

FONDAMENTI DI PROGRAMMAZIONE

Giovanni Pellegrini

JAVASCRIPT VARIABILI









2024

Instant Book 3

INSTANT BOOK 4

VARIABILI

1.  **VARIABILI** pag. 3
2.  **UNA ANALOGIA CON IL MONDO REALE** pag. 4
3.  **NOMI DELLE VARIABILI** pag. 5
4.  **COSTANTI** pag. 7
5.  **RIEPILOGO** pag. 9
6.  **ESERCIZI**

● VARIABILI

La maggior parte delle volte, le applicazioni JavaScript necessitano di lavorare con informazioni. Vediamo due esempi:

- Un negozio online – le informazioni possono riguardare i beni venduti e il carrello.
- Un applicazione di messaggistica – le informazioni possono riguardare utenti, messaggi e molto altro.

Le variabili vengono utilizzate per memorizzare informazioni.

Una variabile è uno “spazio di memoria con nome” utilizzato per salvare dati.

Per creare una variabile in JavaScript, dobbiamo utilizzare la parola chiave **let**.

L'istruzione sotto crea (in altre parole: dichiara) una variabile identificata dal nome “messaggio”:

```
let message;
```

Adesso possiamo inserirci dei dati utilizzando l'operatore di assegnazione =:

```
let message;  
message = 'Hello'; // memorizzazione della stringa
```

La stringa è adesso salvata nell'area di memoria associata alla variabile. Possiamo accedervi utilizzando il nome della variabile:

```
let message;  
message = 'Hello!';  
alert(message); // mostra il contenuto della variabile
```

Per precisione, potremmo unire la dichiarazione e l'assegnazione in una singola riga:

```
let message = 'Hello!'; // definisce la variabile e gli assegna il valore  
alert(message); // Hello!
```

Possiamo anche dichiarare più variabili in una riga:

```
let user = 'John', age = 25, message = 'Hello';
```

Questo potrebbe risultare più breve, ma è sconsigliato. Per mantenere una migliore leggibilità è meglio dichiarare solamente una variabile per riga.

L'alternativa a più righe è un po' più lunga, ma più facile da leggere:

```
let user = 'John';  
let age = 25;  
let message = 'Hello';
```

Alcuni programmatori scrivono variabili multiple in questo modo:

```
let user = 'John',  
    age = 25,  
    message = 'Hello';
```

... o anche con la virgola su nuova riga:

```
let user = 'John'  
    , age = 25  
    , message = 'Hello';
```

Tecnicamente, tutte queste varianti fanno la stessa cosa. Quindi è una questione di gusto personale ed estetico.



var piuttosto che let

Nei vecchi script si potrebbe trovare **var** piuttosto che **let**:

```
var message = 'Hello';
```

La parola chiave **var** è quasi la stessa cosa di **let**. Dichiara comunque una variabile, ma in un maniera leggermente diversa, “vecchio stile”.

UN' ANALOGIA CON IL MONDO REALE

Possiamo comprendere meglio il concetto di “variabile” se la immaginiamo come una scatola per dati, con attaccata un’etichetta univoca.

Per esempio, la variabile `message` può essere immaginata come una scatola con etichetta “`message`” con il valore “Hello!” al suo interno:



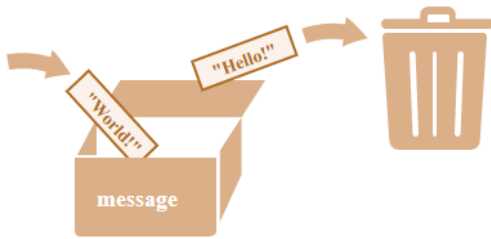
Possiamo inserire qualsiasi valore all’interno della scatola.

Possiamo anche cambiarlo.

Il valore può cambiare tutte le volte in cui sia necessario:

```
let message;  
message = 'Hello!';  
message = 'World!'; // il valore è cambiato  
alert(message);
```

Quando il valore viene cambiato, il dato vecchio viene rimosso dalla variabile:



Possiamo anche dichiarare due variabili e copiare i dati da una all'altra.

```
let hello = 'Hello world!';  
let message;  
// copia 'Hello world' da hello in message  
message = hello;  
// ora le due variabili contengono gli stessi dati  
alert(hello); // Hello world!  
alert(message); // Hello world!
```

Dichiarare una variabile due volte genera un errore

Una variabile dovrebbe essere dichiarata una volta sola.

