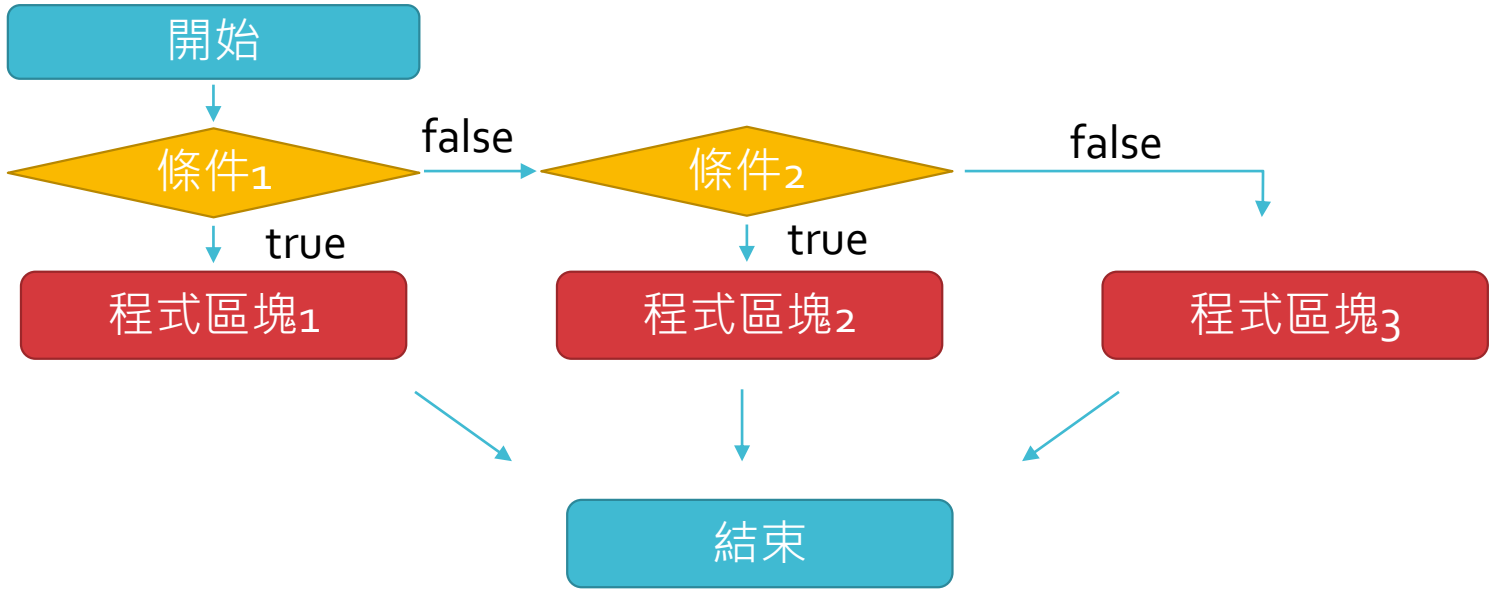
RWD網頁設計

第五次課程

JS if/else if/else 條件控制

- 當程式條件超過三個以上範圍時，可使用if/else if /else條件控制來撰寫程式




```
if (條件式1) {
    //條件式1成立時執行
}else if(條件式2){
    //條件式2成立時執行
}else{
    //所有條件都不成立時執行
}
```

JS if/else if/else 條件控制

- 範例程式：成績分布
- 題目：輸入一正整數(0<=N<=100)，判斷其成績等第。低於60分輸出不及格，60~69為丙等，70~79為乙等，80~89為甲等，90~100優等。

```
var a = prompt("請輸入分數",0);
a = a*1;
if (a>=90) {
    document.write("優等");
}else if (a>=80 && a<90) {
    document.write("甲等");
}else if (a>=70 && a<80) {
    document.write("乙等");
}else if (a>=60 && a<70) {
    document.write("丙等");
}else{
    document.write("不及格");
}
```

