CSS 入門

1. 什麼是 CSS
2. 主要功能
3. 基本語法
4. 套用方式
5. 套用 Bootstrap

什麼是 CSS ？

1. CSS (Cascading Style Sheets) 階層式樣式表
2. 一種用來為結構化文件（如HTML文件或XML應用）添加樣式（字型、間距和顏色等）的電腦語言

基本語法

```
h1 {
    color:red; font-size:14px;
}
```

1. 選擇器（Selector）：h1

套用方式

方式	優點	缺點	重複
style	可以快速套用	需在每個標籤中套用	否
<style>	在 HTML 文件中套用載入	需在 `<style>` 中套用	否
<link>	無需在每個標籤中套用	額外載入（請求）	可
@import	無需在每個標籤中套用	額外載入（請求）	可

實作：使用 `style` 屬性套用 CSS

1. 新增 inlinecss.html
2. 快速鍵 `!` 產生 html
3. 新增 `<h1>` 、 `<p>` 、 `<span>`
4. 設定屬性

實作：使用 `<style>` 套用 CSS

1. 新增 stylecss.html
2. 快速鍵 `!` 產生 html
3. 新增 `<h1>`、`<p>`、`<span>`
4. 新增 `<style>` 並設定屬性

實作：使用 `<link>` 套用 CSS

1. 新增 linkcss.html
2. 快速鍵 `!` 產生 html
3. 新增 `<h1>`、`<p>`、`<span>`
4. 新增 `link.css` 並設定屬性
5. 加入 `<link>`

實作：使用 `@import` 套用 CSS

1. 新增 linkcss.html
2. 快速鍵 `!` 產生 html
3. 新增 `<h1>` 、 `<p>` 、 `<span>`
4. 新增 `p.css` 、 `h.css` 、 `span.css` 並設定屬性
5. 新增 `im.css` 引用 `@import h.css、p.css、span.css` 加入 `<link>`

CSS 選擇器

1. 通用：`* { color: black; }`
2. 元素：`h1 { font-size: 20px; }`
3. ID：`#header { background: blue; }`
4. class：`.button { padding: 10px; }`
5. 屬性：`[type="text"] { border: 1px solid; }`
6. 文件結構：`body > p { margin: 20px; }`
7. 群組或組合：`h1, h2 { color: red; }`

實作：通用選擇器與元素選擇器

1. 新增 starh.html
2. 快速鍵 `!` 、 `h$*6>lorem`
3. 設定 `* { color: black; }`
4. 設定 `h1` 到 `h6` 不同的字型大小，EX：

```
h1 { font-size: 20px; }
```

實作：ID 選擇器與 Class 選擇器

1. 新增 idclass.html
2. 快速鍵 `!` 、 `h$*6>lorem.hc$#hi$`
3. 設定 `.h1` - `.h6` css
4. 設定 `#h1` - `#h6` css

實作：屬性,文件結構,群組或組合的

1. 新增 attrgroup.html
2. 快速鍵 `!` 、 `h$*6>lorem.hc$#hi$`
3. 快速鍵 `form>input[type="text"]#a+input[type="text"]#b`
4. 設定 `[type="text"] { border: 1px solid; }`
5. 設定 `form > input { margin: 20px; }`
6. 設定 `h1, h2 { color: red; }`

CSS 顏色（color）

1. 顏色名稱：`red`、`green`、`blue`
2. HEX：`#F00`、`#0F0`、`#00F` 與 `#FF0000`、`#00FF00`、`#0000FF`
3. RGB 與 ARGB：`rgb(255, 0, 0)`、`rgba(255, 0, 0, 0.5)`，最後一個值 A 代表透明度

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

CSS 字型（font-family）

1. 常見的字型類型有：serif（襯線體，如 Times New Roman）、sans-serif（無襯線體，如 Arial）、monospace（等寬字體，如 Courier New）。
2. 使用 font-family 屬性來指定文字的字型。不止於單一字型，你可以提供一個字型系列，當第一個字型不可用時，瀏覽器會依序選用後面的字型。

```
p {
  font-family: "Helvetica", "Arial", sans-serif;
}
```

CSS 字型（font-size）

1. `px` 像素
2. `em` 相對單位（所在區塊）
3. `rem` 根相對單位（通常是HTML）
4. `%` 百分比
5. 其他

```
p {
  font-size: 13px;
}
```

