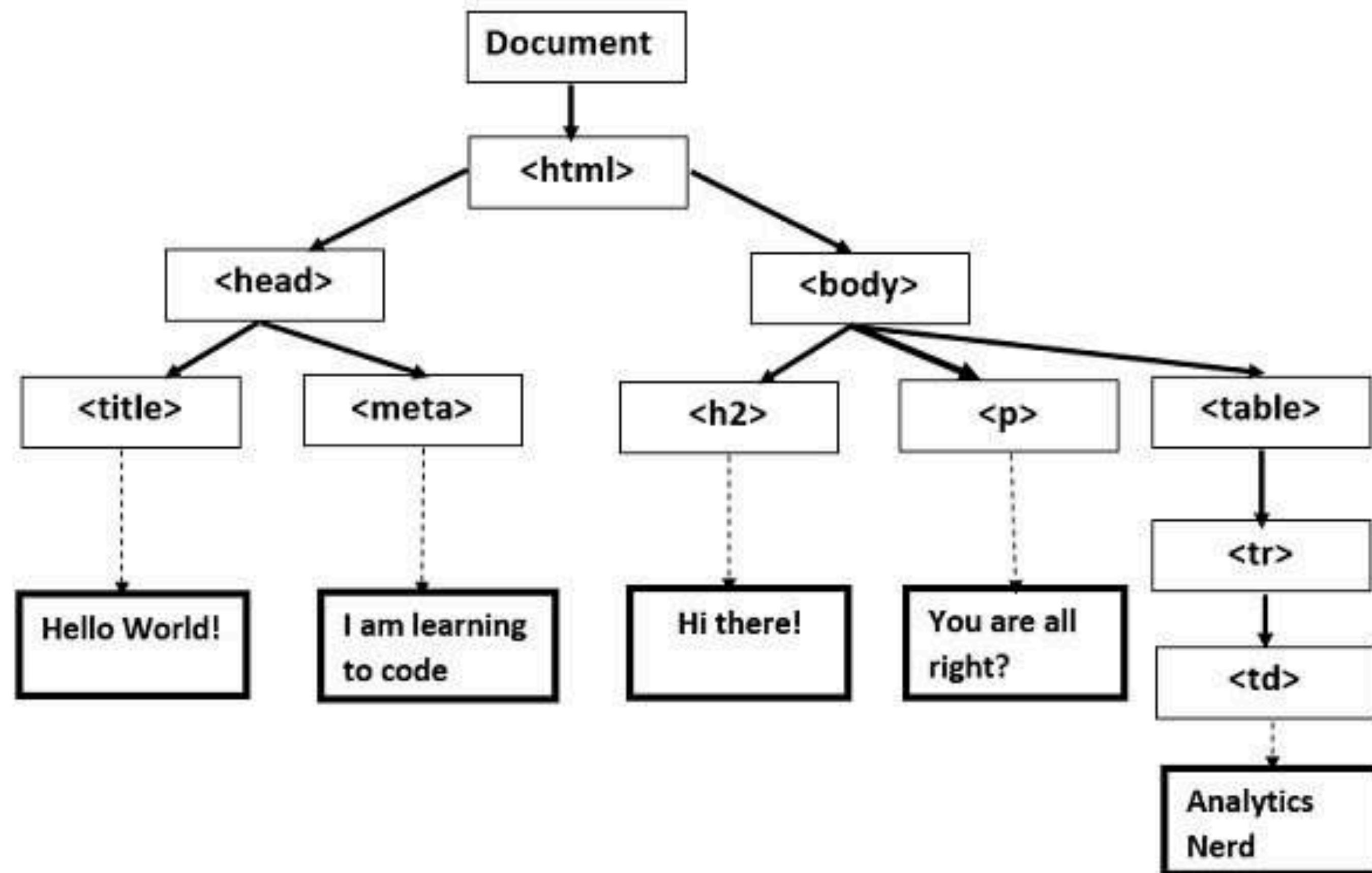
如果你期待的網頁/網路服務設計是  
可以用拖拉、what you type is  
what you get (ref: no-code)  
那你可能走錯棚，可以按左鍵離開了



# Introducing DOM (Document Object Method)

(wiki) ...is a cross-platform and language-independent application programming interface that treats an HTML, XHTML, or XML document as a **tree structure** wherein each node is an object representing a part of the document.

# DOM Example



```
<html lang="en">
<head>
  <title>Document</title>
  <meta text="I am learning to code">
</head>
<body>
  <h2>Hi there!</h2>
  <p>You are all right?</p>
  <table>
    <tr>
      <td>Analytics Nerd</td>
    </tr>
  </table>
</body>
</html>
```

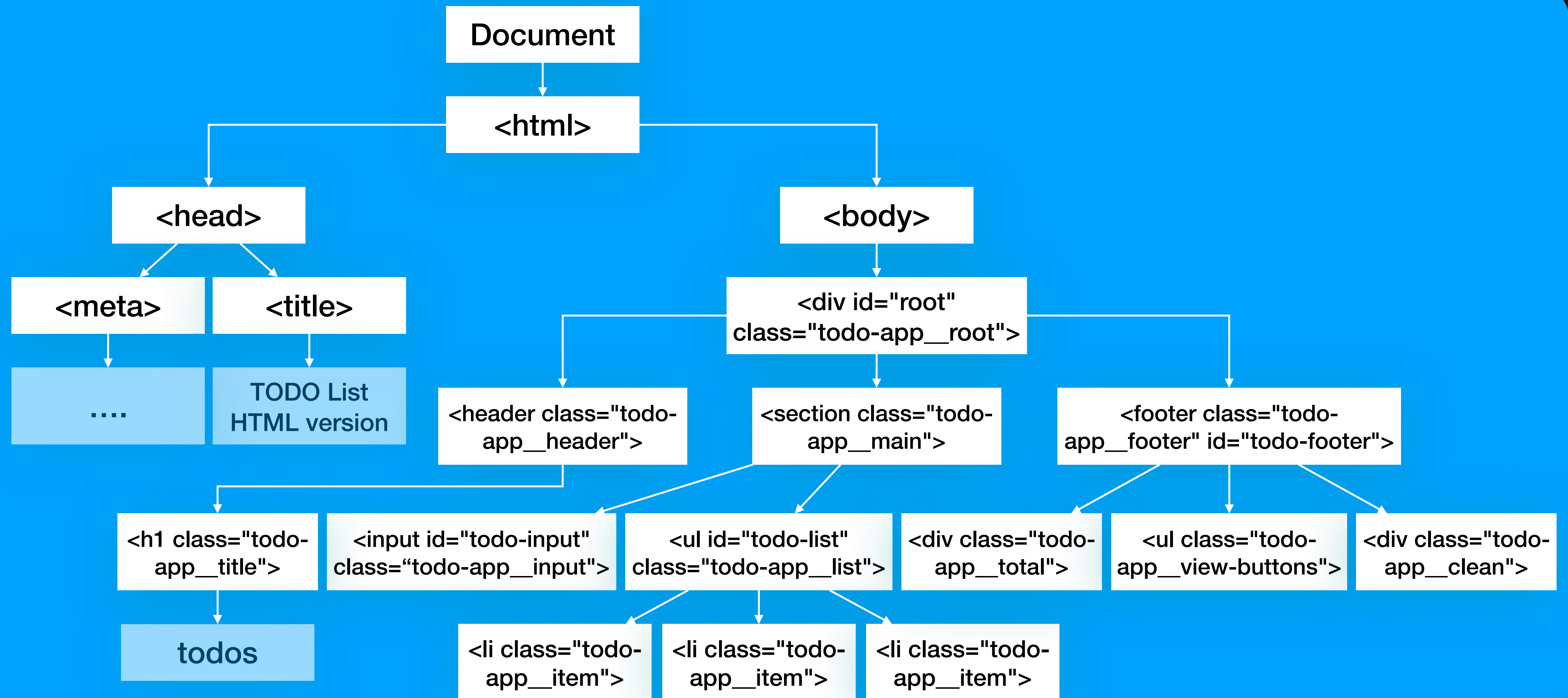
**Hi there!**

You are all right?

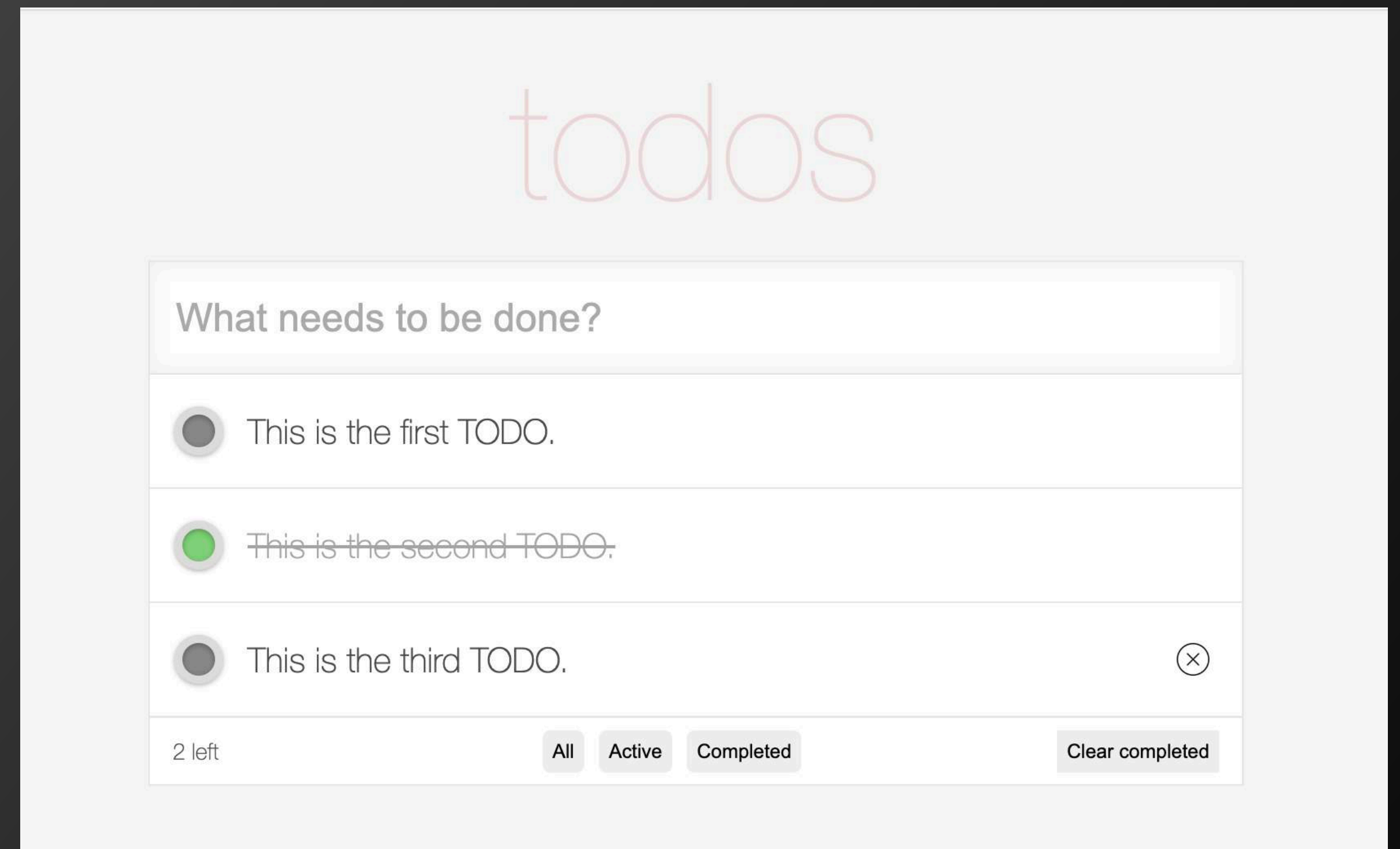
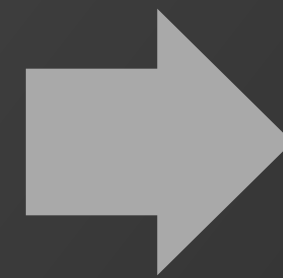
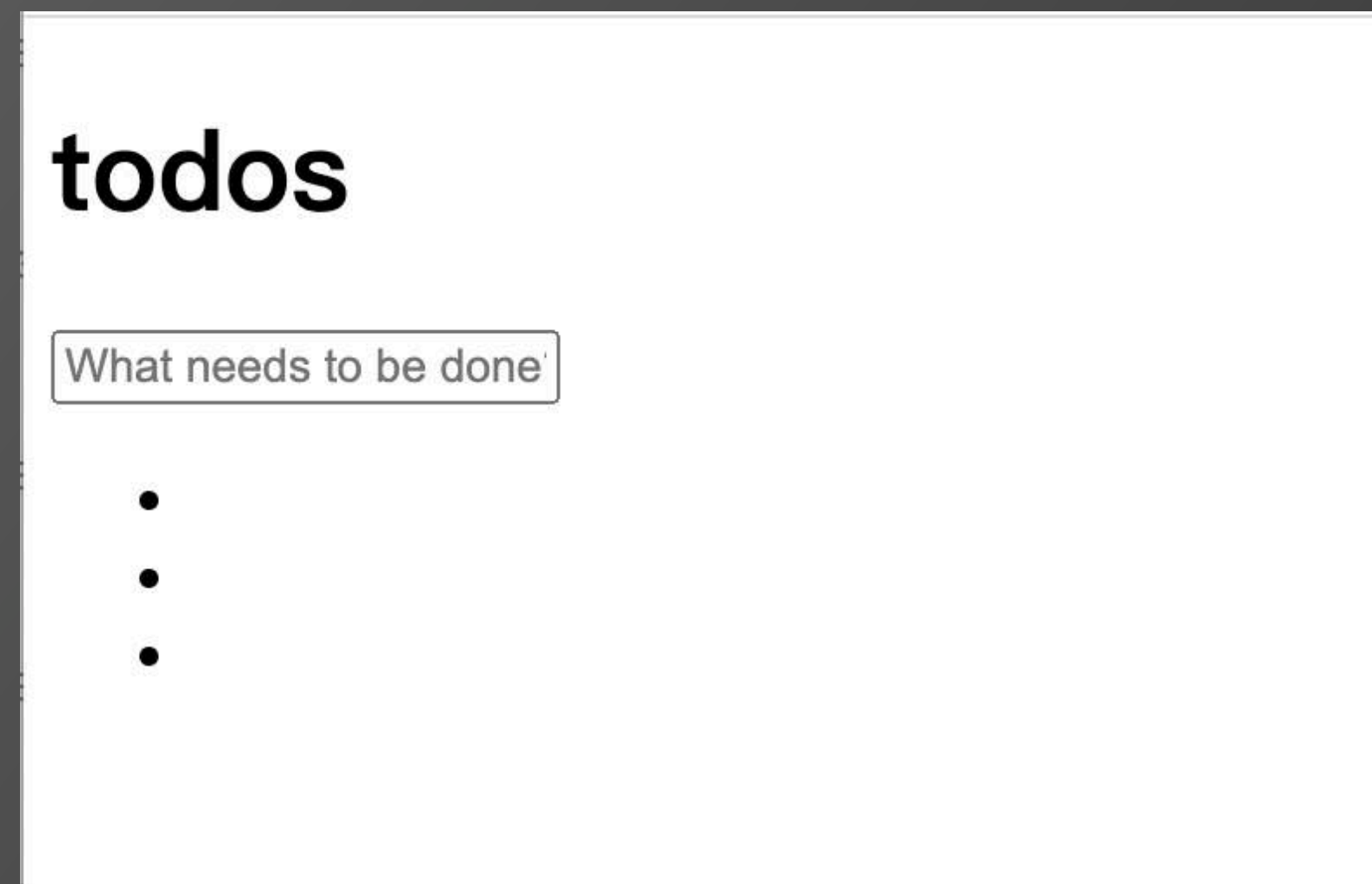
Analytics Nerd



