



Web Development Fundamentals

网站开发系列 - JavaScript 基础

Aug 2020

Microsoft Reactor | Ryan Chung

```

    self.load_image("kg.png")

    def __init__(self):
        # Initialize Dog object and create Text object
        self.image = Dog.image
        self.x = games.mouse.x
        self.bottom = games.screen.bottom

        self.score = games.Text(value = 0, size = 24,
                                top = 5, right = games.screen.right)
        self.screen.add(self.score)

        self.lives = games.Text(value = 0, size = 24,
                                top = 5, left = games.screen.left)
        self.screen.add(self.lives)

```



Ryan Chung

Instructor / DevelopIntelligence
Founder / MobileDev.TW

@ryanchung403 on WeChat
Ryan@MobileDev.TW



Web On-line Workshop agenda 网页开发在线研讨会议程

Intro to JavaScript 网页开发入门 – JavaScript

19:30	Welcome 开场
19:35	Introduction to JavaScript JavaScript 基本介绍
19:55	Event Trigger and Handling 事件触发与处理
20:15	Object-Oriented and DOM 面向对象与DOM模型探索
20:30	5-minute break 中场休息
20:35	Lab Practice 网站综合练习
21:00	Event end 研讨会结束



Reactor



developer.microsoft.com/reactor/
[@MSFTReactor](#) on Twitter

JavaScript 简介

概要

- 语法简介
- 浏览器Console输出
- 函数
- 物件

JavaScript 简介

- 小道消息
 - JavaScript语言是由Brendan Eich在Netscape工作时所设计出来的，听说他只花了11天的时间就完成。
- 微软于很早就开始支持JS!
 - IE3版本(1996年)
- 分工角色
 - HTML 画面组件结构
 - CSS 样式、排版
 - JavaScript 互动、程序逻辑



开发环境

The image shows the Visual Studio Code website on the left and a screenshot of the Visual Studio Code application interface on the right.

Visual Studio Code Website:

- Header: Visual Studio Code, Docs, Updates, Blog, API, Extensions, FAQ, Search Docs, Download.
- Message: Version 1.42 is now available! Read about the new features and fixes from January.
- Section: Code editing. Redefined.
- Text: Free. Built on open source. Runs everywhere.
- Buttons: Download for Windows (Stable Build), Other platforms and Insiders Edition.
- Text: By using VS Code, you agree to its license and privacy statement.

Visual Studio Code Application Interface:

- Extensions: Marketplace (@sort:installs)
- Extensions List:
 - Python 2019.6.24221 (4.9M, 4.5 stars) - Install
 - GitLens — Git superpower (9.8.5, 23.1M, 5 stars) - Install
 - C/C++ 0.24.0 (23M, 3.5 stars) - Install
 - ESLint 1.9.0 (21.9M, 4.5 stars) - Install
 - Debugger for Chrome 4.11.6 (20.6M, 4 stars) - Install
 - Language Support for Java 0.47.0 (18.7M, 4.5 stars) - Install
 - vscode-icons 8.8.0 (17.2M, 5 stars) - Install
 - Vetur 0.21.1 (17M, 4.5 stars) - Install
 - C# 1.21.0 (15.6M, 4 stars) - Install
- Editor: JS serviceWorker.js - create-react-app - Visual Studio Code - In...
 - Code:

```
src > JS serviceWorker.js > register > window.addEventListener('load') callback
checkValidServiceWorker(swUrl, config);

// Add some additional logging to localhost, pointing toward the
// service worker/PWA documentation.
navigator.serviceWorker.ready.then(() => {
  product
  productSub
  removeSiteSpecificTrackingException
  removeWebWideTrackingException
  requestMediaKeySystemAccess
  sendBeacon
  serviceWorker (property) Navigator.serviceWorker...
  storage
  storeSiteSpecificTrackingException
  storeWebWideTrackingException
  userAgent
  vendor
})

function registerValidSW(swUrl, config) {
  navigator.serviceWorker
    .register(swUrl)
    .then(registration => {
```
 - Terminal: 1: node

```
You can now view create-react-app in the browser.

Local:      http://localhost:3000/
On Your Network: http://10.211.55.3:3000/

Note that the development build is not optimized.
```

<https://code.visualstudio.com/>

Microsoft

Reactor

安装扩充套件

- 按下左边 Extensions图示 或 Ctrl + Shift + X
 - Chinese (Simplified) Language Pack for Visual Studio Code
 - Live Server
- 设定Ctrl+鼠标滚轴控制编辑器字号
 - editor.mouseWheelZoom
- 设定编辑时自动储存
 - 档案 -> 自动储存



Chinese (Simplified) Language Pack for Visual Studio Code

Microsoft | 4,835,250 | ★★★★★ | 儲存庫

Language pack extension for Chinese (Simplified)

安裝



Live Server ritwickdey.liveserver

Ritwick Dey | 5,128,585 | ★★★★★ | 儲存庫 | 授權

Launch a development local Server with live reload feature for static & dynamic pages

停用▼

解除安裝

This extension is enabled globally.

≡ 設定 ×

editor.mouseWheelZoom

使用者 工作區

▼ 文字編輯器 (1)