CSS 字型（font-weight）

1. 數字 100 - 900
2. 關鍵字 normal、bold 等

```
p {
  font-weight: bold;
}
```

CSS 字型（font）

1. Shorthand Property：可在單一屬性內設定多個屬性
2. 必要：`font-style`、`font-family`
3. Options：`font-variant`、`font-weight`、`font-size/line-height`

```
/* font-style font-size/line height font-family */
font: italic bold 30px/2 "Helvetica", "Arial", sans-serif;
```

CSS 對齊（text-align）

1. `left` 靠左
2. `right` 靠右
3. `center` 置中
4. `justify` 左右對齊

```
p {
    text-align:center;
}
```

CSS 首行縮排（text-indent）

1. 用來設定第一行的縮排空間
2. 單位：`px`、`em`、`rem`、`%`
3. 正負值：正值第一行向右縮排。負值縮排向左。

```
p {
    text-indent: 20px;
}
```

CSS 文字間隔（letter-spacing）

1. 文字之間的距離
2. 單位：`px`、`em`、`rem`、`%`、`pt`
   （點，1 點等於 1/72 英寸，印刷用）
3. 正負值：正值第間距變大。負值間距變小。

```
p {
    letter-spacing: 2px;
}
```

CSS 行高（line-height）

1. 控制行與行之間的距離
2. 單位：
   - `數值`：如 `line-height: 1.5` 為 1.5 倍的字型大小
   - `%`：如 `line-height: 150%` 為 150% 的字型大小
   - 固定單位：`px`、`em`、`rem`、`%`
3. 正負值：正值第間距變大。負值間距變小。

```
p {
    line-height: 1.5em;
}
```

CSS 垂直對齊（vertical-align）

1. `baseline`：一般
2. `top`：置頂
3. `middle`：置中
4. `bottom`：置底
5. `super`、`sub`：上標、下標

```
p {
    vertical-align: middle;
}
```

CSS 文字陰影（text-shadow）

為文字添加陰影效果可依序設定四個值：

1. 水平位移 (horizontal offset)：正向右，負向左。
2. 垂直位移 (vertical offset)：正向下，負向上。
3. 模糊半徑 (blur radius)：值越大，陰影越模糊。
4. 陰影顏色 (shadow color)。

```
p {
    text-shadow: 2px 2px 5px rgba(0, 0, 0, 0.5);
}
```

CSS 背景（background-color）

```
div {
    background-color: #ff0000;
}
```

CSS 背景（background-image）

```
div {
    background-image: url('image.jpg');
}
```

CSS 背景（background-repeat）

1. repeat：重複
2. repeat-x：X軸重複
3. repeat-y：Y軸重複
4. no-repeat：不重複

```
div {
    background-image: url('image.jpg'); /* 通常會搭配背景圖使用 */
    background-repeat: repeat;
}
```

CSS 背景（background-position）

1. 可以使用關鍵字（如：top, right, center）或像素值/百分比。
2. DEMO

```
div {
    background-position: right 30% bottom 40%;
}
```

CSS 背景（background-size）

1. 依序為長寬的值：像素值/百分比。
2. `cover`：圖片會被縮放到完全覆蓋元素的背景區域，可能會裁剪掉一部分圖像以確保背景區域沒有空白。
3. `contain`：圖片會被縮放到在保持縱橫比的情況下，完全置入元素的背景區域。這意味著圖片可能會留白以適應元素的寬高。

```
div {
  background-image: url('image.jpg');
  background-size: cover;
}
```

CSS 背景（background-attachment）

1. `scroll`：背景圖片隨頁面滾動
2. `fixed`：背景圖片固定。
3. `local`：背景圖片與元素的滾動一致。

```
div {
  background-image: url('image.jpg');
  background-attachmentsize: fixed;
}
```

CSS 背景漸層（linear-gradient）

說明

```
div {
  background: linear-gradient(to right, red, orange, yellow, green, blue, indigo, violet);
}
```

CSS 背景（background）

```
/* background: color image position/size repeat attachment; */
p {
    background: #ff0000 url('image.jpg') no-repeat fixed center;
}
```

CSS Width 與 Height

1. `width` and `height`
2. `min-width` and `min-height`
3. `max-width` and `max-height`

CSS Margin

1. margin 是元素外部的空白區域
2. 屬性：
    - margin-top: 元素上方的空間。
    - margin-right: 元素右方的空間。
    - margin-bottom: 元素下方的空間。
    - margin-left: 元素左方的空間。

CSS Margin

```css
.element {
  margin: 10px; /* 所有方向使用相同的邊距 */
}
.element {
  margin: 10px 5px; /* 垂直方向使用 10px，水平方向使用 5px */
}
.element {
  margin: 10px 5px 15px; /* 上 10px，水平 5px，下 15px */
}
.element {
  margin: 10px 5px 15px 20px; /* 上、右、下、左 */
}
```