在一個瀏覽器的頁面 “window” 是唯一的全域物件，  
而 “document” 是它的一個屬性



# What “todo-app” looks like now...



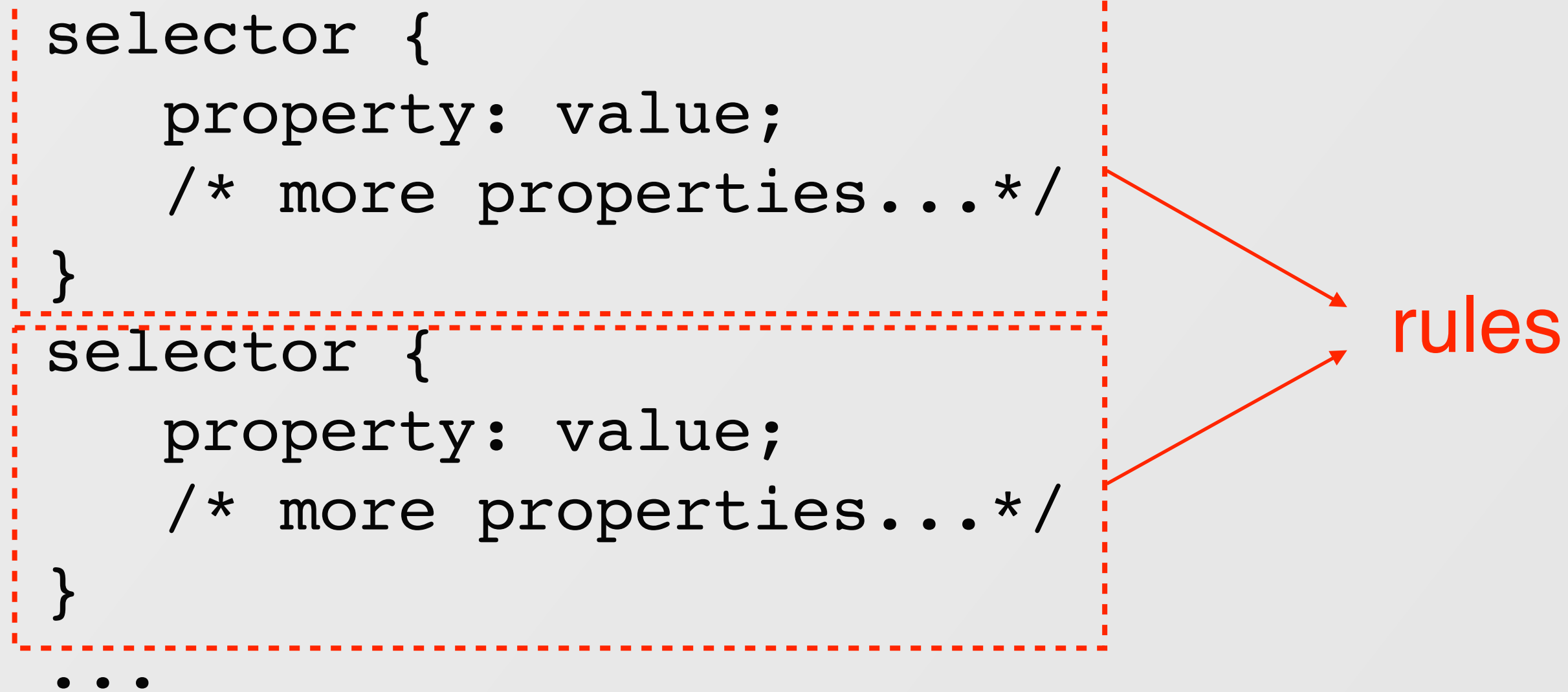
排版、樣式、風格！！



- 前面講的各種 HTML elements, 可以讓你建置網頁的各種樣式與內容
- 至於網頁的外觀與排版，則可以靠 tags 裡頭各式 attributes 來控制
  - align, bgcolor, border, height, size, width...
- 不過把外觀的控制散落到各個 elements 將會使得 code 很難維護，也很難保持整體外觀的一致性

# CSS。化妝網頁的衣裳

- CSS: Cascading Style Sheets
  - 「階層式」、「結構化」地定義了 HTML document 的外觀樣式
- CSS 檔案 ≡ 許多 rules 所組成



```
selector {  
    property: value;  
    /* more properties...*/  
}  
selector {  
    property: value;  
    /* more properties...*/  
}  
...
```

rules

- HTML 檔案裡頭 element 的樣式就由這些 rules 來定義



## CSS。化妝網頁的衣裳

- Rule precedence
  - Rule with more specific selector > less specific selector
  - 後面的 rule > 前面的 rule
- Comments are in `/* ... */`
- Web dev playground: <https://codepen.io/>

# CSS。應用方式

## 1. 直接在 <head> 裡頭 include .css file

- `<link rel="stylesheet" type="text/css" href="path/to/style.css">`

## 2. 將一段 CSS code 嵌入到 HTML 裡頭

- `<style>`  
    `a { color: purple; }`  
    `</style>`

## 3. 直接寫在 tag 的 attribute

- `<div style="border: 1px solid red;">`



# CSS • Basic Selectors

Given an HTML element

```
<div class="class1 class2" id="anID"  
      attr="value" otherAttr="en-us foo bar" />
```

We can decorate it with one of the following CSS selectors —

1. Select by one or more class names

- `.class1 { }`
- `.class1, .class2 { }`

2. Select by the tag name

- `div { }`

3. Select by its id

- `#anID { }`

# CSS • Properties and Values

```
selector {  
  
    /* Units of length can be absolute or relative. */  
  
    /* Relative units */  
    width: 50%;           /* percentage of parent element width */  
    font-size: 2em;       /* multiples of element's original font-size */  
    font-size: 2rem;      /* or the root element's font-size */  
    font-size: 2vw;       /* multiples of 1% of the viewport's width (CSS 3) */  
    font-size: 2vh;       /* or its height */  
    font-size: 2vmin;     /* whichever of a vh or a vw is smaller */  
    font-size: 2vmax;     /* or greater */  
  
    /* Absolute units */  
    width: 200px;         /* pixels */  
    font-size: 20pt;      /* points */  
    width: 5cm;           /* centimeters */  
    min-width: 50mm;      /* millimeters */  
    max-width: 5in;       /* inches */  
}
```



```

/* Colors */
color: #F6E; /* short hex format */
color: #FF66EE; /* long hex format */
color: tomato; /* a named color */
color: rgb(255, 255, 255); /* as rgb values */
color: rgb(10%, 20%, 50%); /* as rgb percentages */
color: rgba(255, 0, 0, 0.3); /* as rgba values (CSS 3) Note: 0 <= a <= 1 */
color: transparent; /* equivalent to setting the alpha to 0 */
color: hsl(0, 100%, 50%); /* as hsl percentages (CSS 3) */
color: hsla(0, 100%, 50%, 0.3); /* as hsl percentages with alpha */

/* Borders */
border-width: 5px;
border-style: solid;
border-color: red; /* similar to how background-color is set */
border: 5px solid red; /* this is a short hand approach for the same */
border-radius: 20px; /* this is a CSS3 property */

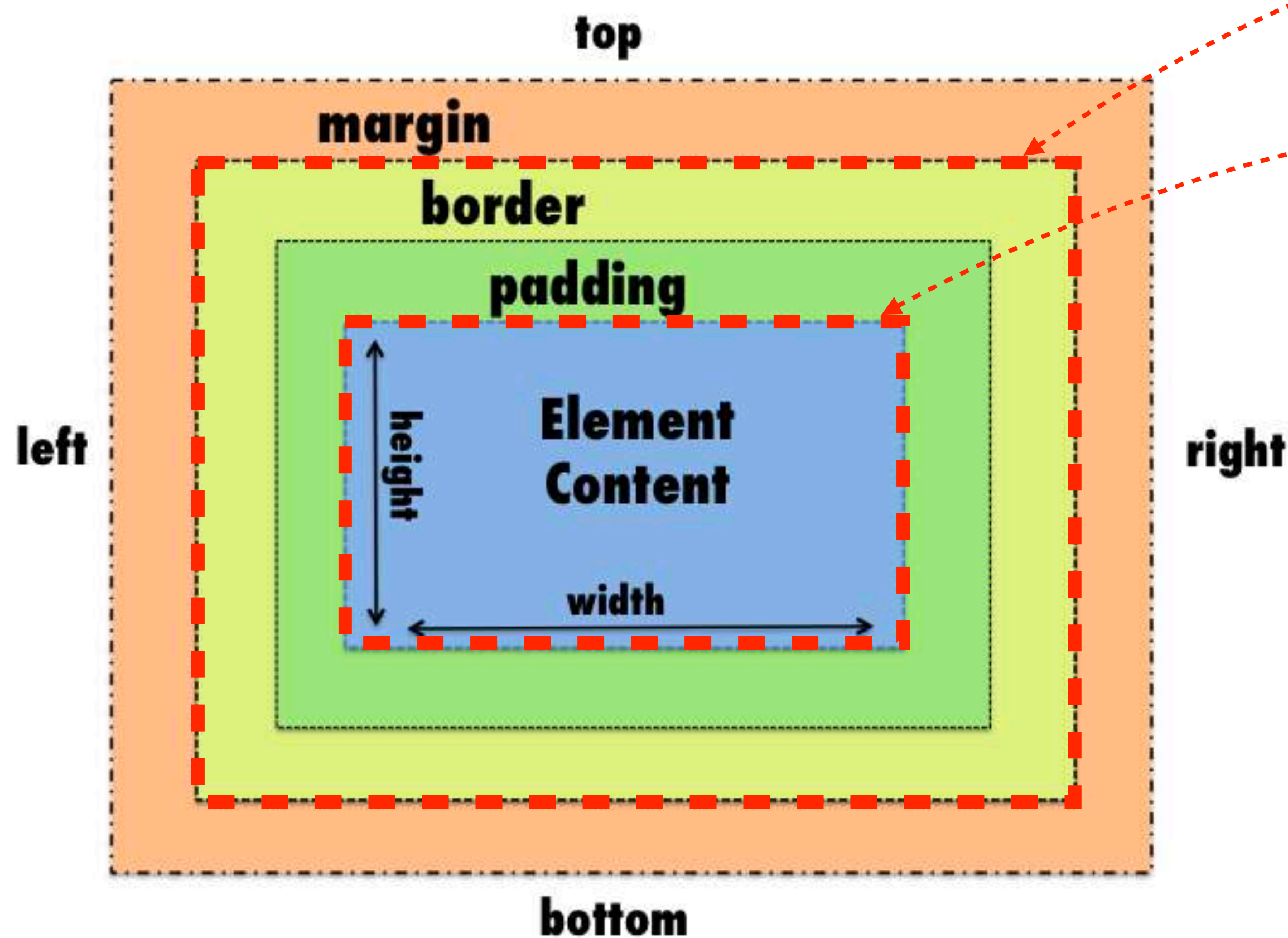
/* Images as backgrounds of elements */
background-image: url(/img-path/img.jpg); /* quotes inside url() optional */

/* Fonts */
font-family: Arial;
/* if the font family name has a space, it must be quoted */
font-family: "Courier New";
/* if the first one is not found, the browser uses the next, and so on */
font-family: "Courier New", Trebuchet, Arial, sans-serif;
}

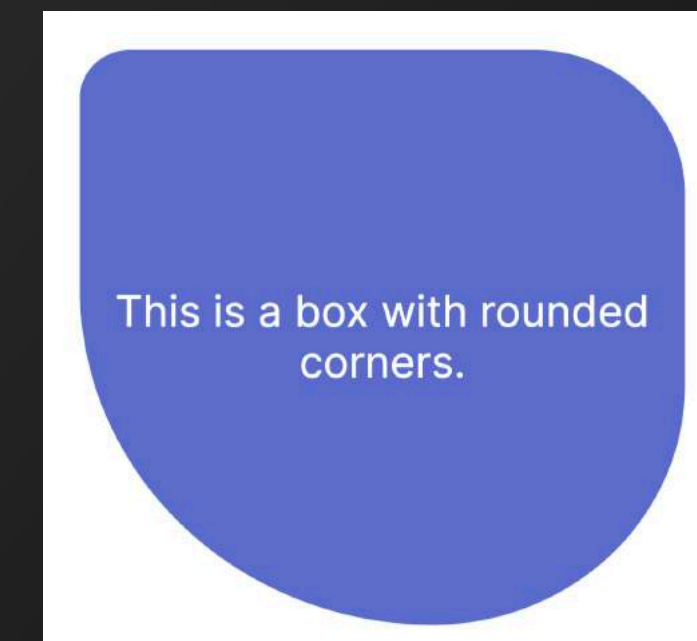
```



# CSS • Box Model



- box-sizing: **border-box** | **content-box**  
=> See examples
- border-radius:  
10%, 30%, 50%, 70%



=> How to draw a circle?