JS switch...case 條件控制

- 說明：當有明確之數值比較時，可以使用switch，相當等於if判斷裡面的==(比較運算子)
- 基本程式碼

```
switch(變數){
    case 判斷值1:
        //變數等於判斷值1時執行的程式內容
        break;
    case 判斷值2:
        //變數等於判斷值2時執行的程式內容
        break;
    case 判斷值3:
        //變數等於判斷值3時執行的程式內容
        break;
    default:
        //當變數值與所有判斷值皆不相等時執行的內容
        break;
}
```

JS switch...case 條件控制

- 範例程式：付款方式選擇
- 題目：由使用者輸入付款方式的代號，再依代號的內容判斷要顯示的訊息。

```
var pay_type = prompt("請選擇付款方式:1.ATM 2.刷卡 3.貨到付款");
switch(pay_type){
    case "1":
        document.write("您選的付款方式為ATM");
        break;
    case "2":
        document.write("您選的付款方式為刷卡");
        break;
    case "3":
        document.write("您選的付款方式為貨到付款");
        break;
    default:
        document.write("無此付款方式");
        break;
}
```

JS switch...case 條件控制

- 練習程式：判斷工作日
- 題目：輸入一正整數(1<=N<=7)代表星期，並判斷值為工作日、休息日、例假日
- 輸入：輸入一正整數(1<=N<=7)
- 輸出：判斷輸入數字後，輸出對應的結果
- P.S.周六(6)為休息日，周日(7)為例假日

JS
迴圈

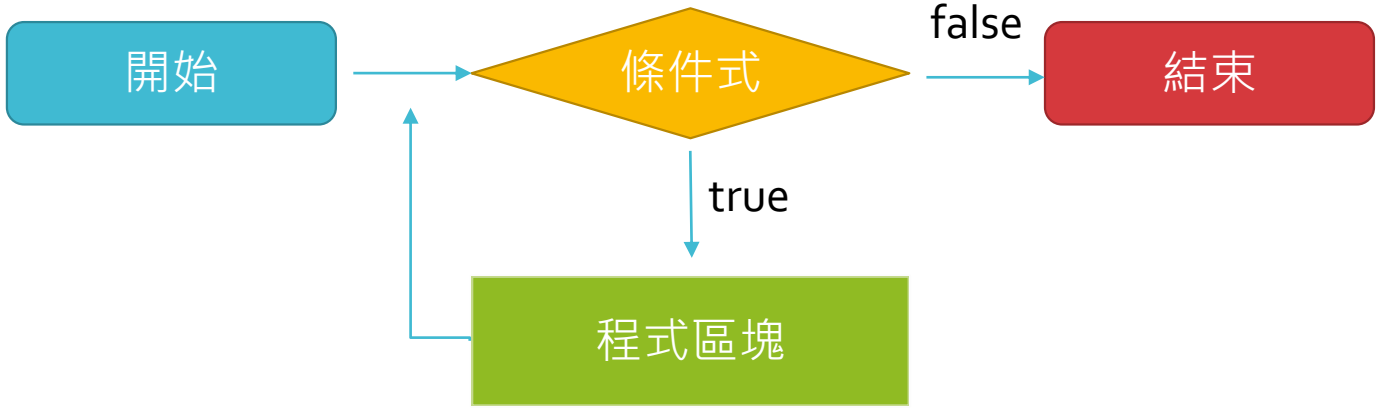
- 迴圈：在程式語言裡面，是一種常見的控制流程方法。迴圈即為在一段程式中出現一次，但可能重複執行多次的方法。它可用來執行特定重複次數，或是達到指定條件後才停止迴圈。JS常見的迴圈有for、while、do迴圈。3種迴圈架構雖不同，但是原理一樣，都能做到相同的功能。

- 禁忌：迴圈最忌諱的是出限無窮迴圈的狀態，容易造成系統崩潰，因此須謹慎操作。

JS
while迴圈

- while迴圈：進入迴圈時，程式會先判斷條件式是否吻合條件，若吻合，則重複執行迴圈內容，否則跳出。

```
while(條件式){
    //條件式達成時執行的內容
}
```



JS
while迴圈

- 範例題目：輸出1~10數字

```
var a=0;
while(a<=10){
    a=a+1;
    document.write(a+' ');
}
```

