Web Programming 03/07

* Introduction to Web Programming
* HTML/CSS/JavaScript Basics
* Document Object Model (DOM)
* Wireframe

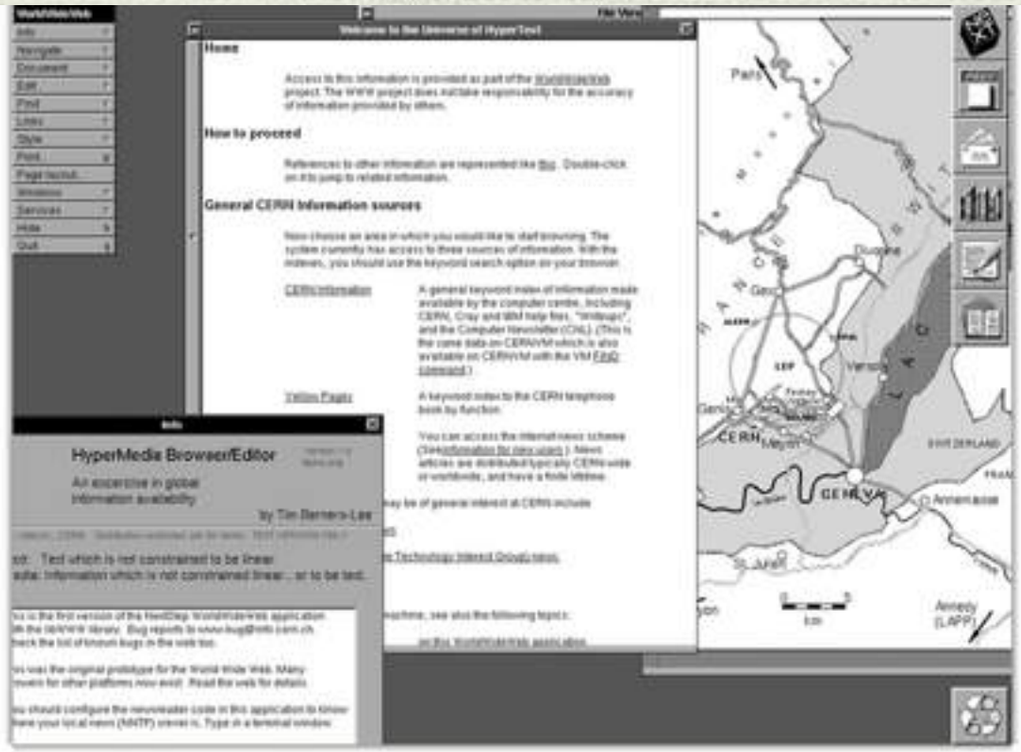
Web ◦ Programming

- 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(WebBrowser)、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

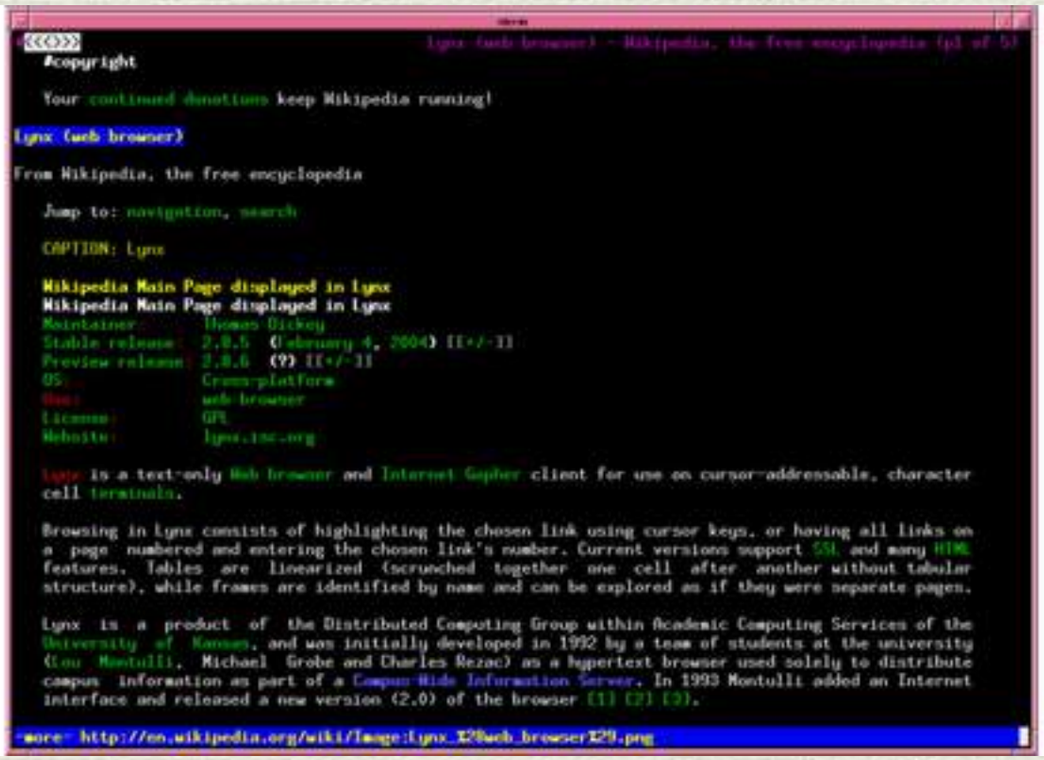
前端。後端

- 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)

前端。瀏覽器 (Web Browser)



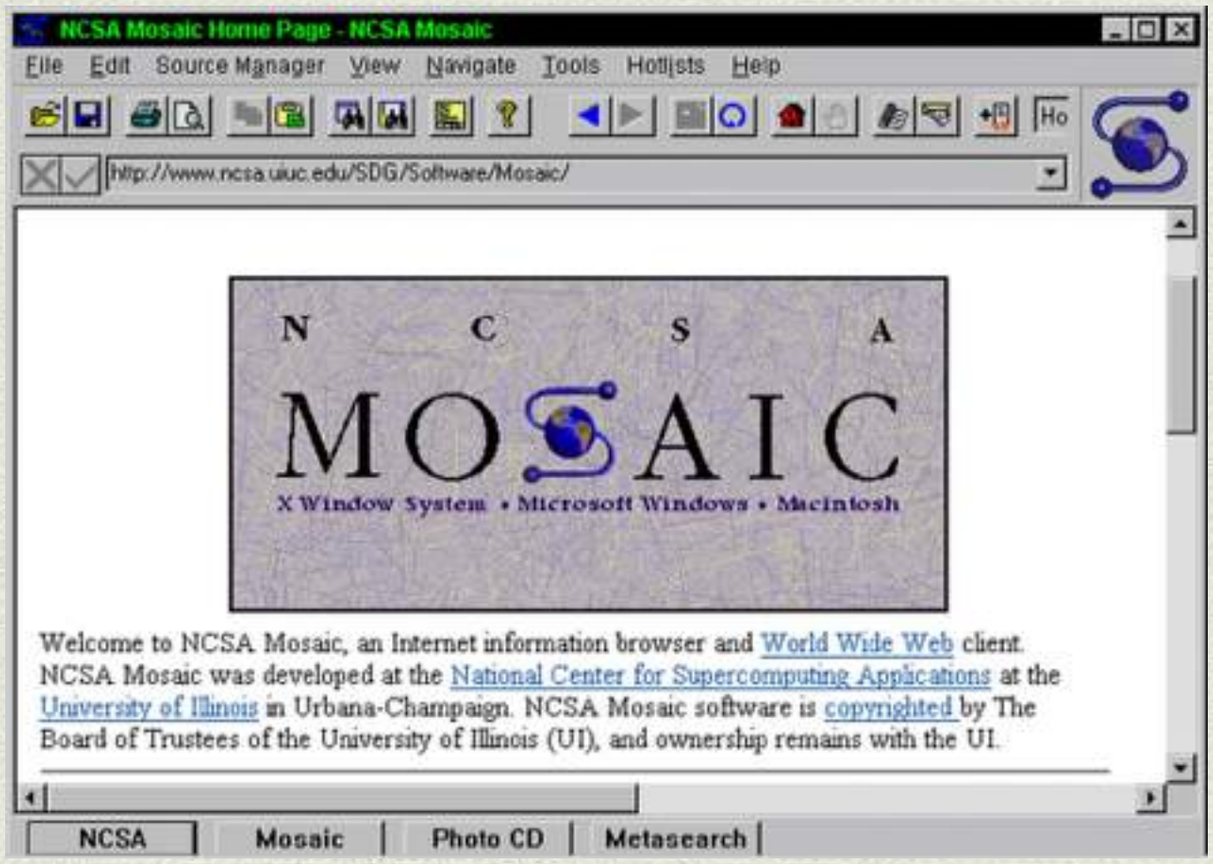
- First web browser: WorldWideWeb (aka. Nexus), by Tim Berners Lee, 1990



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

Ref: https://www.telegraph.co.uk/technology/microsoft/11577364/Web-browsers-a-brief-history.html