- margin: **calc(...)**



# CSS • Useful Resource for Text Fonts

The screenshot displays the Google Fonts website interface. At the top, the 'Google Fonts' logo is on the left, and navigation links for 'Browse fonts', 'Featured', 'Articles', and 'About' are on the right. Below the navigation bar is a search bar with a magnifying glass icon and the text 'Search'. To the right of the search bar are dropdown menus for 'Sentence' and 'Type something', and a font size selector set to '40px' with a slider. Further down are filters for 'Categories', 'Language', and 'Font properties', along with a checkbox for 'Show only variable fonts'. The main content area shows '1003 of 1003 families' and a 'View' selector set to 'Grid'. The font families displayed are:

- Roboto** by Christian Robertson (12 styles): Sample text 'Almost before we knew it, we had left the ground.'
- Noto Sans TC** by Google (6 styles): Sample text '他們所有的設備和儀器彷彿都是有生命的。'.
- Ranchers** by Impallari Type (1 style): Sample text 'Almost before we knew it, we had left the ground.'
- Kumbh Sans** by Saurabh Sharma (3 styles): Sample text 'Almost before we knew it, we had left'.
- Open Sans** by Steve Matteson (10 styles): Sample text 'Almost before we knew it, we had left'.
- Noto Serif TC** by Google (7 styles): Sample text '他們所有的設備和儀器彷彿都是有生命的。'.

# CSS for todo-app (1)

- 先設定一些全域（整個網頁）的樣式

```
* {  
  box-sizing: border-box;  
}  
  
html,  
body {  
  height: 100%;  
  font: 14px 'Helvetica Neue', Helvetica, Arial, sans-serif;  
  background: rgb(245, 245, 245);  
  overflow: auto;  
}
```



# CSS for todo-app (1)

- For the root element

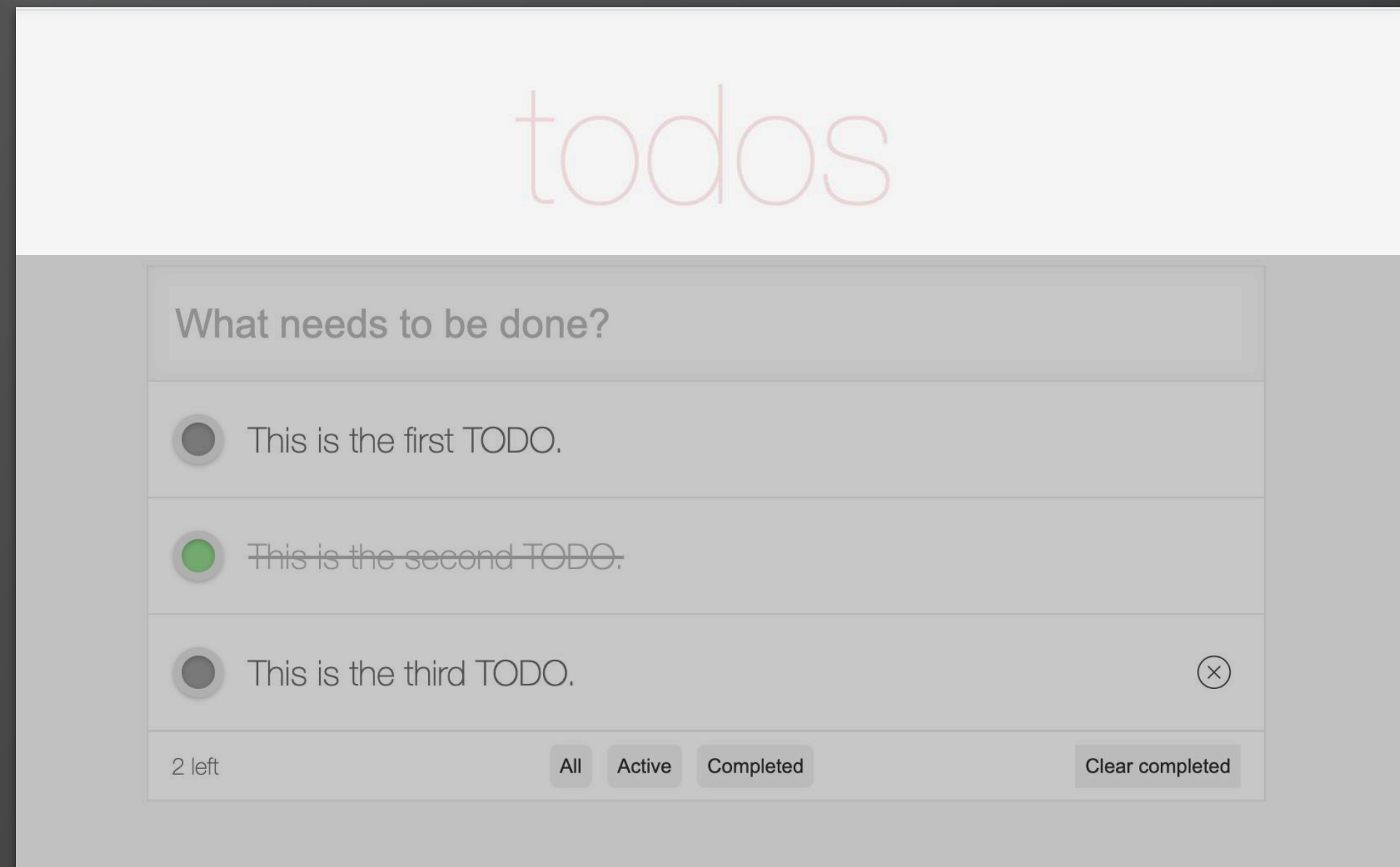
```
.todo-app__root {  
  margin: 0 auto;    // 上下, 左右  
  width: 50em;  
  display: flex;  
  flex-direction: column;  
  justify-content: flex-start;  
  align-items: center;  covered later  
  padding: 1em;  
  color: #4d4d4d;  
  font-weight: 300;  
}
```

- 存檔成 “styles.css” , 然後在 <head> 裡頭加上 :

```
<link rel="stylesheet" type="text/css" href="./styles.css" />
```

# CSS for todo-app (2)

```
<header class="todo-app__header">
  <h1 class="todo-app__title">todos
</h1>
</header>
```



- 先修好 <header> :
  - 字體要放大，顏色要改變，文字要置中

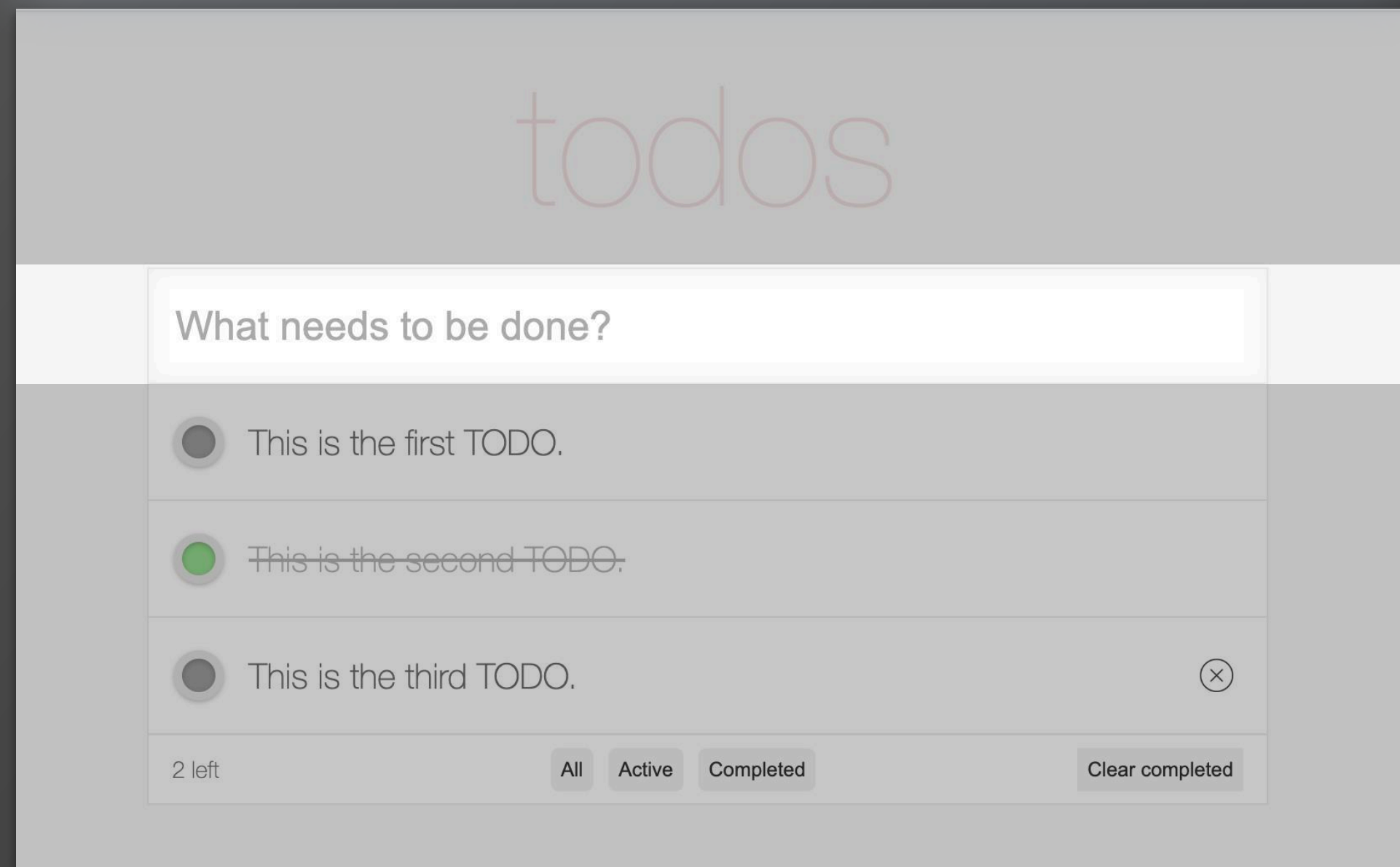
```
.todo-app__header {
  margin-bottom: 1em;
}

.todo-app__title {
  font-size: 100px;
  font-weight: 100;
  text-align: center;
  color: rgba(175, 47, 47, 0.15);
  margin: 0;
  padding: 0;
}
```



# CSS for todo-app (3)

```
<section class="todo-app__main">
  <input
    id="todo-input"
    class="todo-app__input"
    placeholder=
      "What needs to be done?"
  />
  ...
</section>
```



- 再來修 <input> :
  - 寬/高，padding，color, font
  - 淡淡的 border, 往內縮的模糊陰影

```
.todo-app__main { width: 100%; }
.todo-app__input {
  width: 100%;
  height: 5rem;
  padding: 16px;
  border: 1px solid rgba(0, 0, 0, 0.089);
  color: inherit;
  font-size: 24px;
  font-weight: 200;
  line-height: 1.4em;
  box-shadow: inset 0px 0px 10px 2px
              rgba(0, 0, 0, 0.048);
}
```

## CSS • Inherited Properties ([ref1](#), [ref2](#))

- CSS properties can be categorized in two types:
  - **inherited properties**, which by default are set to the computed value of the parent element (e.g. color)
  - **non-inherited properties**, which by default are set to initial value of the property (e.g. border)
- CSS values for controlling inheritance:  
**inherit, initial, revert, revert-layer, and unset**



# CSS • Box-shadow Properties [\(Ref\)](#)[\(Example\)](#)

Syntax —

- `box-shadow: none | inset | initial | inherit | h-offset v-offset blur spread color ;`

See also —

- `text-shadow`: shadow around text

[Test yourself] How to use "text-shadow" to create the border effect as below?

**Border around text!**

```
h1 {  
  color: coral;  
  text-shadow: -1px 0 black,  
               0 1px black,  
               1px 0 black,  
               0 -1px black;  
}
```

Update "styles.css".  
Do you see any problem?

輸入文字時，有很醜的外框

Placeholder 的字顏色也太深了...