Editor: Mouse Wheel Zoom



使用滑鼠滾輪並按住 **Ctrl** 時，縮放編輯器的字型

Microsoft

Reactor

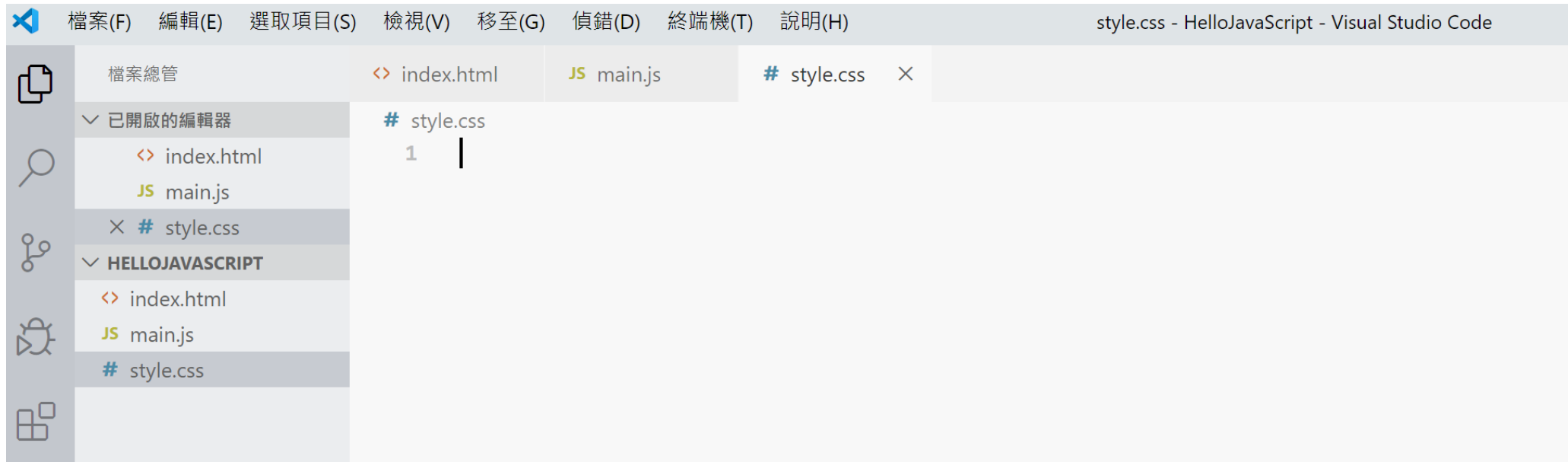
第一个 JS 专案

- 在左边区块点击右键，新增档案，命名为index.html



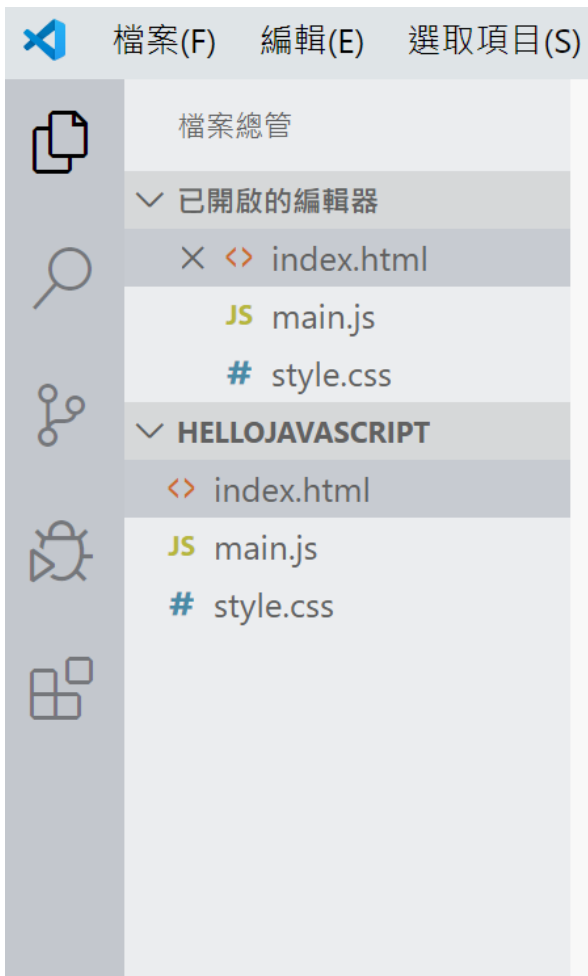
第一个 JS 专案

- 在左边点击右键，新增档案，再增加两个档案
 - main.js
 - style.css



第一个 JS 专案

- 回到index.html，编辑档案如下

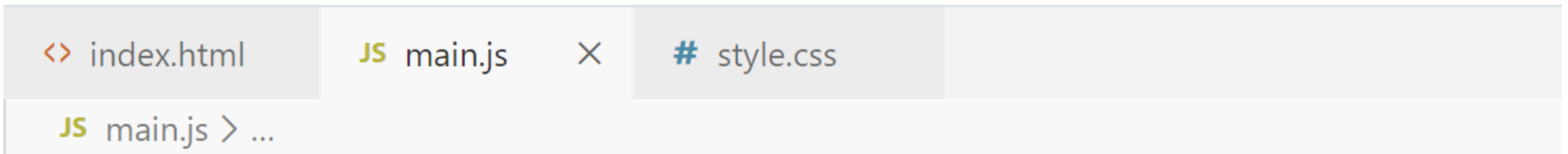


```
<!DOCTYPE html>
<head>
  <meta charset="utf-8">
  <title></title>
  <link rel="stylesheet" href="style.css">
  <script src="main.js"></script>
</head>
<body>

</body>
</html>
```

第一个 JS 专案

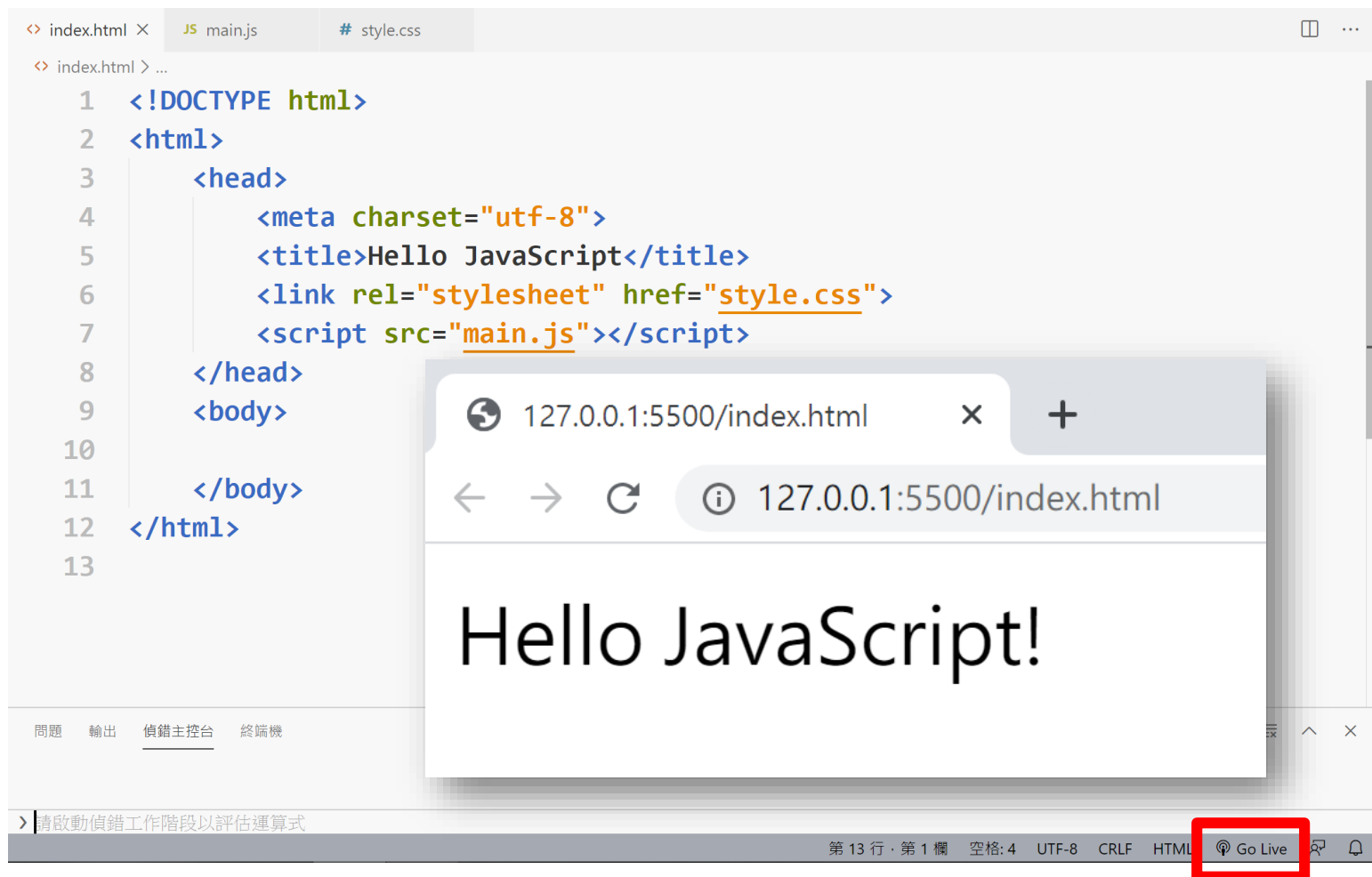
- 再到main.js，编辑档案如下



```
window.onload = function(){  
    document.write("Hello JavaScript!");  
};
```


测试执行

- 按下右下方的Go Live
- 或在左边档案区 index.html右键单击 Open with Live Server



什么是JavaScript

- 增加HTML页面上的互动性
- 是一种scripting language
- 功用
 - 增加网页上的动态效果
 - 可以根据特定事件对应执行动作
 - 可以读取与改写HTML组件
 - 可以拿来进行窗体验证
 - 侦测使用者的浏览器
 - 建立与存取cookie

摆放位置

1. 放在<body>..</body>

- 一般要直接执行的JavaScript会放在</body>前面
- 先让整个页面的HTML组件加载

2. <head>...</head>

- 特定事件触发才做的函数内容会放在<head>section

3. 独立成一个档案

<head> <script src="xxx.js"> </script> </head>

```
<script type="text/javascript">
```

```
//就写在这里面
```

```
document.write("<p>" + Date() + "</p>");
```

```
</script>
```

陈述句 / 批注方式

- 顺序执行
- 大小写有区分
- 通常以分号结尾

- 单行 //

- 多行

/* Comments more than
one line

*/

变数

- 变量名称
 - 大小写有区分
 - 第一个字必须是英文字母或底线
- 变量宣告
 - var (逐渐减少使用)
 - let **NEW** (建议使用)
 - const **NEW** (建议使用)

变量范围 Variables Scope

- 全局 Global Scope
 - 整个网页程序执行都可以使用
- 函数内 Function Scope
 - 只在函数范围内可以使用
- 区块内 Block Scope **NEW**(ECMAScript 2015后)
 - 只在区块范围内可以使用
 - var不支援

跳出窗口

- Alert box
 - `alert("sometext");`
- Confirm box
 - `confirm("sometext");`
 - OK : true or Cancel : false
- Prompt box
 - `prompt("sometext","defaultvalue");`
 - 取得使用者输入的值进行动作

