Javascript

- 【bootstrap-table】相關連結
- 【Bootstrap-table】bs_table_common.js
- 【ES6】其他新語法
- 【ES6】物件拷貝
- 【ES6】物件拷貝
- 【ES6】非同步 與 promise
- 【ES6】展開
- 【ES6】陣列方法
- 【ES6】箭頭函式
- 【ES6】賦值解構
- 【ES6】async / await
- 【ES6】ESModual
- 【ES6】import / export
- 【JS】相關資源
- 【JS】iframe 自適應內容高度
- 【ES6】null 判斷
- 【Tool】常用方法

【bootstrap-table】相關連結

- bootstap-table 官網
- bootstrap table filterBy□据刷□□□器和查询条件 - itxst.com

【Bootstrap-table】bs_table_common.js

```
var bootstrapTableBaseConfig = {
    height: getHeight(),
    idField:"dv_no",
    toolbar:"#toolbar",
    minimumCountColumns:2, //最小顯示欄位數量
    pagination:true,
    pageSize:'10000000', //預設row筆數
    pageList:['ALL'],
    queryParams:getQueryParams,
    search:false,
    showColumns:true,
    showFooter:false,
    showPaginationSwitch:true,
    showRefresh:true,
    showExport:true, //匯出按鈕
    showToggle:false,
    detailView:false,
    detailFormatter:detailFormatter,
    // responseHandler:responseHandler,
    virtualScroll:false,
    sidePagination:"server",
    contextMenu: '#bootstraptable-context-menu',
    contextMenuButton: '.bootstraptable-button',
    classes:'table table-bordered table-hover table-sm table-condensed nowrap',
}


/** common function begin */

function ajaxUpdata(config){

    var url = config.url || "";
    var update_type = config.update_type || "更新";
    var update_data = config.update_data || {};
    var ajax_type = config.ajax_type || "POST";
    var ajax_data_type = config.ajax_data_type || "json";
    var after_success_func = config.after_success_func || function(){};
    var after_error_func = config.after_success_func || function(){};
    // var bs_table = config.bs_table || $table;

    console.log('ajaxUpdata_config',config);

    var success_func = config.success_func || function(response){
        console.log("response",response);
        if (response.msg != undefined && (response.msg == "success")) {
            if(config.bs_table !== undefined) config.bs_table.bootstrapTable('refresh');
            alert(update_type+"資料成功!!");
        } else {
            alert((response.msg == undefined)?update_type+"失敗請洽系統管理員":response.msg);
        }
        after_success_func();
    }

    var error_func = success_func || function(response){
        var err_msg = response.msg || "";
        alert(err_msg+","+update_type+"失敗請洽系統管理員!!");
        // dialog.close();  //關閉對話視窗
        after_error_func();
    }
```

```
    var ajax_config = {
            url:  url,
            type: ajax_type,
            data: update_data,
            error: error_func,
            success: success_func,
            dataType: ajax_data_type
            };
    console.log(ajax_config);
    //寫入資料庫
    $.ajax(ajax_config);
}


function getEditDataStr(beforDate,afterData){
    var resultTextArr = [];
    for(var key in beforDate){
        if (afterData[key] !== undefined && beforDate[key] != afterData[key]
        ){
            if (beforDate[key] == null && afterData[key] == '') continue;
            resultTextArr.push(key+":"+beforDate[key]+"=>"+afterData[key]);
        }
    }
    if(resultTextArr.length > 0){
        return resultTextArr.join(",");
    }else{
        return "";
    }
}

function getDataByClass(class_name,perfix){
    var data = {};
    _perfix = perfix || "";
    $('.'+class_name).each(function(){
        var el = this;
        console.log(el.id);
        var id = (el.id).replace(_perfix,'');
        var value = $(el).val();
        data[id] = value;

    });
    return data;
}


function initTypeahead(config){

    console.log('initTypeahead typeahead',typeof jQuery().typeahead);
    console.log('initTypeahead Bloodhound',typeof Bloodhound);
    //check typeahead exist
    if(typeof jQuery().typeahead !== 'function') return;
    if(typeof Bloodhound !== 'function') return;

    console.log("initTypeahead");

    var selectFunc = (config && config.selectFunc) || function(event, data){
        $("#USER_ID").val(data['USER_ID']);
        $("#USER_NAME").val(data['USER_NAME']);
    }
    var userInfo = new Bloodhound({
        datumTokenizer: Bloodhound.tokenizers.obj.whitespace('value'),
        queryTokenizer: Bloodhound.tokenizers.whitespace,
        remote: {
            url: '/db/php/getOisUser.php?search=%QUERY',
            wildcard: '%QUERY'
        }
    });

    $('#typeahead_input').typeahead({
```

```
        // classNames: {
        //  input: 'Typeahead-input',
        //  hint: 'Typeahead-hint',
        //  selectable: 'Typeahead-selectable'
        // }
       // },
       hint: true,
       highlight: true,
       minLength: 1
       },
       {
       name: 'best-pictures',
       display: function(data){
          return  data['USER_ID']  ;
          // return  data['USER_ID']  + '-' + data['AD_ID']  + '-' + data['USER_NAME'];
       },
       source: userInfo,
       templates: {
          //查無資料顯示
          empty: [
          '<div class="empty-message">',
             '查無使用者',
          '</div>'