## CSS • Pseudo Class Selectors

- 有的時候會希望能在 `mouse` 做一些動作的時候，指定或是改變一些 `properties` 的值
  - `selector:pseudo-class {  
    property: value;  
}`
- `pseudo-class := hover, active, focus, first-child, nth-child(n), not(s), visited, link...`

```
.todo-app__input:focus {  
    outline: none;  
}
```

## CSS • Pseudo Element Selectors

- 如何更改 "placeholder" 的樣式呢？
  - 有時候，我們只想修飾一個 element 的某一部分 (如：第一行)
  - `selector::pseudo-element {  
 property: value;  
}`
- Pseudo-element := `first-line`, `first-letter`, `marker`, `selection`, `placeholder`, `target-text`...





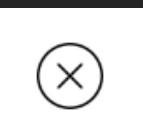
```
.todo-app__input::placeholder {  
    font-weight: 100;  
    opacity: 0.6;  
}
```



# CSS for todo-app (4)

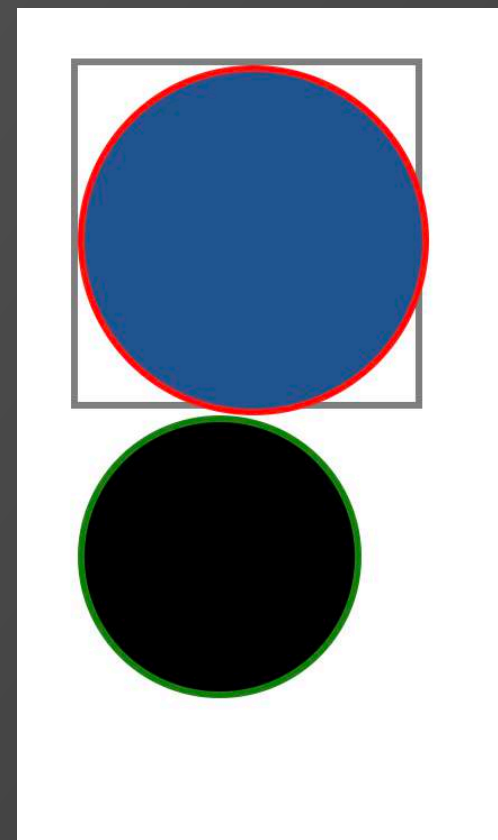
```
<section class="todo-app__main">
  ...
  <ul id="todo-list"
    class="todo-app__list">
    <li class="todo-app__item"> </li>
    <li class="todo-app__item"> </li>
    <li class="todo-app__item"> </li>
  </ul>
</section>
```



- 如何讓 TODO items 由上而下排在 TODO list 裡頭，而且排好排滿？
- 要如何畫出：
- 要如何畫出一個圓？
- 要如何畫出同心圓，但又可以 click 而變色？ → 
- 如何讓  , TODO detail, 以及  排成一行？

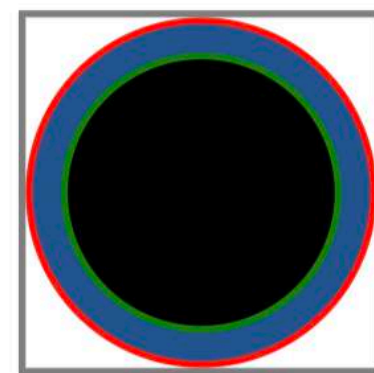
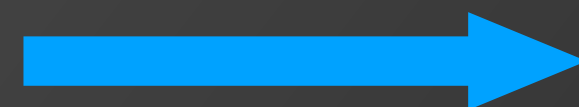
# Try this —

```
<body>
  <div id="target">
    <div id="outer"></div>
    <div id="inner"></div>
  </div>
</body>
```



為什麼排成這樣？

How?



如何排成這樣？

```
#target {
  width: 50px;
  height: 50px;
  border: 1px solid gray;
}
#target > div {
  border-radius: 50%;
}
#outer {
  width: 100%;
  height: 100%;
  border: 1px solid red;
  background-color: #225591;
}
#inner {
  width: 80%;
  height: 80%;
  border: 1px solid green;
  background-color: black;
}
```



# CSS ◦ Element Layout

- By default, HTML 的 elements 是由上到下，靠左一行一行排下來
- Hierarchical elements 裏頭 child elements 是從 parent element 的左上開始排的
- 有些 element 是 inline (如 : `<span>`) 則不會換行
- CSS property "display" to control element layout ([ref](#))

```
.element {  
  display: block | inline;      // (外) 這個 element 如何排列  
  display: flex;                // (內) element 的內容如何排列  
  display: none;                // 不 display;  
                                // 與 "visibility: hidden" 不同  
}
```

學好 CSS Flex Box  
排版沒煩惱～



# CSS ◦ Flexible Box Layout

- 設定 flex box mode 來控制其內容如何排列

```
display: flex;           // 元件 block; 內容 flex-box  
display: inline-flex;    // 元件 inline; 內容 flex-box
```

學習重點：

1. Main axis vs. Cross axis
2. 對齊 & 分佈方式
3. 縮放

# CSS ◦ flex-direction

- 決定 main axis 的方向

```
flex-direction: row;      // 水平為 main axis, 且以左端點為 flex-start
flex-direction: row-reverse; // 同上, 但以右端點為 flex-start
flex-direction: column;   // 垂直為 main axis, 且以上端點為 flex-start
flex-direction: column-reverse; // 同上, 但以下端點為 flex-start
```

- Main axis 決定後，內容物從 flex-start 往 flex-end 排，如果超過範圍，則由 flex-wrap 決定：

```
flex-wrap: no-wrap;      // 不自動換行
flex-wrap: wrap;          // 往 cross axis 方向換行
flex-wrap: wrap-reverse; // 往 cross axis 的反方向換行
```



# CSS。內容物對齊與分佈方式

- 決定 content items 周邊的空白在 main axis 的分佈方式

<b>justify-content:</b>	// 決定 items 間的空白 如何在 main axis 分佈
flex-start	// 從 flex-start 開始緊密排列
flex-end	// 從 flex-end 開始緊密排列
center	// 置中緊密排列
space-between	// 對齊 flex-start 與 flex-end, 中間空白平均分配
space-around	// 將剩下來的空白平均分配給每個 item 的周圍
space-evenly;	// 讓 items 之間與 items 旁邊的空白一樣大

- 決定 content items 周邊的空白在 cross axis 的分佈方式

<b>align-content:</b>	// 決定 items 間的空白 如何在 cross axis 分佈
flex-start	// 靠 flex-start 緊密排列
flex-end	// 靠 flex-end 緊密排列
center	// 置中緊密排列
space-between	// 對齊 flex-start 與 flex-end, 中間空白平均分配
space-around	// 將剩下來的空白平均分配給每個 item 的周圍
stretch;	// 將可變的 items 沿著 cross axis 等量拉長不留空白

# CSS。內容物對齊與分佈方式

- 決定 content **items** 在 **cross axis** 的**對齊**方式

<b>align-items:</b>		// 決定 items 如何在 cross axis 上對齊
flex-start		// 靠 flex-start 邊界對齊
flex-end		// 靠 flex-end 邊界對齊
center		// 置中對齊
stretch;		// 將可變的 elements 沿著 cross axis 拉長對齊

- 例如：如果 flex-direction: row 的話，align-items: flex-start 就是靠上方對齊
- 決定 element **自己** 在 **cross axis** 的**對齊**方式

<b>align-self:</b>		// 決定 element 自身如何在 cross axis 上對齊
flex-start		// 靠 flex-start 邊界對齊
flex-end		// 靠 flex-end 邊界對齊
center		// 置中對齊
stretch;		// 將 element 自身沿著 cross axis 拉長對齊

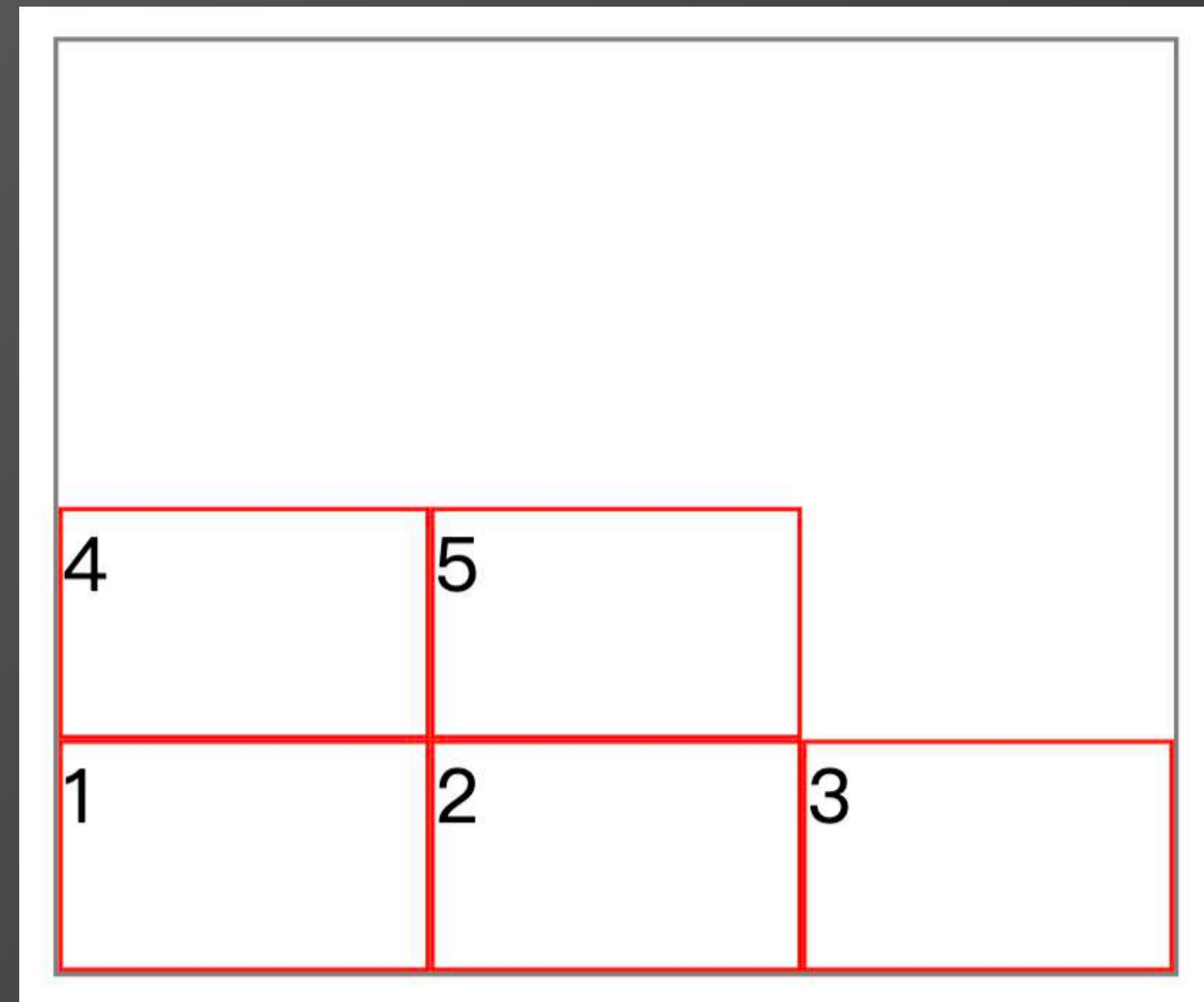


## CSS。內容物對齊與分佈方式

- 關於 CSS 的排版，根據不同的 "display" mode, 有各種不同的設定規則，十分複雜
- 建議把 "flexbox" 排版學好為要，可從這篇比較精簡的說明開始，再去延伸閱讀：<https://mzl.la/3AfzPg0>
- 一些值得去查找資料/搞清楚的問題：
  - "start/end" 與 "flex-start/flex-end" 有何不同？([ref](#))
  - 有沒有 "justify-items" 或者是 "justify-self"?
  - 上兩頁教的 properties 還有哪些 values？何時會用到？

# CSS Flexbox 對齊與排列 • Test Yourself (1)

- 先來個簡單的：



- Copy the ref codes to CodePen and try!!

