Scope statico e scope dinamico

Definizioni preliminari

- Definiamo lo **scope** come la porzione di codice in cui una variabile è accessibile e può essere utilizzata.

- Definiamo l’**ambiente** come una struttura dati che memorizza tutte le variabili e i loro valori disponibili durante l'esecuzione. Rappresenta lo "stato" delle variabili visibili in un dato momento e contesto del programma.

Ambienti principali

- Ambiente globale
- Ambiente locale

Scope globale vs. Scope locale

```
1  let global_x = 10;
2
3  function f(){
4    let local_x = 20;
5
6    console.log("Variabile locale", local_x);
7  }
8
9  console.log("Variabile globale", global_x);
10 f();
```

```
Variabile globale 10
Variabile locale 20
```

Scope globale vs. Scope locale

```
1  let x = 10;
2
3  function f(){
4    let x = 20;
5
6    console.log("In f x vale", x);
7  }
8
9  console.log("x vale", x);
10 f();
```

Cosa stampa questo codice?

Scope globale vs. Scope locale

```
1  let x = 10;
2
3  function f(){
4    let x = 20;
5
6    console.log("In f x vale", x);
7  }
8
9  console.log("x vale", x);
10 f();
```

```
x vale 10
In f x vale 20
```

Scope globale vs. Scope locale

```
1  let x = 10;
2
3  function f(){
4    let x = 20;
5    console.log("In f x vale", x);
6  }
7
8  function g(){
9    console.log("In g x vale", x);
10 }
11
12 f();
13 g();
```

Cosa stampa questo codice?

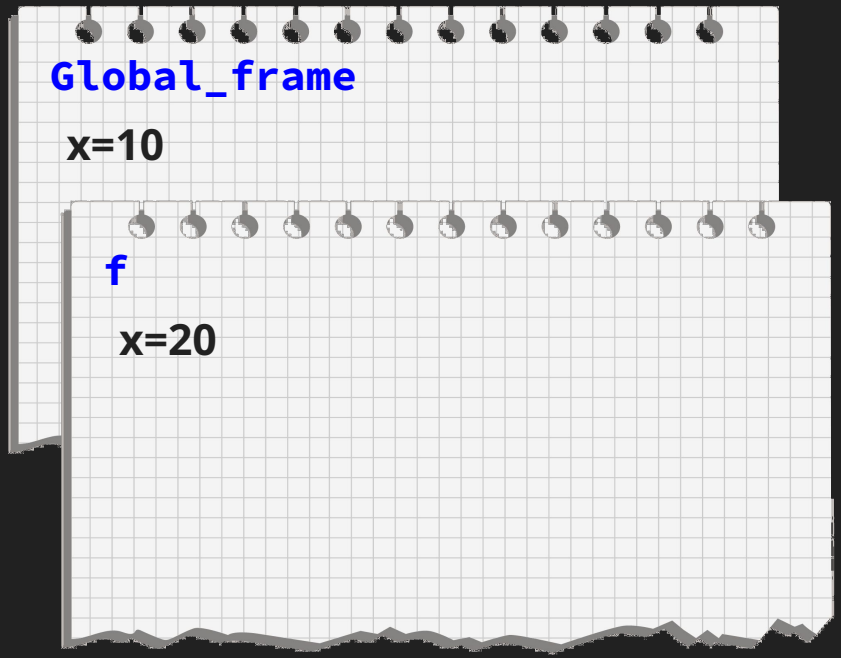
Scope globale vs. Scope locale

```
1  let x = 10;
2
3  function f(){
4    let x = 20;
5    console.log("In f x vale", x);
6  }
7
8  function g(){
9    console.log("In g x vale", x);
10 }
11
12 f();
13 g();
```

```
In f x vale 20
In g x vale 10
```