          ].join('\n'),

          suggestion: function(data) {
             return '<p><strong>' + data['USER_ID']  + '-' + data['AD_ID']  + '-' + data['USER_NAME'] + ' #' + data['EXT'] + '</strong>
</p>';
          }
       }
    });

    $('.typeahead').on('typeahead:select', function(event, data) {
       selectFunc(event, data);
    });

    $('.typeahead').on('typeahead:change ', function(event, data) {
       // console.log('change : ' , data);
    });

    $('.js_typeahead_reset_btn').on('click',function(){
       $('.js_typeahead_reset').each(function(index,el){
          $(el).val('');
       });
    });

    // for(var i = 1;i<=5;i++){
    //     if($('.js_typeahead_reset_btn_'+i).length > 0){
    //         $('.js_typeahead_reset_btn_'+i).on('click',function(){
    //             $('.js_typeahead_reset_'+i).each(function(index,el){
    //                 $(el).val('');
    //             });
    //         });
    //     }
    // }
}

function isEmpty(str){
    if(str === undefined || str.trim() === "" || str === null){
       return true;
    }
    return false;
}


function getEditDataStr(beforDate,afterData){
    var resultTextArr = [];
```

```
    for(var key in beforDate){
        if (afterData[key] !== undefined && beforDate[key] != afterData[key]
        ){
            if (beforDate[key] == null && afterData[key] == '') continue;
            resultTextArr.push(key+":"+beforDate[key]+"=>"+afterData[key]);
        }
    }
    if(resultTextArr.length > 0){
        return resultTextArr.join(",");
    }else{
        return "";
    }
}
/*
================================================================================================


function responseHandler(res) {
    $.each(res.rows, function (i, row) {
        row.state = $.inArray(row.dv_no, selections) !== -1;
    });
    return res;
}

function getHeight(_height) {
    var _height = _height || 110;
    return $(window).height() - $(".panel-heading").height() - _height;
}

// function getQueryParams(params){

//     var queryParams = {};
//     $(".query").each(function(i,e){
//         var id = (e.id).replace("q_","");
//         queryParams[id]=$(e).val();
//     });
//     $.extend(queryParams,params);
//         console.log("getQueryParams",queryParams);

//     return queryParams;
// }


function getQueryParams(params){
    var queryParams = {};
    $(".query").each(function(i,e){
        var id = (e.id).replace("q_","");
        queryParams[id]=$(e).val();
    });
    $.extend(queryParams,params);
    return queryParams;
}



/* bootsrap table formatter */
function detailFormatter(index, row) {
    var html = [];
    var trans = [];
    $.each(row, function (key, value) {
        if (key != 'total' && key != 'check'){
            var name=(trans[key] === undefined) ? key: trans[key];
            var value=(value === null || value === undefined ) ? '': value;
            html.push('<p><b>' + name + ':</b> ' + value + '</p>');
        }
    });
    return html.join('');
}
```

```
function operateFormatter(value, row, index) {

    return [
        '<a class="add" href="javascript:void(0)" title="add">',
        '<i class="glyphicon glyphicon-plus text-success"></i>',
        '</a> ',
        '<a class="edit" href="javascript:void(0)" title="edit">',
        '<i class="glyphicon glyphicon-edit"></i>',
        '</a> ',
        '<a class="remove" href="javascript:void(0)" title="remove">',
        '<i class="glyphicon glyphicon-remove text-danger"></i>',
        '</a>'
    ].join('');
}

function editFormatter(value, row, index){
    return [
    '<a class="edit" href="javascript:void(0)" title="edit">',
    '<i class="glyphicon glyphicon-edit"></i>',
    '</a>'
    ].join('');
}

function yesNoFormatter(value,row,index){
    return (value == 1)?'是':'否';
}

function snFormatter(value,row,index) {
    return index+1;
}

function scanNumberFormatter(data) {
    return (data == 1)?'是':'否';
}

function totalTextFormatter(data) {
    return 'Total';
}

function totalNameFormatter(data) {
    return data.length;
}

function nowrapFormatter(){
    return { css:{"whiie-space":"nowrap"}} ;
}

function totalPriceFormatter(data) {
    var total = 0;
    $.each(data, function (i, row) {
        total += +(row.price.substring(1));
    });
    return '$' + total;
}


 // function buildTable($el) {
        //     var i, j, row,
        //             columns = [],
        //             data = [];
        //     var opt = $el.bootstrapTable('getOptions');
        //     var fixedColumnsOption = { fixedColumns: true,fixedNumber: +$('#fixedNumber').val()};
        //     $.extend(opt,fixedColumnsOption);
        //     console.log("bootstrap_option",opt);
        //     $el.bootstrapTable('destroy').bootstrapTable(opt);
        // }
```

```
$(window).resize(function () {

    var $table = $table || null;
    if($table !== null){

        $table.bootstrapTable('resetView', {
            height: getHeight()
        });
    }
});
```

