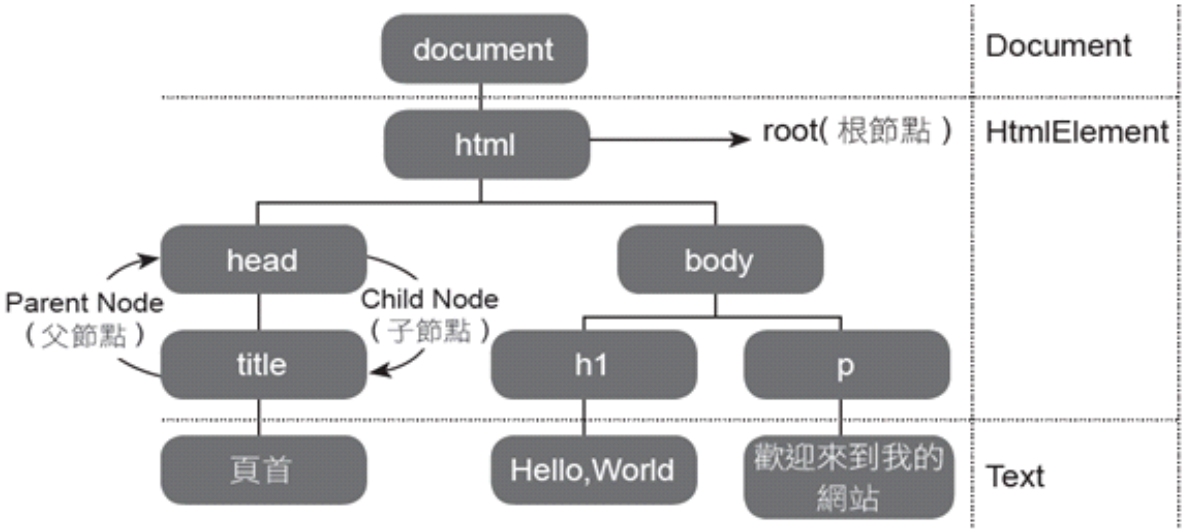
21.JavaScript - DOM

一、認識 DOM（Document Object Model）

簡單來說，在 DOM 的標準下，文件中所有的標籤定義，都是一個物件，這些物件以文件定義的結構，形成了一個樹狀結構。例如：

```
<html>
    <head>
        <title> 首頁 </title>
    </head>
    <body>
        <h1>Hello,World</h1>
        <p> 歡迎來到我的網站 </p>
    </body>
</html>
```

這份HTML 文件，會形成以下樹狀的物件結構：



在這個 HTML 的結構當中，每一個標籤及內容，都可以視為一個節點(Node)，各個節點相連出一個樹狀結構。document 代表整個文件，所有節點的起點為 html，一般稱為「根節點」(Root Node)。由 html 以下的每一層節點之間都有相對應的關係的，在任一節點的上層稱為「父節點」(Parent Node)，下層稱為「子節點」(ChildNode)，而每一個HTML 標籤內可能都還有屬性、文字等內容，這些都會被各自視為一個節點。

二、Window Object

JavaScript 與「瀏覽器」溝通的窗口，可以存取和操作瀏覽器的視窗，不涉及網頁內容。

常見 Window Object 方法：

- window.alert()
- window.EventListener()
- window.clearInterval()
- window.prompt()
- window.setInterval()

```
<script>
    function sayHi(){
        console.log("Hi");
    }

    setInterval(sayHi, 3000);
</script>
```

常見 Window Object 屬性：

- Console
- Document
- LocalStorage
- SessionStorage

三、尋找元素節點

- addEventListener()
- createElement() -新增指定元素標籤。
- getElementById() -尋找指定id元素節點。
- getElementByClassName() -尋找指定class元素節點。
- querySeletor() -尋找指定元素節點。
- querySeletorAll() -尋找指定元素節點。

```
<h1 id="first">歡迎光臨</h1>
<p class="second">勞動部勞動力發展署中彰投分署</p>
<script>
    let myH1 = document.getElementById("first");
    let myP = document.getElementsByClassName("second");
    console.log(myH1);
</script>
```

四、Arrow Functions 箭頭函數

ES6 新增用箭頭 => 來簡短定義函數，這種語法稱作 Arrow Functions。

```
function sayH1(){
    console.log("H1");
}

let sayH1 = () => {
    console.log("H1");
}
```

```
let TCNR = {
    name: "勞動部勞動力發展署中彰投分署",
    myName(){
        console.log("TCNR是" + this.name + "。");
    },

    address: () => {
        console.log("TCNR的地址是台中市西屯區工業區一路100號");
    }
}

TCNR.myName();
TCNR.address();
```