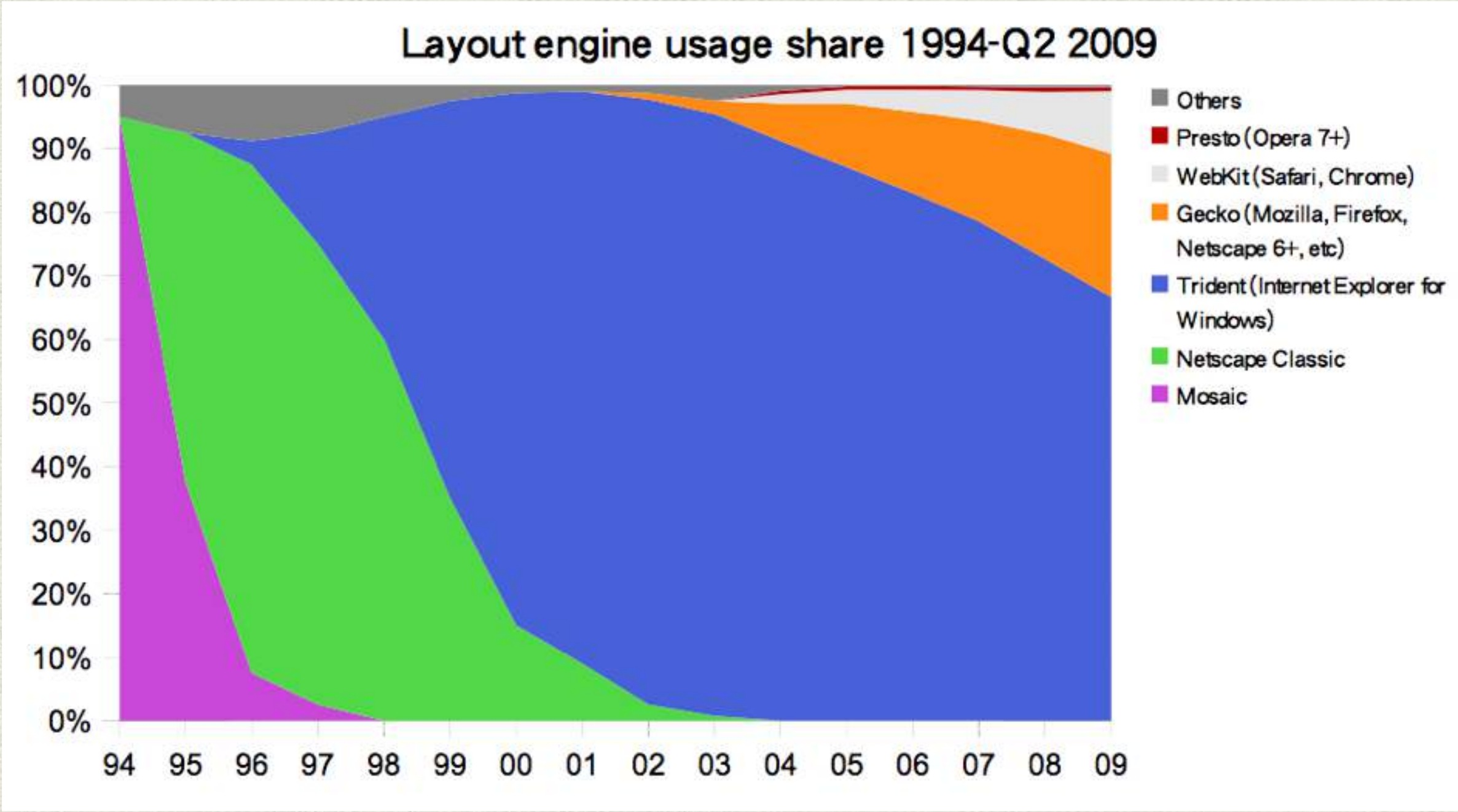
前端。瀏覽器 (Web Browser)



- Mosaic, often referred 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 90's

前端。瀏覽器 (Web Browser)

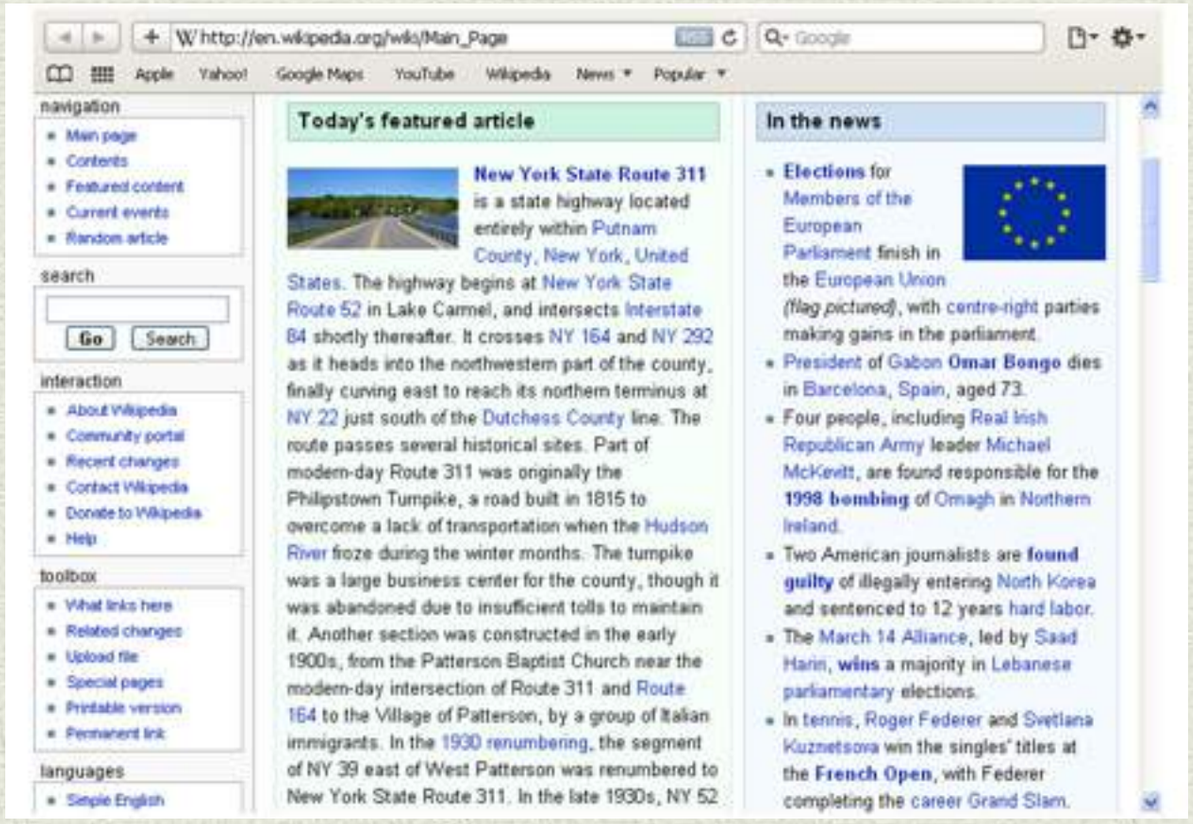


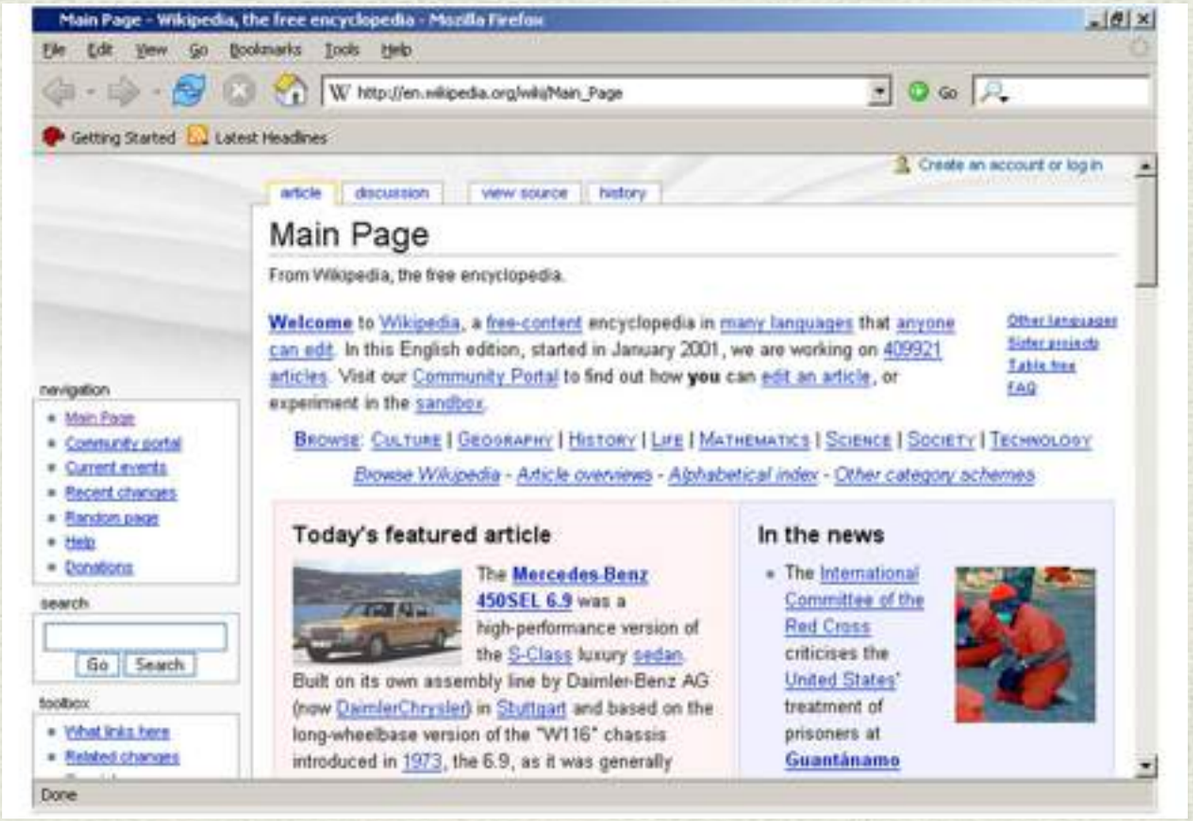
前端。瀏覽器 (Web Browser)