- CSS reference:

```
.container {  
  width: 240px;  
  height: 200px;  
  border: 1px solid gray;  
}  
.container > div {  
  width: 80px;  
  height: 50px;  
  border: 1px solid red;  
}
```

- HTML reference:

```
<body>  
  <div class="container">  
    <div id="one"> 1</div>  
    <div id="two"> 2</div>  
    <div id="three">3</div>  
    <div id="four"> 4</div>  
    <div id="five"> 5</div>  
  </div>  
</body>
```



## CSS • More on Class Selectors

```
<div class="one"> 1
  <div class="two"> 2in1
    <div class="three"> 3in2in1 </div>
  </div>
  <div class="three"> 3in1
    <div class="two"> 2in3in1 </div>
  </div>
</div>
<div class="one two"> 1and2 </div>
```

- `.one.two { /* Select 1and2 */ }` // AND
- `.one .two { /* Select 2in1, 2in3in1 */ }` // in
- `.one > .two { /* Select 2in1 */ }` // followed
- `.one,.two { /* Select 1, 2in1, 1and2 */ }` // OR

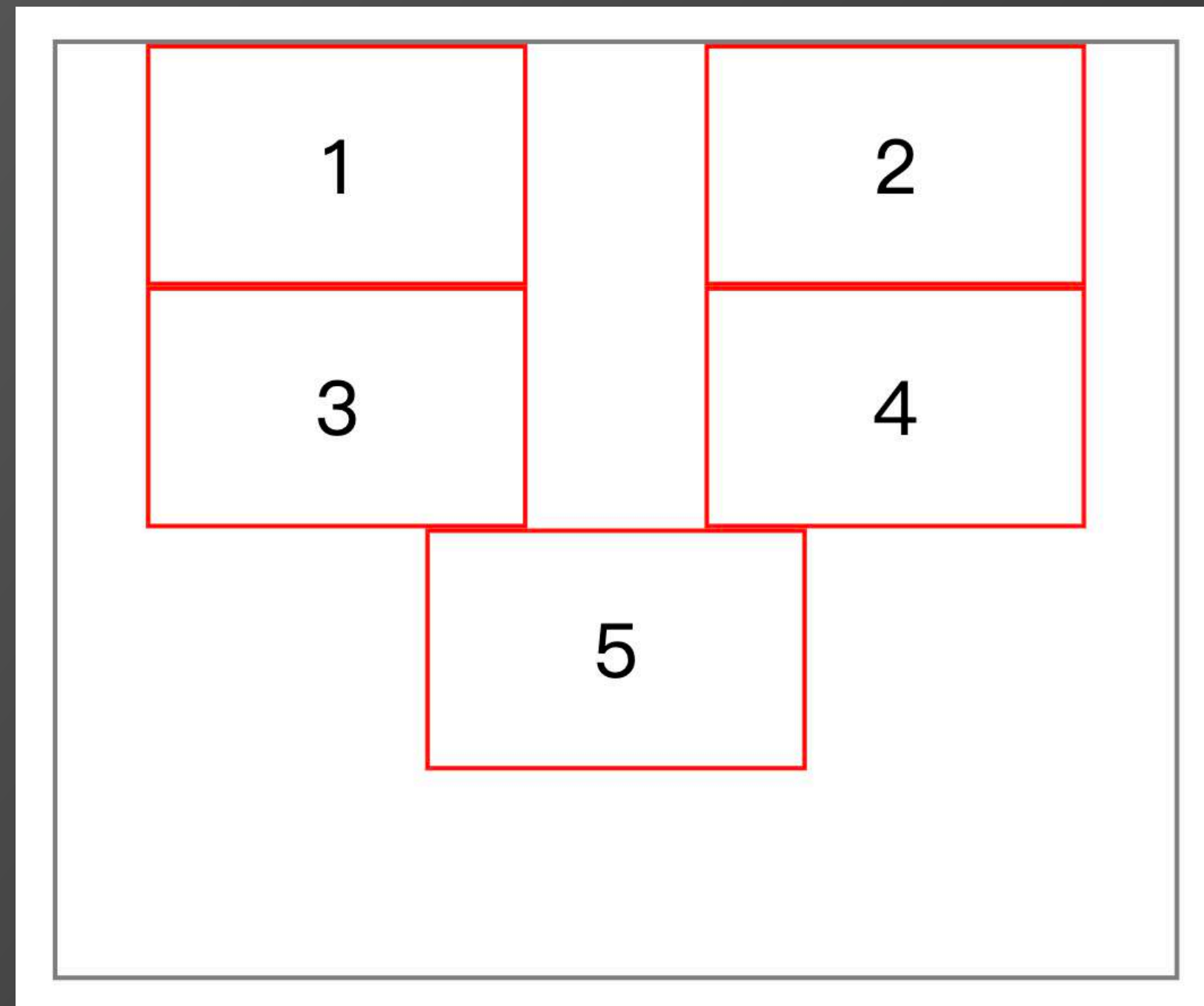
## CSS • Other Selectors

- \* : Select all elements
  - `* { }` `/* all elements */`
  - `.parent * { }` `/* all descendants */`
  - `.parent > * { }` `/* all children */`
- Select an element based on its adjacent sibling
  - `.i-am-just-before + .this-element { }`
- Select an element based on any sibling preceding it
  - `.i-am-any-element-before ~ .this-element { }`

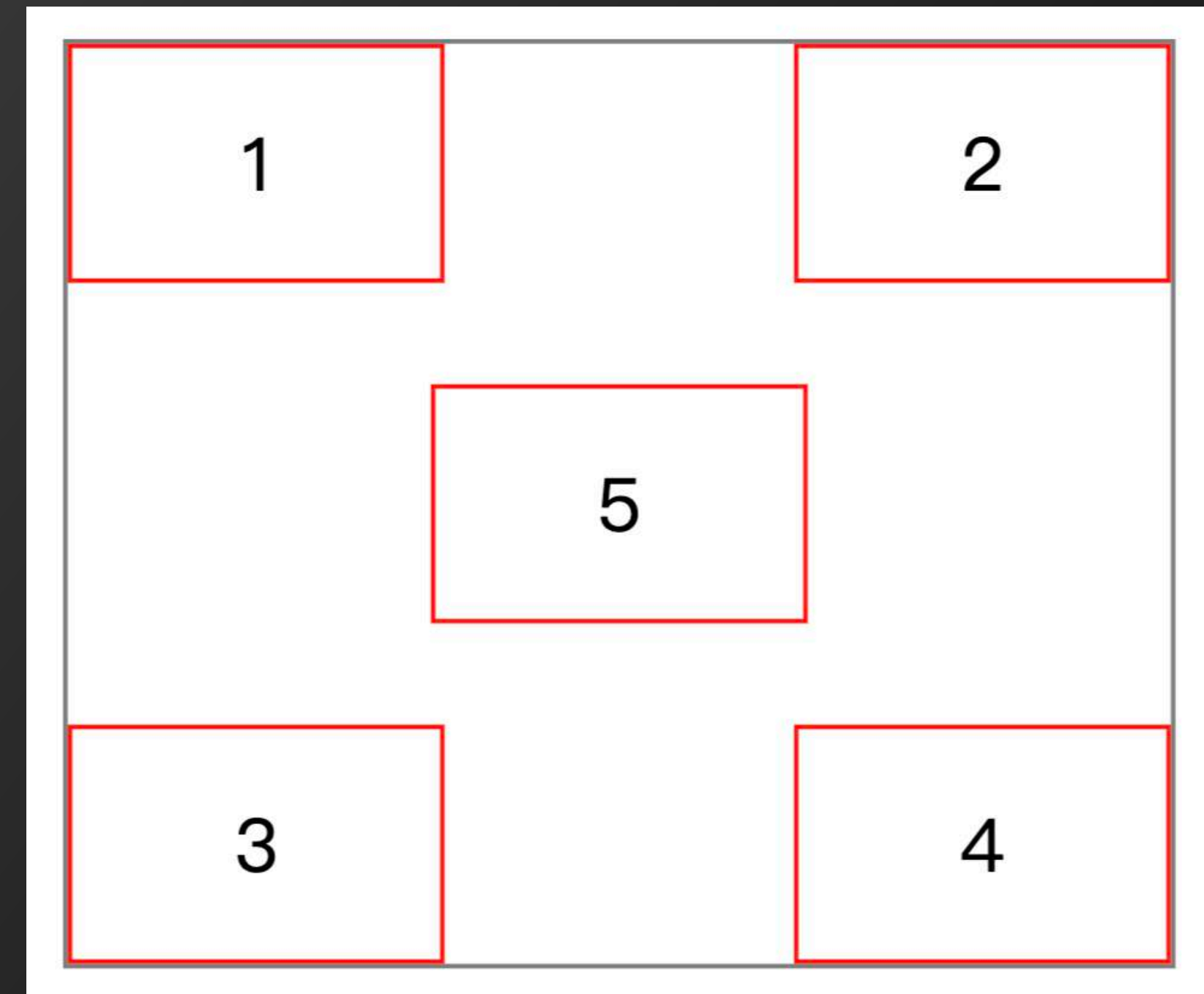


## CSS Flexbox 對齊與排列。Test Yourself (2)

- 把 blocks 與字都置中



- 把 blocks 擺在角落與中間



# CSS · Positioning

- Positioning 的模式

`position:`

<code>static</code>		// (default) 根據 normal flow 擺放 (忽略 top/down/left/right)
<code>relative</code>		// Normal flow + top/down/left/right
<code>absolute</code>		// 從 normal flow 拿掉，然後以 closest positioned ancestor // (container) 為基準，再根據 top/down/left/right 移位擺放
<code>sticky</code>		// 不隨視窗捲動而移位 (see <a href="#">doc</a> )
<code>fixed</code>	;	// 固定在視窗上面 (see <a href="#">doc</a> )

- 調整 x-y-軸的位移

`top | down | left | right : [offset-value];`

- 定義 z 軸的順序 (與螢幕垂直) ([ref](#))

`z-index: auto | <integer> | inherit ;`



## CSS • Element Transform

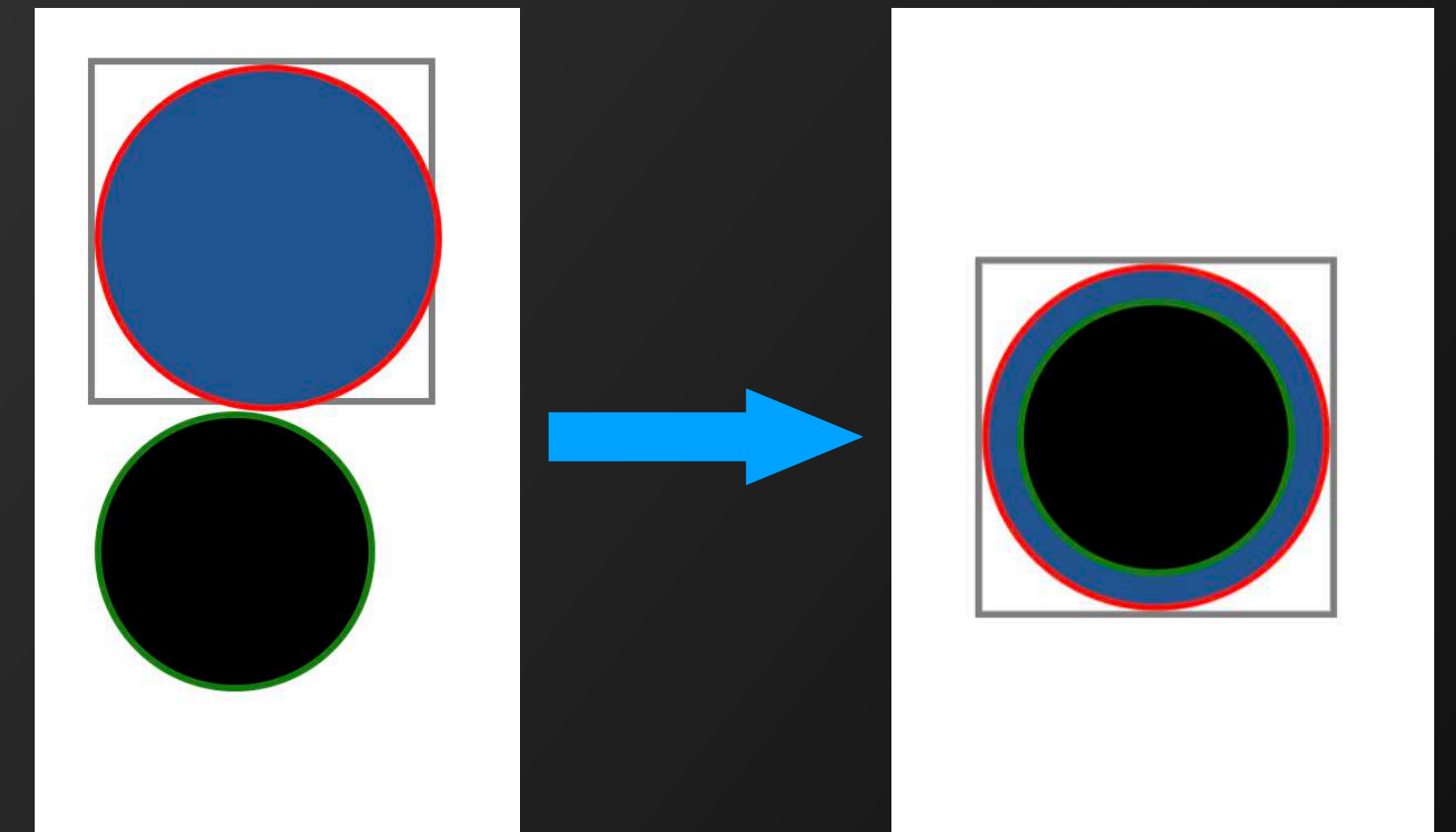
Syntax - `transform : method(...)`

- `translate(x-offset, y-offset);`
- `rotate(45deg); rotate(1.57rad);`
- `scaleX(1.5); scaleY(2.8); scale(1.5, 2.8);`
- `skew(x-angle, y-angle);`
- `rotateX(angle); rotateY(angle);`  
`rotateZ(angle);`

## How to modify the example in p64?

- 先把 #target <div> 放在畫面中央

```
#target {  
  ...  
  position: absolute;  
  top: 50%;  
  left: 50%;  
  transform: translate(-50%, -50%);  
}
```



- #outer 以及 #inner 目前擺放在哪裡？ (Hint: 目前 position: static)
- #outer 的 border 是否超過 #target 的 bounding box?
- 再把 #inner 用同樣的方式搬到 #target 的中央
  - 可以加一個 "z-index: 1" 以確保 #inner 會在最上面



## (Back to) CSS for todo-app (4)

```
<section class="todo-app__main">
  ...
  <ul id="todo-list"
    class="todo-app__list">
    <li class="todo-app__item"> </li>
    <li class="todo-app__item"> </li>
    <li class="todo-app__item"> </li>
  </ul>
</section>
```