選項設定預設值

```
var DEFAULTS = {
    height: undefined,
    classes: 'table table-bordered table-hover',
    buttons: {},
    theadClasses: '',
    headerStyle: function headerStyle(column) {
      return {};
    },
    rowStyle: function rowStyle(row, index) {
      return {};
    },
    rowAttributes: function rowAttributes(row, index) {
      return {};
    },
    undefinedText: '-',
    locale: undefined,
    virtualScroll: false,
    virtualScrollItemHeight: undefined,
    sortable: true,
    sortClass: undefined,
    silentSort: true,
    sortName: undefined,
    sortOrder: undefined,
    sortReset: false,
    sortStable: false,
    rememberOrder: false,
    serverSort: true,
    customSort: undefined,
    columns: [[]],
    data: [],
    url: undefined,
    method: 'get',
    cache: true,
    contentType: 'application/json',
    dataType: 'json',
    ajax: undefined,
    ajaxOptions: {},
    queryParams: function queryParams(params) {
      return params;
    },
    queryParamsType: 'limit',
    // 'limit', undefined
    responseHandler: function responseHandler(res) {
      return res;
    },
    totalField: 'total',
    totalNotFilteredField: 'totalNotFiltered',
    dataField: 'rows',
    footerField: 'footer',
    pagination: false,
    paginationParts: ['pageInfo', 'pageSize', 'pageList'],
    showExtendedPagination: false,
    paginationLoop: true,
    sidePagination: 'client',
    // client or server
```

```
totalRows: 0,
totalNotFiltered: 0,
pageNumber: 1,
pageSize: 10,
pageList: [10, 25, 50, 100],
paginationHAlign: 'right',
// right, left
paginationVAlign: 'bottom',
// bottom, top, both
paginationDetailHAlign: 'left',
// right, left
paginationPreText: '&lsaquo;',
paginationNextText: '&rsaquo;',
paginationSuccessivelySize: 5,
// Maximum successively number of pages in a row
paginationPagesBySide: 1,
// Number of pages on each side (right, left) of the current page.
paginationUseIntermediate: false,
// Calculate intermediate pages for quick access
search: false,
searchHighlight: false,
searchOnEnterKey: false,
strictSearch: false,
regexSearch: false,
searchSelector: false,
visibleSearch: false,
showButtonIcons: true,
showButtonText: false,
showSearchButton: false,
showSearchClearButton: false,
trimOnSearch: true,
searchAlign: 'right',
searchTimeOut: 500,
searchText: '',
customSearch: undefined,
showHeader: true,
showFooter: false,
footerStyle: function footerStyle(column) {
  return {};
},
searchAccentNeutralise: false,
showColumns: false,
showColumnsToggleAll: false,
showColumnsSearch: false,
minimumCountColumns: 1,
showPaginationSwitch: false,
showRefresh: false,
showToggle: false,
showFullscreen: false,
smartDisplay: true,
escape: false,
filterOptions: {
  filterAlgorithm: 'and'
},
idField: undefined,
selectItemName: 'btSelectItem',
clickToSelect: false,
ignoreClickToSelectOn: function ignoreClickToSelectOn(_ref) {
  var tagName = _ref.tagName;
  return ['A', 'BUTTON'].includes(tagName);
},
singleSelect: false,
checkboxHeader: true,
maintainMetaData: false,
multipleSelectRow: false,
uniqueId: undefined,
cardView: false,
detailView: false,
detailViewIcon: true,
