

JavaScript 基礎課程

@網站前端課程

在網頁裡使用javascript

00.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>在網頁裡使用javascript</title>
</head>
<body>
  <script>
    //coding
  </script>
</body>
</html>
```

變數宣告

var：變數可以提升且可重複宣告及重新賦值，容易引發意外行為，應避免使用。

let：不能重複宣告，但能重新賦值，是推薦的變數宣告方式。

const：類似 **let**，但宣告時必須賦值，且不能重新賦值，適合用於常量和不可變的數據。

變數命名規則

- 字母、數字、下劃線 (_)、或美元符號 (\$)
- 數字不能作為變數名稱的開頭
- 區分大小寫的， `myVariable` 和 `myvariable` 是兩個不同的變數。
- 禁止使用關鍵字或保留字，例如 `let`、`function`、`class`

資料形式

形式	說明
Number	數值型資料，既可以是整數，也可以是浮點數。
String	字串資料，使用單引號或雙引號來定義。
Boolean	只有 true 或 false 兩個值
array	存儲有序集合 <code>let fruits = ['apple', 'banana', 'orange'];</code>
object	複合型資料，包含鍵值對 <code>let person = { name: 'John', age: 30 };</code>

宣告變數

01.html

```
const a='const創建後，就不能再賦值'  
let b='let創建後，可再賦值';
```

```
//a='再複寫內容會(值)出錯'  
b='可再複寫內容(值)';
```

```
console.log(a)  
console.log(b)
```

02.html

```
<script>
  let txt = 'const創建後，就不能再賦值'
  let num1 = 1;
  let num2 = 2;
  let arr = ["apple", "banana", "tangerine"]
  const obj = { "name": "王小明", "age": 20, "job": "Web Designer" };
  //obj.name="張大壯"
  //obj['name']="張大壯"
  arr[2]="西瓜"
  console.log(txt);
  console.log(num1);
  console.log(arr[0]);
  console.log(obj['name'],obj.name);
  console.log(num1+num2);
  console.log(num1+txt);
  console.log(arr);
</script>
```

字串組合

03.html

```
<script>
```

```
const txt = "王建中就讀中和國中";  
const a = 3, b = '中'
```

```
console.log("a="+a +"並且b="+ b)  
console.log(`a=${a}並且b=${b}`)
```

```
console.log("王建中就讀中和國中這句話裡，有3個\"中\"字");  
console.log(txt + "這句話裡，有" + a + "個\"" + b + "\"字");  
console.log(`${txt}這句話裡，有${a}個\"${b}\"字`);
```

```
</script>
```


選取網頁中元素

`document.getElementById('myElementId')`

`document.getElementsByClassName('ClassName')` 返回一個類似陣列的物件HTMLCollection

`document.getElementsByTagName('div')`

`document.querySelector('#myClass')`

`document.querySelectorAll('.myClass')` 返回一個 NodeList，其中包含所有符合 .myClass (即具有該類名) 的元素。
NodeList 是類似陣列的物件，可以通過索引 (如 `odelist[0]`) 來訪問元素。

04.html

```
<script>
  console.log(document.getElementById('firstH2'))

  console.log(document.getElementsByClassName('myh2'))

  console.log(document.getElementsByClassName('myh2')[2].innerHTML)

  console.log(document.getElementsByTagName('h2'))

  console.log(document.getElementsByTagName('h2')[0].innerHTML)

  console.log(document.querySelector("h1").style.color)

  console.log(document.querySelector("h2.myh2").innerHTML = "我變了內容")

  console.log(document.querySelectorAll("h2.myh2")[2].innerHTML = "我2變了內容")

</script>
```

05.html

```
const a = 1
if (a > 1) {
    console.log('a > 1');
}
else if(a == 1){
    console.log('a = 1');
}
else {
    console.log('a < 1');
}
```

```
const a = 1,b='小'
if (a > 1 && b=='大') {
    console.log('a > 1 '+' b= 大' );
}
else if(a == 1 || b=='中'){
    console.log('a = 1, b='+ b);
}
else {
    console.log(`a=${a},b=${b}`);
}
```

使用function()

06.html

