LE VARIABILI

Come per ogni linguaggio, anche con JavaScript possiamo utilizzare **le variabili** per memorizzare valori o oggetti durante l’esecuzione degli script. Una variabile è identificata da un nome che deve rispettare alcune regole:

* non deve coincidere con una delle parole chiave del linguaggio;
* non può iniziare con un numero;
* non può contenere caratteri speciali come ad esempio:
  + lo spazio,
  + il trattino (-),
  + il punto interrogativo (?),
  + il punto (.),

Sono però ammessi:

l’underscore (\_)

il simbolo del dollaro ($).

JavaScript non prevede la dichiarazione obbligatoria delle variabili, cioè un’operazione esplicita di creazione. Il semplice utilizzo di un identificatore indica all’engine di creare implicitamente la variabile, se non esiste già. Tuttavia questa pratica può favorire errori ed ambiguità.

Una variabile non dichiarata viene automaticamente creata al primo utilizzo ed è accessibile da qualsiasi punto di uno script. Questo significa che in script particolarmente complessi si può correre il rischio di utilizzare involontariamente in punti diversi variabili con lo stesso nome generando comportamenti non previsti.

È opportuno pertanto **dichiarare sempre una variabile** tramite la parola chiave **var**:

var miaVariabile;

avendo così la possibilità di stabilire un ambito di accessibilità o **scope**, come vedremo in seguito.

## Operatori aritmetici

Per quel che riguarda il tipo di dato numerico, abbiamo gli operatori aritmetici, che consentono la combinazione di valori numerici. Si tratta di operatori binari corrispondenti ai classici operatori matematici:

|  |  |
| --- | --- |
| Operatore | Nome |
| + | addizione |
| - | sottrazione |
| / | divisione |
| \* | moltiplicazione |
| % | modulo o resto |

Gli operatori aritmetici seguono le regole di precedenza matematiche e, come in matematica, possiamo utilizzare le parentesi (ma soltanto quelle tonde) per modificare il loro ordine di valutazione:

Gli operatori aritmetici prevedono anche tre operatori unari:

|  |  |
| --- | --- |
| Operatore | Nome |
| - | negazione |
| ++ | incremento |
| -- | decremento |

La negazione consente di ottenere un valore negativo di un numero.

Gli operatori di incremento e decremento sono applicabili soltanto a variabili e consentono di ottenere un valore rispettivamente aumentato o diminuito di uno. È molto importante la posizione dell’operatore. Infatti, se l’operatore si trova a sinistra dell’operando, l’operazione di incremento o decremento viene eseguita prima della valutazione dell’espressione, mentre se si trova a destra l’operazione viene eseguita dopo.

## Operatori relazionali

Gli operatori relazionali sono quegli operatori utilizzati per il confronto tra valori (e stabilire una “relazione d’ordine”):

|  |  |
| --- | --- |
| Operatore | Nome |
| < | minore |
| <= | minore o uguale |
| > | maggiore |
| >= | maggiore o uguale |
| == | uguale |
| != | diverso |
| === | strettamente uguale |
| !== | strettamente diverso |

Le espressioni che utilizzano gli operatori relazionali restituiscono un valore booleano in base all’esito del confronto. Ad esempio:

4 > 2 // true

5 != 4 + 1 //false

Il significato della maggior parte degli operatori dovrebbe essere abbastanza intuitivo. Per quanto riguarda il significato degli operatori === e !==, questi verranno approfonditi quando parleremo delle conversioni di tipo.

Gli operatori logici consentono la combinazione di espressioni booleane. JavaScript prevede due operatori binari e un operatore unario:

|  |  |
| --- | --- |
| Operatore | Nome |
| && | and |
| || | or |
| ! | not |

Le seguenti sono espressioni che utilizzano operatori logici:

5 > 2 && 3 != 4 // true

true || 4 >= 6 // true

!5==5 // false

È però opportuno utilizzare le parentesi per indicare esplicitamente l’ordine di valutazione e per renderle più leggibili.

Operatori di assegnamento

Un discorso a parte va fatto per gli **operatori di assegnamento**. Abbiamo già avuto modo di vedere *informalmente* l’operatore = che ci consente di assegnare il valore di un’espressione ad una variabile. L’espressione di assegnamento restituisce essa stessa un valore: il valore dell’espressione assegnata alla variabile. Il valore del seguente assegnamento, ad esempio:

x = 3 + 5;

è lo stesso valore dell’espressione 3 + 5. Questo vuol dire che possiamo avere espressioni analoghe alla seguente:

y = (x = 3 + 5);

dove y assumerà lo stesso valore di x, cioè 8.

GLI ARRAY

Gli **array** consentono di associare più valori ad un unico nome di variabile (o identificatore). In genere i valori contenuti in un array hanno una qualche affinità (ad esempio l’elenco dei giorni della settimana). L’uso degli array evita di definire più variabili e semplifica lo svolgimento di operazioni cicliche su tutti i valori. Il seguente è un esempio di array:

var giorniDellaSettimana = [

"lunedì",

"martedì",

"mercoledì",

"giovedì",

"venerdì",

"sabato",

"domenica"];

Come si vede in questa dichiarazione i valori dell’array sono delimitati da parentesi quadre e separati da virgole. Questo è un modo di definire un **array basato sulla rappresentazione letterale**.

Una volta definito un array possiamo accedere ai singoli elementi facendo riferimento al nome della variabile e all’indice corrispondente all’elemento, ricordandoci che **la numerazione degli indici parte da zero**.

Così, ad esempio, se vogliamo accedere al primo elemento dell’array dei giorni della settimana scriveremo:

var primoGiorno = giorniDellaSettimana[0];

Naturalmente l’indice scalerà allo stesso modo anche per gli altri giorni:

var terzoGiorno = giorniDellaSettimana[2];

var settimoGiorno = giorniDellaSettimana[6];

Dopo l’assegnamento ciascuna variabile conterrà la stringa corrispondente all’indice.

Un array può essere vuoto, cioè senza elementi, ed è rappresentato dalle sole parentesi quadre [].

Se definiamo un array nel seguente modo:

var myArray = [,"elemento"];

otterremo un array con due elementi di cui il primo è undefined. Se invece l’elemento finale non è indicato, come nel seguente esempio:

var myArray = ["elemento", "altroelemento",];

JavaScript ignorerà l’ultima virgola generando un array di due soli elementi.

Gli elementi di un array possono essere di qualsiasi tipo di dato previsto da JavaScript, quindi possiamo avere array con elementi di diverso tipo:

var myArray = [123, "stringa", true, null];

Un elemento di un array può anche essere un altro array:

var myArray = [123, "stringa", ["a", "b", 99]];

In questo caso, per accedere ad esempio all’elemento 99 dovremo accedere prima al terzo elemento dell’array principale e poi al terzo elemento dell’array contenuto in esso:

var novantanove = myArray[2][2];

## Array Multidimensionali

La presenza di array come elementi di altri array ci consente di definire **array multidimensionali o matrici**. Ad esempio, quella che segue è la definizione di una matrice 3×3 di interi:

var matrice = [[24, 13, 1], [48, 92, 17], [8, 56, 11]]

Per accedere al numero 48 indicheremo le sue coordinate nel seguente modo:

var quarantotto = matrice[1][0];