CSS Padding

1. padding 是元素內部的空白區域
2. 屬性
    - padding-top: 內容與上邊框的空間。
    - padding-right: 內容與右邊框的空間。
    - padding-bottom: 內容與下邊框的空間。
    - padding-left: 內容與左邊框的空間。

CSS Padding

```css
.element {
  padding: 10px; /* 所有方向使用相同的內距 */
}
.element {
  padding: 10px 5px; /* 垂直方向使用 10px，水平方向使用 5px */
}
.element {
  padding: 10px 5px 15px; /* 上 10px，水平 5px，下 15px */
}
.element {
  padding: 10px 5px 15px 20px; /* 上、右、下、左 */
}
```

CSS box-sizing

1. 屬性：
   - **content-box**: 預設值。元素的 `width` 和 `height` 僅包括內容區，不包括 `padding` 和 `border`。
   - **border-box**: 元素的 `width` 和 `height` 包含內容、內邊距 `padding`、和邊框 `border` 的尺寸。

CSS box-sizing

```
<style>
.content-box { box-sizing: content-box; width: 200px;
padding: 20px; border: 5px solid #000; background-color: lightgrey; }
/* 200px (內容) + 40px (內邊距) + 10px (邊框) = 250px */

.border-box { box-sizing: border-box; width: 200px;
padding: 20px; border: 5px solid #000; background-color: lightblue; }
/* 寬度依然保持 200px */
</style>
...
<div class="content-box">content-box範例</div>
<div class="border-box">border-box範例</div>
```

CSS float

1. 屬性：
    - `none`：預設
    - `left`：向左浮動
    - `right`：向右浮動
2. `clear` 清除：
    - `both`
    - `left`
    - `right`

CSS float

```
<div class="container">
  <div class="float-left">浮動在左側</div>
  <div class="float-right">浮動在右側</div>
  <p>這是一段文字</p>
</div>
```

CSS float

```css
.float-left {
  float: left;
  width: 200px;
  background-color: #f0f0f0;
}
.float-right {
  float: right;
  width: 200px;
  background-color: #f0f0f0;
}
```

CSS @media

`@media` 是用來實現響應式設計的關鍵技術。它允許我們根據不同的設備特性（如螢幕大小、解析度等）來指定不同的樣式。

```
body { font-size: 16px; }
@media (max-width: 600px) {
  body { font-size: 14px; }
}
@media (min-width: 601px) and (max-width: 1200px) {
  body { font-size: 18px; }
}
@media (min-width: 1201px) {
  body { font-size: 20px; }
}
```

和 AI 聊聊 CSS

1. 選擇器的優先權
2. 同一個屬性重複設定
3. 什麼是 `display` go

Bootstrap 簡介與設定

1. **Bootstrap 簡介**
   - 什麼是 Bootstrap？
   - 為什麼要使用 Bootstrap？
2. **Bootstrap 安裝與設定**
   - 透過 CDN 引入 Bootstrap（須留意引用版本）
   - 下載與本地安裝
   - **AI：常用 HTML 範本結構**

裝置尺寸

1. **Extra small** `xs`: 寬度通常少於 576px。
2. **Small** `sm`: 寬度通常在 576px 到 768px 之間。
3. **Medium** `md`: 寬度通常在 768px 到 992px 之間。
4. **Large** `lg`: 寬度通常在 992px 到 1200px 之間。
5. **Extra large** `xl`: 寬度超過 1200px。
6. **Extra extra large** `xxl`: 寬度超過 1400px。

布局系統

1. 固定寬度容器 `.container`
2. 流體容器 `.container-fluid`

網格系統 (Grid System)

1. 行 `.row`
2. 列 `.col`：一個 12 列的系統，這意味著一行可以被分成最多 12 個等份。
3. 響應式佈局 (Responsive Layout) 的概念：使得網頁能適應不同的裝置尺寸。

網格系統 (Grid System)

```
<div class="container"><div class="row">
    <div class="col-xs-12 col-sm-6 col-md-4 col-lg-3">
      Column 1
    </div>
    <div class="col-xs-12 col-sm-6 col-md-4 col-lg-3">
      Column 2
    </div>
    <div class="col-xs-12 col-sm-6 col-md-4 col-lg-3">
      Column 3
    </div>
    <div class="col-xs-12 col-sm-6 col-md-4 col-lg-3">
      Column 4
    </div>
</div></div>
```

顏色（預先定義的顏色樣式）

1. 文字顏色： `.text-muted1`, `.text-primary`, `.text-success`, `.text-info`, `.text-warning`, `.text-danger`, `.text-secondary`, `.text-white`, `.text-dark` ...
2. 背景顏色： `.bg-primary`, `.bg-success`, `.bg-info`, `.bg-warning`, `.bg-danger`, `.bg-secondary`, `.bg-dark` and `.bg-light`.