```

```
    detailViewByClick: false,
    detailViewAlign: 'left',
    detailFormatter: function detailFormatter(index, row) {
      return '';
    },
    detailFilter: function detailFilter(index, row) {
      return true;
    },
    toolbar: undefined,
    toolbarAlign: 'left',
    buttonsToolbar: undefined,
    buttonsAlign: 'right',
    buttonsOrder: ['paginationSwitch', 'refresh', 'toggle', 'fullscreen', 'columns'],
    buttonsPrefix: CONSTANTS.classes.buttonsPrefix,
    buttonsClass: CONSTANTS.classes.buttons,
    icons: CONSTANTS.icons,
    iconSize: undefined,
    iconsPrefix: CONSTANTS.iconsPrefix,
    // glyphicon or fa(font-awesome)
    loadingFontSize: 'auto',
    loadingTemplate: function loadingTemplate(loadingMessage) {
      return "<span class=\"loading-wrap\">\n      <span class=\"loading-text\">".concat(loadingMessage, "</span>\n      <span
class=\"animation-wrap\"><span class=\"animation-dot\"></span></span>\n      </span>\n    ");
    },
    onAll: function onAll(name, args) {
      return false;
    },
    onClickCell: function onClickCell(field, value, row, $element) {
      return false;
    },
    onDblClickCell: function onDblClickCell(field, value, row, $element) {
      return false;
    },
    onClickRow: function onClickRow(item, $element) {
      return false;
    },
    onDblClickRow: function onDblClickRow(item, $element) {
      return false;
    },
    onSort: function onSort(name, order) {
      return false;
    },
    onCheck: function onCheck(row) {
      return false;
    },
    onUncheck: function onUncheck(row) {
      return false;
    },
    onCheckAll: function onCheckAll(rows) {
      return false;
    },
    onUncheckAll: function onUncheckAll(rows) {
      return false;
    },
    onCheckSome: function onCheckSome(rows) {
      return false;
    },
    onUncheckSome: function onUncheckSome(rows) {
      return false;
    },
    onLoadSuccess: function onLoadSuccess(data) {
      return false;
    },
    onLoadError: function onLoadError(status) {
      return false;
    },
    onColumnSwitch: function onColumnSwitch(field, checked) {
      return false;
```

```
  },
  onPageChange: function onPageChange(number, size) {
    return false;
  },
  onSearch: function onSearch(text) {
    return false;
  },
  onToggle: function onToggle(cardView) {
    return false;
  },
  onPreBody: function onPreBody(data) {
    return false;
  },
  onPostBody: function onPostBody() {
    return false;
  },
  onPostHeader: function onPostHeader() {
    return false;
  },
  onPostFooter: function onPostFooter() {
    return false;
  },
  onExpandRow: function onExpandRow(index, row, $detail) {
    return false;
  },
  onCollapseRow: function onCollapseRow(index, row) {
    return false;
  },
  onRefreshOptions: function onRefreshOptions(options) {
    return false;
  },
  onRefresh: function onRefresh(params) {
    return false;
  },
  onResetView: function onResetView() {
    return false;
  },
  onScrollBody: function onScrollBody() {
    return false;
  },
  onTogglePagination: function onTogglePagination(newState) {
    return false;
  }
};
```

