LA SINTASSI

In questa lezione e nelle successive concentreremo l’attenzione su tipi di dato, variabili, istruzioni, array e funzioni. Ma prima facciamo alcune considerazioni sugli aspetti sintattici.

## Il punto e virgola in JavaScript

Anzitutto ricordiamo il codice JavaScript è composto da una sequenza di istruzioni che viene interpretata ed eseguita dall’engine. In questa sequenza, ciascuna istruzione (o blocco di istruzioni) è delimitata da un *punto e virgola* (;), come nel seguente esempio:

var x = 5;

x = x + 1;

Terminare un’istruzione con **il punto e virgola non è obbligatorio** in JavaScript: possiamo scrivere la sequenza precedente anche così:

var x = 5

x = x + 1

In base alle specifiche del linguaggio, il parser effettua un inserimento automatico del punto e virgola. Tuttavia **è buona norma inserire sempre la punteggiatura di terminazione** per evitare ambiguità e risultati inattesi. Tra le altre cose, mettere il punto e virgola in fondo ad ogni istruzione ci consente di scrivere più istruzioni sulla stessa riga:

var x = 5; x = x + 1;

## JavaScript è case sensitive

Un altro aspetto sintattico da tenere in considerazione è il fatto che JavaScript è *case sensitive*, cioè fa **distinzione tra maiuscole e minuscole** nei nomi di istruzioni, variabili e costanti.

## Commenti in JavaScript

Come ogni linguaggio, JavaScript prevede delle sequenze di caratteri per **inserire commenti nel codice**. Tutto ciò che viene marcato come commento non viene preso in considerazione dall’interprete JavaScript. Possiamo inserire due tipi di commenti nel codice:

* commento per singola riga;
* commento multiriga.

Il **commento a singola riga** (inline) inizia con i caratteri // (doppio slash), come mostrato nel seguente esempio:

// Questo è un commento

var x = 5;

Tutto ciò che si trova a destra dei caratteri // fino alla fine della riga verrà considerato commento.

Il **commento multiriga** (multiline) prevede la sequenza iniziale /\* (slash e asterisco) e si conclude con \*/ (asterisco e slash). Tutto ciò che viene inserito all’interno di questa coppia di terminazioni sarà considerato un commento, indipendentemente dal numero di righe utilizzate:

/\* Questo è un commento sulla prima riga

Questo è un commento sulla seconda riga

Questa è l'ultima riga del commento \*/

var x = 5;

## Spazi bianchi

Gli spazi bianchi in JavaScript non assumono significati particolari nelle espressioni e possiamo utilizzarli per aumentare la leggibilità del codice. Unica eccezione va fatta per gli spazi all’interno delle stringhe (che vedremo).

TIPI DI DATO PRIMITIVI

## Stringhe in JavaScript

Una **stringa in JavaScript** è una sequenza di caratteri delimitata da **doppi o singoli apici**. Le seguenti sono esempi di stringhe:

"stringa numero uno"

'stringa numero due'

Un tipo speciale di stringa è la **stringa vuota**, cioè una stringa senza caratteri. Essa può essere rappresentata indifferentemente come "" oppure ''.

Per inserire caratteri speciali all’interno di una stringa si fa ricorso al **carattere di escaping** \ (backslash). Ad esempio, per inserire un ritorno a capo possiamo utilizzare la sequenza \n, come mostrato nel seguente esempio:

"Ieri pioveva.\n Oggi c'è il sole."

La stampa di questa stringa avrà il seguente risultato:

Ieri pioveva.

Oggi c'è il sole.

Tra i caratteri speciali che possiamo inserire tramite backslash c’è lo stesso backslash, come nel seguente esempio:

"La sequenza \\n indica il ritorno a capo"

Il primo backslash è il carattere di escaping mentre il secondo è il carattere da inserire nella stringa. Il risultato sarà quindi:

La sequenza \n indica il ritorno a capo

## Numeri in JavaScript

JavaScript ha un unico tipo di dato numerico, cioè non c’è distinzione formale, ad esempio, tra intero e decimale. Internamente tutti i valori numerici sono rappresentati come numeri in virgola mobile, ma se non è specificata la parte decimale il numero viene trattato come intero.

I seguenti sono esempi di rappresentazione di numeri interi:

var interoNegativo = -10;

var zero = 0;

var interoPositivo = 123;

Per rappresentare un numero non intero facciamo ricorso al punto come separatore della parte decimale:

var numeroDecimale = 0.52;

var altroNumeroDecimale = 12.34;

var decimaleNegativo = -1.2;

var decimaleZero = 1.0;

Ogni valore che va al di fuori dell’intervallo rappresentabile non genera un’eccezione ma viene rappresentato da due valori speciali:

* **Infinity**, detto anche infinito positivo,
* **-Infinity**, detto infinito negativo.

Ogni valore che ha una precisione maggiore di quella rappresentabile viene considerata uguale a zero.

Un altro valore numerico speciale è **NaN**, acronimo di *Not a Number*, che indica un valore numerico non definito. Nel seguente esempio il valore della variabile x dopo l’assegnamento è NaN:

var x = x + 1;

## I tipi di dato null, undefined e booleano

JavaScript prevede due tipi di dato speciali per rappresentare valori nulli e non definiti.

Il tipo di dato **null** prevede il solo valore null, che rappresenta un valore che non rientra tra i tipi di dato del linguaggio, cioè non è un valore numerico valido, né una stringa, né un oggetto.

È possibile assegnare il valore null ad una variabile come nel seguente esempio:

var x = null;

Da notare che il valore null è diverso dal valore numerico 0 o dalla stringa vuota. Infatti, questi ultimi appartengono ai tipi numero e stringa, mentre null è un tipo a se stante.

Il tipo di dato **undefined** rappresenta un valore che non esiste. Anche questo tipo di dato contiene un solo valore: undefined. Questo è il valore di una variabile non inizializzata, a cui non è stato assegnato nessun valore, nemmeno null.

**Il tipo di dato booleano** prevede due soli valori: true (vero) e false (falso).

var miaVariabile = 0;