- Internet Explorer (IE), by Microsoft, in 1995

- Opera, by Telenor, in 1996

前端。瀏覽器 (Web Browser)

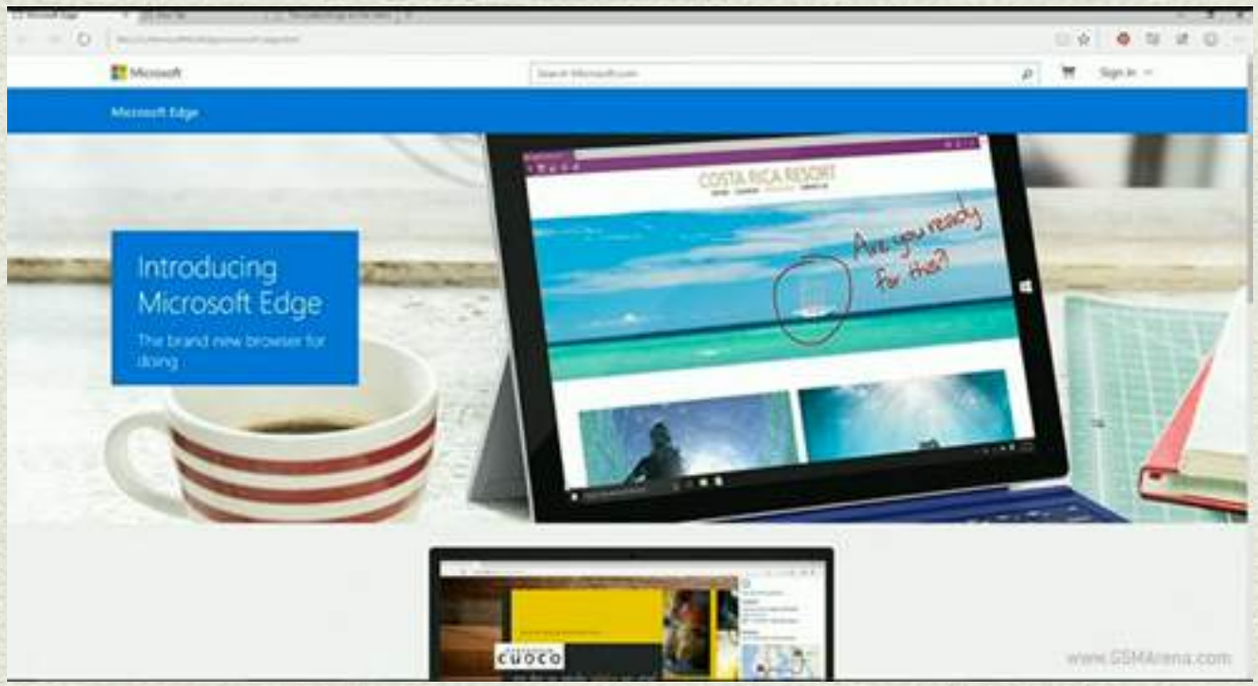




- Safari, by Apple, in 2003
- Firefox, by Mozilla, in 2004

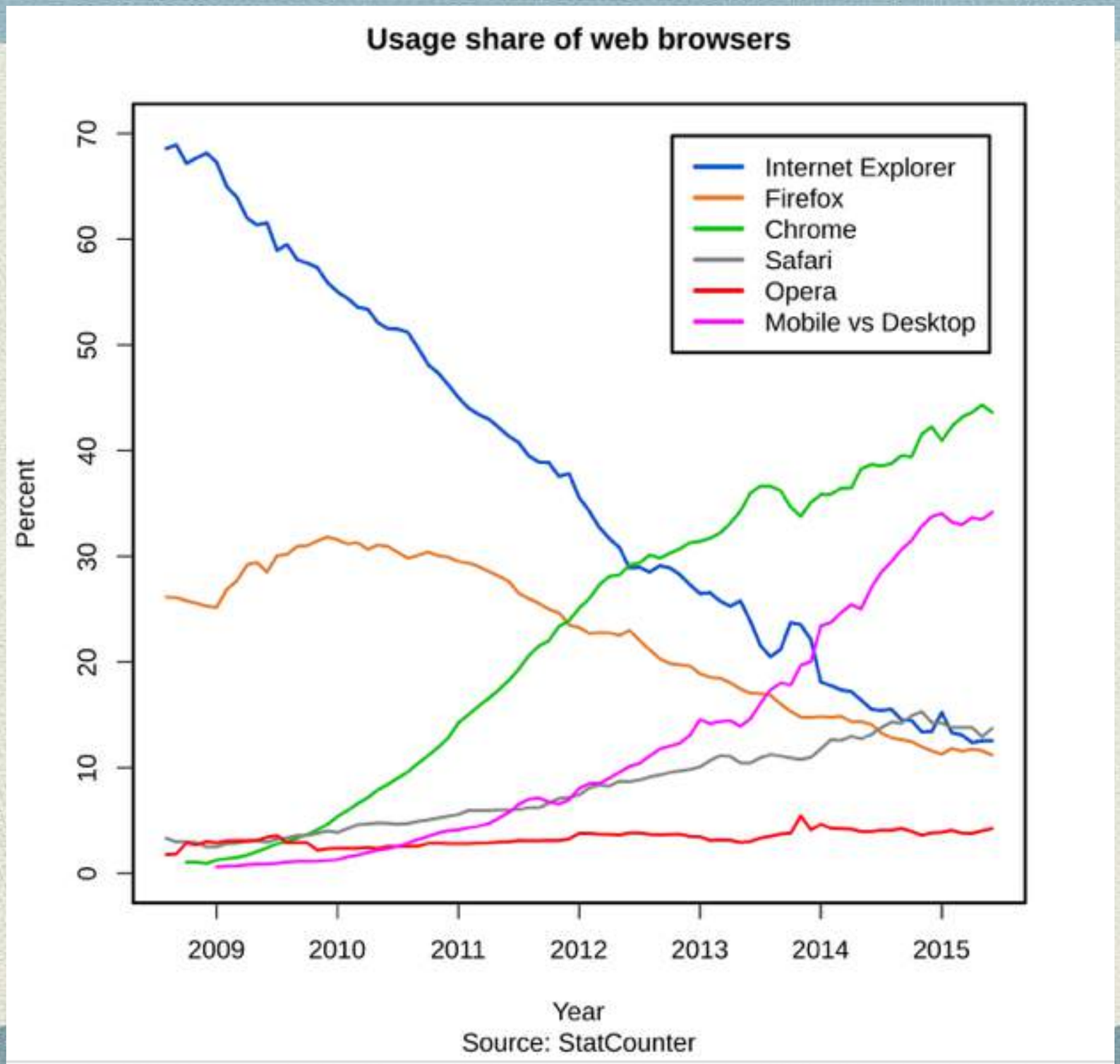
前端。瀏覽器 (Web Browser)

- Chrome, by Google, in 2008



- Edge, by Microsoft, in 2015

前端。瀏覽器 (Web Browser)