【ES6】其他新語法

預設值

```
function sum(a, b = 1) { // 加入預設值避免錯誤
  return a + b;
}
console.log(sum(1));
```

【ES6】物件拷貝

淺層拷貝(只複製一層，內容物件還是參考至同一位址)

```
const person = {
  name: '小明',
  obj: {}
}

const person2 = {...person} //方法1
const person3 = Object.assign({},person); //方法2
```

深層拷貝

```
const person = {
  name: '小明',
  obj: {}
}

// 使用josn轉成字串，再轉回物件
const person2 = JSON.parse(JSON.stringify(person))
```

【ES6】物件拷貝

淺層拷貝(只複製一層，內容物件還是參考至同一位址)

```
const person = {
  name: '小明',
  obj: {}
}

const person2 = {...person} //方法1
const person3 = Object.assign({},person); //方法2
```

深層拷貝

```
const person = {
  name: '小明',
  obj: {}
}

// 使用josn轉成字串，再轉回物件
const person2 = JSON.parse(JSON.stringify(person))
```

【ES6】非同步 與 promise

非同步

```
function getData() {
  setTimeout(() => {
    console.log('... 已取得遠端資料');
  }, 0);
}
// 請問取得資料的順序為何
function init() {
  console.log(1);
  getData();
  console.log(2);
}
init();
////////////// result
1
2
... 已取得遠端資料
```

premise 使程式流程同步

```
const promiseSetTimeout = (status) => {
  return new Promise((resolve, reject) => {
    setTimeout(() => {
      if (status) {
        resolve('promiseSetTimeout 成功')
      } else {
        reject('promiseSetTimeout 失敗')
      }
    }, 0);
  })
}

// #2-1 基礎運用 (成功用 then 捕捉, 失敗用 catch捕捉)
// 呼叫成功
console.log(1)
promiseSetTimeout(1)
.then((res)=>{
    console.log(res);
})
.catch((err)=>{
    console.log(err);
})
console.log(2)
/** result
1
promiseSetTimeout 成功
2
*/

// 呼叫失敗
console.log(1)
promiseSetTimeout(2)
.then((res)=>{
    console.log(res);
})
.catch((err)=>{
    console.log(err);
})
console.log(2)
/** result
1
promiseSetTimeout 失敗
2
```

```
*/
```

promise 串接（不使用巢狀寫法)

```
promiseSetTimeout(1)
.then((res)=>{
    console.log(res);

    // 使用return 發起第二次非同步
    return promiseSetTimeout(1);
})
// 第二次非同步成功
.then((res) => {
    console.log("第二次非同步成功");
})
```

【ES6】展開

合併陣列

```
const groupA = ['小明', '杰倫', '阿姨'];
const groupB = ['老媽', '老爸'];
const groupC = [...groupA,...groupB]; //'小明', '杰倫', '阿姨', '老媽', '老爸'
```

繼承物件

```
const methods = {
  fn1() {
    console.log(1);
  },
  fn2() {
    console.log(1);
  },
}
const methods2 = {
  ...methods,
  fn3(){
    console.log(1);
  }
}
```

轉換為純陣列

```
// NodeList 或 HTMLCollection 或 Arguments 是「類陣列」的物件
// 無法使用 map 等純物件的方法
const doms = document.querySelectorAll('li');
console.log(doms);
// 轉為純陣列
const newDom = [...doms];
newDom.map((item,i)=>{
  console.log(i);
});
```