练习：点击文字取得字符数

- HTML
- CSS
- JavaScript

127.0.0.1:5500 顯示

JavaScript有10個字元

確定

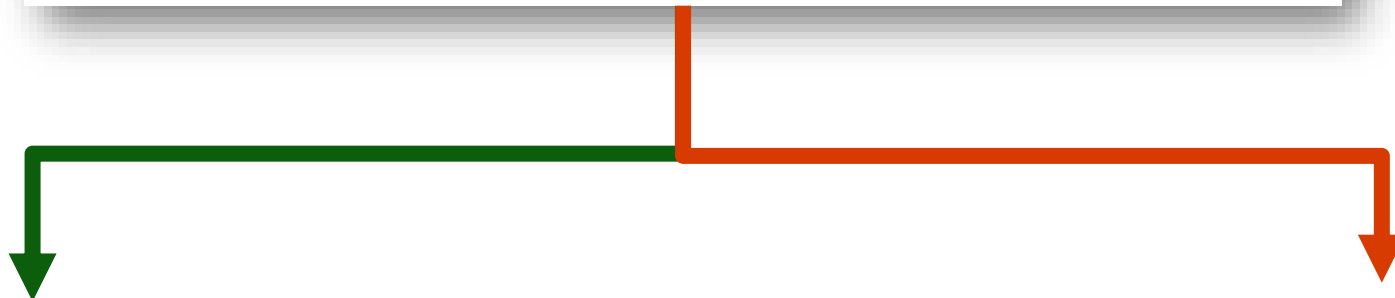
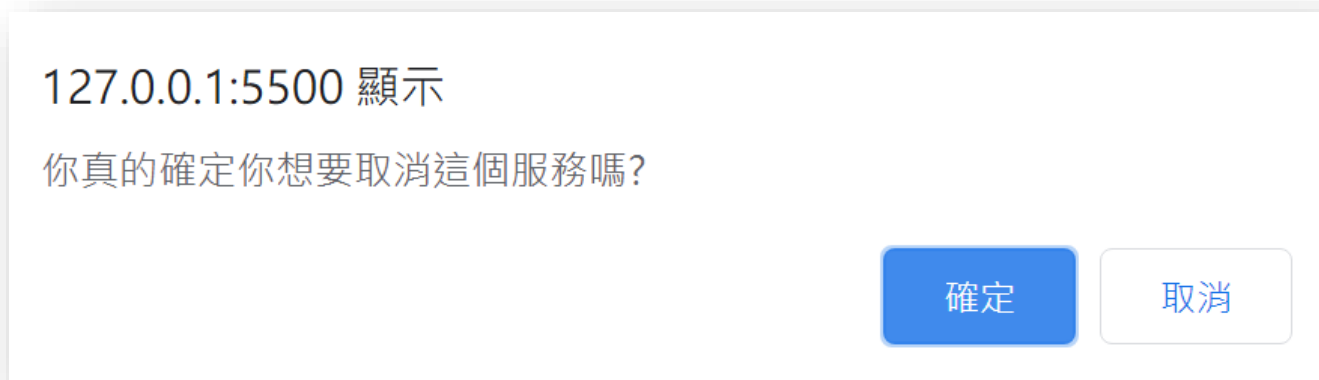
index.html

```
1  <!DOCTYPE html>
2  <html>
3      <head>
4          <meta charset="utf-8">
5          <title></title>
6          <link rel="stylesheet" href="style.css">
7          <script src="main.js"></script>
8      </head>
9      <body>
10         <ul>
11             <li>HTML</li>
12             <li>CSS</li>
13             <li>JavaScript</li>
14         </ul>
15     </body>
16 </html>
```

main.js

```
1 window.onload = function(){
2     document.onclick = function(e){
3         alert(e.target.innerHTML + "有" +
4             e.target.innerHTML.length+"個字元");
5     };
6 };
```


Confirm Box 确认



服務已取消

繼續使用本服務

index.html

```
<!DOCTYPE html>
  <head>
    <meta charset="utf-8">
    <title></title>
    <link rel="stylesheet" href="style.css">
  </head>
  <body>
    <h1>XXXXX</h1>
    <!--
    <ul>
      <li>HTML</li>
      <li>CSS</li>
      <li>JavaScript</li>
    </ul>
    -->
    <script async defer src="main.js"></script>
  </body>
</html>
```

main.js

```
let confirmAnswer = confirm("你真的确定你想要取消这个服务吗?");
let thisH1 = document.getElementsByTagName("h1")[0];
if(confirmAnswer){
    thisH1.innerHTML = "服务已取消";
}else{
    thisH1.innerHTML = "继续使用本服务";
}
```

Prompt Box

127.0.0.1:5500 顯示

小明家裡有三個小孩，他兩個哥哥叫張一、張二、第三個小孩叫什麼？

張三

確定

取消

张三

小明

李四

那小明是誰？

聰明

你想多了

index.html

```
<!DOCTYPE html>
  <head>
    <meta charset="utf-8">
    <title></title>
    <link rel="stylesheet" href="style.css">
  </head>
  <body>
    <h1 id="Response">XXXXX</h1>
    <script async defer src="main.js"></script>
  </body>
</html>
```


main.js

```
let promptAnswer = prompt("小明家里有三个小孩，他两个哥哥叫张一、张二、第三个小孩叫什么？","张三");
```

```
let thisH1 = document.getElementsByTagName("h1")[0];
```

```
switch(promptAnswer){  
  case "张三":  
    thisH1.innerHTML="那小明是谁?";  
    break;  
  case "小明":  
    thisH1.innerHTML="聪明";  
    break;  
  default:  
    thisH1.innerHTML="你想多了";  
}
```

函数

- 执行时机
 - 直接被呼叫
 - 事件发生时触发
 - Callback
- 语法

```
function functionname(var1, var2, ..., varX)  
{  
  some code  
}
```

函数的呼叫 – index.html

```
<!DOCTYPE html>
  <head>
    <meta charset="utf-8">
    <title></title>
    <link rel="stylesheet" href="style.css">
  </head>
  <body>
    <h1 id="Response">XXXXXX</h1>
    <button>Click Me</button>
    <script async defer src="main.js"></script>
  </body>
</html>
```

函数的呼叫 – main.js

```
function showAlert(){  
    thisH1.innerHTML = "Hello!";  
}
```

```
let thisButton = document.getElementsByTagName("Button")[0];  
let thisH1 = document.getElementsByTagName("h1")[0];  
thisButton.onclick = function(){  
    showAlert();  
};
```



Break and Continue in For loop

- break : 离开循环
- continue : 跳过这一圈

```
0 for(let i=0;i<10;i++){
1   if(i==3){
2     break;
3   }
4   console.log(i);
5 }
6
7
8
9
>
```

```
0 for(let i=0;i<10;i++){
1   if(i==3){
2     continue;
3   }
4   console.log(i);
5 }
6
7
8
9
>
```

```
0
1
2
4
5
6
7
8
9
>
```

For...in

- 巡访对象中的每一个属性与方法

```
let person = {  
  firstName: "Ryan",  
  lastName: "Chung",  
  height: 178  
};  
  
for(x in person){  
  console.log(person[x]);  
}
```