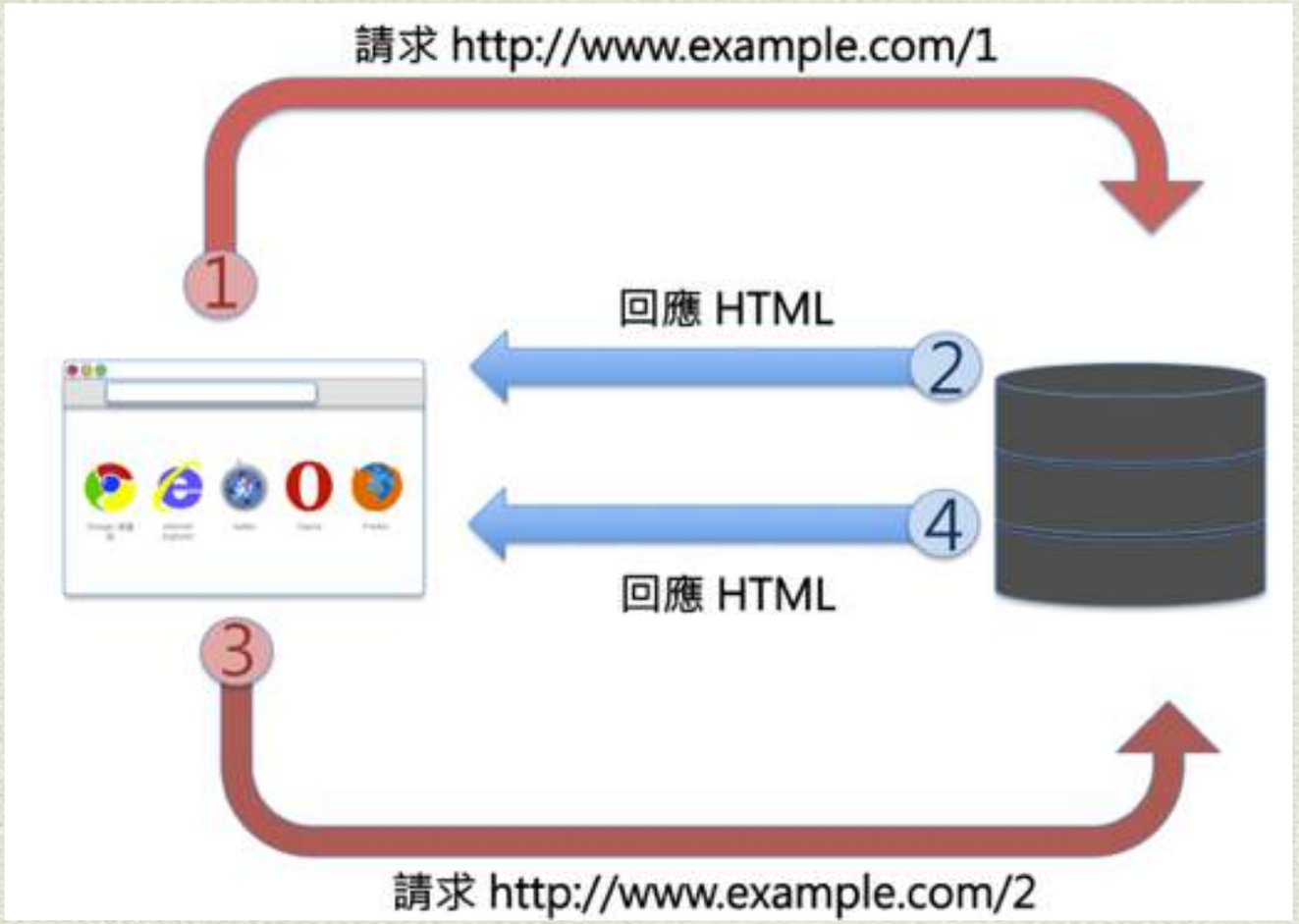


前端。網頁

- 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

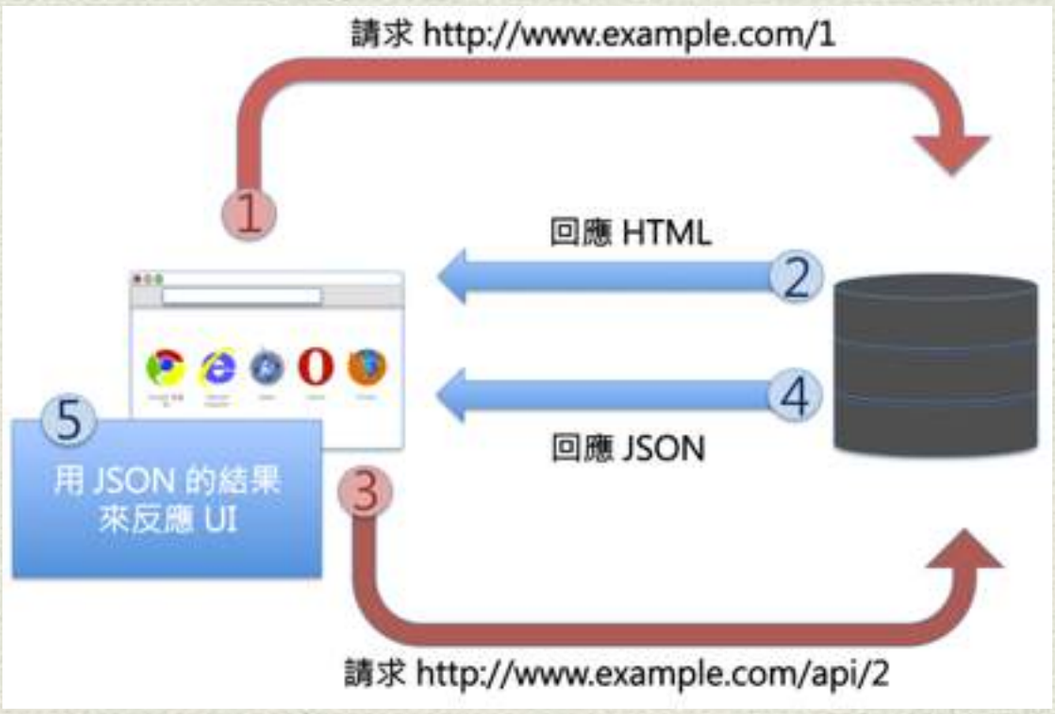
靜態。動態網頁

- **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，是跳掉別的網址，網頁內容會重新下載
- 換頁時會有「白畫面」



靜態。動態網頁

- **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)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