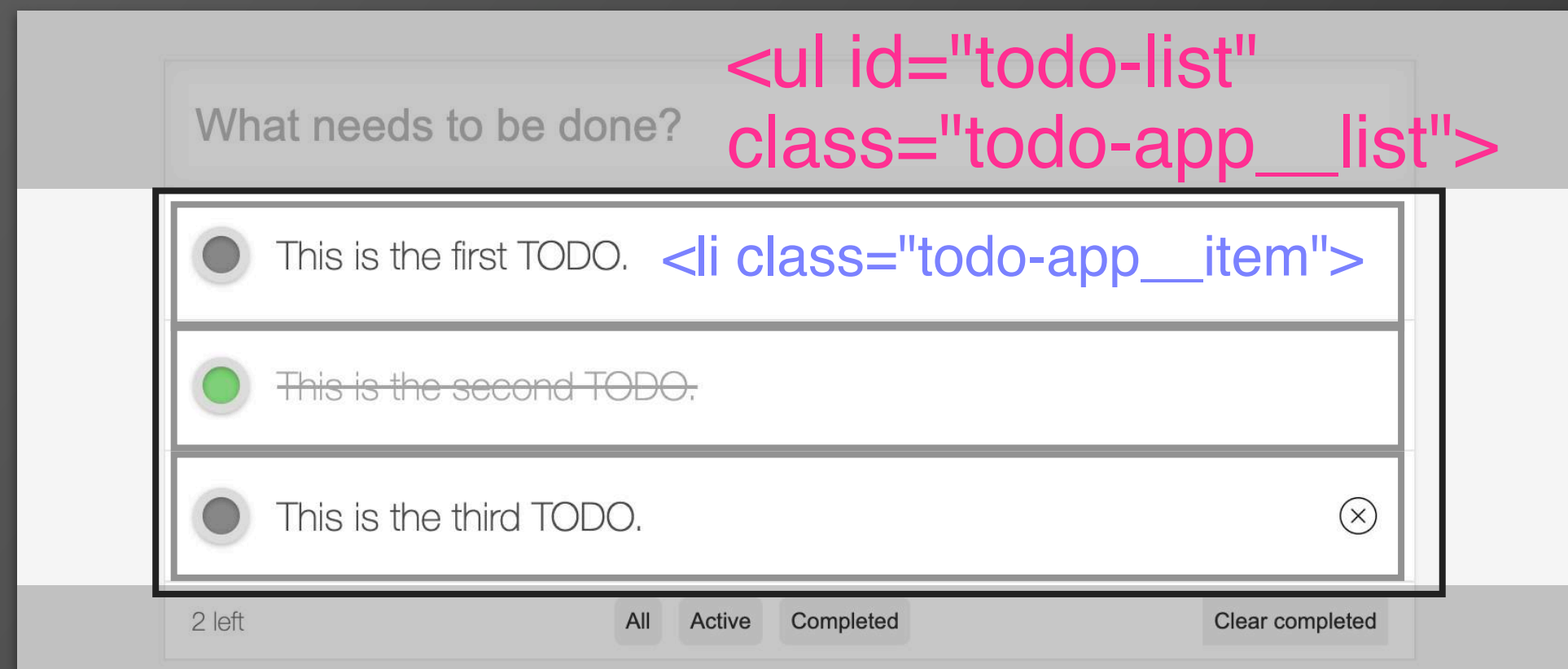
- 如何讓 TODO items 由上而下排在 TODO list 裡頭，而且排好排滿？



`<ul id="todo-list" class="todo-app__list">`

`<li class="todo-app__item">`

# CSS for todo-app (4)



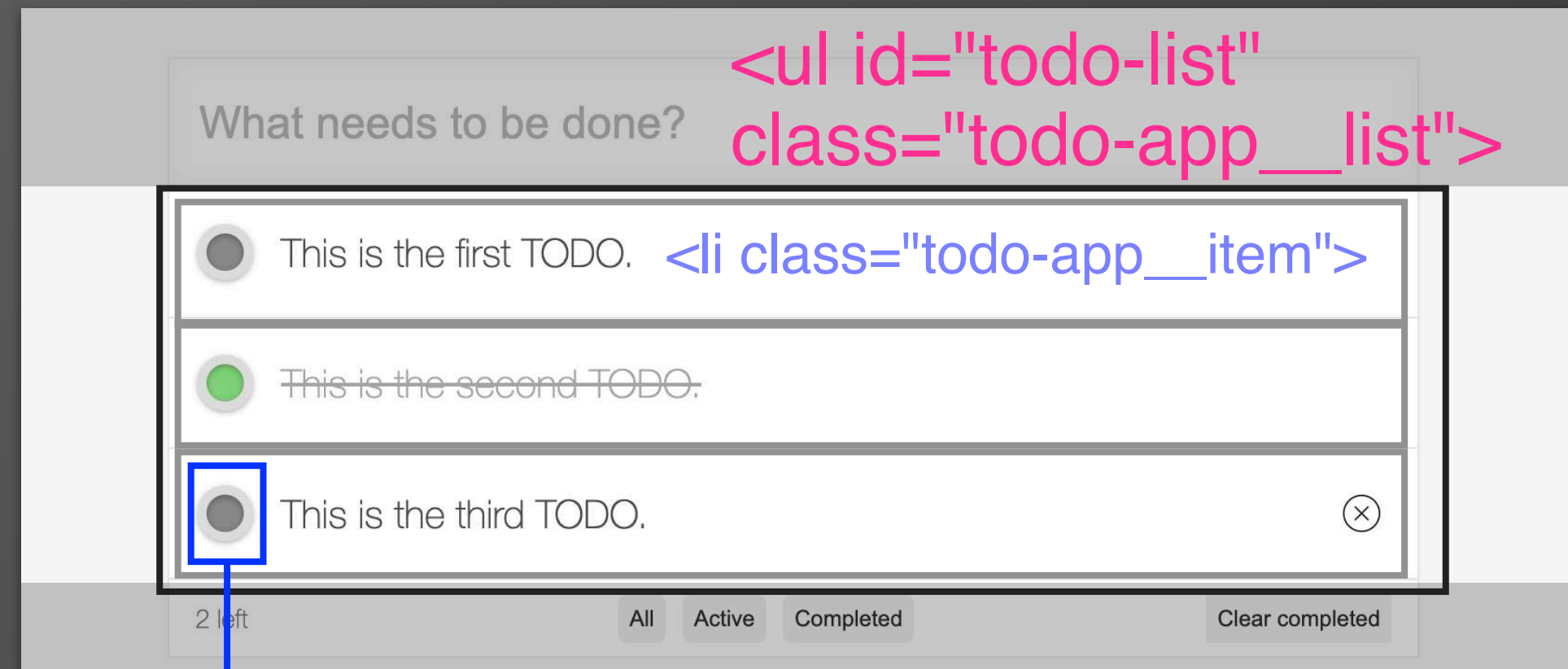
- 如何讓 app\_\_item 由上而下排在 app\_\_list 裡頭，而且排好排滿？
  - 定義 app\_\_list 以及 app\_item 的 max/min-height，讓(未來)新增/刪除 app\_\_item 時 app\_\_list 還是可以把 app\_\_item 包裝好
- 如何讓 check\_box, input, 以及 x 排成一列？
  - flexbox + position

```
.todo-app__list {  
  list-style: none;  
  padding: 0;  
  margin: 0;  
  max-height: 23em;    // for ~5 items  
  overflow: auto;      // 自動 scroll  
}
```



```
.todo-app__item {  
  width: 100%;  
  min-height: 5em;    // 5x5 > 23  
  background: white;  
  border: 1px solid rgba(0, 0, 0, 0.089);  
  border-top: 0;      // 讓 items 靠近一點  
  position: relative;  
  display: flex;  
  flex-direction: row;  
  justify-content: flex-start;  
  align-items: center;  
}
```



# CSS for todo-app (4)



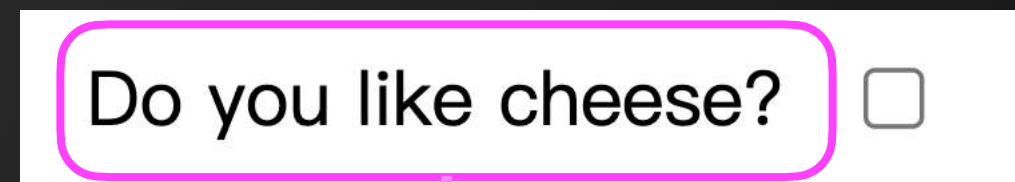
```
<li class="todo-app__item">
  <div class="todo-app__checkbox">
    <input id="0" type="checkbox" />
    <label for="0"></label>
  </div>
</li>
```

- 要如何畫出：
- 要如何畫出一個圓？
  - `border-radius: 50%;`
- 要如何畫出同心圓，但又可以 click 而變色？
  - `<input type="radio" />?`
  - 在 `<div>` 內包一個 radio `<input>?`
  - 用 `<label> + <input>!!`

## <label> with <input>

- <label> represents a **caption** for an item in an user interface (e.g. <input>, <button>...)
  - Guide users what to do
  - Increase the area for activating the input

```
<div class="preference">  
  <label for="cheese">Do you like cheese?</label>  
  <input type="checkbox" name="cheese" id="cheese">  
</div>
```



Clicking here also works!

- Explicitly associate <input>'s "id" with <label>'s "for" or Implicitly associate them by putting <input> as <label>'s child
- <label> can be a box and thus a circle!



# CSS for todo-app (4)

- 要如何畫出同心圓，但又可以 click 而變色？ 

```
<li class="todo-app__item">
  <div class="todo-app__checkbox">
    <input id="0" type="checkbox" />
    <label for="0"></label>
  </div>
</li>
```

```
.todo-app__checkbox {
  width: 30px;
  height: 30px;
  background: #ddd;
  margin: 15px;
  border-radius: 50%;
  position: relative;
  box-shadow: 0px 1px 3px
              rgba(0, 0, 0, 0.312);
}
```

```
.todo-app__checkbox label {
  width: 20px;
  height: 20px;
  border-radius: 50%;
  background: rgba(99, 99, 99, 0.698);
  box-shadow: inset 0px 1px 3px
              rgba(0, 0, 0, 0.312);
  transition: all 0.5s ease;
  cursor: pointer;
  position: absolute;
  top: 5px;
  left: 5px;
  z-index: 1;
}
```

```
.todo-app__checkbox input[type="checkbox"] {
  visibility: hidden;
}

.todo-app__checkbox
  input[type="checkbox"]:checked + label {
  background: #26ca299b;
}
```

# CSS ◦ Transition Property

讓 CSS 的 styling 變化看起來更 smooth 一些 —

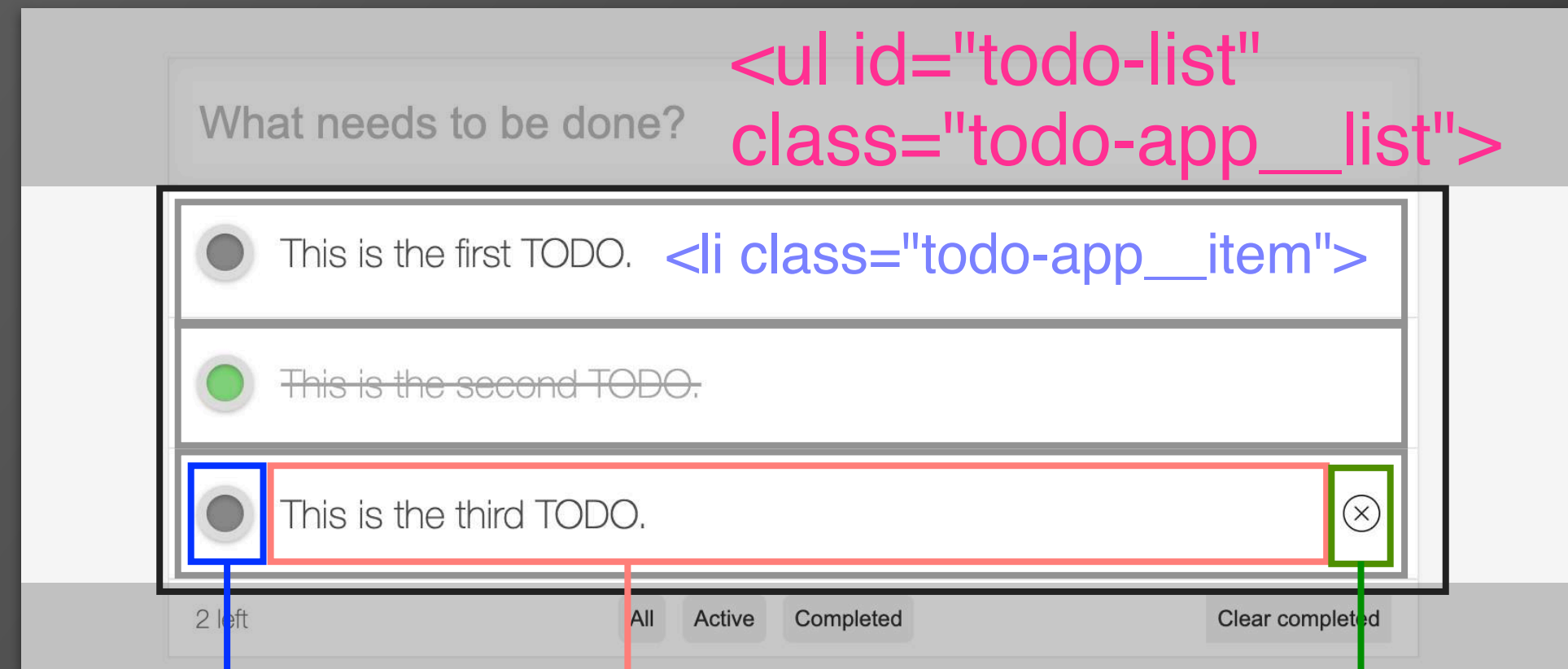
- Properties: `transition`, `transition-delay`, `transition-duration`, `transition-property`, `transition-timing-function`