Ryan

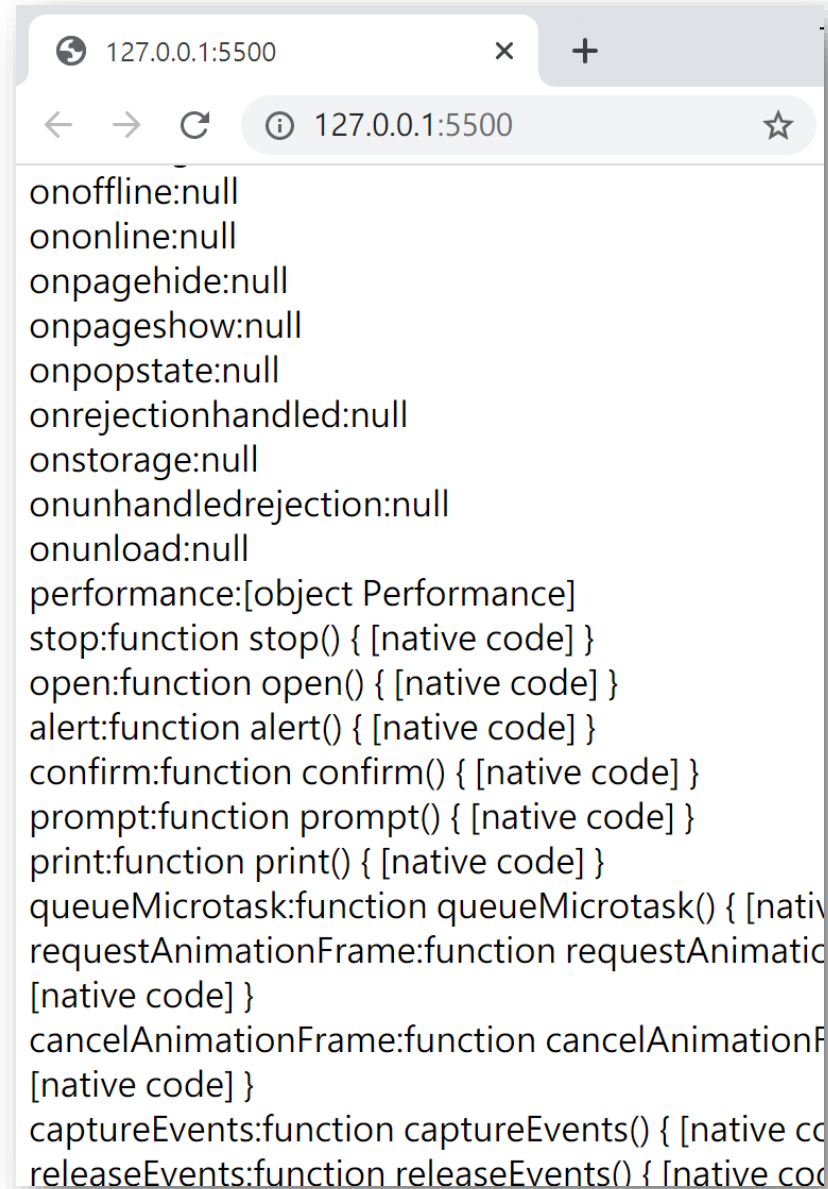
Chung

178



For...in Lab

- 巡访window物件



A screenshot of a web browser window with the address bar showing '127.0.0.1:5500'. The browser's developer console or a text area displays the contents of the `window` object, listing various properties and methods. The list includes: `onoffline:null`, `ononline:null`, `onpagehide:null`, `onpageshow:null`, `onpopstate:null`, `onrejectionhandled:null`, `onstorage:null`, `onunhandledrejection:null`, `onunload:null`, `performance:[object Performance]`, `stop:function stop() { [native code] }`, `open:function open() { [native code] }`, `alert:function alert() { [native code] }`, `confirm:function confirm() { [native code] }`, `prompt:function prompt() { [native code] }`, `print:function print() { [native code] }`, `queueMicrotask:function queueMicrotask() { [native code] }`, `requestAnimationFrame:function requestAnimationFrame() { [native code] }`, `cancelAnimationFrame:function cancelAnimationFrame() { [native code] }`, `captureEvents:function captureEvents() { [native code] }`, and `releaseEvents:function releaseEvents() { [native code] }`.

```
onoffline:null
ononline:null
onpagehide:null
onpageshow:null
onpopstate:null
onrejectionhandled:null
onstorage:null
onunhandledrejection:null
onunload:null
performance:[object Performance]
stop:function stop() { [native code] }
open:function open() { [native code] }
alert:function alert() { [native code] }
confirm:function confirm() { [native code] }
prompt:function prompt() { [native code] }
print:function print() { [native code] }
queueMicrotask:function queueMicrotask() { [native code] }
requestAnimationFrame:function requestAnimationFrame() { [native code] }
cancelAnimationFrame:function cancelAnimationFrame() { [native code] }
captureEvents:function captureEvents() { [native code] }
releaseEvents:function releaseEvents() { [native code] }
```

事件触发方法与常见事件

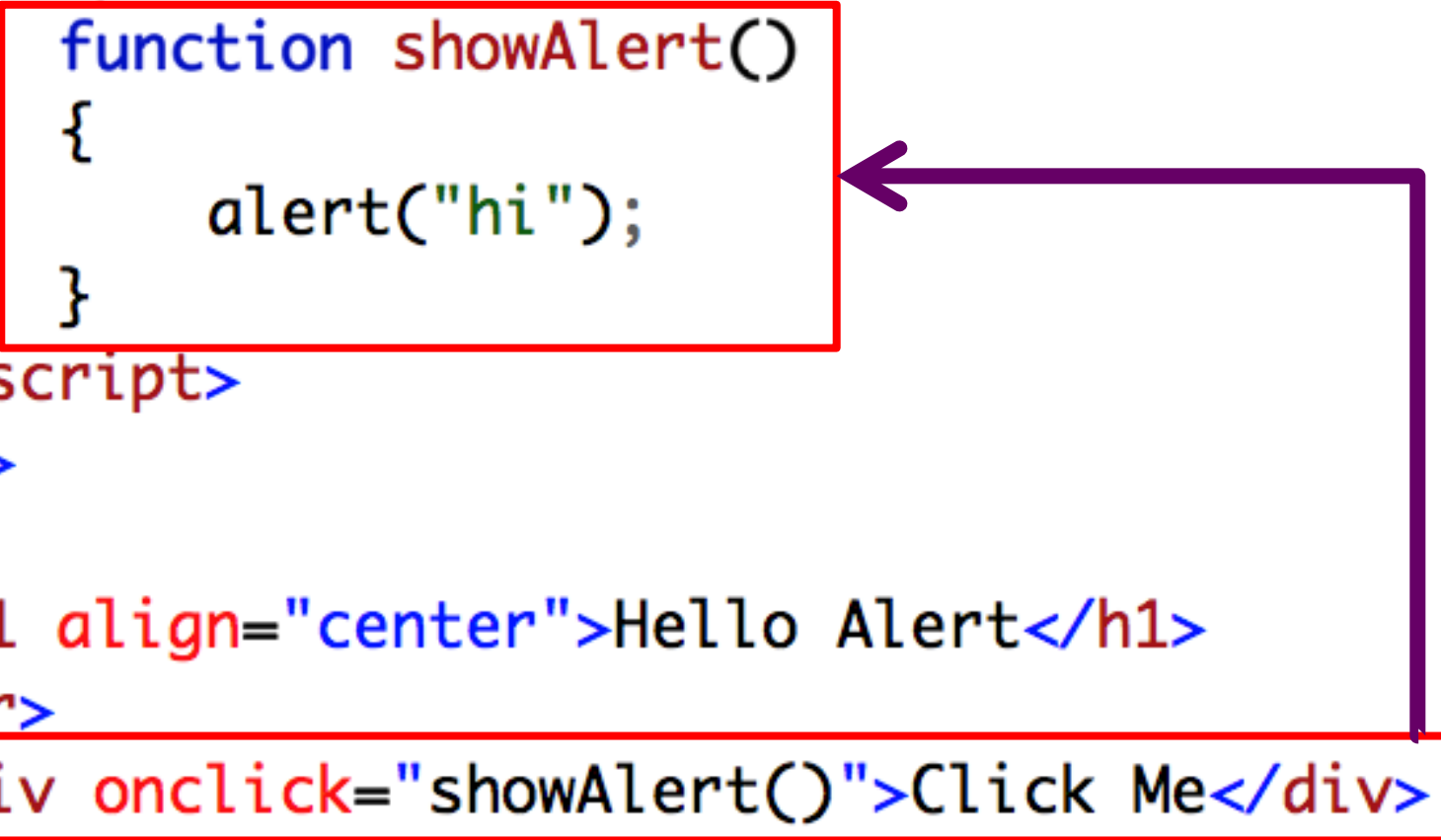
- 事件是JavaScript可以侦测得到的一种动作
- HTML组件上的动作会触发JavaScript的事件
 - 按下按钮组件 --> onClick事件
 - 网页页面载入 --> onLoad事件
 - 鼠标光标在某个组件上 --> onMouseOver事件

发生什么事？要做什么因应措施？

事件触发处理方法1.HTML Attribute

- 直接加入在HTML的属性中

```
20<script>
21  function showAlert()
22  {
23      alert("hi");
24  }
25</script>
26</head>
27<body>
28  <h1 align="center">Hello Alert</h1>
29  <hr>
30  <div onclick="showAlert()">Click Me</div>
31</body>
```