排版

1. `.h1` - `.h6` 對應 `<h1>` - `<h6>`
2. `.small` 對應 `<small>`
3. `.mark` 對應 `<mark>`
4. `.display-1` - `.display-6`: more `weight` and `bigger` heading

文字樣式

1. `font-size` ： `.fs-1` - `.fs-6`
2. `font-weight` ： `fw-bold` 、 `fw-bolder` 、 `fw-normal` 、 `fw-light` 、 `fw-lighter`
3. `font-style` ： `fst-italic` 、 `fst-normal`
4. `lint-height` ： `lh-1` 、 `lh-sm` 、 `lh-base` 、 `lh-lg`

…

文字對齊

1. `text-start`：LTR 靠左
2. `text-end`：LTR 靠右
3. `text-center`

間距

```
.mt-0 { margin-top: 0 !important; }
.ms-1 { margin-left: ($spacer * .25) !important; }
.px-2 {
   padding-left: ($spacer * .5) !important;
   padding-right: ($spacer * .5) !important;
}
.p-3 { padding: $spacer !important; }
```

1. Property：
   - `m-` ： `margin`
   - `p-` ： `padding`

間距

2. Sides：
   - `t`：`top`
   - `b`：`bottom`
   - `s`：LTR `left`
   - `e`：LTR `right`
   - `x`：同時設定 `*-left` 與 `*-right`
   - `y`：同時設定 `*-top` 與 `*-bottom`

間距

3. Size：
   - `0` - `5` 與 `auto`

水平置中

```
<div class="mx-auto" style="width: 200px;">
  Centered element
</div>
```

表格

1. 基本表格樣式 `.table`
2. Striped（條紋） `.table-striped`
3. Bordered（編框） `.table-bordered`
4. Hoverable（滑鼠游標停留）： `.table-hover`

   ...

表單 (Forms)

1. Inputs `.form-control`
2. Labels `.form-label`
3. Input Group `.input-group`
4. Checkboxes、Radios：`.form-check` 、`.form-check-input` 與 `.form-check-label`
5. Select `.form-select`

   ...

按鈕（Buttons）

1. `<button>` 與 `<a>` 加上 `class='btn btn-*'` 或 `btn btn-outline-*`
2. 尺寸：`.btn-lg`、`.btn-sm`
3. 狀態：`.disable` 與 `.active`

   ...

圖片（Images）

1. 響應式圖片：`.img-fluid`
2. 縮圖：`.img-thumbnail`

導覽列（Navbar）

```
<nav class="navbar navbar-expand-lg navbar-light bg-light">
  <a class="navbar-brand" href="#">Logo</a>
  <button class="navbar-toggler"
  type="button"
  data-toggle="collapse"
  data-target="#navbarSupportedContent" aria-controls="navbarSupportedContent"
  aria-expanded="false"
  aria-label="Toggle navigation">
    <span class="navbar-toggler-icon"></span>
  </button>
  <!-- collapse navbar-collapse Next Page -->

</nav>
```

導覽列（Navbar）

```
<div class="collapse navbar-collapse" id="navbarSupportedContent">
    <ul class="navbar-nav mr-auto">
      <li class="nav-item active">
        <a class="nav-link" href="#">首頁 <span class="sr-only">(current)</span></a>
      </li>
      <li class="nav-item">
        <a class="nav-link" href="#">關於我們</a>
      </li>
      <li class="nav-item">
        <a class="nav-link" href="#">聯絡我們</a>
      </li>
    </ul>
  </div>
```

其他

1. 導覽列 (Navbar)
2. 導航條 (Navs)
3. Breadcrumbs
4. 卡片 (Cards)
5. 模態框 (Modals)
6. 警示框 (Alerts)

...