Ref: http://indus3.org/json-and-game-data/#.WpuZ1xNuYnM

Ajax:
Asynchronous JavaScript and XML

- XML vs. JSON

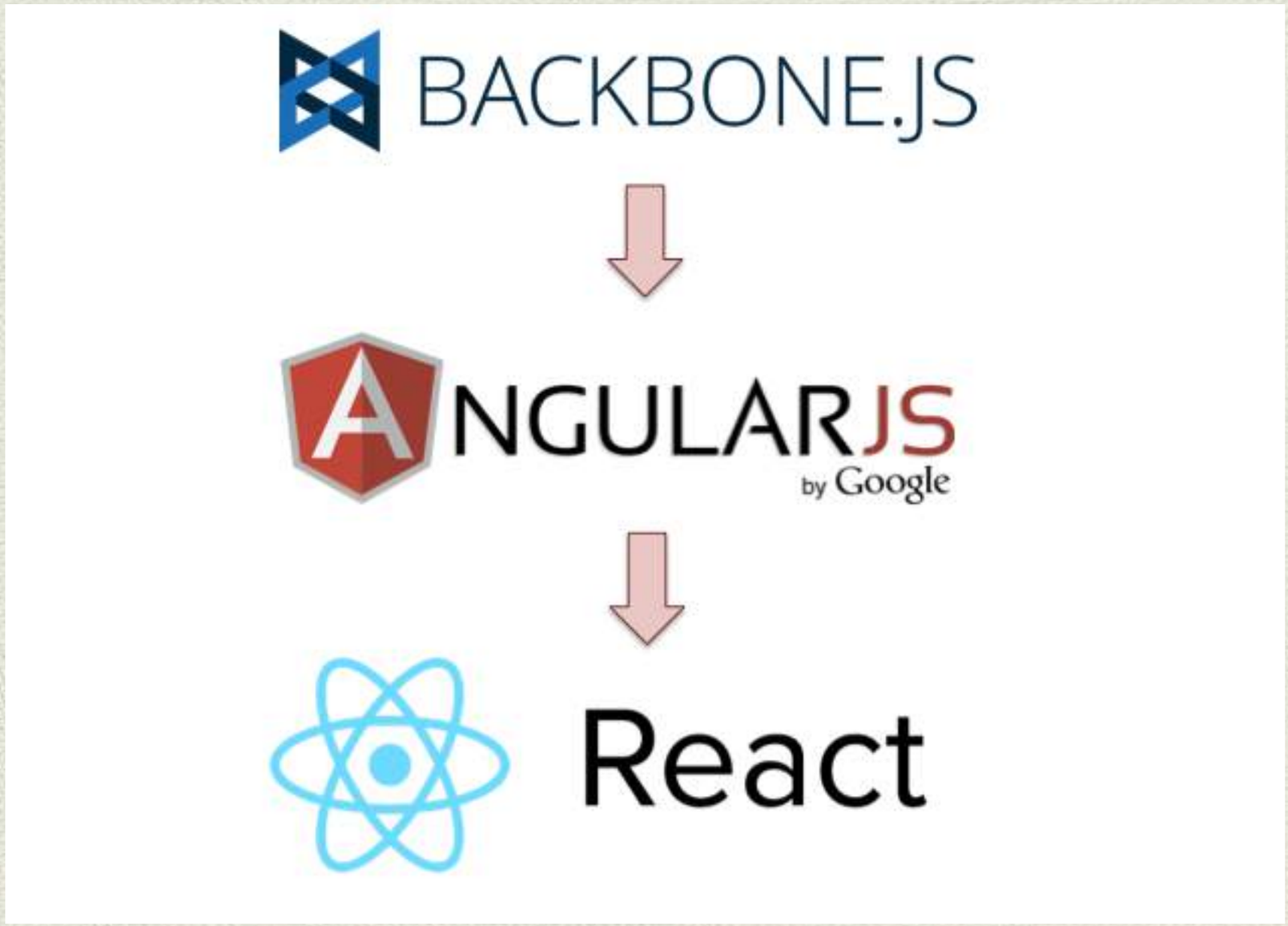
```
<?xml version="1.0" encoding="iso-8859-1" ?>
- <department>
  - <employee>
      <name>John Doe</name>
      <job>Software Analyst</job>
      <salary>2000</salary>
    </employee>
  - <employee>
      <name>Jane Fletcher</name>
      <job>Designer</job>
      <salary>2500</salary>
    </employee>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

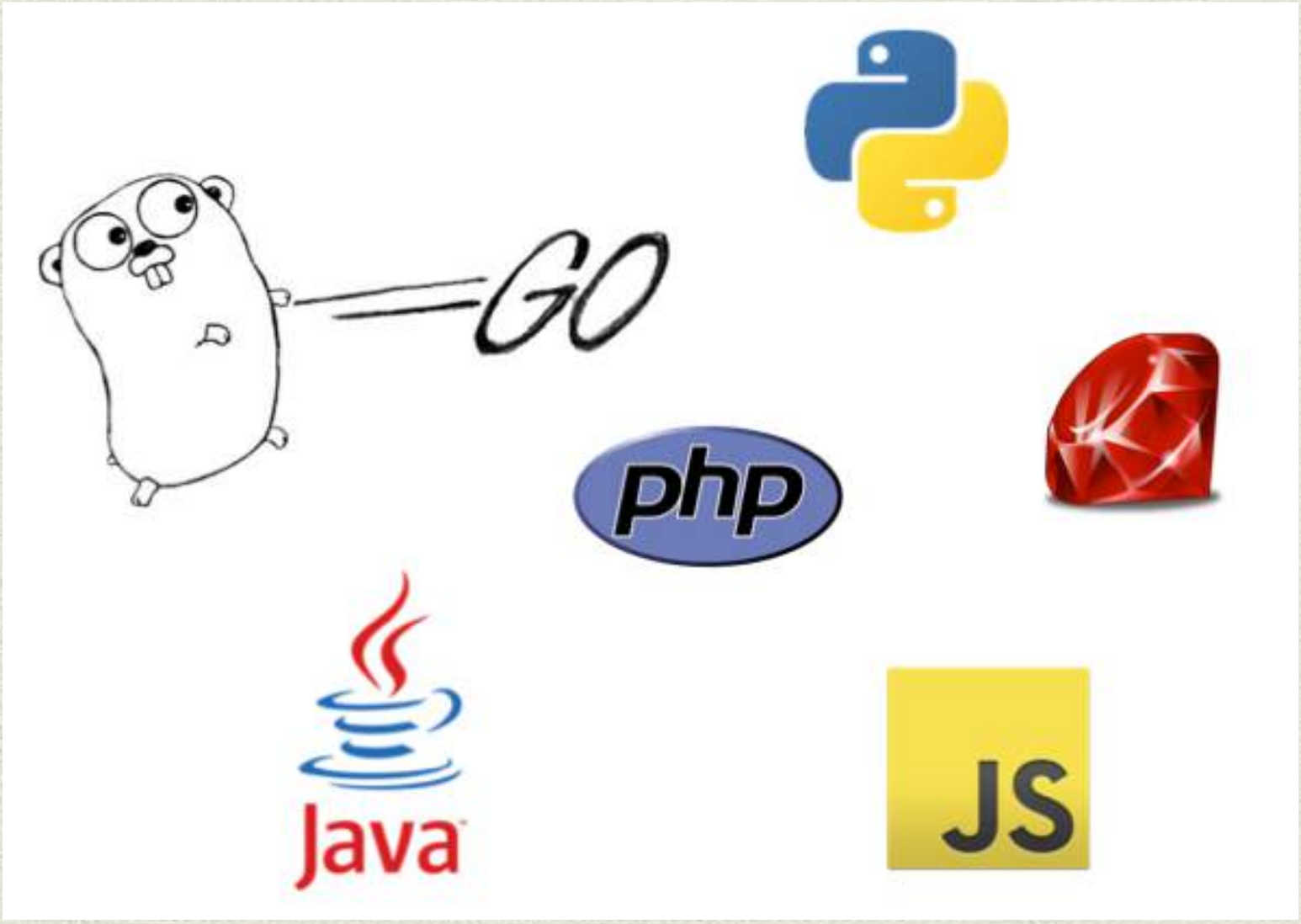


React.js。元件式開發 + 狀態決定 UI

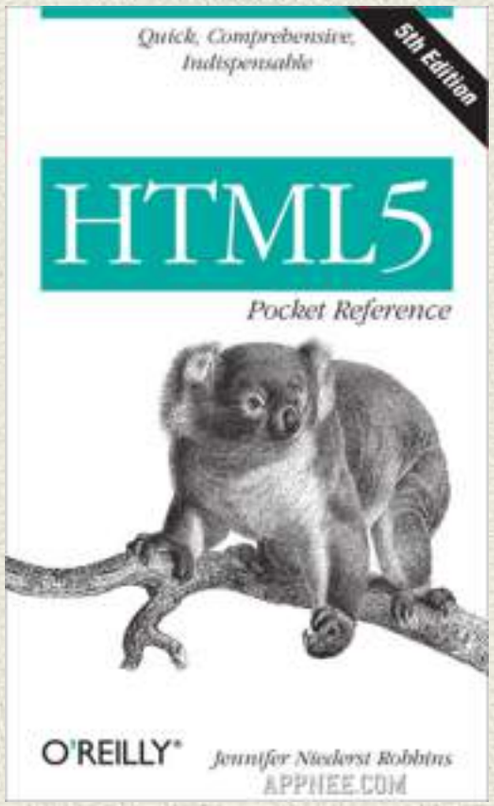
- Component-based implementation with states —
  - 以前：某個 UI 呈現出來的 bug 可能要數個操作步驟來重現，debug 過程可能會讓你很崩潰
- With React.js components/states：
  ```
  var AppUI = AppComponent(state);
  ```
  => 讓資料儲存在單一地方 (i.e. single source of truth)，所有的改變會立即反應在 component 上面
  => 除了 debug 較為輕鬆之外，頁面更新的速度也可以變快 (more to cover later)
  // more to cover in later meetings

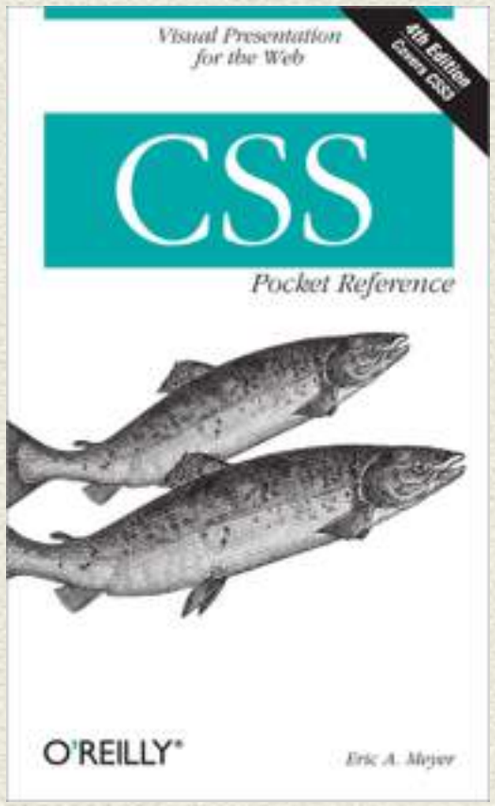
後端語言更是百花齊放~

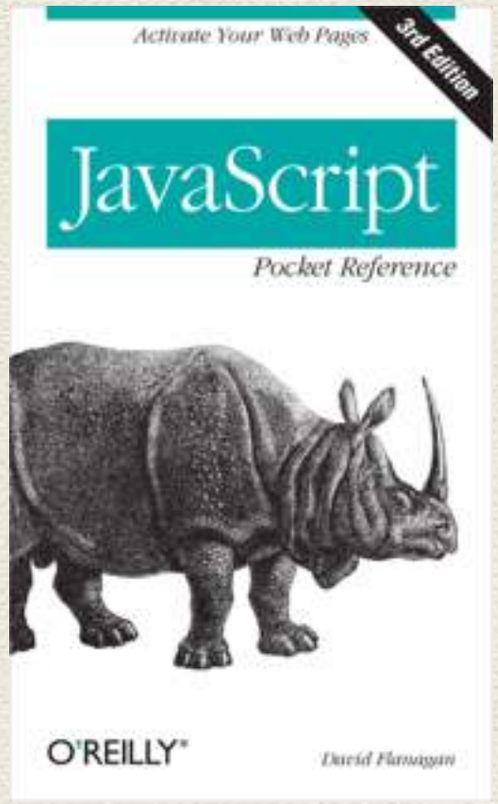
- 可以跑在伺服器 (電腦) 的程式語言都可以用



HTML/CSS/JavaScript Basics



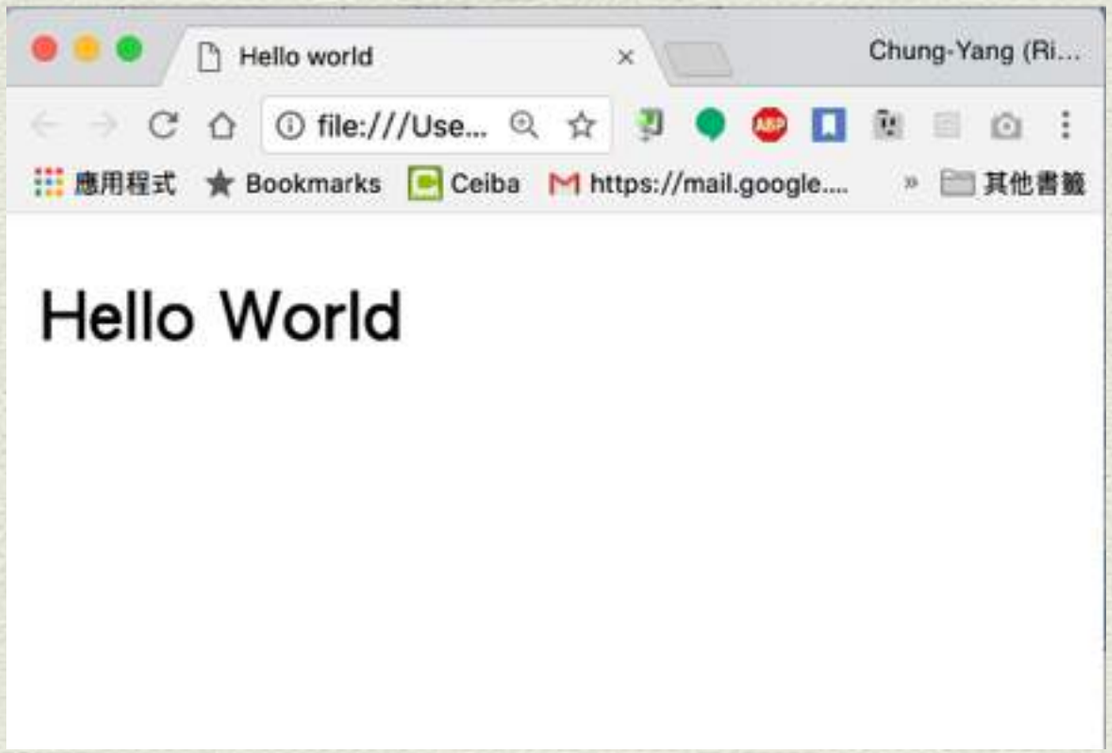




HTML (HyperText Markup Language)

Source: hello.html

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Hello world</title>
  </head>
  <body>
    <p> Hello World </p>
  </body>
</html>
```