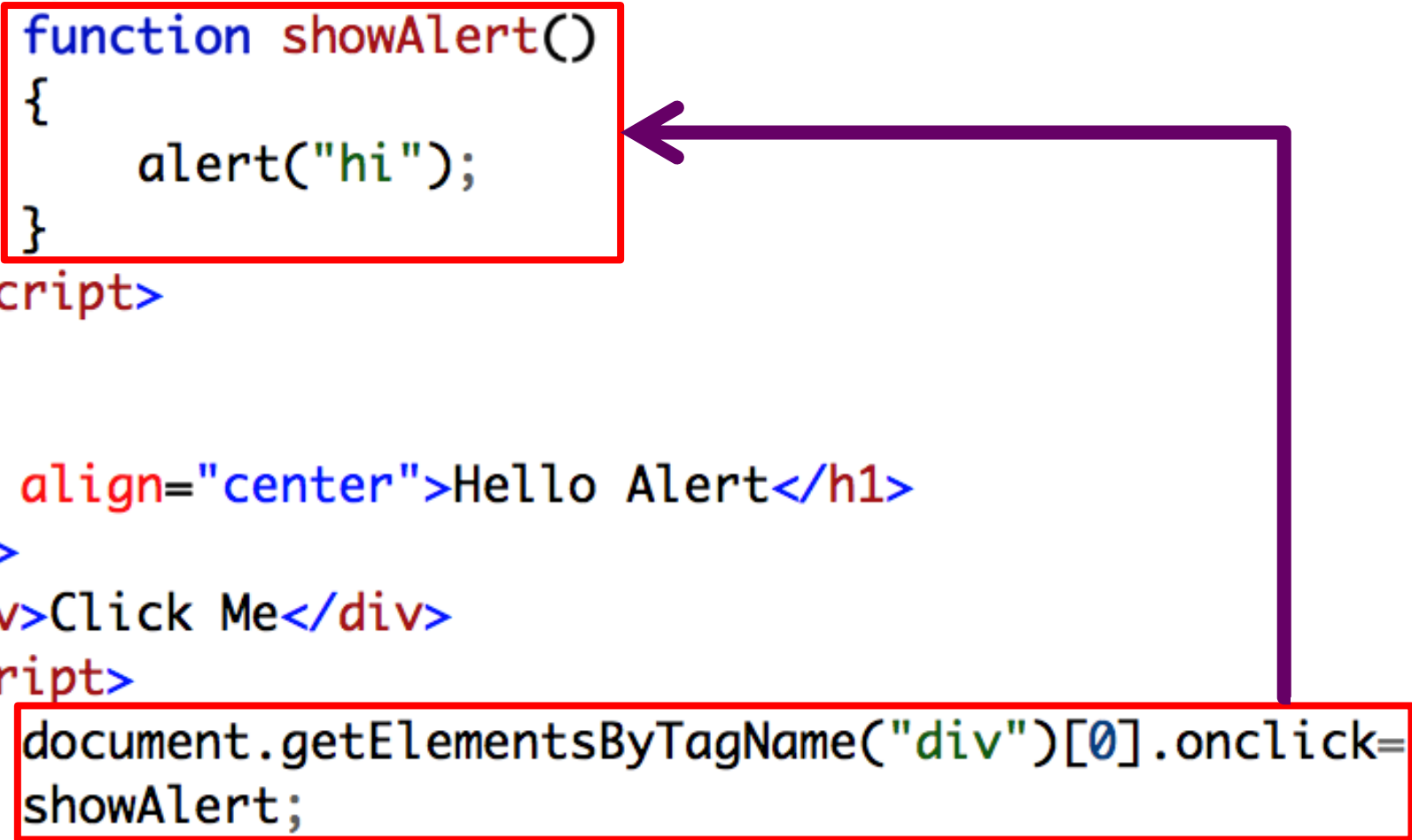


A diagram consisting of a red rectangular box around the JavaScript function `showAlert()` in the script section and another red rectangular box around the `onclick="showAlert()"` attribute in the HTML `<div>` tag. A purple arrow points from the `showAlert()` call in the HTML attribute to the `showAlert()` function definition in the script block.

事件触发处理方法2.Event Function

- 使用JavaScript语法，定义onclick要执行的动作

```
20 <script>
21   function showAlert()
22   {
23     alert("hi");
24   }
25 </script>
26 </head>
27 <body>
28   <h1 align="center">Hello Alert</h1>
29   <hr>
30   <div>Click Me</div>
31   <script>
32     document.getElementsByTagName("div")[0].onclick=
33     showAlert;
34   </script>
```

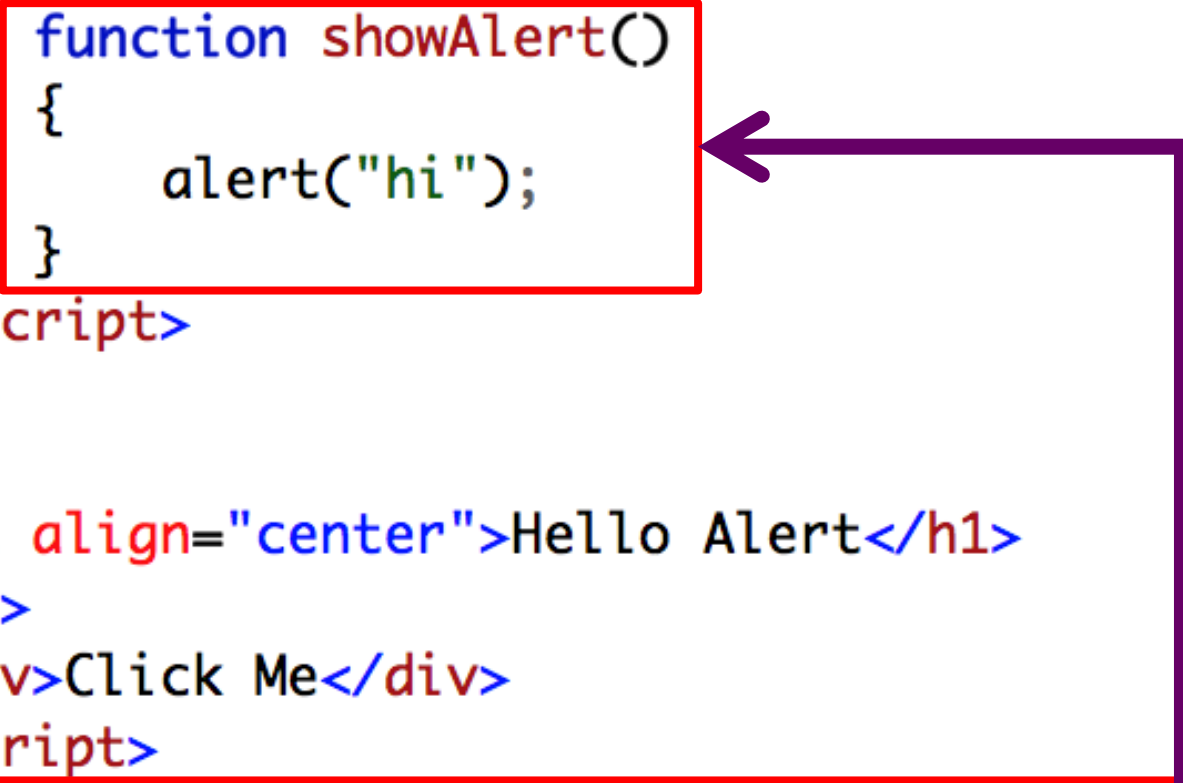


A diagram consisting of two red rectangular boxes. The first box, located between lines 21 and 24, contains the JavaScript function definition: `function showAlert() { alert("hi"); }`. The second box, located between lines 32 and 33, contains the event assignment: `document.getElementsByTagName("div")[0].onclick= showAlert;`. A purple arrow originates from the right side of the second box and points horizontally to the left, ending at the function definition in the first box, indicating that the function is being referenced by the event handler.

事件触发处理方法3.Event Listener

- 监听该组件，是否有XX事件发生

```
20 <script>
21   function showAlert()
22   {
23     alert("hi");
24   }
25 </script>
26 </head>
27 <body>
28   <h1 align="center">Hello Alert</h1>
29   <hr>
30   <div>Click Me</div>
31 <script>
32   document.getElementsByTagName("div")[0].
33   addEventListener("click",showAlert);
34 </script>
```



A diagram consisting of a red rectangular box around the `function showAlert()` block in the script section (lines 21-24) and another red rectangular box around the `addEventListener` call in the body script section (lines 32-33). A purple arrow originates from the right side of the first box and points to the `showAlert` parameter in the `addEventListener` call of the second box, illustrating how the function is passed as a callback to the event listener.