[Example]

```
div {  
    ... /* width, height, background, etc */  
    transition: width 2s, height 2s, transform 2s;  
}  
div:hover {  
    ... /* width, height, etc */  
    transform: rotate(180deg);  
}
```



# CSS for todo-app (4)



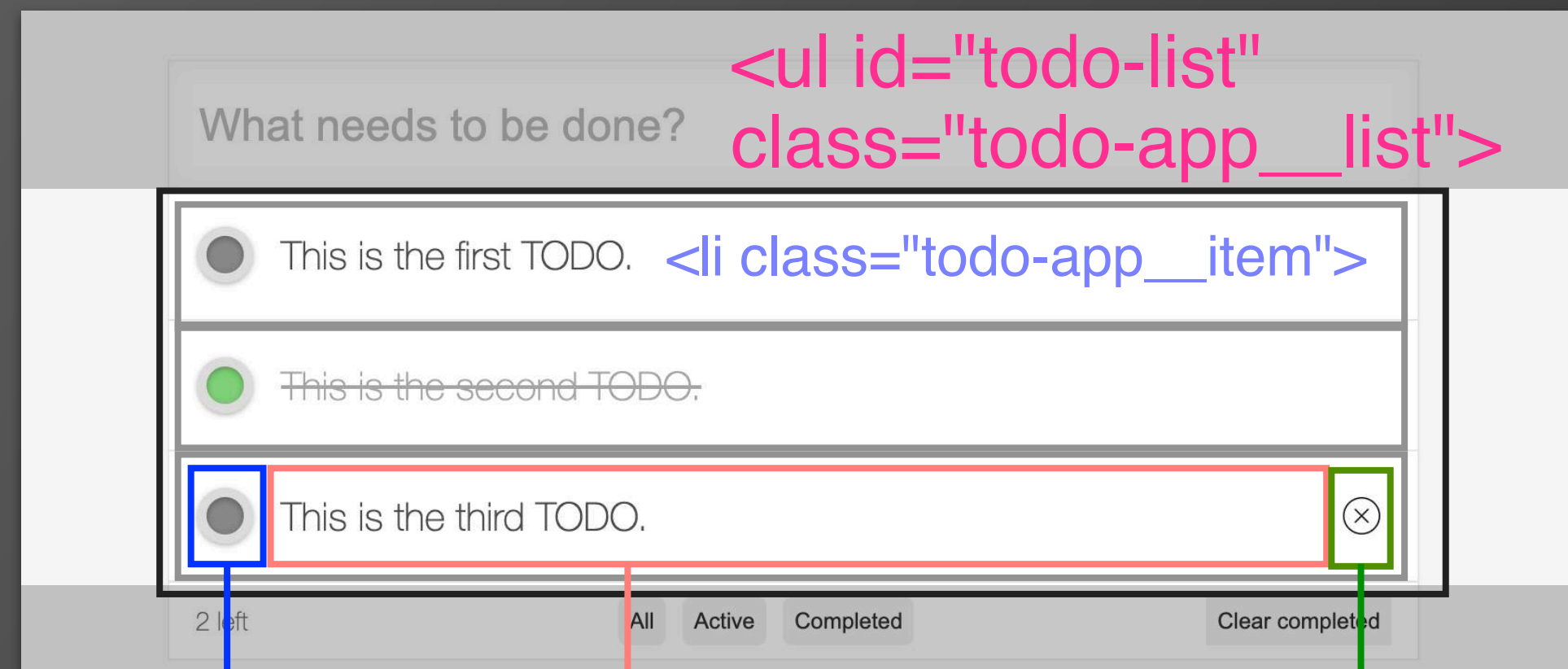
```
<li class="todo-app__item">
  <div class="todo-app__checkbox">
    <input id="0" type="checkbox" />
    <label for="0"></label>
  </div>
  <h1
    class="todo-app__item-detail">
    This is the first TODO.
  </h1>
  
</li>
```

- 如何讓 mouse over todo-app\_\_item 時, 才讓  出現, 然後當 mouse over  時, 會讓他微微變大。


- 利用 pseudo-class selector ":hover"

```
.todo-app__item-x {
  display: none; // 看不見, 但會佔位置
}
.todo-app__item:hover
  .todo-app__item-x {
    display: inline;
    position: absolute;
    max-width: 20px;
    max-height: 20px;
    right: 20px;
  }
.todo-app__item:hover
  .todo-app__item-x:hover {
    max-width: 25px;
    max-height: 25px;
    right: 17.5px;
  }
```

# CSS for todo-app (4)



```
<li class="todo-app__item">
  <div class="todo-app__checkbox">
    <input id="0" type="checkbox" />
    <label for="0"></label>
  </div>
  <h1
    class="todo-app__item-detail">
    This is the first TODO.
  </h1>
  
</li>
```

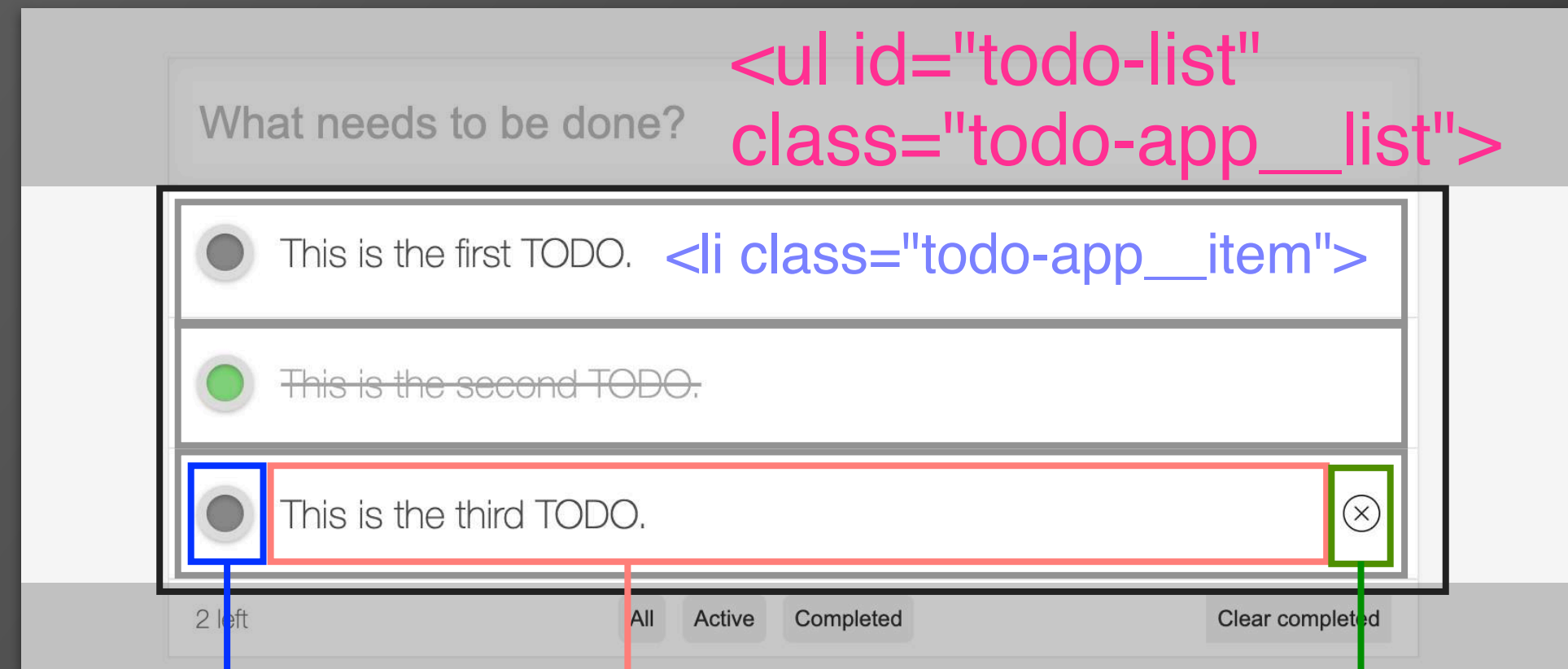
- 如何讓 , TODO detail, 以及  排成一行？
- 設定 todo-app\_\_item 的 flex box

```
.todo-app__item {
  ...
  position: relative;
  display: flex;
  flex-direction: row;
  justify-content: flex-start;
  align-items: center;
}
```

```
.todo-app__item-detail {
  font-weight: 300;
  overflow-wrap: break-word;
  width: 90%;
  transition: opacity 0.3s;
  height: 100%;
}
```



# CSS for todo-app (4)



```
<li class="todo-app__item">
  <div class="todo-app__checkbox">
    <input id="0" type="checkbox" />
    <label for="0"></label>
  </div>
  <h1
    class="todo-app__item-detail">
    This is the first TODO.
  </h1>
  
</li>
```

- 如何讓第二個 todo-app\_\_item 畫上刪除線，並且顏色變淡？
  - 給予另一個 CSS class

```
<h1 id="1detail"
  class="todo-app__item-detail
        todo-app__item-done">
  This is the second TODO.
</h1>
```

```
.todo-app__item-done {
  text-decoration: line-through;
  opacity: 0.5;
}
```

# CSS for todo-app (5)



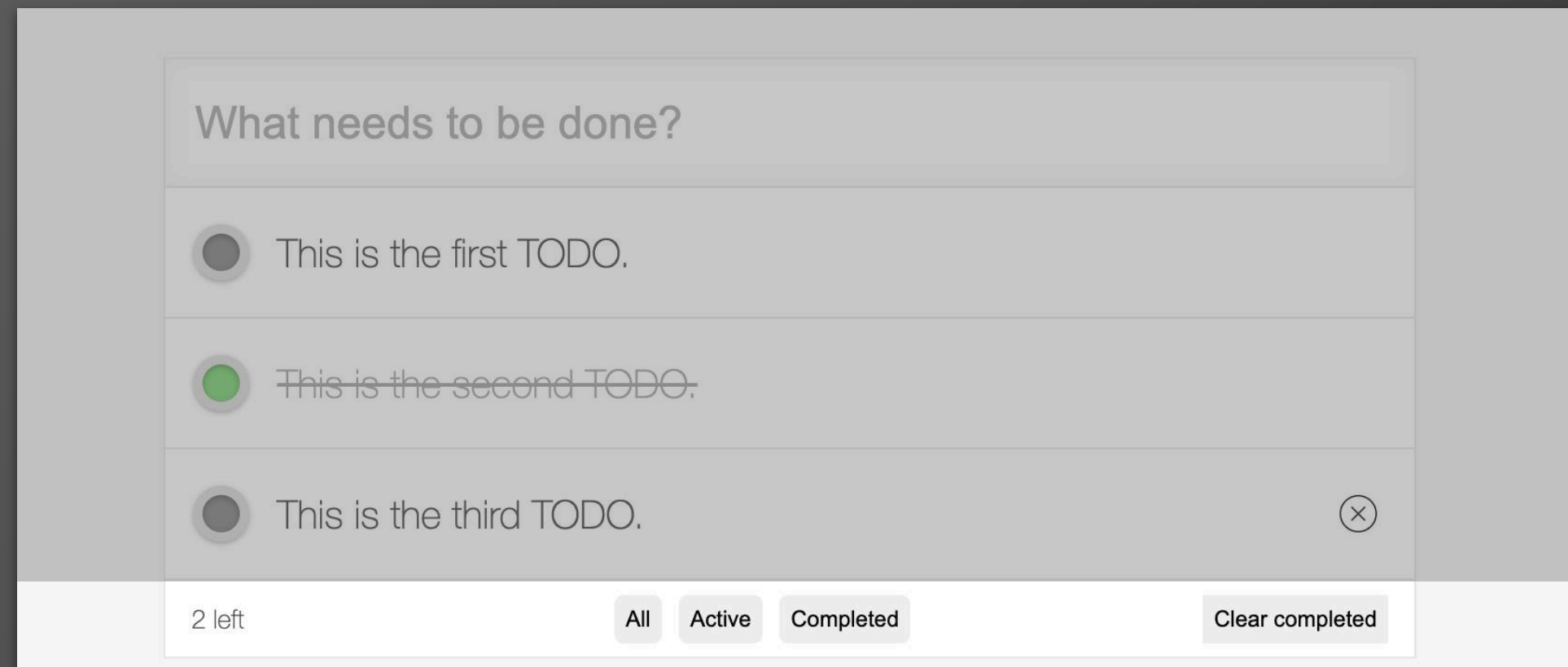
- How to set up the flex box for todo-app\_\_footer?

```
.todo-app__footer {  
  display: flex;  
  flex-direction: row;  
  justify-content: space-between;  
  align-items: center;  
}
```

```
<footer class="todo-app__footer"  
  id="todo-footer">  
  <div class="todo-app__total">  
    <span id="todo-count">2</span>  
    left  
  </div>  
  <ul class="todo-app__view-buttons">  
    <li><button id="todo-all">All</button></li>  
    <li><button id="todo-active">Active</button></li>  
    <li><button id="todo-completed">Completed</button></li>  
  </ul>  
  <div class="todo-app__clean">  
    <button id="todo-clear-complete">Clear completed</button>  
  </div>  
</footer>
```