- Try….
  - 在 “Hello World” 前後加入空白或是換行
  - 試試看 , 等其他 text elements

HTML (HyperText Markup Language)

[Syntax]

- For most of the tags…
  - `<tag attribute1="value" attribute2="value">`
    `Some text to display </tag>`
- Some exception (no closing tag)
  - `<img src="someURL" alt="someText">`
- 字串(value)可以用單引號，也可以用雙引號
- Comments are in <!--...-->

HTML Elements (1/5)

- **Main root**
  - **<html>**
- **Document metadata** (in <head> section)
  - **<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
- **Sectioning root**
  - **<body>** // **<head>** is optional

Ref: https://developer.mozilla.org/en-US/docs/Web/HTML/Element

HTML Elements (2/5)

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

Ref: https://developer.mozilla.org/en-US/docs/Web/HTML/Element

HTML Elements (3/5)

- **Inline text semantics**
  - **<a** href="…"**>**: hyper link
  - ****
  - ****, **<i>**, **<mark>**, **<q>**, **<s>**, ****
  - **<cite>**, **<code>**, **<time>**
  - **
**
- **Image and multimedia**
  - **<img** src="…" alt="…"**>**
  - **<audio>**, **<video>**, **<track>**
  - **<map>**, **<area>**: to define image map

Ref: https://developer.mozilla.org/en-US/docs/Web/HTML/Element

HTML Elements (4/5)

- **Scripting**
  - **<script** type="text/javascript" src="someSource.js"**>**
  - **<noscript>**: in case a script type is unsupported
  - **<canvas>**: for canvas or webGL APIs
- **Table content**
  - **<table>**
  - **<caption>**
  - **<th>, <tr>, <td>**: header, row, data cell
  - **<map>, <area>**: to define image map
- **Interactive elements**
  - **<dialog>**
  - **<menu>, <menuitem>**

Ref: https://developer.mozilla.org/en-US/docs/Web/HTML/Element

HTML Elements (5/5)

- **Forms**
  - **<form>**
  - **<label** for="…"**>**
  - **<input** type="…" value="…"**>**
  - **<output** name="…"**>**
  - **<button** name="…"**>**
  - **<select>, <option>**
  - **<fieldset>, <datalist>, <optgroup>**
  - **<meter>, <progress>**
- **Embedded content** (e.g. <applet>)

Ref: https://developer.mozilla.org/en-US/docs/Web/HTML/Element

網頁。外觀 / 排版

- 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

Ref: https://developer.mozilla.org/en-US/docs/Web/HTML/Attributes

CSS — Cascading Style Sheets

- CSS「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
  - rule ≡ selector { property: value; /* more properties...*/ }
  - HTML 檔案裡頭 element 的樣式就由這些 rules 來定義
- Precedence issue：如果 rules 之間有重疊或是衝突怎麼辦？
  - Rule with more specific selector > less specific selector
  - 後面的 rule > 前面的 rule
- Comments are in /* … */
- CSS playground: http://dabblet.com/

CSS — Cascading Style Sheets

- CSS 有三種應用方式
  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
     -

CSS 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 both class names
   - .class1 { }
   - .class1.class2 { }
2. Select by the tag name
   - div { }
3. Select by its id
   - #anID { }

Ref: https://learnxinyminutes.com/docs/css/

CSS Selectors

4. Select by the fact that it has an attribute
   - [attr] { }
5. the attribute has a specific value
   - [attr='value'] { }
6. starts or ends with a value
   - [attr^='val'] { }
   - [attr$='ue'] { }
7. contains a value in a space-separated list
   - [otherAttr~='foo'] { }
8. contains a value in a dash-separated list
   - [otherAttr|='en'] { }

Ref: https://learnxinyminutes.com/docs/css/

More Advanced CSS Selectors

1. Combine different selectors to create a more focused selector
   /* no space is allowed */
   - `div.class1[attr$='ue'] { }`
2. Select an element which is a child (direct descendant) of another element
   - `div.some-parent > .class-name { }`
3. Select a descendant of another element
   /* note a space between parent and its decedent */
   - `div.some-parent .class-name { }`
4. Select an element based on its adjacent sibling
   - `.i-am-just-before + .this-element { }`
5. Select an element based on any sibling preceding it
   - `.i-am-any-element-before ~ .this-element { }`

Ref: https://learnxinyminutes.com/docs/css/

More Advanced CSS Selectors

6. Select an element only when it is in a particular state
   - `selector:someState { }`
     /* hover, visited, link, focus, first-child, last-child, before, after */
7. In certain proper cases, a wildcard (*) can be used to select every element
   - `* { }` /* all elements */
   - `.parent * { }` /* all descendants */
   - `.parent > * { }` /* all children */

Ref: https://learnxinyminutes.com/docs/css/

Defining CSS Selector Properties

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

```css
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 */
```

Ref: https://learnxinyminutes.com/docs/css/

Defining CSS Selector Properties

```
    /* 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;
}
```

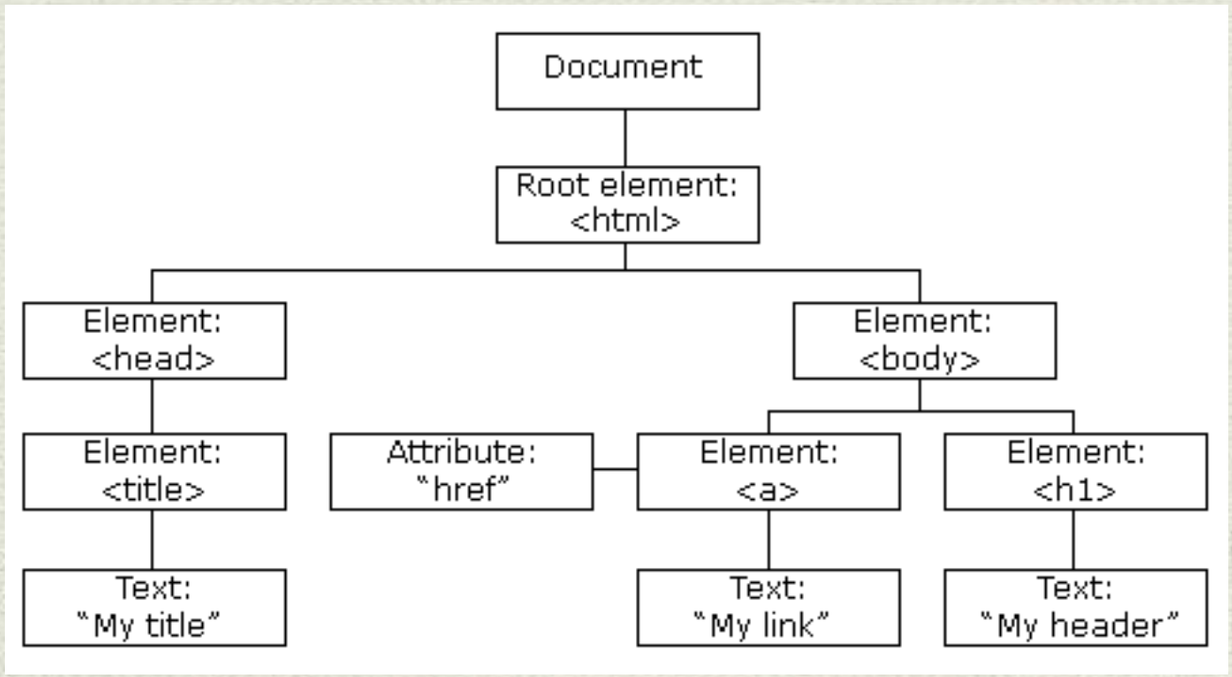
Ref: https://learnxinyminutes.com/docs/css/

在學習完 HTML & CSS 之後，你應該是有能力可以刻出一個靜態網頁(雖然 CSS 的 state selector 也是可以做出一定程度的動態網頁啦！)

但為了接下來理解如何利用 JavaScript 製作出動態網頁，我們必須先瞭解何謂 DOM (Document Object Model)

DOM (Document Object Method)

- (Ref: 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.
- JavaScript 就是去 listen event 來操作 DOM 的內容，以達到動態網頁的目的
- 試試看！



JavaScript Basics

- console.log()
- Types / Variables / Objects
- Function
- Array
- Control statements
- DOM manipulation & Event listening

JavaScript。與 C++ 比較

- 由於 Brendan Eich 當初在設計 JavaScript 的時候是參考 C/C++ 語言，希望可以降低語言切換時的學習門檻，因此，許多的語法跟 C/C++ 基本上是一模一樣(有些部分是像 Java)。
- 不一樣的地方 (舉例)：
  - 弱型別，所有的變數都是 var (or let/const)
    - var a = “38”; var b = false; console.log(a + !b);
- Statement 的後面可以不用加 “;”
- 數字都是 “double“
  - 0.1 + 0.2; / / = 0.30000000000000004
- 除了 ==, 還有 ===
- “prototype-based” object construction. 不用先定義 class 即可建置物件，且一旦一個物件被建置好，後續的物件可以用它來當作原型來建置類似的物件
- ”first-class functions“，也就是說 functions 被當成是一般的變數(物件)，可以當成其他 function 的參數或是回傳值，也可以被 assigned 給別的變數