五、元素屬性

- childran - 回傳 HTMLCollection。
- childNode -回傳 NodeList。
- parentElement -回傳指定元素的上層元素。
- innerHTML- 插入指定 HTML 標籤。
- innerText -插入指定文字。
- appenChind() -插入指定元素屬性。
- classList -回傳類別屬性清單。
  - add() -新增指定類別屬性。
  - remove() -移除指定類別屬性。
  - toggle() -新增或移除指定類別屬性。
  - contains() -回傳是否包含指定類別屬性。
- getAttribute() -回傳指定元素屬性。
- querySeletor() -尋找指定元素屬性。
- querySeletorAll() -尋找指定元素屬性。
- remove() -移除指定元素。
- style - 控制 CSS style。

```
<h1 class="myH1"></h1>
<script>
    let h1 = document.querySelector("h1.myH1");
    h1.innerText = "勞動部勞動力發展署中彰投分署";
    console.log(h1);
</script>
```

```
let body = document.querySelector("body");

let myH1 = document.createElement("h1");
myH1.innerText = "勞動部勞動力發展署中彰投分署";

body.appendChild(myH1);
```

```
    <style>
        .red{
            background-color: red;
        }
        .blue{
            background-color: blue;
        }
        .bold{
            font-weight: 600;
        }
    </style>
</head>
<body>
    <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Molestiae
    praesentium sapiente repellendus debitis ullam tenetur mollitia illo
    voluptates vel accusantium? Sint provident minus placeat necessitatibus iusto
    aperiam qui ipsum voluptas?</p>
<script>
    let myP = document.querySelector("p");

    myP.classList.add("red");
</script>
```

```
    <a title="google" href="https://www.google.com.tw">Google</a>
<script>
    let a = document.querySelector("a");
    console.log(a.getAttribute("href"));
</script>
```

```
    <section>
        <p class="red">
            Lorem ipsum dolor, sit amet consectetur adipisicing elit. Impedit ut
            dignissimos magnam iure amet at incidunt et quos autem ducimus?
        </p>
        <p class="red">
            Lorem ipsum dolor sit amet, consectetur adipisicing elit. Numquam,
            quas?
        </p>
    </section>

    <p class="red">
        Lorem ipsum dolor sit amet consectetur, adipisicing elit. Neque, porro.
    </p>
<script>
    let section = document.querySelector("section");
    let redP2 = section.querySelectorAll("p.red");
    console.log(redP2);
</script>
```

```
    <h1>中彰投分署</h1>
<script>
    let heading = document.querySelector("h1");
    heading.remove();
</script>
```

```
    <h1>中彰投分署</h1>
<script>
    let h1 = document.querySelector("h1");
    h1.style.backgroundColor = "red";
    h1.style = "background-color: red;";
```

六、Events

使用者在瀏覽網頁時會觸發很多事件 (events) 的發生，例如按下滑鼠是一種事件、按下鍵盤按鍵是一種事件、網頁或圖片完成下載時是一種事件、表單欄位值被改變是一種事件...等等。

- addEventListener(event type, callback) 。

  event type - event 的樣式。

  callback - 執行的函式。

```
window.addEventListener("click", e => {
    console.log(e);
});
```

改寫 project3 的 header 滾動陰影效果。

- window.pageXOffset - 回傳頁面在視窗左上角水平方向滾動的位置（px）。
- window.pageYOffset - 回傳頁面在視窗左上角垂直方向滾動的位置（px）。

```
let header = document.querySelector("header");
window.addEventListener("scroll", () => {
    if(window.pageYOffset != 0){
        header.style = "box-shadow: 3px 3px 5px 1px #757474";
    }
});
```

七、Event Bubbling

指的是當某個 Events 發生時（如：click），這個 Events 會觸發本身的 callback，接下來會再觸發他的 父層的 callback，以及父層的父層的 callback，直到最上層。

```
<style>
    .a{
        width: 300px;
        height: 300px;
        background-color: red;
    }
    .b{
        width: 150px;
        height: 150px;
        background-color: blue;
    }
</style>
```

```
    <div class="a">
        <div class="b"></div>
    </div>
<script>
    let a = document.querySelector("div.a");
    let b = document.querySelector("div.b");

    a.addEventListener("click", () =>{
        alert("a的callback正在執行");
    })
    b.addEventListener("click", () =>{
        alert("b的callback正在執行");
    })
</script>
```

停止 Event Bubbling -stopPropagation()。

```
b.addEventListener("click", e =>{
    e.stopPropagation();
    alert("b的callback正在執行");
})
```

七、Storage

Storage 指的是瀏覽器中儲存資料的地方，但它並不是資料庫。Storage 是以 key & value 方式

儲存資料，需注意的是 Storage 僅可儲存字串型態資料。

Storage 有 localStroage 與 sessionStorage 兩種，最大的差別在於 sessionStorage 所儲存資料在瀏覽器關閉後即清空，但 localStroage 所儲存資料則會保存到使用者執行清空指令。

- setItem(key, value) - 新增資料存入Storage。
- getItem(key - 取得 Storage 內指定資料。
- removeItem(key) - 刪除 Storage 內指定資料。
- clear() - 清空 Storage。

```
<script>
    localStorage.setItem("name", "Thor");
    localStorage.setItem("age", "25");
    console.log(localStorage);

    let myAge = localStorage.getItem("age");
    console.log(myAge);
</script>
```