【ES6】陣列方法

https://www.youtube.com/watch?v=_vFuDQ_6Xt8

【ES6】箭頭函式

改寫原 functoin

```
// 傳統寫法
function fn(a, b) {
  return a * b;
}
// 箭頭函式
const fn = (a, b) => {
  return a * b;
}
// 省略 return
const fn = (a, b) => {
  a * b;
}
//////////////////////////////
function fn(name) {
  return "Hello"+name;
}
// 單行省略 return , {}
const fn = (name) => "Hello "+name;
```

this 指向不會指向全域，而是指向自己

```
var name = '全域'
const person = {
  name: '小明',
  callName: function () {
    console.log('1', this.name); // 1 小明
    setTimeout(function () {
      console.log('2', this.name); // 2 全域
      console.log('3', this); // 3 window
    }, 10);
  },
}
```

【ES6】賦值解構

取出特定值作為變數

```
// #1 取出特定值作為變數
const person = {
  name: '小明',
  age: 16,
  like: '鍋燒意麵'
}
const {name, age} = person;
console.log(name, age); // 小明, 16
```

陣列解構

```
const arr = ['小明', '杰倫', '漂亮阿姨'];
const [Ming, Jay ] = arr;
console.log(Ming, Jay); // '小明', '杰倫'
```

解構加展開

```
const people = {
  Ming: {
    name: '小明',
    age: 16,
    like: '鍋燒意麵'
  },
  Jay: {
    name: '杰倫',
    age: 18,
    like: '跑車'
  },
  Auntie: {
    name: '漂亮阿姨',
    age: 22,
    like: '名牌包'
  }
}
const { Ming,...other} = people;
console.log(Ming, other );
// Ming
// {
//     name: '小明',
//     age: 16,
//     like: '鍋燒意麵'
// }

// other
// {
//   Jay: {
//     name: '杰倫',
//     age: 18,
//     like: '跑車'
//   },
//   Auntie: {
//     name: '漂亮阿姨',
//     age: 22,
//     like: '名牌包'
//   }
// }
```

解構加上重新命名

```
// 冒號：別名
const { Ming:AAA,...other} = people;
console.log(AAA, other );
```

函式參數解構

```
const person = {name: "Sam", age: 18};

const fn = function({name, age}){
console.log(name,age)
}

fn(person); // Sam 18
```

解構在React應用

```
const { useState } = React;
```

【ES6】async / await

使用方式

```
// #1 當資料有順序性的時候
const fn1 = () =>{
  axios.get("https://randomuser.me/api/")
  .then((res)=> {
    console.log("fn1 seed",res.data.info.seed);
    const seed = res.data.info.seed;
    return axios.get(`https://randomuser.me/api/?seed=${seed}`)
  })
  .then((res)=>{
    console.log("fn1 seed2",res.data.info.seed)
  })
}

// #2 使用 async await 來修改
const fn2 = async ()=> {
  const res = await axios.get("https://randomuser.me/api/")
  const seed = res.data.info.seed;
  console.log("fn2 seed",res.data.info.seed);
  const res2 = await axios.get(`https://randomuser.me/api/?seed=${seed}`);
  const seed2 = res2.data.info.seed;
  console.log("fn2 seed2",res2.data.info.seed);
}


// #3 立即函式的寫法
(async function(){
  const res = await axios.get("https://randomuser.me/api/")
  const seed = res.data.info.seed;
  console.log("fn3 seed",res.data.info.seed);
  const res2 = await axios.get(`https://randomuser.me/api/?seed=${seed}`);
  const seed2 = res2.data.info.seed;
  console.log("fn3 seed2",res2.data.info.seed);
})();
```

錯誤處理

```
const fn4 = async () => {
  try {
    const res = await axios.get("https://randomuser.me/api/")
    return res
  } catch (error) {
    throw error
  }
}

fn4()
.then((res)=>{
  console.log(res);
})
.catch((err)=>{
  console.log(err);
});
```