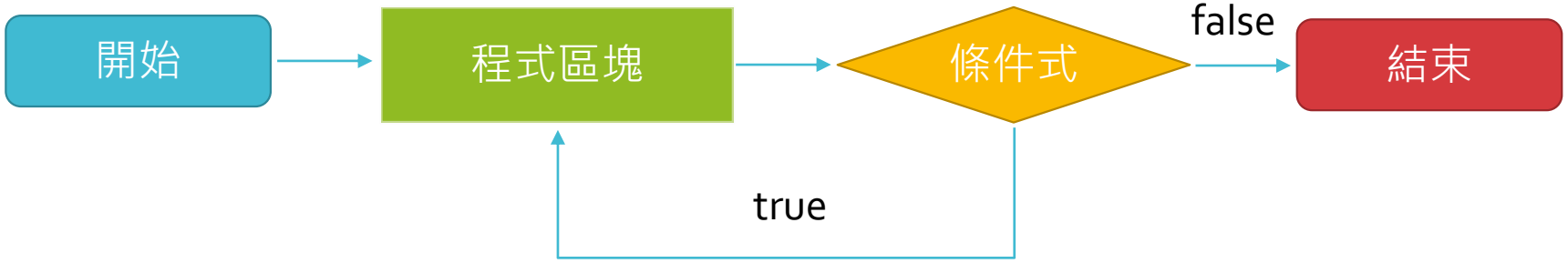
JS
while迴圈

- 練習題目：由使用者輸入一數值(需大於1)，並輸出1至該數值之間的所有偶數。如：輸入10，擇輸出2 4 6 8 10。

JS
do迴圈

- do迴圈：進入迴圈時，程式會先執行迴圈內容後，再判斷條件式是否吻合條件，若吻合，則重複執行迴圈內容，否則跳出。

```
do{
    執行的程式內容
}while(條件式)
```



JS
do迴圈

- 範例題目：輸出1~10數字

```
var sum = 0;
do{
    sum++;
    document.write(sum+" ");
}while(sum<10);
```

JS
do迴圈

- 練習題目1：由使用者輸入一數值(需大於1)，並輸出1至該數值之間的所有奇數。如：輸入10，則輸出1 3 5 7 9。

- 練習題目2：輸入一個正整數，並將所有數字倒轉之後輸出，如輸入12345，則輸出54321。(開頭為0不需要輸出)
- *提示:使用Math.floor函數

JS
do迴圈

- 練習1解答

```
var input = prompt();
var sum = -1;
do{
    sum+=2;
    document.write(sum+" ");
}while(sum<input-1);
```

- 練習2解答

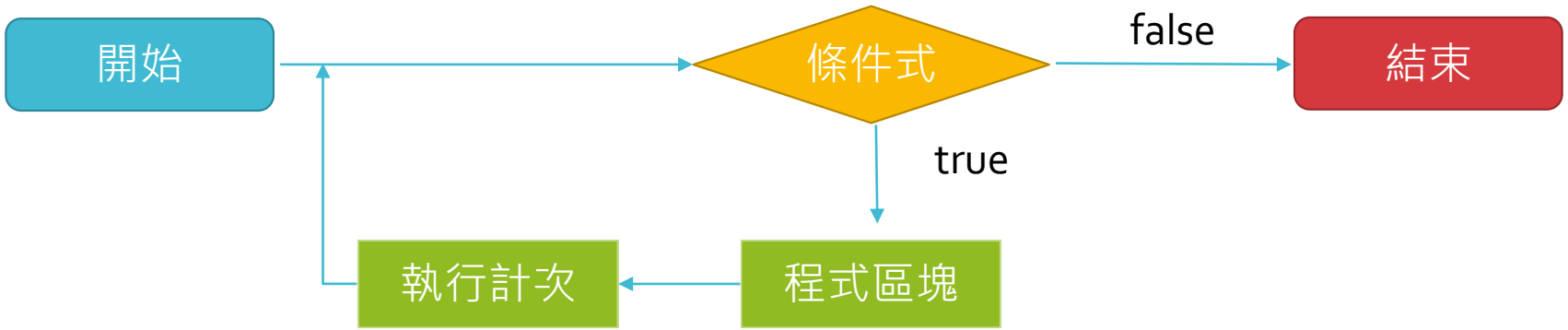
```
var input = prompt();
do{

    document.write(input%10);
    input = Math.floor(input/10);
}while(input>0);
```