常见鼠标/键盘事件

事件	描述
onclick	点击
ondblclick	双击
onmousedown	在该组件上按下鼠标键
onmousemove	在该组件上移动鼠标
onmouseover	鼠标进入该组件范围
onmouseout	鼠标离开该组件范围
onmouseup	在该组件上释放鼠标键

事件	描述
onkeydown	实体键盘被按下
onkeypress	有字符被输入
onkeyup	实体键盘被放开

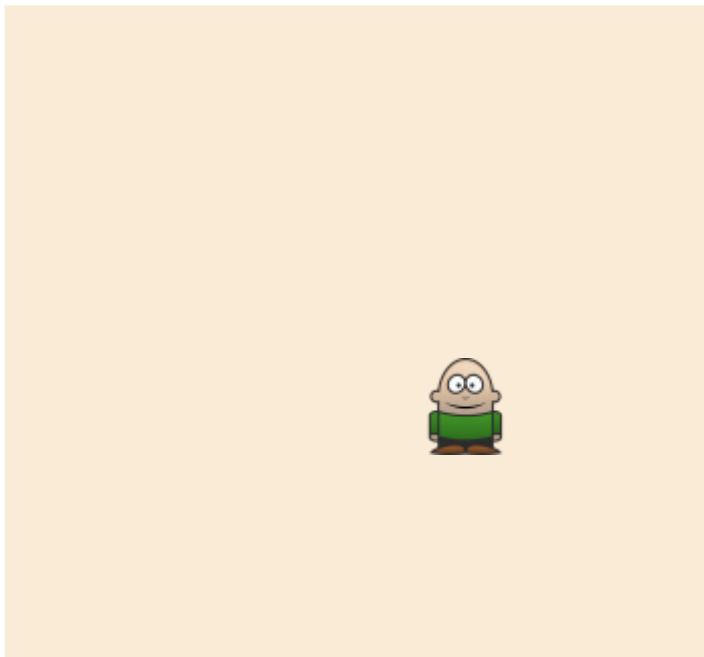
框架/对象事件

事件	描述
onload	文件、对象被加载
onscroll	文件滚动
onabort	取消载入
onerror	加载错误
onresize	大小被改变
onunload	离开该页面、进入新页面、重新整理

窗体事件

事件	描述
onblur	用户离开聚焦在该项目
onchange	内容被改变(选项勾选、下拉选单...)
onfocus	用户聚焦在该项目
onreset	恢复默认资料
onselect	选取了一些文字
onsubmit	送出窗体

练习：小人物回家



你進來了

你在裡面走來走去



index.html

```
<!DOCTYPE html>
  <head>
    <meta charset="utf-8">
    <title></title>
    <link rel="stylesheet" href="style.css">
  </head>
  <body>
    <div></div>
    <h1></h1>
    <p></p>
    <script async defer src="main.js"></script>
  </body>
</html>
```


style.css

```
div{  
    width: 30%;  
    height: 200px;  
    background-color: antiquewhite;  
    cursor: url(man.png), auto;  
}  
p{  
    height: 30px;  
    font-size: large;  
}
```

main.js

```
let thisH1 = document.getElementsByTagName("h1")[0];
let thisP = document.getElementsByTagName("p")[0];
let thisDiv = document.getElementsByTagName("div")[0];

function mouseIn(){
    thisH1.innerHTML="你进来了";
}

function mouseOut(){
    thisH1.innerHTML="你出去了";
    thisP.innerHTML="";
}

function mouseMove(e){
    thisP.innerHTML="你在里面走来走去。位置 : "+e.clientX+", "+e.clientY;
}

thisDiv.addEventListener("mouseover", mouseIn);
thisDiv.addEventListener("mouseout", mouseOut);
thisDiv.addEventListener("mousemove", mouseMove);
```

String 字符串对象

- 属性
 - `text.length` : 取得字符串长度
- 方法
 - `text.big()` : 将字符串字体放大
 - `text.charAt(x)` : 回传x位置的字符
 - `text.concat(string2,string3,...)` : 串接字符串
 - `text.indexOf(string)` : 回传第一个出现该字符串的位置
 - `text.lastIndexOf(string)` : 回传最后一个出现该字符串的位置
 - `text.replace("subString","newString")` : 字符串替换
 - `text.slice(begin, end)` : 取出部份字符串 (可输入一个负值从后面取x字符)
 - `text.split(separator)` : 依特定符号进行切割，并放入数组
 - `text.substr(begin,length)` : 取出部份字符串
 - `text.substring(from,to)` : 取出部份字符串
 - `text.toLowerCase()` : 全部转小写

String Lab

- 字符串 Hello World!
 - 秀出字符串长度
 - 找到World的位置
 - 依空格切割，秀出Hello与World!

index.html

```
1  <!DOCTYPE html>
2  <html>
3      <head>
4          <meta charset="utf-8">
5          <title></title>
6      </head>
7      <body>
8          <h1>Hello World!</h1>
9          <script async defer src="main.js"></script>
10     </body>
11 </html>
```

main.js

```
1  let thisH1 = document.getElementsByTagName("h1")[0];
2  thisH1.addEventListener("click", showAlert);
3
4  function showAlert(){
5      alert("字串長度 : "+thisH1.innerHTML.length+"\n"+
6          "World在第"+thisH1.innerHTML.indexOf("World")+"位置"+"\\n"+
7          "第一個字"+thisH1.innerHTML.split(" ")[0]+"\\n"+
8          "第二個字"+thisH1.innerHTML.split(" ")[1]);
9  }
```

Date 日期时间对象

- 取得目前时间
 - `var d=new Date();`
 - `d.getFullYear()` 目前年
 - `d.getMonth()` 目前月(0~11)
 - `d.getDate()` 目前日(1~31)
 - `d.getDay()` 目前周(0~6)
 - `d.getHours()` 目前时(0~23)
 - `d.getMinutes()` 目前分(0~59)
 - `d.getSeconds()` 目前秒(0~59)

取得目前日期

`d.toLocaleDateString()`

取得目前时间

`d.toLocaleTimeString()`

取得目前日期加时间

`d.toLocaleString()`

数组对象

- 建立数组

```
let myFriends = new Array();  
myFriends[0]="John";  
myFriends[1]="Mary";  
myFriends[2]="David";
```

```
let myFriends =  
new Array("John", "Mary", "David");
```

```
let myFriends = ["John", "Mary", "David"];
```

- 存取数组

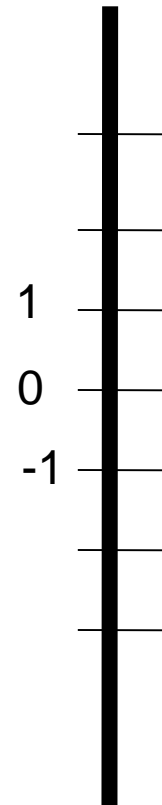
Array 数组对象

- 将数组的各元素以特定的链接符号组合成字符串
 - `arrayName.join(separator)`

```
//陣列的三種建立方式
/*
let myFriends = new Array();
myFriends[0]="John";
myFriends[1]="Mary";
myFriends[2]="David";
*/
//let myFriends = new Array("John","Mary","David");
let myFriends = ["John","Mary","David"];
//console.log(myFriends[1]);
//把陣列的各元素組合成一個字串
console.log(myFriends.join(" and "));
```

Math 数学对象

- `Math.PI` : PI值
- `Math.abs(number)` : 取绝对值
- `Math.floor(number)` : 向下整数
- `Math.ceil(number)` : 向上整数
- `Math.round(number)` : 取整数(四舍五入)
- `Math.max(number,number,number,...)` : 最大值
- `Math.random()` : 产生随机数 (介于0 ~ 1之间)



window Object

- 在浏览器中开启一个窗口即建立一个窗口对象
- 属性
 - `window.closed` : 该窗口关闭即为true
 - `window.name` : 该窗口名称
 - `history` (物件) : 记录下用户在该窗口所去过的网址
 - `navigator` (物件) : 浏览器相关信息
 - `document` (物件) : 当该窗口加载一份HTML文件时即产生

window Object

- 方法

- 跳出窗口

- alert()
 - confirm()
 - prompt()

- 定时器

- setInterval()
 - setTimeout()
 - clearInterval()
 - clearTimeout()

- 开新窗口

- window.open(URL, name, specs)
 - URL : 显示网页
 - name : 窗口名称
 - specs : 规格
 - height
 - width
 - left
 - top

document Object

- 属性

- domain : 传回目前文件所在的域名
- title : 传回目前文件定义的title
- URL : 传回目前文件的完整网址路径
- cookie : 传回目前文件的cookie信息

- 方法

- getElementById() : 存取第一个id名称相符的组件
- getElementsByName() : 存取所有name相符的组件
- getElementsByTagName() : 存取所有该卷标名称的组件
- write() : 写入文件
- writeln() : 写入文件并带上换行符号

document Object

- Collections 筛选
 - anchors[] 找到所有页面上的anchor
 - forms[] 找到所有页面上的form
 - images[] 找到所有页面上的image
 - links[] 找到所有页面上的link

Document Object Model

- W3C标准
- 用来存取HTML或XML文件
- Core DOM
 - 任何结构化文件的标准模型
- XML DOM
 - XML文件的标准模型
- HTML DOM
 - HTML文件的标准模型

HTML DOM

- HTML的标准对象模型
- HTML的标准程序界面
- 可于各种平台、语言使用
- 用来取得、改变、新增或删除HTML组件

Node 节点

- HTML文件中，所有的事物都是一个节点
- 整个文件：文件节点
- HTML组件：组件节点
- HTML组件中的文字：文字节点
- HTML中的属性：属性节点
- HTML中的批注：批注节点