實作：使用 AI 產生 Html5 的文件 並套禦 Bootstrap

1. 新增 bootstrap.html
2. 快速鍵 `!`
3. 請AI生成將以下標記 `<header>` 、 `<nav>` 、 `<main>` 、 `<section>` 、 `<article>` 、 `<aside>` 、 `<figure>` 、 `<footer>` 加入 `<body>` 中 ，並產生內容。
4. 套用 Boostrap5 使用 npm 的 CDN

Java Script 入門

1. 什麼是 Java Script？
2. 資料
3. 變數
4. 運算符號
5. 流程控制
6. DOM 操作

什麼是 Java Script ？

1. 一種腳本語言
2. 可以實現複雜的前端網頁網頁功能
3. Java 與 Javascript 是不同的
4. 簡稱 JS

實作：Hello Javascript

1. 新增 js.html
2. 快速鍵 `!` 、
3. 位置： `</body></html>` 前
4. 輸入快速鍵 `script`
5. 在底部放置 script 的優點是可以確保 HTML 結構先行載入，提升頁面載入速度。缺點是可能在 DOM 未完全載入時無法操作。

```
<body>...
<script>console.log("hello javascript!"); // console.log 改成 alert</script>
</body></html>
```

Java Script 資料

1. 數字：`42`, `3.14`
2. 布林：`true`, `false`
3. 字串：`"hello"`, `'world'`
4. 空值：`null`, `undefined`
5. 物件：`{}`
6. 陣列：`[]`
7. 實作

Java Script 變數和常數

1. 保留字：`var`、`let`、`const`
2. 無宣告：直接使用變數，JS 會自動宣告
3. 建議使用 let 與 const
4. 實作：設定並輸出

```
let x = 1; // var x = 1;
const y = "y";
z = "z";
```

Scope（作用域）

var

- `var` （Function Scope），即在函數內是屬於同一個變數的。
- `let` 和 `const` （Block Scope），即在括號 `{}` 內是同一個變數的。
- 無宣告的變數會自動成為全域變數（Global）。

Reassignment（重新指派）：

- `var` 和 `let` 變數都允許重新指派值。
- `const` 變數一旦宣告並賦值後就不能再重新指派新的值。
- 無宣告的變數可以重新指派值。

Redeclaration（重新宣告）

1. `var`
   - 重新宣告的變數值會被覆蓋掉，新的值會被賦予。
   - 允許在同一作用域內重新宣告。
2. `let` 和 `const`
   - 重新宣告的變數值不會被覆蓋掉，仍然保持原來的值。
   - 不允許在相同的作用域內重新宣告。
3. 無宣告的變數沒有宣告的過程，因此不適用。

Hoisting（提升）

```
console.log(a); // 輸出：undefined
var a = 5;
console.log(b); // ReferenceError: b is not defined
let b = 10;
```

- 所有變數（`var`, `let`, `const`）都會被 Hoisting，但行為有所不同。
- `var` 會被提升且初始化為 `undefined`。
- `let` 和 `const` 會被提升，但在提升之前使用會導致參照錯誤（Reference Error）。
- 無宣告的變數，由於不涉及宣告，因此 Hoisting 不適用。

變數的 scope 與特性

	var	let	const	無宣告
Scope	Function	Block	Block	Global
重新指派	可以	可以	不可	可以
重新宣告	可以	不可	不可	不適用
Hoisting	*有	有	有	不適用

Javascript 物件（Object）

在 JavaScript 中，物件是一種用來存儲相關資料與功能的集合。物件由 key-value（鍵值對）組成，而鍵是字串，值可以是任何資料類型。

```
let student = {
    name: "John", age: 21, grade: "A",
    study: function() {  // 方法（函數）作為值
        console.log(this.name + " is studying.");
    }
};
// 輸出物件的屬性
console.log(student.name);   // 輸出：John
console.log(student.age);    // 輸出：21
// 呼叫物件的方法
student.study();  // 輸出：John is studying.
```

Javascript 陣列（Array）

```
// 創建一個包含數字的陣列
let numbers = [10, 20, 30, 40, 50];
// 訪問陣列的元素
console.log(numbers[0]);  // 輸出：10
console.log(numbers[3]);  // 輸出：40
// 修改陣列的元素
numbers[1] = 25;
console.log(numbers[1]);  // 輸出：25
// 陣列的方法示範
numbers.push(60);         // 在陣列末尾添加元素
console.log(numbers);     // 輸出：[10, 25, 30, 40, 50, 60]
let lastNumber = numbers.pop();  // 從陣列末尾移除元素
console.log(lastNumber);         // 輸出：60
console.log(numbers);            // 輸出：[10, 25, 30, 40, 50]
```

Javascript 物件與陣列混用

```
let students = [
{name: "John", age: 21, grade: "A"},
{name: "May", age: 22, grade: "C"},
{name: "An", age: 23, grade: "D"}
];
console.log(students);
```