JS
for迴圈

- for迴圈：進入迴圈時，需先設定變數初值，再設定計次條件，最後設定計次方式。程式會先判斷條件式是否吻合條件，若吻合，則重複執行迴圈內容，直至條件不符合才結束。

```
for (變數初值;條件式;變數計次方式) {
    執行的程式內容
}
```



JS
for迴圈

- 範例題目：輸出1~10數字

```
var i;
for (i=1;i<=10;i++) {
    document.write(i+" ");
}
```

JS
for迴圈

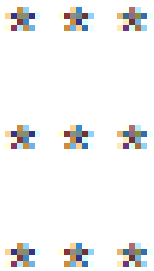
- 練習題目：由使用者輸入一數值(需大於1)，並輸該數值至0之間(包含0及該數)的所有整數。如：輸入，則輸出5 4 3 2 1 0。

```
var input = prompt();
for (input; input>=0; input--) {
    document.write(input+" ");
}
```

JS多重迴圈

- 多重迴圈：由迴圈內再包一至多個迴圈
- 範例：透過迴圈輸出右圖

```
for(var i=0;i<3;i++){
    for(var j=0;j<3;j++){
        document.write("* ");
    }
    document.write("<br>");
}
```



JS多重迴圈

- 練習：透過雙重迴圈輸出九九乘法表

JS
自訂函數

- 隨著程式開發內容越多，在操作或判斷時會經常使用某些重複或相似的程式，這些程式就可以整理成一段函數，讓程式更精簡、修改維護更容易。
- ※函數都必須經過呼叫才會執行。
- ※函數內之變數僅能在該函數內使用。

```
function 函數名稱(參數1,參數2) {
    執行的程式內容
    return 回傳值;
}
```

- ※參數、return回傳值為非必要項目

JS
自訂函數

- 範例1：透過呼叫函數輸出1-10所有數

```
output();//呼叫函數

function output() {
    var i;
    for(i=0; i<=10; i++){
        document.write(i+" ");
    }
}
```

- 範例2：透過呼叫函數並傳入參數(需大於1)，輸出1至該參數之間所有整數。

```
output(10);//呼叫函數

function output(i) {
    for(i; i>=1; i--){
        document.write(i+" ");
    }
}
```

JS
自訂函數

- 範例3：設定一變數，並將該變數值設為函數運算結果後輸出。函數程式內容為1-10加總。

```
var sum = output();//呼叫函數
document.write(sum);

function output() {
    var i;
    var sum = 0;
    for(i=1; i<=10; i++){
        sum+=i;
    }
    return sum;
}
```

JS
自訂函數

- 練習：設定一函數，由使用者輸入攝氏溫度後，該函數可運算攝氏溫度轉換為華氏溫度後回傳並輸出。

```
var dc = prompt();
function dctodf(dc) {
    return dc*1.8+32;
}
document.write("華氏溫度為"+dctodf(dc));
```