async function 本身就是 promise

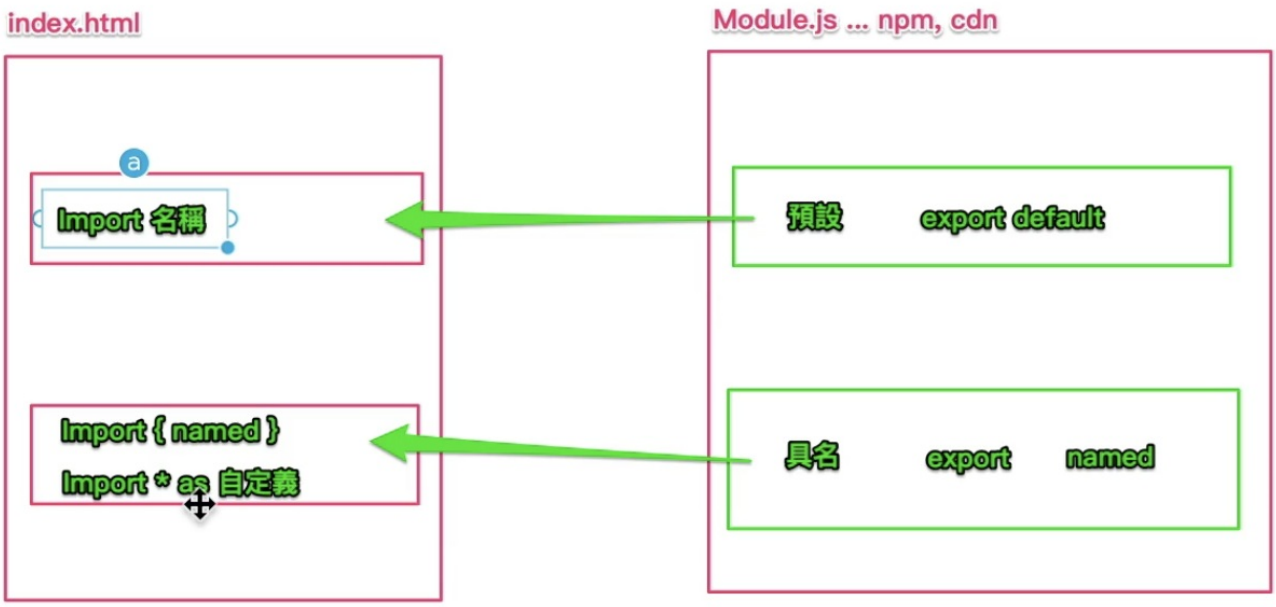
async Function 與 React

```
// async Function 與 React
const App = () => {
  const [data, setData] = React.useState({})

  React.useEffect(() => {
    (async () => {
      const res = await axios.get('https://jsonplaceholder.typicode.com/todos/1');
      console.log(res);
      setData(res.data);
    })();
  }, []);

  return <div>
    {data.title}
  </div>
}
```

【ES6】ESModual



什麼樣的匯出對應什麼樣的匯入

預設匯入(一個匯出檔只能有一個)

常見的匯出方式，通常用於匯出物件，在 React 開發中可用來匯出元件

```
/** 匯出頁 exportDefault.js */
export default function fn(){
 console.log("default export")
}

/** 匯入 index.html */
/** export ${name}(隨便取) from ${path} */
<script type="module">
 export aa from "./exportDefault.js";
 aa();
</script>
```

```
/** 匯出頁 exportDefault.js */
export default {
 name: "sam",
   func: function(){
    console.log("預設方法...")
   }
}

/** 匯入 index.html */
/** import ${name}(隨便取) from ${path} */
<script type="module">
 import aa from "./exportDefault.js";
 console.log(aa.name);
   aa.func();
</script>
```

具名匯入(可多個)

可用於匯出已定義的變數、物件、函式，專案開發中通常用於 “方法匯出”
第三方的框架、函式、套件很常使用具名定義 “方法”

```
/** 匯出頁 exportModule.js */
export const myName = "Jhon"
```

```
export function fn1() {
 console.log("function 1");
}

/** 匯入 index.html */
/** import { 解構名稱 } from ${path} */
<script type="module">
 import {fn1, myName} from "./exportDefault.js";
   console.log("myName",myName);
 fn1();
</script>
```

全部匯出(不常用)

```
/** 匯出頁 exportModule.js */
export const myName = "Jhon"
export function fn1() {
 console.log("function 1");
}

/** 匯入 index.html */
/** import * as ${別名} from ${path} */
<script type="module">
 import * as bb from "./exportDefault.js";
   console.log("myName",bb.myName);
 bb.fn1();
</script>
```

sideEffect (舊函式庫，立即執行)

```
/** 匯出頁 sideEffect.js */
(function(global){
 global.$ = "I am jquery";
})(window)

/** 匯入 index.html */
/** import ${path} */
<script type="module">
 import "./sideEffect.js";
   console.log("$",$); // I am jquery
 bb.fn1();
</script>
```