```
<script>
  let a = 1
  let b = 2
  handlerA()
  handlerB(3,4)
  //handlerC(3,4);
  function handlerA() {
    console.log(a + b);
  }
  function handlerB(num1, num2) {
    console.log("handlerB:" +( num1 + num2));
  }

  const handlerC=(num1,num2)=>{
    console.log(`handlerB: ${num1*num2}`);
  };
  handlerC(3,4);
</script>
```



元素.addEventListener("事件", 動作)

常用事件

滑鼠事件 (Mouse Events)

dblclick: 雙擊滑鼠
mousedown: 按下滑鼠按鍵
mouseup: 放開滑鼠按鍵
mousemove: 滑鼠移動
mouseover: 滑鼠移到元素上方
mouseout: 滑鼠移出元素
contextmenu: 右鍵點擊 (顯示上下文選單)

鍵盤事件 (Keyboard Events)

keydown: 按下鍵盤按鍵
keyup: 放開鍵盤按鍵
keypress: 按下任意鍵 (包括重複按下的鍵)
表單事件 (Form Events):

submit: 提交表單

change: 表單元素改變 (如 <input> 、<select>)
focus: 元素獲得焦點
blur: 元素失去焦點
input: 輸入時觸發，適用於 <input> 、<textarea>

視窗事件 (Window Events)

resize: 視窗大小改變
scroll: 捲動視窗或元素
load: 頁面資源載入完成
unload: 使用者離開頁面
beforeunload: 使用者即將離開頁面

觸控事件 (Touch Events)

touchstart: 手指觸摸到螢幕
touchmove: 手指在螢幕上移動
touchend: 手指離開螢幕
touchcancel: 觸控事件被取消

剪貼板事件 (Clipboard Events)

copy: 複製內容
cut: 剪下內容
paste: 貼上內容

拖放事件 (Drag and Drop Events)

drag: 開始拖動元素
dragstart: 開始拖動
dragend: 拖動結束
dragenter: 拖動的元素進入放置區域
dragover: 元素被拖動到可放置區域上方
dragleave: 拖動元素離開放置區域
drop: 元素被放置

更多 : <https://developer.mozilla.org/zh-CN/docs/Web/API>

事件行為

```
<script>
  const btnHanlder = () => {
    console.log(document.querySelector("#password").value);
    if (document.querySelector("#password").value == "0000") {
      //document.getElementsByTagName('img')[0].style.display="block"
      document.querySelector('#errMsg').innerHTML = ""
      document.querySelector('.img-fluid').style.display = "block"
    }else{
      document.querySelector('#errMsg').innerHTML = "密碼錯誤"
      document.querySelector('.img-fluid').style.display = "none"
    }
  }
  document.querySelector("#btn").addEventListener("click", btnHanlder)
</script>
```

改變樣式

work-js-1/index.html

```
const nav = document.querySelector('nav')
window.addEventListener("scroll", () => {
  if (window.scrollY >= 120) {
    nav.style.height = '60px';
  } else {
    nav.style.height = '120px';
  }
})
```

```
const nav = document.querySelector('nav')
window.onscroll = function () {
  if (window.scrollY >= 120) {
    nav.classList.add('short');
  } else {
    nav.classList.remove('short');
  }
}
```

```
nav.style.setProperty('--nav-h', '60px');
```


手機般選單(DOM.classList)

work-js-1/index.html

```
mclick.addEventListener('click', () => {  
    mclick.classList.toggle('checked')  
})
```

for迴圈

```
const totalNum=9

for (let i = 1; i <= totalNum; i++) {

    console.log(i);

    //自動執行9次動作

}
```

setTimeout

09.html

```
const showPicture=document.querySelector("#showPicture");
const mask = document.querySelector("#mask");
const btnClose = document.querySelector("#btnClose");
const setTimeout= () => {
    setTimeout(() => {
        mask.classList.toggle('d-none')
        showPicture.classList.toggle('blur')
    }, 3000);
}
setTimeout()
btnClose.addEventListener('click', () => {
    mask.classList.toggle('d-none');
    showPicture.classList.toggle('blur')
    setTimeout();
})
```

- 用來 延遲執行
- 只會執行一次

setInterval

10.html