JavaScript ◦ console.log()

- console.log() 是你 debug 的好朋友，會將訊息印在 “console”
  - (e.g.) In Chrome on Mac, press “command-option-i”
  - var a = 1;
    console.log(a);

JavaScript。Types / Variables / Objects

- JavaScript 只有五種內建型別(primitive types)，其他都是物件(objects)

型態	範例
Undefined (未定義)	undefined
Null (空值)	null
String (字串)	“哎呀”
Boolean (布林值)	true, false
Number (數字)	3.1415926

JavaScript ◦ More about Objects

- “In JavaScript, almost everything is an object.”
  - var a = new Number(3); // 不建議
    var b = Number(3); // What's the difference?
  - 內建物件型別：Date, Math, Function, Event, Array, RegEx, Error…
- “Object” in an object // Recall: “prototype-based” object construction
  - var o = new Object;
    o.name = “Ric”;
    o.score = 100;
  - console.log(o);

JavaScript ◦ Creating Objects

- 三種產生 object 的方法

1. Object literal
   - var a = { name:”Ric”, score:100 };
2. “new” operator
   - var b = new Date;
3. Constructor function
   - function Student(name, score) {
     this.name = name;
     this.score = score;
     }
   - var c = new Student(“Ric”, 100);
   - What happens if we do “var c = Student(“Ric”, 100);”?

JavaScript ◦ Assignment

- Compare these two different assignments…

1. Primitive variable assignment makes a "copy"
   - var a = 3;
     var b = a; // {a, b} = {3, 3}
     b = 4; // {a, b} = {3, 4}
     a = 5; // {a, b} = {5, 4}
2. Object variable assignment pass the "reference" to the object's address
   - var i = { a:3 };
     var j = i;
     j.a = 4; // i.a = 4
     i.a = 5; // j.a = 5

- var a =3; // a is a primitive
  var b = Number(3); // b is a primitive
  var c = new Number(3); // c is an object

JavaScript ◦ Function as an Object

- Recall: Functions in JavaScript is “first-class”. Function as an object.
- Function 的 arguments 沒有必要加上 “var”
  - function add(a, b) { return a + b; }; // NOT: function add(var a, var…)
- Function 的 assignment 如同 object assignment (i.e. pass by reference)
  - var f = function add(a, b) { return a + b; }; // f is an reference to add
  - console.log(f(3,4)); // print out the returned value of f(3,4)
    console.log(f); // print out f itself
    console.log(add); // Error (why?)
- Function can be anonymous
  - var add = function(a, b) { return a + b; }; // recommended

JavaScript ◦ Return a function

- Compare —
  - var f = function(a, b) { return a + b; };
    f(3,5);
    f(f(1,2),3);
  - var f = function(s) {
        return s? function(a,b) { return a+b; }:
                  function(a,b) { return a-b; } };
    var f1 = f(true);
    f1(3,5);
    var f2 = f(false);
    f2(3,5);

JavaScript ◦ Methods in Objects

- Recall: “prototype-based” object construction.
  - So, this is OK —
    > var a = { name:”Ric”, score:100 };
    > a.gender = “Male”;
- Since function is also an object, we can define “method functions” in an object as:
  - var a = {
      name:”Ric”,
      score:100
      report: function() { console.log(this.name + “ got ” + this.score); }
    };
    a.report(); / / Ric got 100
  - a.isPass = function() { console.log(this.score >= 60? “Yes”: “No”); };
    a.isPass() / / Yes

JavaScript ◦ Array Objects

- To define an array…

1. Using array literal
   - var students = [“John”, “Mary”, “Ric”];
2. Using new Array // not recommended
   - var students = new Array(“John”, “Mary”, “Ric”);

- Data in an array can be of any types
  - var a = [“Ric”, 100, function() { return “Hello!”; }];
    a[2](); // Hello!

- ”length” property and “push” method in Array
  - var students = [ ]; // initialized as an empty array
    students.push(“John”); // recommended
    students[students.length] = “Mary”; // length is now 2
    students[3] = “Mary”; // OK, but create an “undefined” in [2]

JavaScript ◦ Array or Object?

- Array is a special kind of object
  - var students = [“John”, “Mary”, “Ric”];
    typeof students; // object
    Array.isArray(students); // true

- Array elements MUST BE accessed by numbers.
  雖然一些 object 的行為看起來很像 array, 但嚴格來說，他們是不一樣的
  - var a = { name:”Ric”, score:100 };
    a[“name”]; // “Ric”
    a[0]; // undefined
    Array.isArray(a); // false
  - var students[0] = “John”;
    students[“score”] = 100; // Array.isArray(students) —> true
    students.length; // length is 1, not 2
    students[1]; // undefined
    students[1] = “Mary”; // length is now 2; “score” as a property of students

JavaScript ◦ Control Statements

- Control statements in JavaScript are pretty much the same as in C/C++
- Comparison in JavaScript use “===“ and “!==“
  - "5" === 5; // = false
  - null === undefined; // = false
- “==“ 跟 “!=“ 會雞婆地幫你做型別轉換
  - "5" == 5; // = true
  - null == undefined; // = true
- 更多令人崩潰的型別轉換/比較會在下一章補充

JavaScript。Variable / Function Hoist

- Variable 定義的位置會自動抬升到最前面
  以下兩種寫法是一樣的(雖然前者不太正常)：
  1. a = 1;
     var a = 2;
  2. var a = 1;
     a = 2;
- Function declaration定義的位置會被抬升到最前面
  - sum(3,5); / / This is OK!
    function sum(a, b) { return a + b; };
- 但是用 expression 定義的 function 變數則不會被抬升
  => 如果在定義之前就被使用，就會是 error
  - sum(3,5); / / Error
    var sum = function(a, b) { return a + b; };

JavaScript ◦ Variable Scope

- 事實上，“var” 沒有寫也會過，只是會被視為全域變數 (global variable)
- 此外，”var” 如為區域變數，其範圍並非 block, 而是 function scope
  - function() {
    for (var i = 0; i < 10; i++) console.log(i);
    }
    等同於
    function() {
    var i;
    for (i = 0; i < 0; i++) console.log(i);
    }
- 在同一個範圍內重複用 “var” 宣告同名字的變數，後者會被無視，而只進行 assignment
  - var a = 0;
    { var a = 10; }   / / 仍屬同一範圍之 ‘a’. a = 10 now
    var b = function() { var a = 20; }
    b();   / / local ‘a’ 並非 global ‘a’. 所以 a 仍然為 10

JavaScript ◦ "let" and "const"

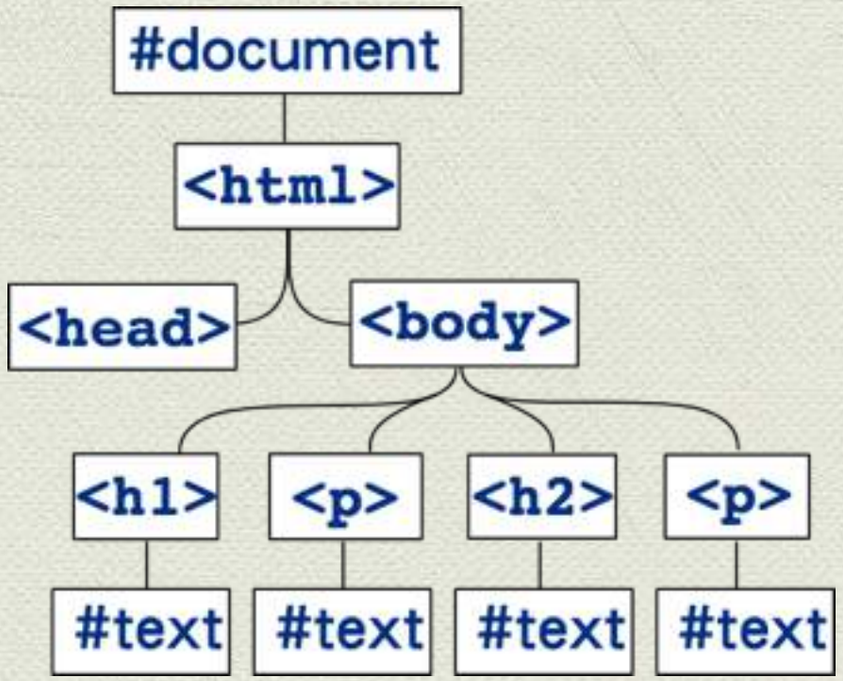
- 2015 年發表的 ES6 版本加入了 "let" 跟 "const"
- "let" 跟 C/C++ 的變數一樣，採取 block scope
  - 同一個 scope 不能宣告相同名稱的變數
  - 不能在宣告變數之前就使用 (i.e. 用 let 宣告的變數不會被抬升 (hoist) 到 scope 的最前面)
  - 在 global scope 用 let 宣告的變數並不是真正的 global variable ==> 並不會變成 "window" 這個全域物件底下的屬性，因此，像是用 module 載入的程式碼並看不到這個 global scope 的 let 變數
- "const" 亦跟 C/C++ 一樣，表示 "read-only" 的常數變數
  - 養成習慣，多多使用 const

DOM Manipulations

- 選擇節點
- 瀏覽節點
- 增減節點
- 修改節點內容
- 事件監聽綁定

DOM Manipulations

- 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



- 如前所述，所謂的動態網頁就是在 event 觸發之後，選擇某個/某些 DOM 節點，然後改寫其內容或是周邊的節點 (node/element)