# CSS for todo-app (5)



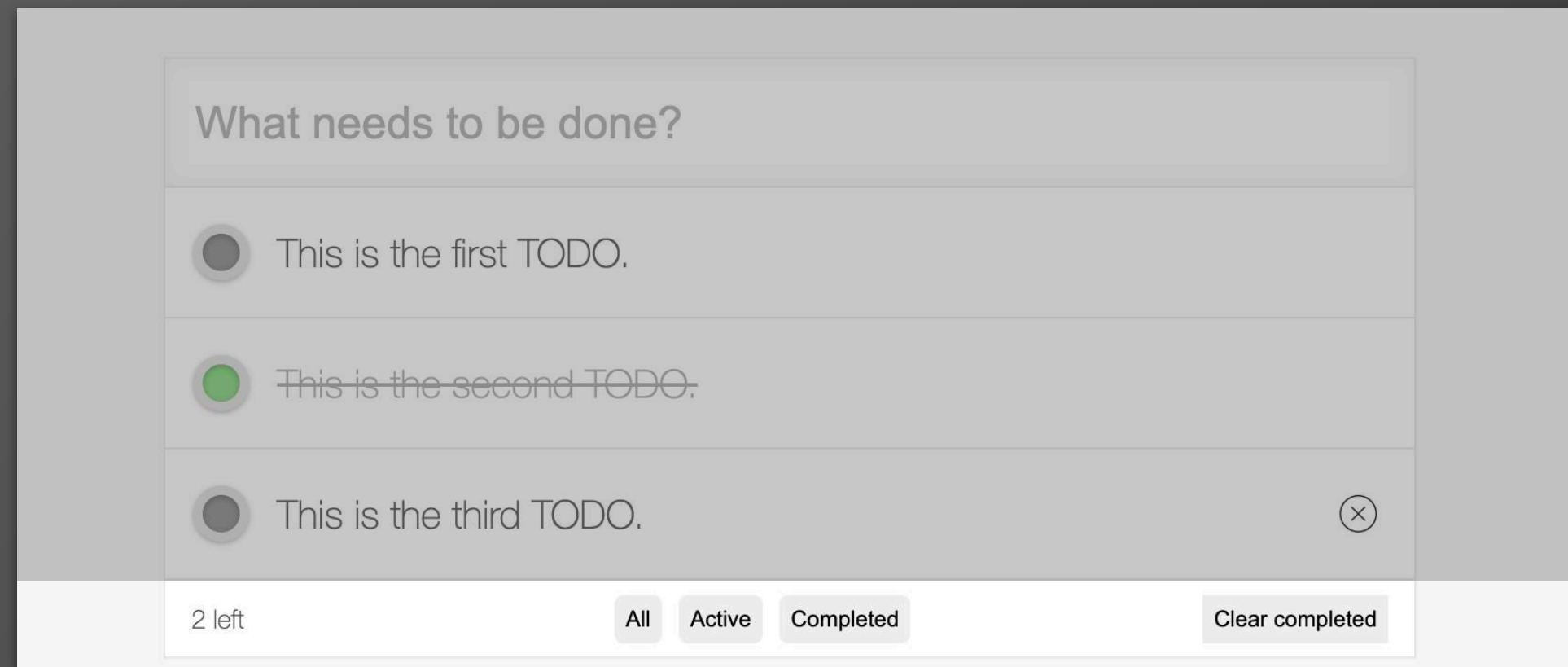
```
<footer class="todo-app__footer"
  id="todo-footer">
  <div class="todo-app__total">
    <span id="todo-count">2</span>
    left
  </div>
  <ul class="todo-app__view-buttons">
    <li><button id="todo-all">All</button>
    <li><button id="todo-active">Active
    <li><button id="todo-completed">Completed
  </ul>
  <div class="todo-app__clean">
    <button id="todo-clear-complete">Clear completed
  </div>
</footer>
```

- 完整 CSS code for todo-app\_\_footer (part I) —

```
.todo-app__footer {
  width: 100%;
  height: 3em;
  padding: 1em;
  background: white;
  border: 1px solid
    rgba(0, 0, 0, 0.089);
  border-top: 0.5px solid
    rgba(0, 0, 0, 0.089);
  display: flex;
  flex-direction: row;
  justify-content: space-between;
  align-items: center;
}

.todo-app__total {
  min-width: 3em;
}
```

# CSS for todo-app (5)



```
<footer class="todo-app__footer"
  id="todo-footer">
  <div class="todo-app__total">
    <span id="todo-count">2</span>
    left
  </div>
  <ul class="todo-app__view-buttons">
    <li><button id="todo-all">All</button>
    <li><button id="todo-active">Active
    <li><button id="todo-completed">Completed
  </ul>
  <div class="todo-app__clean">
    <button id="todo-clear-complete">Clear completed
  </div>
</footer>
```

- 完整 CSS code for todo-app\_\_footer (part II) —

```
.todo-app__view-buttons {
  margin: 0;
  margin-left: 2em;
  padding: 0;
  width: 12em;
  list-style: none;
  display: flex;
  flex-direction: row;
  justify-content: space-around;
  align-items: center;
}
.todo-app__view-buttons button {
  border: none;
  padding: 5px;
  font-size: 0.8em;
  font-weight: inherit;
  border: 1px solid transparent;
  border-radius: 5px;
  cursor: pointer;
}
.todo-app__view-buttons button:focus {
  outline: none;
}
```



# CSS for todo-app (5)



```
<footer class="todo-app__footer"
  id="todo-footer">
  <div class="todo-app__total">
    <span id="todo-count">2</span>
    left
  </div>
  <ul class="todo-app__view-buttons">
    <li><button id="todo-all">All</but
    <li><button id="todo-active">Activ
    <li><button id="todo-completed">Co
  </ul>
  <div class="todo-app__clean">
    <button id="todo-clear-complete">C
  </div>
</footer>
```

- 完整 CSS code for todo-app\_\_footer (part III) —

```
.todo-app__clean {
  vertical-align: middle;
}

.todo-app__clean button {
  border: none;
  padding: 5px;
  border: 1px solid transparent;
  font-size: 0.8em;
  font-weight: inherit;
  cursor: pointer;
}

.todo-app__clean button:focus {
  outline: none;
}

.todo-app__clean button:hover {
  text-decoration: underline;
}
```

# CSS • Filter and Box-shadow Properties

Syntax —

- filter: none | blur() | brightness() | contrast() | drop-shadow() | grayscale() | hue-rotate() | invert() | opacity() | saturate() | sepia() | url();
- box-shadow: none | h-offset v-offset blur spread color | inset | initial | inherit;



## CSS • @keyframes and Animation

```
@keyframes rulename { /* 定義動畫規則 */  
    40% { transform: rotate(0deg); }  
    45% { transform: rotate(-5deg); }  
    55% { transform: rotate(5deg); }  
    60% { transform: rotate(0deg); }  
}  
someSelector {  
    animation: rulename 3s infinite;  
}
```

# Responsive Web Design (RWD) ([ref](#))

- 簡單的說，就是隨著瀏覽器放大、縮小，或者是在不同的載具上面 (PC, notebook, cell phone, pad)，你的網頁排版都不應該爛掉！
- 區塊重新排版、文字自動換行，圖片自動放大縮小, etc



# RWD – Practice Example

- Edit an "index.html"

```
<head>
  <meta charset="utf-8">
  <style>
    body { text-align: center; }
  </style>
</head>
<body>
  <h3> Love </h3>
  <p>
    Love is real, real is love <br>
    Love is feeling, feeling love<br>
    Love is wanting to be loved<br>
  <br>
```

```
Love is touch, touch is love<br>
Love is reaching, reaching love<br>
Love is asking to be loved<br>
<br>
Love is you<br>
You and me<br>
Love is knowing<br>
we can be<br>
<br>
Love is free, free is love<br>
Love is living, living love<br>
Love is needing to be loved<br>
<br>
<p> by <em> John Lennon </em> </p>
</body>
</html>
```

## RWD — Getting Started

- To start an RWD design, you need to first include a meta viewport tag in the head of the document.

```
<meta name="viewport" content="width=device-width,  
initial-scale=1">
```

- Include "width=device-width" to match the screen's width in device-independent pixels.
- Include "initial-scale=1" to establish a 1:1 relationship between CSS pixels and device-independent pixels.



## RWD — Getting Started

- **Media queries** allow the page to use different CSS style rules based on characteristics of the device the site is being displayed on, most commonly the width of the browser.
- The **fluid grid** concept calls for page element sizing to be in relative units like percentages, rather than absolute units like pixels or points.
- **Flexible images** are also sized in relative units, so as to prevent them from displaying outside their containing element.

# RWD — @media queries

```
@media (query) {  
  /* CSS Rules used when query matches */  
}
```

<b>min-width</b>	Rules applied for any browser width greater than the value defined in the query.
<b>max-width</b>	Rules applied for any browser width less than the value defined in the query.
<b>min-height</b>	Rules applied for any browser height greater than the value defined in the query.
<b>max-height</b>	Rules applied for any browser height less than the value defined in the query.
<b>orientation=portrait</b>	Rules applied for any browser where the height is greater than or equal to the width.
<b>orientation=landscape</b>	Rules for any browser where the width is greater than the height.



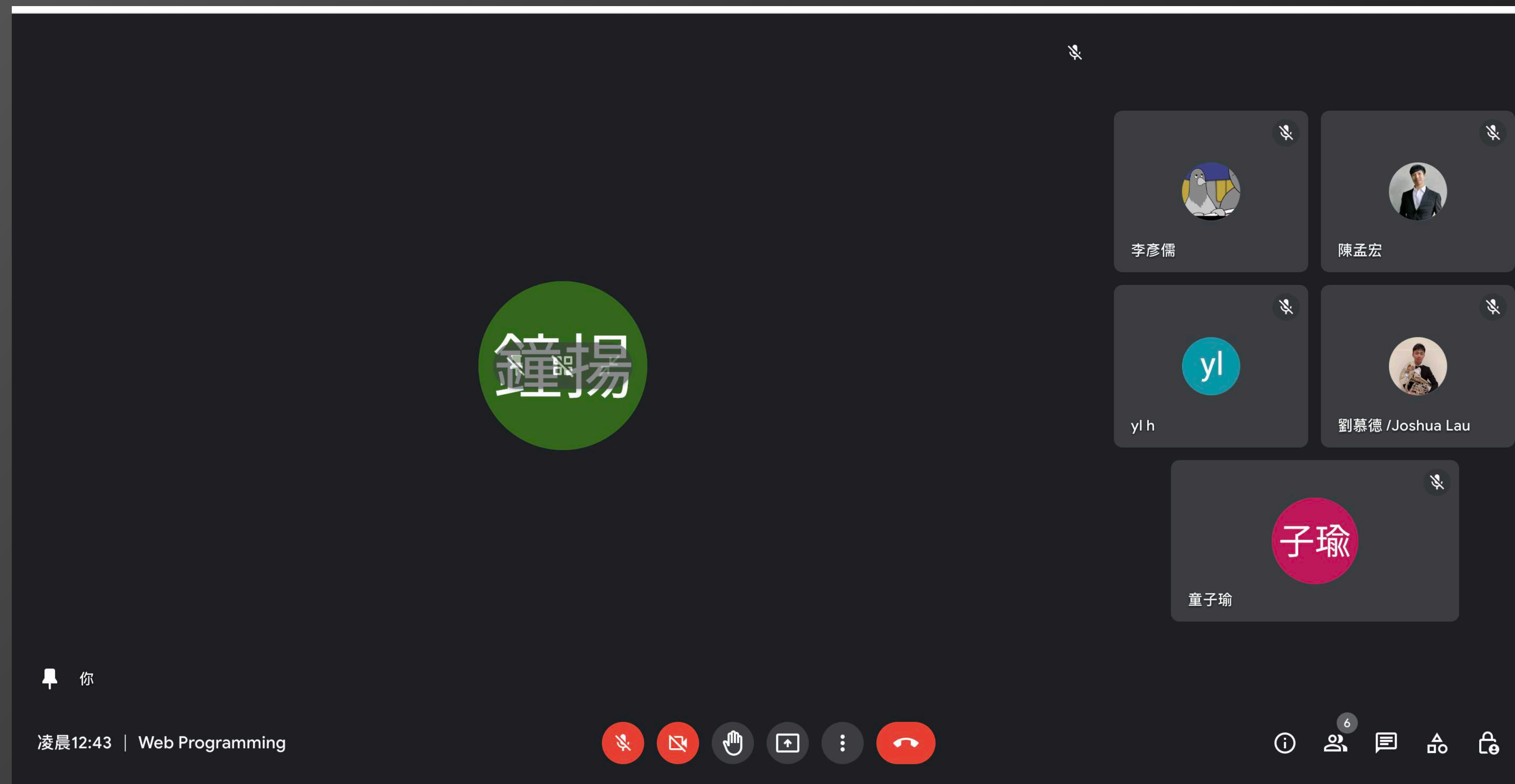
# RWD – Practice Example

```
@media (max-width: 400px) {  
  h3 {  
    color: DodgerBlue;  
    font-size: 16px;  
  }  
  p {  
    color: DodgerBlue;  
    font-size: 12px;  
  }  
}  
  
@media (max-width: 600px) {  
  h3 {  
    color: Tomato;  
    font-size: 24px;  
  }  
  p {  
    color: Tomato;  
    font-size: 16px;  
  }  
}
```

試著把這些 code 放到適當的地方...

# Take-Home Practice #1

- Design a (fake) static Google Meet webpage using HTML and CSS



# 感謝聆聽！

Ric Huang / NTUEE

(EE 3035) Web Programming

© 2022 - Ric Huang ALL RIGHTS RESERVED