<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge" />

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <script>

        const person = {

            myname: '小明',

            age: 16,

            like: '鍋燒意麵',

            url: 'https://randomuser.me/api',

        };

        //取出特定值(解構)

        //傳統:(程式碼很長)

        const mynameOld = person.myname;

        console.log(mynameOld);

        //現在:(較簡單，可取出多值)

        const { myname, age, like } = person;

        console.log(myname, age, like);

        //陣列的解構:

        const people = ['小明', '杰倫', '漂亮阿姨'];

        const [Ming, Jay, Auntie] = people;

        console.log(Ming, Jay, Auntie);

        //重新命名

        const { myname: newname } = person;

        console.log(newname);

        //預設值:

        const { like: favoriteFood = '拉麵' } = person;

        console.log(favoriteFood);

        //連接api(用到箭頭函數，以便更簡潔的使用語法)

        fetch('https://randomuser.me/api')

            .then(res => res.json())

            .then(json => {

                //(基本方式)

                console.log(json.results[0]);

                //(解構方式)

                const [result] = json.results;

                console.log(result);

                //針對特定值取出

                const { email, phone } = result;

                console.log(email, phone);

            });

        // Vue, React > props

        function fn(props) {

            console.log(props);

        }

        fn(person);

        function fn2({ myname, like }) {

            console.log(myname, like);

        }

        fn2(person);

        function fn3() {

            return {

                myname: '小明',

                age: 16,

                like: '鍋燒意麵',

                url: 'https://randomuser.me/api',

            };

        }

        const { url } = fn3();

        console.log(fn3().url);

        //陣列的展開

        const people1 = ['小明', '杰倫'];

        const people2 = ['媽媽', '爸爸'];

        const people3 = [...people1, ...people2];

        console.log(people3);

        const person2 = { ...person };

        person2.myname = '杰倫';

        console.log(person.myname); // 輸出結果為'小明'

        console.log(person === person2); // 輸出結果為false

        function fn4(n1, n2) {

            console.log(n1, n2);

        }

        fn4(...people1); // 會將小明和杰倫放到n1和n2的位置

        function fn5() {

            console.log(arguments); // 類陣列

        }

        fn5(1, 2, 3, 4, 5);

        //展開修改後可使用forEach:

        function fn6() {

            const args = [...arguments];

            args.forEach(a => {

                console.log(a);

            });

        }

        fn6(1, 2, 3, 4, 5);

        // 如果有許多 <p> 元素

        const dom = document.querySelectorAll('p');

        console.log(dom); // 結果仍為類陣列

        const domArr = [...document.querySelectorAll('p')];

        console.log(domArr); // 會變為純陣列

        const dom2 = [...domArr];

        dom2.map(a => { console.log(a); }); // 這樣map就可以運作

        //其餘參數

        function fn7(...args) {

            console.log(args);

        }

        fn7(1, 2, 3, 4, 5);

        //增加a(型態為純陣列，可使用forEach將資料一筆筆帶出)

        function fn8(a, ...args) {

            console.log(a, args);

            args.forEach(arg => {

                console.log(arg);

            });

        }

        fn8(1, 2, 3, 4, 5); // 輸出結果1會被a圈住，args則表示[2, 3, 4, 5]

        //let: 宣告變數，能重新賦予數值

        let x = 10;

        if (x === 10) {

            let x = 20; // 區塊作用域內的新變數

            console.log(x); // 20

        }

        console.log(x); // 10

        //const:宣告常數，不能重新賦予數值。

        const y = 30;

        // y = 40; // 非法，會拋出錯誤

        const arr = [1, 2, 3];

        arr.push(4); // 合法，數組內容可以改變，但不能重新賦值

        console.log(arr); // [1, 2, 3, 4]

        const obj = { a: 1 };

        obj.b = 2; // 合法，對象內容可以改變，但不能重新賦值

        console.log(obj); // {a: 1, b: 2}

        //物件縮寫方式

        const App = () => {

            //Read value, Set Value Method

            const [val, setVal] = React.useState(0);

            return <div>

                <button onClick={() => { setVal(val + 1); }}>按我</button>

                {val}

            </div>;

        };

        const myname2 = '小明';

        const callname = function () {

            console.log(this.myname);

        };

        //傳統(會寫得較詳細)

        const person3 = {

            myname: '小明',

            callname: function () {

                console.log(this.myname);

            },

        };

        //縮寫後:

        const person4 = {

            myname: '小明',

            callname() {

                console.log(this.myname);

            },

        };

        person4.callname();

        //屬性縮寫:

        const person5 = {

            myname,

            callname,

        };

        person5.callname();

        console.log(person5);

        const person6 = {

            ...person5,

            method() {

                console.log('這是一個新的方法');

            },

        };

        console.log(person6 === person5); // false

        person6.method(); // 輸出這是一個新的方法

        // person.method(); // 沒有method方法

        //可選串連

        const person7 = {

            myname: '小明',

            age: 16,

            like: '鍋燒意麵',

            url: 'https://randomuser.me/api',

            friend: {

                Jay: {

                    name: '杰倫',

                    id: 1,

                },

                Auntie: {

                    name: '漂亮阿姨',

                    id: 2,

                },

            },

        };

        //正常:

        console.log(person7.friend.Jay.id); // 如果沒有資料時會出現錯誤，但會因為錯誤導致後續無法執行

        console.log(person7.friend?.Jay?.id); // 這樣會顯示undefined，但可以使後續繼續執行(避免生成錯誤)

        if (person7?.friend?.Auntie?.id) {

            console.log(person7.friend.Auntie);

        } // 當增加?時，可避免錯誤

    </script>

</body>

</html>

使用網頁

<https://www.youtube.com/watch?v=y7sgpXhH0Vc>

<https://www.casper.tw/development/2021/09/14/js-console/>