DOM Manipulations・選擇節點

- 從 top level (i.e. “document”) 尋找 DOM nodes
  // <div id=“target">
  //
  - // 回傳 unique DOM node
    ```
    var target = document.getElementById('target');
    ```
  - // 回傳一個 array of DOM nodes
    ```
    document.getElementsByClassName(”a-class”);
    document.getElementsByTagName(“div”);
    ```
  - // 用 CSS 的一部份 selector 來選取
    ```
    document.querySelector(“#test”);
    document.querySelectorAll(“div.highlighted > p”);
    ```

DOM Manipulations。瀏覽節點

- 從某個 DOM node 走到上 / 下一層的節點

```
// These are properties, not functions
```

- thisNode.parentNode
- thisNode.firstChild
- thisNode.lastChild

DOM Manipulations・增減節點

- 在 document 底下 或是 某個節點前後 增減節點
  - var newElement = document.createElement(“div”);
    var newText = document.createTextNode(“Hello!”);
  - // become the last child
    var appendedNode = parentNode.appendChild(childNode);
    // removed from parent
    var removedNode = parentNode.removeChild(childNode);
    // insert before refNode
    var insertedNode = parentNode.insertBefore(newNode, refNode);
    // replace child node
    var replacedNode = parentNode.replaceChild(newNode, oldNode);

DOM Manipulations。修改節點內容

- 修改節點內容、屬性、樣式... 等
  // These are properties, not functions
  - // get the serialized HTML code for all of its children
    const childrenHTMLCode = thisNode.innerHTML;
    // replace all of its children using the HTML code
    thisNode.innerHTML = childrenHTMLCode;
- const cssStyleObj = thisElement.style;
  thisElement.style.color = “red”;
- const className = thisElement.className;
  thisElement.className = “new-class1 new-class2”;
- const elementArray = thisElement.classList;
  thisElement.classList.{add,remove,…}(className);

DOM Manipulations。事件監聽綁定

- targetElement.addEventListener(eventType, listenerFunction);

eventType	說明
click	點擊
focus	開始在輸入框輸入的時候
blur	離開輸入框的時候
change	輸入值改變的時候
keydown	鍵盤按下去的時候
keyup	鍵盤按下去的鍵上來的時候
mouseenter	滑鼠進到元素裡面的時候
mouseleave	滑鼠移出元素的時候

DOM Manipulations。事件監聽綁定

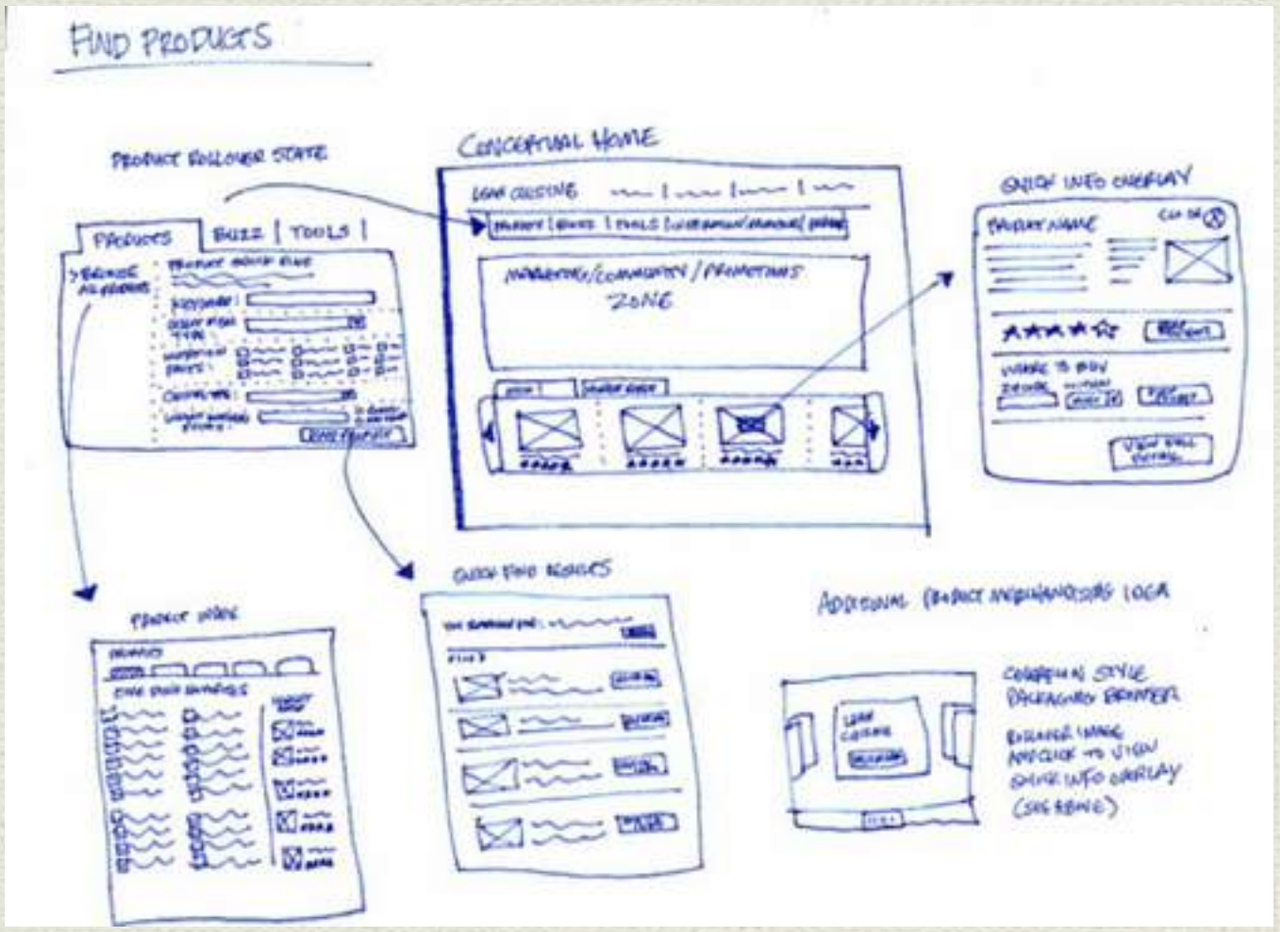
- / / <button id="target">是個按鈕</button>

```
var targetElement = document.getElementById(“target”);
targetElement.addEventListener(“click”, function() {
    / / ..做一些事，任何事，你希望按鈕按了要幹麻？
    alert('你希望按鈕按了要幹麻？');
});
```

這週介紹了 Web Programming 的基礎知識，以及 HTML、CSS、JavaScript 的基本語法，大家應該要有能力刻出一個 client 端的 web service prototype

WebPrototype設計・Wireframe

- **Information** — what info to be displayed efficiently and clearly?
- **Navigation** — how to move from page to page?
- **Interface** — How the layout of the page be presented?



Web Prototype 設計・Wireframe 工具

- 最簡單實用 — Pen and paper
- 單機版 — Omnigraffle
- 協作版 — Lucidchart
- 便宜版 — Sketch
- 討論文章 on Inside: https://goo.gl/5BMta1

After Class ◦ Week #1

- **Homework assignment**
  - **目標**：發想一個簡單、有一點用處的網路服務(e.g. 簡易網頁相簿、GRE 單字自我測驗、簡單的雙人網路遊戲... etc)，這星期的目標是要刻一個簡單的 prototype website (只有前端畫面，後端/資料暫時寫死在 code 裡面)，所以不要想得太複雜，害死自己 :)
  - **Wireframe**：先畫 wireframe, 力求清晰完整，除了要當作是自己的規格設計之外，下星期上課我們將會隨機交換 wireframe designs, 當作是下星期的回家作業。所以，下星期上課時記得把你的 wireframe design 印出來，帶到課堂上。Wireframe 紙本記得留下姓名，以及聯絡方式，免得對方(苦主)看不懂你在畫什麼。
- **Coding 練習**：利用這個星期教的 HTML/CSS/JavaScript/DOM basics，刻一個符合 wireframe 設計的網頁。如同前面所述，我們只要做前端的畫面，而且由於時間有限，你們可以不用刻完，minimum requirement 是一個完整頁面，並且至少有一項動態網頁的功能。至於後端服務以及資料庫請先寫死在 code 裡面就好。
- **繳交**：請將 wireframe (如果是用紙筆，照相即可) 以及上面的 codes 放在同一個母目錄 (yourStudentID_hw1), 壓縮 (tar zcvf yourStudentID_hw1.tgz yourStudentID_hw1), 再將壓縮檔上傳到 Ceiba (只開放有修課同學).
- **Deadline**: 5pm, 03/14(Wed). 遲交將會有嚴重的 penalty.

After Class ◦ Week #1

- **Readings /Other actions**
  - **JavaScript tutorials/references**
    - Learning X in Y minutes: https://learnxinyminutes.com/docs/javascript/
    - MDN Web Docs: https://developer.mozilla.org/en-US/docs/Learn/JavaScript
    - Codecademy: https://www.codecademy.com/
  - **安裝 Git, 申請 Github 帳號**
    - 請安裝 command line, 避免使用 GUI 工具：
      https://git-scm.com/book/zh-tw/v2 (chap 1.4 開始)
  - **Git tutorial**
    - (30 mins video) https://www.youtube.com/watch?v=HVsySz-h9r4
    - Codeschool Try Git: https://www.codeschool.com/courses/try-git
  - **安裝 ESLint (for 你習慣的編輯器)**
  - **加入 FB 社團**
    - For 修課同學 only, https://www.facebook.com/groups/NTURicWebProg/