# [PILLOLE PHP] Lo scope


Un argomento che può risultare ostico al pricipiante (o che perlomeno può indurlo a commettere degli errori banali o a perdere tempo) é senz'altro lo scope, ovvero l'ambito di validità di una variabile. Nell'esecuzione di uno script infatti **una variabile potrebbe avere diversi significati a dipendenza di dove viene dichiarata.**.
Iniziamo dunque a vedere quali sono i tre possibili scope di una variabile:

- **Globale**: una variabile si dice globale quando é dichiarata nello script principale (all'interno delle funzioni non avrà nessun valore);
- **Locale**: si dice locale una variabile dichiarata all'interno di una funzione (all'esterno della funzione non avrà nessun valore);
- **Super-globale**: le variabili super-globali sono quelle gestite da PHP ($_SESSION, $_SERVER, $_POST, $_GET, $_REQUEST, $_COOKIE, $_ENV, $_FILES, $GLOBALS). L'ambito di validità delle variabili super-globali é totale, cioé sono valide sia all'interno che all'esterno delle funzioni. Inoltre va considerato che anche le costanti hanno un abito di validità super-globale.

Ma facciamo subito degli esempi.

## Le variabili globali non sono disponibili in ambito locale

Come abbiamo già detto, le variabili globali non sono disponibili in ambito locale.
Guarda l'esempio spiegato di seguito:

## Le variabili locali non sono disponibili in ambito globale

Analizziamo l'esempio che segue per capire cosa intendiamo quando diciamo che le variabili locali non sono disponibili in ambito globale:

Quindi riassumendo possiamo dire che una funzione, per quanto concerne le variabili, é chiusa ermeticamente rispetto al resto del codice.

## Le variabili super-globali sono disponibili ovunque

Adesso, andiamo a dare uno sguardo alle variabili super-globali:

## Come far cadere le barriere tra ambito globale ed ambito locale?

E' possibile fare in modo che una **variabile globale** sia resa disponibile anche **all'interno di una funzione** tramite la keyword *global*; vediamo come:

E' possibile ottenere lo stesso risultato utilizzando l'array $GLOBALS, difatti l'array $GLOBALS è una variabile automaticamente disponibile in tutti gli ambiti senza doverla dichiarare globale per accedervi dall'interno di una funzione.

Considera comunque che rompere gli argini tra ambito locale ed ambito globale **é un'operazione da evitare finché é possibile.**

## Perché l'utilizzo di global é da evitare?

Come detto, per quanto possibile, é meglio evitare l'utilizzo delle tecniche appena descritte.
La netta distinzione di ambiti tra il locale e il globale, che al profano può sembrare un'inutile seccatura, ha delle ragioni molto precise.
Le funzioni generalmente rappresentano una parte di codice molto specifica della quale intravvediamo spesso dei possibili riutilizzi.
Dal momento che:

1. non sappiamo quando e in che circostanza potrebbe tornarci utile questa funzione;
2. nel caso in cui stiamo lavorando in team ad un grande progetto, potrebbe anche accadere che questa funzione venga riutilizzata da qualcun'altro;

è bene che la funzione sia il più robusta possibile. **Global inevitabilemente introduce delle ambiguità** e potrebbe portare all'interno della funzione dei valori non corretti.
Una funzione invece, di principio, riceve i dati come argomenti e li restituisce attraverso la keyword *return*. E questo basta per garantirle la robustezza che auspichiamo.

Per concludere vediamo come passare correttamente gli argomenti ad una funzione.

## Argomenti, argomenti facoltativi, argomenti arbitrari

Vediamo brevemente e in pratica come passare gli argomenti ad una funzione in tre diversi casi.

**SOMMA DI DUE NUMERI**

**SOMMA DI DUE O TRE NUMERI (argomenti facoltativi)**

**SOMMA DI X NUMERI (argomenti arbitrari)**

## Conclusione

In questo articolo abbiamo visto alcune caratteristiche fondamentali del liguaggio. La logica degli ambiti di validità delle variabili riveste infatti una notevole importanza nello sviluppo di un codice robusto. Spero di aver chiarito questo argomento che, come ho detto all'inizio, risulta spesso essere un punto critico per i principianti.