La ripetizione della dichiarazione di una stessa variabile porterà ad un errore:

```
let message = "This";  
// 'let' ripetuto genererà un errore  
let message = "That"; // SyntaxError: 'message' has already been declared  
Quindi, dovremmo dichiarare una variabile una volta sola e farne riferimento senza la  
parola chiave let.
```

● NOMI DELLE VARIABILI

In JavaScript ci sono solo due limitazioni per il nome delle variabili:

Il nome deve contenere solo lettere, numeri, simboli \$ e _

Il primo carattere non può essere un numero.

Esempi di nomi validi:

```
let userName;  
let test123;
```

Quando il nome contiene più parole, viene utilizzato il camelCase. La logica è: le parole vanno una dopo l'altra, ogni parola inizia con lettera maiuscola: myVeryLongName. Una cosa interessante è che il simbolo del dollaro '\$' e l'underscore '_' possono essere utilizzati nei nomi. Sono dei semplici simboli, come le lettere, senza alcun significato speciale.

Ad esempio questi nomi sono validi:

```
let $ = 1; // dichiarata una variabile con nome "$"
let _ = 2; // qui una variabile con nome "_"
alert($ + _); // 3
```

Questi invece non lo sono:

```
let 1a; // non può cominciare con una stringa
let my-name; // '-' non è consentito nei nomi
```



La questione delle lettere

Le variabili apple ed AppLE sono distinte.

Le lettere non latine sono permesse, ma sono sconsigliate

E' possibile utilizzare qualsiasi alfabeto, compreso quello cirillico o addirittura i geroglifici:

```
let ИМЯ = '...';
let ☐ = '...';
```

Tecnicamente, non ci sono errori, questo tipo di nomi sono permessi, ma la tradizione internazionale è di utilizzare l'alfabeto inglese per il nome delle variabili.



Nomi riservati

C'è una lista di parole riservate, che non possono essere utilizzare come nomi di variabili, perché vengono utilizzate dal linguaggio stesso.

Per esempio, le parole let, class, return, function sono riservate.

Questo codice provocherà un errore di sintassi:

```
let let = 5; // non è possibile chiamare una variabile "let", errore!
let return = 5; // nemmeno "return", errore!
```

Attribuire i giusti nomi alle variabili

Parlando di variabili, c'è un'altra cosa estremamente importante.

Il nome di una variabile dovrebbe sempre essere semplice, ovvio e descrittivo del suo contenuto.

Dare i giusti nomi alle variabili è una delle abilità più importanti (e difficili) nella programmazione. Una rapida occhiata ai nomi delle variabili può rivelare se il codice sia stato scritto da un principiante o da uno sviluppatore esperto.

In un progetto reale, la maggior parte del tempo lo si perde a modificare ed estendere del codice già esistente, piuttosto che riscriverne uno nuovo. E quando si ritorna sul codice, dopo aver fatto qualcos'altro, risulta molto più facile trovare informazioni

se sono ben descritte. In altre parole, quando le variabili utilizzano dei nomi efficaci. Quindi è utile spendere del tempo a pensare il giusto nome per una variabile, prima di dichiararla. Questo approccio vi ripagherà.

Alcune regole da seguire:

- Utilizzare nomi leggibili da persone, come `userName` o `shoppingCart`.
- Evitare abbreviazioni o nomi brevi come `a`, `b`, `c`, senza che abbiano veramente senso.
- Rendere il nome il più descrittivo e preciso possibile. Esempi di pessimi nomi sono `data` e `value`. Questo tipo di nomi non dicono niente. Si possono utilizzare eccezionalmente se il contesto renda esplicito il significato.
- Definire delle regole personali o con il team. Se il visitatore del sito viene chiamato “user” allora dovremmo chiamare la relativa variabile come `currentUser` o `newUser`, non `currentVisitor` o `newManInTown`.

Sembra facile ? Infatti lo è, ma trovare dei buoni nomi che siano precisi e descrittivi nella pratica non è sempre così semplice.



Nuovo o Riciclo ?

Come ultima cosa. Ci sono alcuni programmatori un po' pigri, che invece di dichiarare nuove variabili tendono a riutilizzare quelle già esistenti.