```
const galleryArr = ["gallery-1.jpg", "gallery-2.jpg", "gallery-3.jpg", "gallery-4.jpg"];
const gallery = document.querySelector("#gallery")
const btnCntrol = document.querySelector("#btnCntrol")
let Interval; // 在 瀏覽器環境 中，Interval 會顯示一個正整數，如 1、2、3 等等。
let index = 1; // 如果設定為0的話，第1張會多等一輪，1--3sec--1--3sec--2
const galleryPlay = () => {
  Interval = setInterval(() => {
    gallery.src = `./img/${galleryArr[index]}`
    if (index < galleryArr.length - 1) {

      index++
    } else {
      index = 0
    }
  }, 3000);
}
galleryPlay()
btnCntrol.addEventListener('click', (e) => {
  if (e.target.innerText == 'Pause') {
    e.target.innerText = 'Play'
    clearInterval(Interval)
  } else if (e.target.innerText == 'Play') {
    e.target.innerText = 'Pause'
    galleryPlay()
  }
})
})
```

- 重複執行該函數
- 直到被停止。

11-order.html

1. 商品區可自行每個商品選擇數量
2. 即時計算
3. 按下按鈕跑css進度條
4. **SetTimeOut**關閉css進度條並跳出交易完成畫面
5. 清空購物清單

購物清單 object形式

12.html

```
for (let i = 0; i < product.length; i++) {  
    //將每項商品的金額寫進物件  
    product[i].amounts = product[i].count * product[i].price  
    //生成表單原始碼  
    tbodyHTML += `|  
        <td>${product[i].name}</td>  
        <td>${product[i].price}</td>  
        <td>${product[i].count}</td>  
        <td>${product[i].count * product[i].price}</td>  
    </tr>`  
}  
//程式碼注入DOM  
tbody.innerHTML = tbodyHTML;

```

定義資料搭配前端行為

12-store.html

```
<script type="module" src="./12-stores.js"></script>
import { storeslist } from './storeslist.js';
```

```
export const storeslist = [
  {
    "city": "台北市",
    "store": [
      {
        "name": "台北市-BESV 實踐旗艦店 (直營門市)",
        "address": "台北市中山區大直街70號H棟2樓",
        "tel": "02-25333898"
      },
      {
        "name": "台北市-BESV 統一時代 5F (直營門市)",
        "address": "台北市信義區忠孝東路五段8號",
        "tel": "02-25333898"
      }
    ]
  }
]
```

台北市



台北市-BESV 實踐旗艦店 (直營門市)



台北市中山區大直街70號H棟2樓

陣列.find() 方法

find() 找到第一個符合條件的元素後，就會停止遍歷，不會繼續檢查剩下的元素

```
const numbers = [1, 2, 3, 4, 5, 6];
```

```
const foundNumber = numbers.find(value => value > 3);
```

```
console.log(foundNumber); // 輸出：4
```


定義外部資料-json格式

user.json

```
{  
  "name": "王聰明",  
  "gender": "男",  
  "occupation": "UI設計師",  
  "skills": "HTML、CSS、JavaScript",  
  "ishow": true  
}
```

fetch取得外部資料-json

01-fetch-0.html

```
const ul = document.querySelector("ul")
const h2 = document.querySelector("h2")
fetch('user.json')
  .then(response => {
    if (!response.ok) {
      return response.json();
    }
  })
  .then(res => {
    if (res.ishow) {
      h2.innerHTML = '用戶資訊顯示'
      ul.innerHTML += `
        <li>${res.name}</li>
        <li>${res.gender}</li>
        <li>${res.occupation}</li>
        <li>${res.skills}</li>
      `
    } else {
      h2.innerHTML = `用戶資訊隱藏`
    }
  })
  .catch(error => {
    console.error(error);
  });
```

fetch取得外部資料-json

01-fetch-1.html