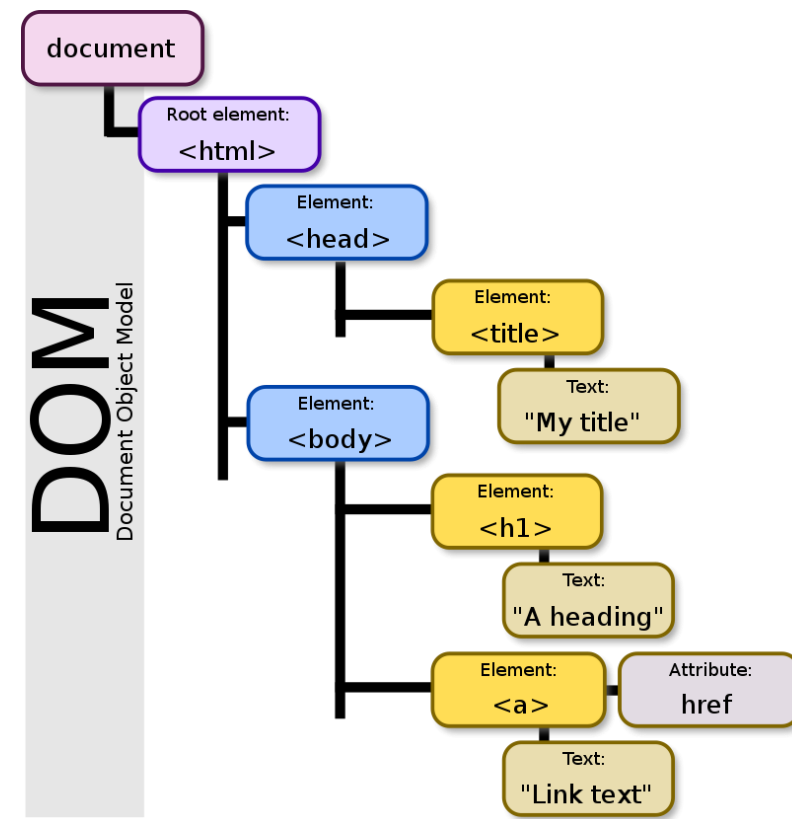
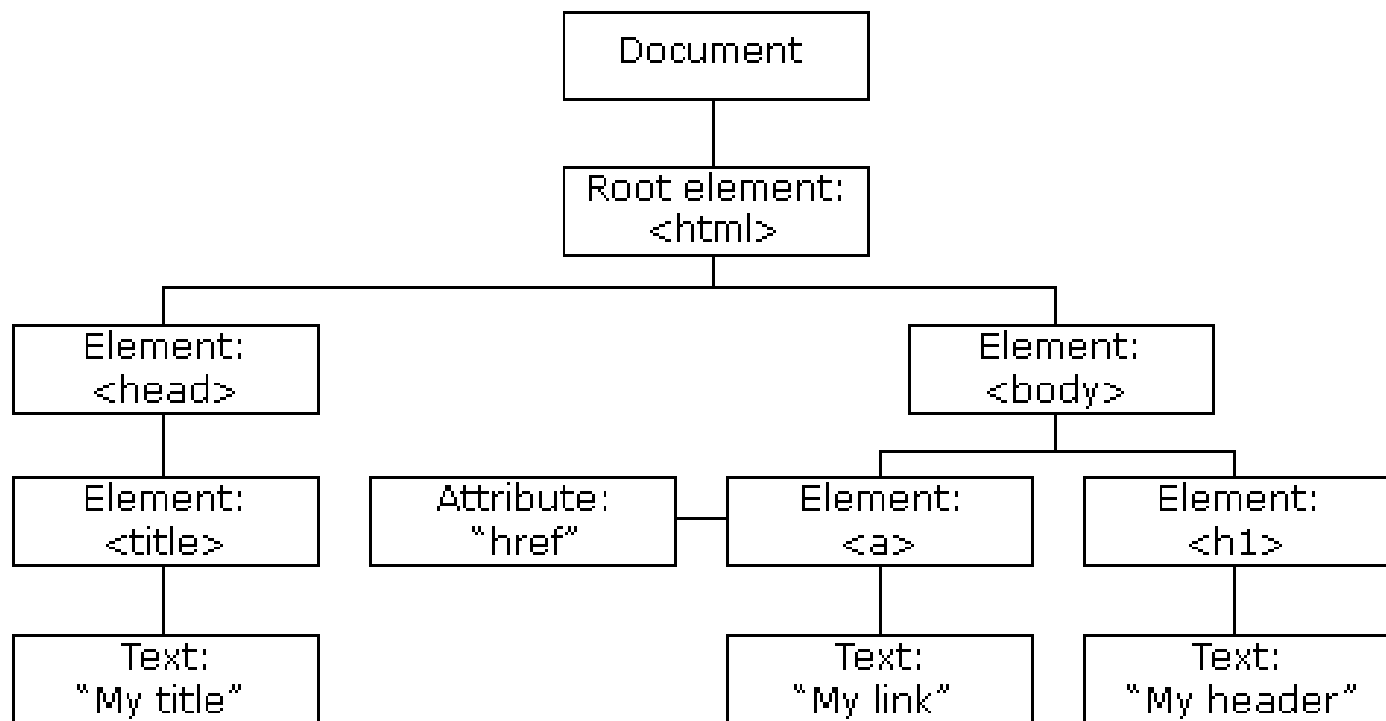
节点分析

- 根节点：`<html>`
- `<html>`节点有两个子节点：`<head>`与`<body>`
- `<title>`节点有一个text子节点：`DOM Tutorial`

```
<html>
  <head>
    <title>DOM Tutorial</title>
  </head>
  <body>
    <h1>DOM Lesson one</h1>
    <p>Hello world!</p>
  </body>
</html>
```

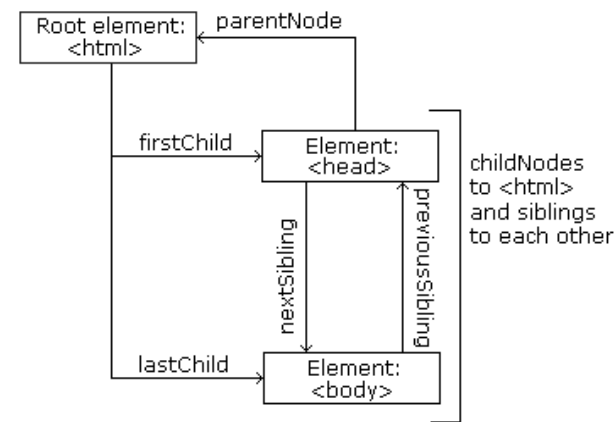
树状结构

- HTML DOM把HTML文件当成一颗节点树



树状结构：父母、小孩与兄弟姊妹

- 最上层的节点称之为根 (root)
- 除了根节点外，每一个节点都有一个父节点
- 一个节点可以有任意数量的子节点
- 没有子节点的节点称之为叶 (leaf)
- 有相同父节点的节点称之为兄弟(sibling)



HTML DOM属性

- innerHTML : 文字值
- nodeName : 名称
- nodeValue : 值
- parentNode : 父节点
- firstChild : 第一个子节点
- lastChild : 最后一个子节点
- nextSibling : 紧邻的兄弟节点
- nodeType : 组件型态

HTML DOM属性

- nodeName

- 组件节点的节点名称即为标签名称(会变成大写)
- 属性节点的节点名称即为属性名称
- 文字节点的节点名称为#text
- 文件节点的节点名称为#document

- nodeValue

- 组件节点的节点值为null
- 文字节点的节点值即为文字本身
- 属性节点的节点值即为属性值

- nodeType

- 1 : Element 组件
- 2 : Attribute 属性
- 3 : Text 文字
- 8 : Comment 批注
- 9 : Document 文件

HTML DOM 方法

- getElementById(id)
- getElementsByTagName(name)
- appendChild(node)
- removeChild(node)
- getAttribute(attributeName)

HTML DOM Collections

- attributes[]
 - 回传该组件的所有属性组成一个数组
- childNodes[]
 - 回传该组件的所有子节点组成一个数组

改变组件

- 改变组件的属性值
 - `document.body.bgColor="yellow";`
- 改变组件内的文字
 - `document.getElementById("p1").innerHTML="Hi!";`
- 改变组件的样式
 - `document.body.style.color="blue";`
 - `document.body.style.backgroundImage`