Il risultato che si ottiene, è che le variabili sono come delle scatole in cui si possono mettere varie cose, senza cambiare l'etichetta. Cosa ci sarà dentro in un dato momento? Chi lo sa... Siamo costretti a controllare manualmente.

Questo genere di programmatori risparmiano qualche bit nella dichiarazione delle variabili ma perdono dieci volte il tempo risparmiato per fare debugging del codice.

Una variabile in più non è necessariamente un male.

I browser moderni e JavaScript minimizzano ed ottimizzano il codice abbastanza bene, quindi non ci saranno problemi di performance. Usare variabili differenti, per valori differenti può addirittura aiutare il motore JavaScript nell'ottimizzazione.



COSTANTI

Per dichiarare una variabile costante (immutabile), dobbiamo utilizzare `const` invece di `let`:

```
const myBirthday = '18.04.1982';
```

Le variabili dichiarate con `const` vengono chiamate “costanti”. Non possono cambiare valore. Se tentassimo di farlo verrebbe sollevato un errore:

```
const myBirthday = '18.04.1982';
```

```
myBirthday = '01.01.2001'; // errore, non è possibile riassegnare la costante!
```

Quando il programmatore è sicuro che il valore della variabile non cambierà mai, può

utilizzare `const` per soddisfare questa esigenza, rendendolo così esplicito.

- **Le costanti maiuscole**

Una pratica molto diffusa è di utilizzare le variabili costanti come alias di valori difficili da ricordare e che sono noti prima dell'esecuzione.

Questo tipo di costanti vengono identificate con lettere maiuscole e underscore (_). Come in questo esempio, creiamo delle costanti nel cosiddetto formato "web" (esadecimale):

```
const COLOR_RED = "#F00";  
const COLOR_GREEN = "#0F0";  
const COLOR_BLUE = "#00F";  
const COLOR_ORANGE = "#FF7F00";
```

```
// ...quando abbiamo bisogno di prelevare un colore  
let color = COLOR_ORANGE;  
alert(color); // #FF7F00
```

Benefici:

- COLOR_ORANGE è più facile da ricordare di "#FF7F00"
- E' più facile commettere errori scrivendo "#FF7F00" piuttosto che COLOR_ORANGE
- Quando leggiamo il codice, COLOR_ORANGE è molto più significativo di #FF7F00

Quando dovremmo utilizzare lettere maiuscole per una costante, e quando invece trattarle come normali variabili? Facciamo un pò di chiarezza.

Essere una "costante" significa che il valore non potrà mai cambiare. Ci sono costanti che sono note prima dell'esecuzione (come la codifica esadecimale del colore rosso), e ci sono quelle che vengono calcolate durante l'esecuzione, ma non cambieranno più dopo che gli sarà stato assegnato un valore.

Per esempio:

```
const pageLoadTime = /* tempo necessario da una pagina web per caricare */;
```

Il valore di pageLoadTime non è noto prima del caricamento della pagina, quindi viene trattato come una normale variabile. Ma rimane comunque una costante, perché non potrà più cambiare dopo che gli sarà stato assegnato un valore.

In altre parole, i nomi delle costanti in maiuscolo vengono utilizzati con variabili dal valore noto prima dell'esecuzione.



RIEPILOGO

Possiamo dichiarare variabili per memorizzare dati.

Le variabili possono essere dichiarate con `var`, `let` o `const`.

- `let` – è una dichiarazione delle variabili più moderna.
- `var` – è una dichiarazione delle variabili più vecchia-scuola.
- `const` – è simile a `let`, ma non consente di cambiare il valore della variabile.

Le variabili dovrebbero avere dei nomi semplici, ovvi e descrittivi.



ESERCIZI

1. Lavorare con le variabili

Dichiarare due variabili: `admin` e `name`.

Assegnare il valore “John” a `name`.

Copiare il valore da `name` su `admin`.

Mostrare il valore di `admin` tramite `alert` (deve stampare “John”).

2. Scegliere il nome opportuno

Creare una variabile con il nome del nostro pianeta. Come chiamereste questo tipo di variabile?

3. Scegliere il nome giusto

Creare una variabile che memorizzi il nome del visitatore corrente. Come chiameresti questa variabile?