Javascript 算數運算：

1. `+`：加法 `a + b`
2. `-`：減法 `a - b`
3. `*`：乘法 `a * b`
4. `/`：除法 `a / b`
5. 實作

Javascript 指定運算：

1. `=` ：賦值 `x = 5`
2. `+=` ：加法賦值 `x += 5`
3. `-=` ：減法賦值 `x -= 5`
4. `*=` ：乘法賦值 `x *= 5`
5. `/=` ：除法賦值 `x /= 5`
6. `%=` ：取餘數賦值 `x %= 5`
7. `**=` ：指數賦值 `x **= 5`
8. 實作

Javascript 比較運算：

1. `==` ：相等 `a == b`
2. `!=` ：不相等 `a != b`
3. `===` ：嚴格相等 `a === b`
4. `!==` ：嚴格不相等 `a !== b`
5. `<` ：小於 `a < b`
6. `>` ：大於 `a > b`
7. `<=` ：小於或等於 `a <= b`
8. `>=` ：大於或等於 `a >= b`

Javascript 單元運算：

1. `+`：正數 `+a`
2. `-`：負數 `-a`
3. `!`：邏輯否定 `!a`
4. 實作

Javascript 邏輯運算：

1. `&&`：邏輯與 `a && b`
2. `||`：邏輯或 `a || b`
3. `!`：邏輯否定 `!a`
4. 實作

JavaScript 自動轉型

1. JavaScript 是一種弱型別語言，這意味著變數可以儲存不同類型的值，而不需要事先聲明其類型。這同時也意味著 JavaScript 會嘗試在某些情況下自動轉換類型，以滿足運算符的需求。
2. 實際操作時易誤踩陷阱，盡量避免自動轉型。

字串與數字的相加

```
let num = 5;
let str = "10";
let result = num + str;
console.log(result); // 輸出 "510"
```

`+` 加法中，如果其中一個 operand 為字串則會將另外一個轉成字串。所以最後結果是 "510"。

字串與數字的減法（非加法）

```
let num = 5;
let str = "10";
let result = num - str;
console.log(result); // 輸出 -5
```

`-` 減法、`*` 乘法、`/` 除法、`%` 餘數、`**` 指數：如果其中一個 operand 是字串，則會將字串轉型為數字。所以最後結果是 -5。

邏輯運算中的自動轉型

```
let val1 = 0;
let val2 = "0";

if (val1 == val2) {
    console.log("相等");
} else {
    console.log("不相等");
}
// 輸出 "相等"
```

在這個範例中，==（寬鬆等於）運算符會進行自動轉型來比較兩個值。這裡，val1 是數字 0，而 val2 是字串 "0"。JavaScript 將字串 "0" 轉為數字 0，然後進行比較，結果兩者相等。

使用 `===` 進行嚴格比較

```
let val1 = 0;
let val2 = "0";

if (val1 === val2) {
    console.log("相等");
} else {
    console.log("不相等");
}
// 輸出 "不相等"
```

Javascript 流程控制 if ... else if ... else

用於根據不同的條件執行不同的代碼塊。

```
let score = 85;
if (score >= 90) {
    console.log("Grade: A");
} else if (score >= 80) {
    console.log("Grade: B");
} else if (score >= 70) {
    console.log("Grade: C");
} else {
    console.log("Grade: D");
}
```

練習：請使用 if 偵測 b ， 大於 0 印出 True 否則 False

Javascript 流程控制 switch

用於根據不同的值執行不同的代碼塊。

```
let value = "A";
switch (value) {
    case "A": { console.log("大於等於 90"); break; }
    case "B": { console.log("大於等於 80 小於 90"); break; }
    case "C": { console.log("大於等於 70 小於 80"); break; }
    case "D": { console.log("小於 70"); break; }
    default: { console.log("Invalid grade"); }
}
```

練習：請使用 switch `1` - `7` 輸出成 `星期一` - `星期日`

Javascript 迴圈 while

用於重複執行代碼直到條件為 false。

```
let count = 0;
while (count < 5) {
    console.log("Count is: " + count);
    count++;
}
```

練習：使用 while 迴圈印出 1-10

Javascript 迴圈 for

用於執行特定次數的迴圈。

```
for (let count = 0; count < 5; count++) {
    console.log("Count is: " + count);
}
```

練習：使用 while 迴圈印出 1-10

`break` 與 `continue`

1. `break` 直接跳離迴圈
2. `continue` 會跳過後到執行下一次迴圈
3. 實作

Javascript 建立函數

用於將多個執行步驟合併為一個單一的可重複使用的程式碼區塊。

```
function greet(name) {
    return "Hello, " + name + "!";
}
```