自定义物件

- [方法一]直接建立一个新对象

```
personObj = new Object();
personObj.firstname = "John";
personObj.lastname = "Doe";
personObj.age = 50;
personObj.eyecolor = "blue";
personObj.smile=function()
{
    document.write("^___^");
};

personObj.smile();

document.write(personObj.firstname+"<br />");
```


自定义物件

- [方法二]直接建立一个新对象(写在一起)

```
personObj={firstname:"John",lastname:"Doe",age:50,eyecolor:"blue",  
smile:function(){document.write("-_-|||")}};  
  
personObj.smile();  
  
document.write(personObj.lastname+"<br />");
```

自定义物件

- [方法三]先宣告，再使用

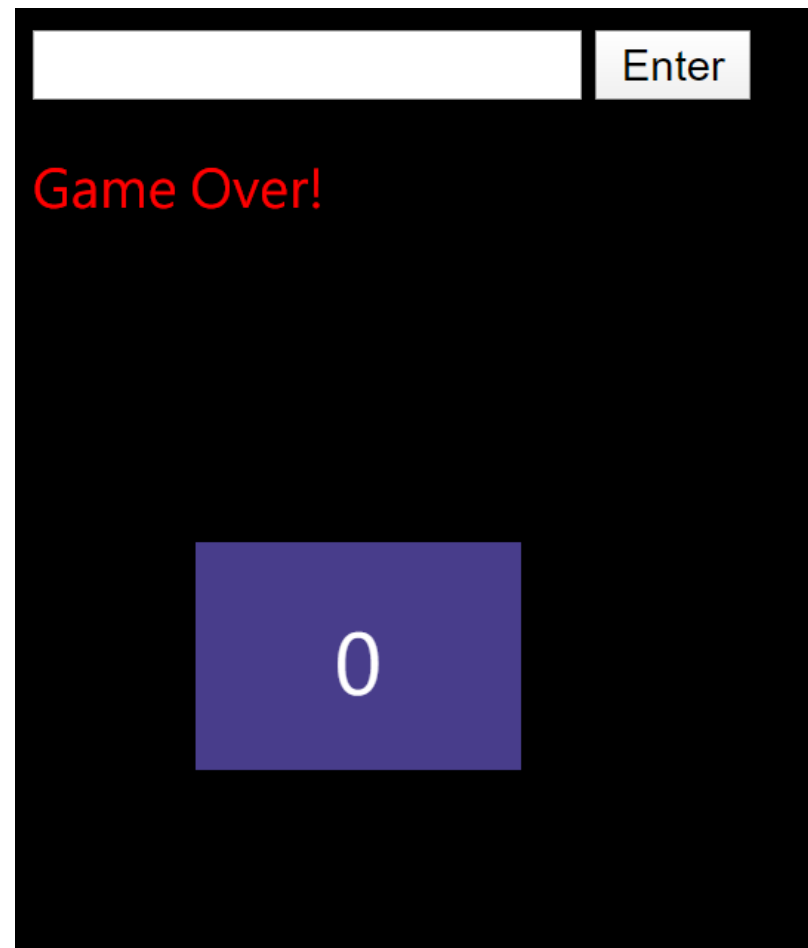
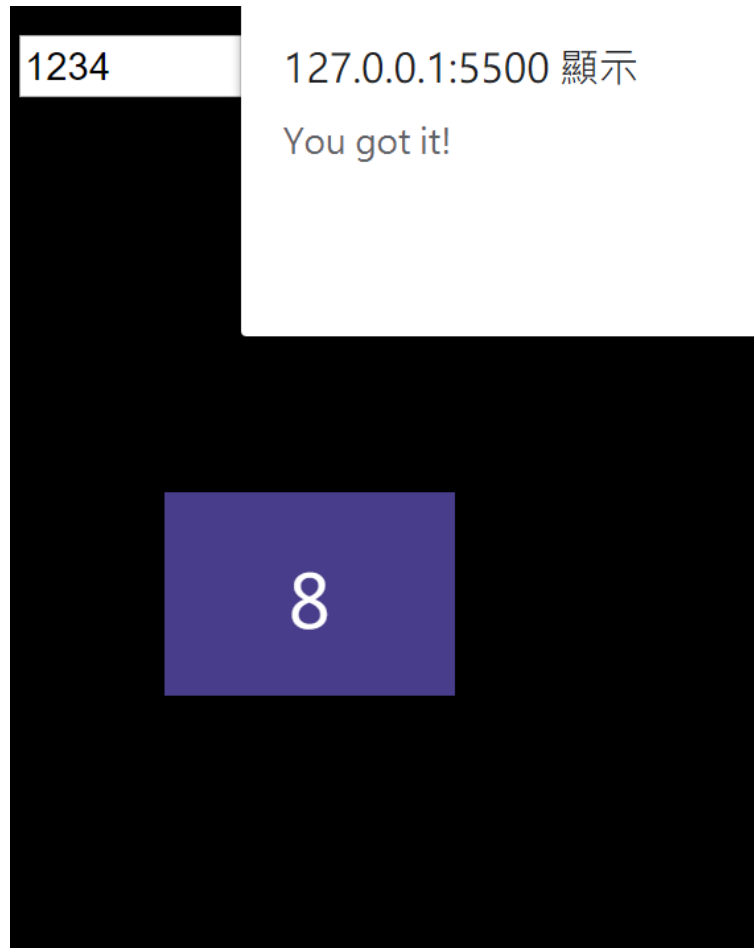
```
function person(firstname, lastname, age, eyecolor) {  
    this.firstname = firstname;  
    this.lastname = lastname;  
    this.age = age;  
    this.eyecolor = eyecolor;  
    this.smile=function(){document.write("^0^")} ;  
}
```

```
var myFather=new person("John","Doe",50,"blue");  
myFather.smile();  
document.write(myFather.lastname+"<br />");
```

定时器

- `setTimeout()` : 特定时间之后做什么事情(一次)
- `clearTimeout()` : 解除`setTimeout`
- `setInterval()` : 每隔多少时间之后做什么事情(反复)
- `clearInterval()` : 解除`setInterval`

练习：密码锁炸弹



index.html

```
<!DOCTYPE html>
  <head>
    <meta charset="utf-8">
    <title></title>
    <link rel="stylesheet" href="style.css">
  </head>
  <body>
    <input id="userInput" type="text">
    <button>Enter</button>
    <p id="hint"></p>
    <div id="timer"></div>
    <script async defer src="main.js"></script>
  </body>
</html>
```

style.css

```
body{
  background-color: black;
}
div{
  width: 100px;
  height: 50px;
  margin: 100px 50px;
  text-align: center;
  padding-top: 20px;
  color: white;
  font-size: 25px;
  background-color: darkslateblue;
}
p{
  color: red;
  height: 20px;
}
```

main.js

```
let timer = document.getElementById("timer");
let userInput = document.getElementById("userInput");
let hint = document.getElementById("hint");
let button = document.getElementsByTagName("button")[0];
let count = 10;
```

```
timer.innerHTML = count;
button.addEventListener("click", checkPassword);
let myVar = setInterval(myTimer, 1000);
```

```
function myTimer(){
    count--;
    timer.innerHTML = count;
    if(count==0){
        hint.innerHTML="Game Over!";
        clearInterval(myVar);
    }
}
```

```
function checkPassword(){
    hint.innerHTML="";
    if(parseInt(userInput.value)==1234){
        alert("You got it!");
        clearInterval(myVar);
    }else{
        hint.innerHTML="Try again!";
    }
    userInput.value = null;
}
```

JavaScript 综整

- 基本认识
 - 简介、功能、摆放位置、批注
- 程序运作
 - 变量、函数、循环、事件触发
- 面向对象
 - 内部对象、DOM、自定义物件





Reactor



developer.microsoft.com/reactor/
[@MSFTReactor](#) on Twitter

Microsoft Reactor 上海 – 网站开发系列

日期	星期	时间	主题
9/02	三	19:30 ~ 21:00	Intro to TypeScript

议程结束 感谢聆听



请记得填写课程回馈问卷
<https://aka.ms/ReactorFeedback>

© 2019 Microsoft Corporation. All rights reserved. The text in this document is available under the Creative Commons Attribution 3.0 License, additional terms may apply. All other content contained in this document (including, without limitation, trademarks, logos, images, etc.) are not included within the Creative Commons license grant. This document does not provide you with any legal rights to any intellectual property in any Microsoft product. You may copy and use this document for your internal, reference purposes.

This document is provided "as-is." Information and views expressed in this document, including URL and other Internet Web site references, may change without notice. You bear the risk of using it. Some examples are for illustration only and are fictitious. No real association is intended or inferred. Microsoft makes no warranties, express or implied, with respect to the information provided here.