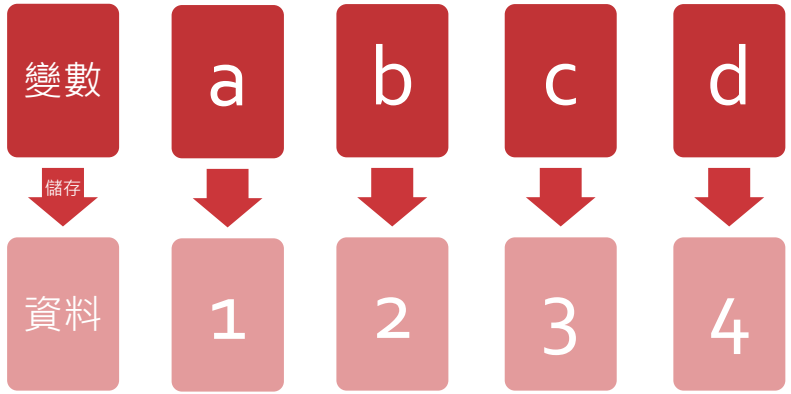
JS陣列

- 在程式設計中，若有大量同類型資料要儲存時，必須宣告大量變數，不僅耗費程式馬，執行效率也不佳。

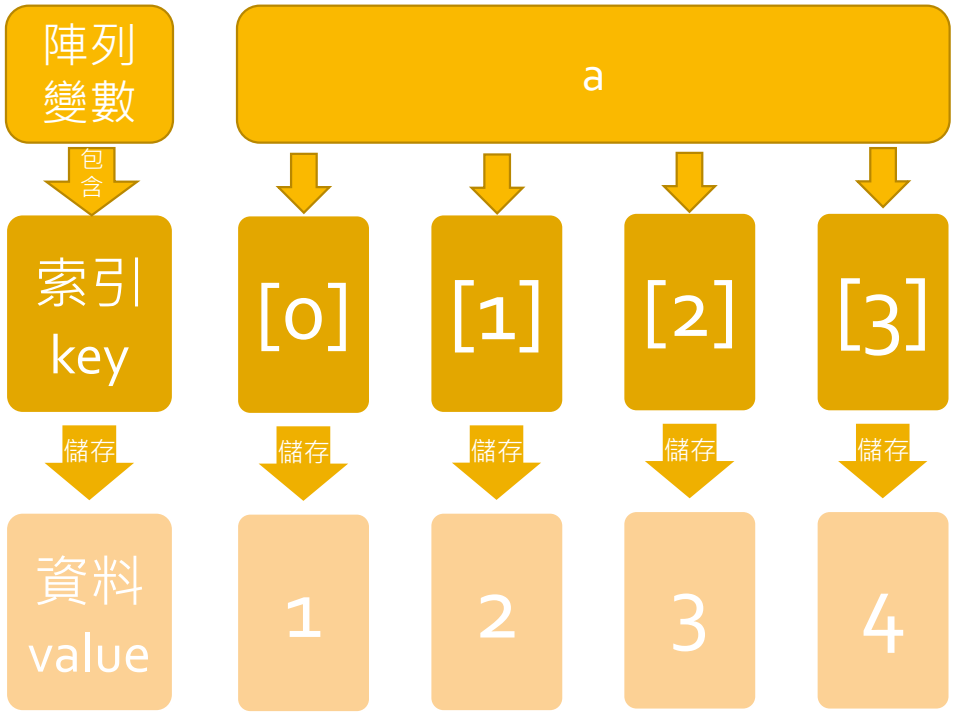
- 陣列：一群性質相同變數的集合。相同陣列中擁有一個變數名稱，作為辨識該陣列的標誌，陣列中的每一份資料為陣列元素(key)，每個元素可以包含一到多筆資料(value)。

JS陣列

一般變數



陣列變數



JS一維陣列

- 陣列宣告方式：new Array()
- 範例：
```
var a = new Array();
```
- 範例2：
```
var a = new Array();
a[0]=1;
a[1]=2;
a[2]=3;
a[4]=3;
```
- 範例3：
```
var a = new Array();
a[0]=1;
a[1]=2;
a[2]=3;
a[4]=3;
document.write(a[2]);
```

陣列a索引為2的值:3

JS一維陣列

- 練習1
  - 宣告一一維陣列，並賦予10個索引(0-9)後，輸出索引為2.4.6.8的值
- 練習2
  - 利用練習一之陣列，透過迴圈方式輸出索引為偶數(2.4.6.8)的值

JS多維陣列

- 多維陣列：許多一維陣列組合起來的變數。
- 範例：

```
var student = new Array();
student[0] = ["David", 95, 60];
student[1] = ["James", 82, 100];
student[2] = ["Penny", 11, 99];
document.write(student);
```

	第1行	第2行	第3行
	姓名	國文	數學
第1列	David	95	60
第2列	James	82	100
第3列	Penny	11	99

JS多維陣列

- 範例：輸出多維陣列

```
var student = new Array();
student[0] = ["David", 95, 60];
student[1] = ["James", 82, 100];
student[2] = ["Penny", 11, 99];
for(var i=0;i<3;i++){
    for(var j=0;j<3;j++){
        document.write(student[i][j]+" ");
    }
    document.write("<br>");
}
```

David 95 60
James 82 100
Penny 11 99