Javascript 呼叫函數

```
console.log(greet("Alice"));
// 輸出: Hello, Alice!
```

Javascript 使用 let 建立匿名函數

可以使用 let 關鍵字將函數賦值給變數，這樣就可以在之後呼叫函數

```
let add = function(a, b) {
    return a + b;
};
```

Javascript 呼叫匿名函數

```
console.log(add(3, 4)); // 輸出: 7
```

Javascript 使用箭頭函數：

```
// 箭頭函數，簡化函數的定義，讓代碼更加簡潔
const multiply = (x, y) => x * y;
/*
const multiply = function(x, y) {
    return x * y;
};
*/
console.log(multiply(2, 5)); // 輸出: 10
```

JS Dom (Document Object Model)

1. 取得 HTML 的 element
   - `document.getElementById`：透過 id 取得 element ID
   - `document.querySelector`：取得 element CSS 選擇器
   - `document.querySelectorAll`：取得 element CSS 選擇器 (多個)
   - `element.parentNode`：取得 element parent element
   - `element.childNodes`：取得 element child elements

JS Dom (Document Object Model)

2. 使用 `element.innerHTML` 來設置 HTML 內容 (innerText 純文字)
3. 使用 `element.style` 來設置 CSS 樣式
4. 使用 `element.addEventListener` 來監聽事件

document.getElementById

```
...
<body>
  <div id="example">Hello, World!</div>
  <script>
    var element = document.getElementById("example");
    console.log(element); // 這會印出 <div id="example">Hello, World!</div>
  </script>
</body>
...
```

document.querySelector

```
...
<body>
  <div class="example">Hello, World!</div>
  <script>
    var element = document.querySelector(".example");
    console.log(element); // 這會印出 <div class="example">Hello, World!</div>
  </script>
</body>
...
```

document.querySelectorAll

```
...
<body>
  <div class="example">Hello!</div>
  <div class="example">World!</div>
  <script>
    var elements = document.querySelectorAll(".example");
    console.log(elements); // 這會印出 NodeList(2) [div.example, div.example]
  </script>
</body>
...
```

element.parentNode

```
...
<body>
  <div id="parent">
    <div id="child">Hello!</div>
  </div>
  <script>
    var childElement = document.getElementById("child");
    var parentElement = childElement.parentNode;
    console.log(parentElement); // 這會印出 <div id="parent">...</div>
  </script>
</body>
...
```

element.childNodes

```
...
<body>
  <div id="parent">
    <div>Child 1</div>
    <div>Child 2</div>
  </div>
  <script>
    var parentElement = document.getElementById("parent");
    var childElements = parentElement.childNodes;
    console.log(childElements); // 這會印出 NodeList(3) [#text, div, div]
  </script>
</body>
...
```

element.innerHTML

```
...
<body>
  <div id="example">原始內容</div>
  <script>
    var element = document.getElementById("example");
    element.innerHTML = "新的內容";
    console.log(element.innerHTML); // 這會印出 "新的內容"
  </script>
</body>
...
```

element.style

```
...
<body>
  <div id="example">設定樣式</div>
  <script>
    var element = document.getElementById("example");
    element.style.color = "red";
    element.style.fontSize = "20px";
    console.log(element.style.color); // 這會印出 "red"
  </script>
</body>
...
```

element.addEventListener

```
...
<body>
  <button id="exampleButton">點我</button>
  <script>
    var button = document.getElementById("exampleButton");
    button.addEventListener("click", function() {
      alert("按鈕被點擊了!");
    });
  </script>
</body>
...
```

JS BOM (Browser Object Model)

1. 核心是 `window` 物件，全域變數
2. 主要屬性
   - `window.document`：Document 物件
   - `window.location`：Location 物件
   - `window.history`：History 物件
   - `window.navigator`：Navigator 物件
   - `window.screen`：Screen 物件

window.document

這是 Document 物件，它代表了整個 HTML 文件，可以用來對 HTML 內容進行操作。

```
console.log(window.document.title); // 輸出當前文件的標題
window.document.title = "新標題"; // 設定文件的新標題
```

window.location

這是 Location 物件，提供了與目前 URL 相關的資訊和方法。

```
console.log(window.location.href); // 輸出當前 URL
// 重定向到新的 URL
window.location.href = "https://www.example.com";
```

window.history

這是 History 物件，允許操作瀏覽器的瀏覽歷史記錄。

```
// 回到上一頁
window.history.back();
// 前進一頁
window.history.forward();
```

window.navigator

這是 Navigator 物件，提供了有關瀏覽器的資訊。

```
console.log(window.navigator.userAgent); // 輸出瀏覽器的 userAgent 字串
console.log(window.navigator.language); // 輸出瀏覽器的語言設置
```

window.screen

這是 Screen 物件，提供了有關用戶屏幕的資訊。

```
console.log(window.screen.width); // 輸出螢幕的寬度
console.log(window.screen.height); // 輸出螢幕的高度
```