【ES6】import / export

匯入的類型要看匯出的形式，所以我用匯入方式來說明

使用default 匯出

匯入就可以取任何名稱

```
import fn from './func.js';
fn();
// 匯出檔案 func.js
// export default function (){
//   console.log("fn")
// }
```

具名匯入

匯入必須使用完整名稱，也可以匯入的時候給予別名

```
import {fn2, fn3 as new_fn3, person} from './func.js';
fn2();
new_fn3();
export const person = {name:"sam",age:18}

// 匯出檔案 func.js
// export function fn2(){
//   console.log("fn2")
// }
// export function fn3(){
//   console.log("fn3")
// }
```

全部匯入

```
import * as obj from './func.js';
obj.fn();
// 匯出檔案 func.js
// export function fn(){
//   console.log("fn")
// }
```

傳統lib匯入

ex.Jquery

```
// import file
import './fn.js';

// module file
(function() {
  console.log('IIFE');
})();
```

【JS】相關資源

教學

- 代 JavaScript 教程

【JS】iframe 自適應內容高度

```
<script type="text/javascript">
function SetCwinHeight() {
   var iframeid = document.getElementById("mainframe"); //iframe id
   if (document.getElementById) {
      if (iframeid && !window.opera) {
         if (iframeid.contentDocument && iframeid.contentDocument.body.offsetHeight) {
            iframeid.height = iframeid.contentDocument.body.offsetHeight;
         } else if (iframeid.Document && iframeid.Document.body.scrollHeight) {
            iframeid.height = iframeid.Document.body.scrollHeight;
         }
      }
   }
}
</script>
```

```
<iframe src="./sourcePage.html" name="mainframe" id="mainframe"
   width="100%" marginwidth="0" marginheight="0"
   scrolling="No" frameborder="0"
   onload="Javascript:SetCwinHeight()">
</iframe>
```

【ES6】null 判斷

https://developer.mozilla.org/zh-TW/docs/Web/JavaScript/Reference/Operators/Optional_chaining

可選串連運算子 `?.` 允許進行深層次的物件值存取，而無需透過明確的物件值串連驗證。`?.` 運算子的操作與 `.` 屬性存取運算子相似，後者會在參照到 nullish (`null` or `undefined`) 的值時出現錯誤，而前者可選串連則回傳 `undefined` 。 當需要存取一個函數，而這函數並不存在時，則會回傳 `undefined`

```
const adventurer = {
  name: 'Alice',
  cat: {
    name: 'Dinah',
  },
};

const dogName = adventurer.dog?.name;
console.log(dogName);
// Expected output: undefined

console.log(adventurer.someNonExistentMethod?.());
// Expected output: undefined
```

當一個物件 `obj` 是巢狀結構時，在沒有可選串連之下，要去查找一深層的屬性值需要驗證每層間的參照連結：

```
let nestedProp = obj.first && obj.first.second && obj.first.second.third;
```

現在可以用?.做串接判斷

```
let nestedProp = obj.first?.second?.third
```

在用 ?? 賦予預設值避免 undefined

```
let nestedProp = obj.first?.second?.third ?? "default"
```

【Tool】常用方法

*判斷為空

```
const isEmpty = (val) => {
    // null or undefined
    if (val == null) return true;

    if (typeof val === "boolean") return false;

    if (typeof val === "number") return !val;

    if (val instanceof Error) return val.message === "";

    // eslint-disable-next-line default-case
    switch (Object.prototype.toString.call(val)) {
        // String or Array
        case "[object String]":
        case "[object Array]":
            return !val.length;

        // Map or Set or File
        case "[object File]":
        case "[object Map]":
        case "[object Set]": {
            return !val.size;
        }
        // Plain Object
        case "[object Object]": {
            return !Object.keys(val).length;
        }
    }

    return false;
};

const isNotEmpty = (val) => {
    return !isEmpty(val);
};
```