```
let tbodyHTML = ""
fetch('products.json')
  .then(response => {
    return response.json();
  })
  .then(res => {
    res.forEach((element) => {
      element.amounts = element.count * element.price
      tbodyHTML += `<tr>
        <td>${element.name}</td>
        <td>${element.price}</td>
        <td>${element.count}</td>
        <td>${element.count * element.price}</td>
      </tr> `
    });
    tbody.innerHTML = tbodyHTML;
  })
  .catch(error => {
    // 攔截錯誤
    console.error(error);
  });
```

Axios取得外部資料-json

01-fetch-2-get.html

```
<script src="https://cdn.jsdelivr.net/npm/axios/dist/axios.min.js"></script>
```

```
let tbodyHTML = ""
axios.get('products.json')
  .then(res => {
    const data=res.data
    console.log(res.data);

    data.forEach((element, index) => {
      element.amounts = element.count * element.price
      tbodyHTML += `<tr>
        <td>${element.name}</td>
        <td>${element.price}</td>
        <td>${element.count}</td>
        <td>${element.count * element.price}</td>
      </tr> `
    });
    tbody.innerHTML = tbodyHTML;
  })
  .catch(error => {
    // 攔截錯誤
    console.error(error);
  });
```

Axios取得外部資料-json

01-fetch-2-post.html

```
// 點擊按鈕後發送 POST 請求
const postButton = document.querySelector('#postButton')
const pre = document.querySelector("#pre")
const result = document.querySelector("#result")
const title = document.querySelector("#title")
const body = document.querySelector("#body")
const userId = document.querySelector("#userId")
postButton.addEventListener('click', () => {
  axios.post('https://jsonplaceholder.typicode.com/posts', {
    title: title.value,
    body: body.value,
    userId:userId.value
  })
  .then(res => {
    console.log('Response data:', res.data);
    pre.innerHTML = JSON.stringify(res.data)
    result.innerHTML = 'POST 請求成功！';
  })
  .catch(error => {
    // 錯誤處理
    console.error('Error:', error);
    result.innerHTML = '發生錯誤，請檢查控制台。';
  });
});
```

fetch也可以使用post方式發送api請求，參考
https://developer.mozilla.org/zh-CN/docs/Web/API/Fetch_API/Using_Fetch

async.. await

01-fetch-3-async.html

```
let tbodyHTML = ""
const handleAsync = async () => {
  try {
    const res = await axios.get('products.json')
    console.log(res.data);
    res.data.forEach((element, index) => {
      element.amounts = element.count * element.price
      tbodyHTML += `<tr>
        <td>${element.name}</td>
        <td>${element.price}</td>
        <td>${element.count}</td>
        <td>${element.count * element.price}</td>
      </tr>`
    });
    tbody.innerHTML = tbodyHTML;
  } catch (error) {
    console.log(error);
  } finally {
    console.log("不管成功或失敗，都會進入finally");
  }
}
handleAsync();
```

天氣api 01-api.html

```
const Authorization = '授權碼'
const api = `https://opendata.cwa.gov.tw/api/v1/rest/datastore/F-C0032-001?Authorization=${Authorization}`
let weatherDate = {}
let minT = ""
let maxT
const handleAsync = async (city) => {
  try {
    showcity.innerHTML=city
    const res = await axios.get(api)
    weatherDate = res.data.records.location;
    weatherDate.forEach(element => {
      if (element.locationName == city) {
        weatherInfo.innerHTML = element.weatherElement[0].time[0].parameter.parameterName
        minT = element.weatherElement[2].time[0].parameter.parameterName
        maxT = element.weatherElement[4].time[0].parameter.parameterName
        temperature.innerHTML = `${minT}°C~${maxT}°C`

        console.log(element);
      }
    });
  } catch (error) {
  }
}
handleAsync('臺北市')
btnCity.forEach(element => {
  element.addEventListener("click", (e) => {
    console.log(e.target.innerHTML);
    handleAsync(e.target.innerHTML)
    body.style.backgroundImage= `url(./img/${e.target.innerHTML}.jpg)`
  })
})
```