BOM 與 DOM 的差別

1. BOM 是**瀏覽器**的物件模型、資訊、屬性和方法
2. DOM 是 **HTML** 文件的物件模型、資訊、屬性和方法。
3. BOM 可以在 HTML 文件外部使用，而 DOM 只能在 HTML 文件內部使用。

jQuery 入門

1. 什麼是 jQuery ？
2. 安裝與套用 jQuery
3. 主要功能

什麼是 jQuery ？

1. jQuery 是一個快速、小巧且功能豐富的 JavaScript 函式庫
2. 使得 HTML 文件遍歷及操作、事件處理、動畫和 AJAX 更簡單，並且提供了許多實用的方法，讓開發者能夠更快速地開發網頁應用程式。
3. 簡化了與 DOM（Document Object Model）的互動。

安裝與套用 jQuery

1. 下載 jQuery 檔案到 `path/to/jquery.min.js`

```
<script src="path/to/jquery.min.js"></script>
```

2. 使用 CDN

```
<script src="https://code.jquery.com/jquery-3.6.0.min.js">
</script>
```

jQuery 選擇器 (Selectors)

jQuery 的選擇器有助於選取和操作 HTML 元素。

```
<!DOCTYPE html><html><head>
    <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
    <script>
        $(document).ready(function(){
            $("p").css("color", "red"); // 選擇所有 <p> 元素並設置它們的文字顏色為紅色
        });
    </script>
</head>
<body><p>這是一段文字。</p><p>這是另一段文字。</p></body>
</html>
```

jQuery Event Handling（事件處理）

jQuery 提供簡單的方法來處理各種事件，如 `click` 、 `dbclick` 、 `hover` 、 `keydown` 、 `keyup` 、 `submit` 等。

```
<script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
<script>
        $(document).ready(function(){
            $("button").on('click', function() {
                alert("按鈕被點擊！");
            });
        });</script>
</head>
<body><button>點我</button> </body>
</html>
```

jQuery DOM Manipulation （操作）

jQuery 提供了一些簡單的方法來添加、刪除或修改 DOM 元素。

```
...
    <script>
        $(document).ready(function(){
            $("#add").on('click', function() {
                $("ul").append("<li>新增項目</li>");
            });
        });
    </script>
...
    <button id="add">新增</button>
    <ul><li>項目 1</li><li>項目 2</li></ul>
...
```

jQuery AJAX

jQuery 的 AJAX 函數是用來處理非同步請求的。

```
$(document).ready(function(){
    $("#loadData").on('click', function() {
        $.ajax({
            type: "GET", url: "https://jsonplaceholder.typicode.com/posts/1",
            success: function(data){$("#result").text(data.title);}
        });
    });
});
```

```
<button id="loadData">載入資料</button><div id="result"></div>
```

jQuery Animations（動態效果）

jQuery 提供簡單的工具來創建動畫效果，比如隱藏和顯示元素。

```
$(document).ready(function(){
    $("#fadeToggle").on('click', function() {
        $(".box").fadeToggle();
    });
});
```

```
<button id="fadeToggle">切換淡入淡出</button>
<div class="box" style="width:100px;height:100px;background:red;"></div>
```

jQuery Form Handling（表單處理）

簡化表單資料的取得和處理。

```
$(document).ready(function(){
    $("form").submit(function(event){
        event.preventDefault();
        alert("姓名: " + $("#name").val());
    });
});
```

```
<form>
    <label for="name">姓名:</label><input type="text" id="name" name="name">
    <input type="submit" value="提交">
</form>
```

jQuery Traversing（遍歷）

遍歷 DOM 結構以查找和選擇元素。

```
$(document).ready(function(){
    $("li").first().css("color", "blue"); // 選擇第一個 <li> 元素
    $("li").last().css("color", "red");
    $("li").css("font-size", "20px");
});
```

```
<ul>
    <li>第一項</li> <li>第二項</li> <li>第三項</li>
</ul>
```

jQuery Attribute Manipulation（屬性操作）

簡單地獲取和設置元素的屬性。

```
$(document).ready(function(){
    $("#changeSrc").click(function(){
        $("img").attr("src", "https://via.placeholder.com/150"); // 更改圖片的 src 屬性
    });
});
```

```
<button id="changeSrc">更改圖片</button>
<img src="https://via.placeholder.com/100" alt="圖片">
```