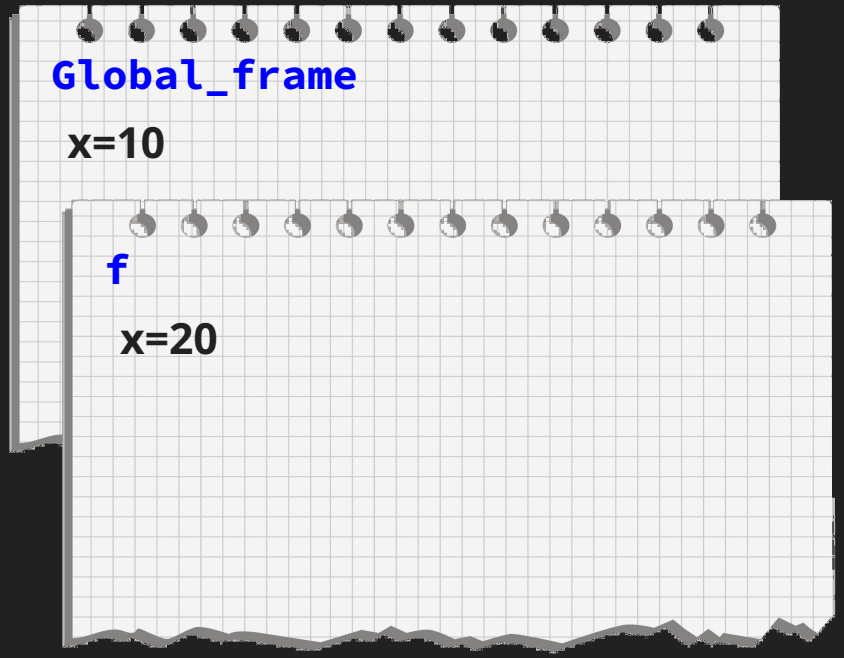
Idea

Ambienti come fogli di carta sovrapposti



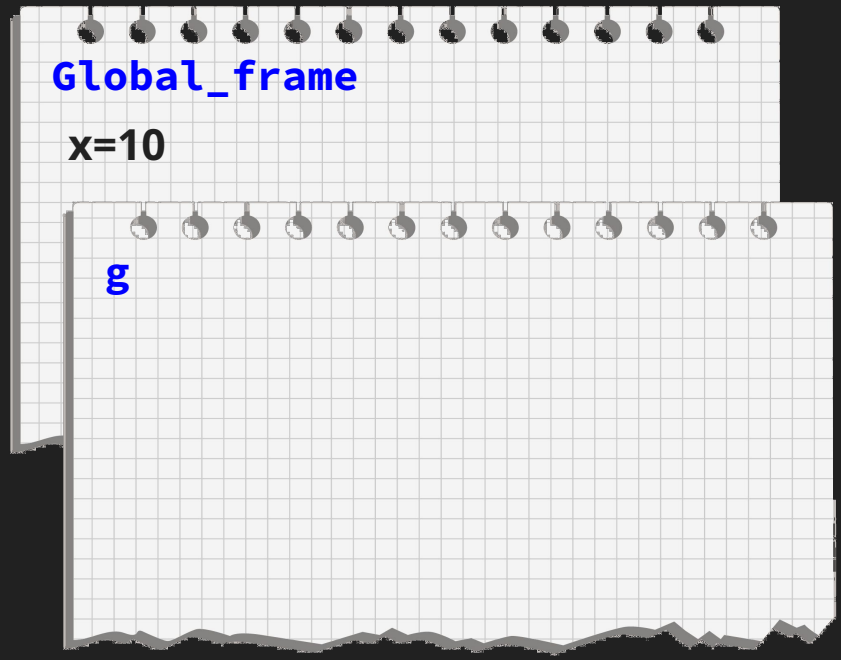
Ambienti come fogli di carta sovrapposti

```
1  let x = 10;
2
3  function f(){
4    let x = 20;
5
6    console.log("In f x vale", x);
7  }
8
9  console.log("x vale", x);
10 f();
```



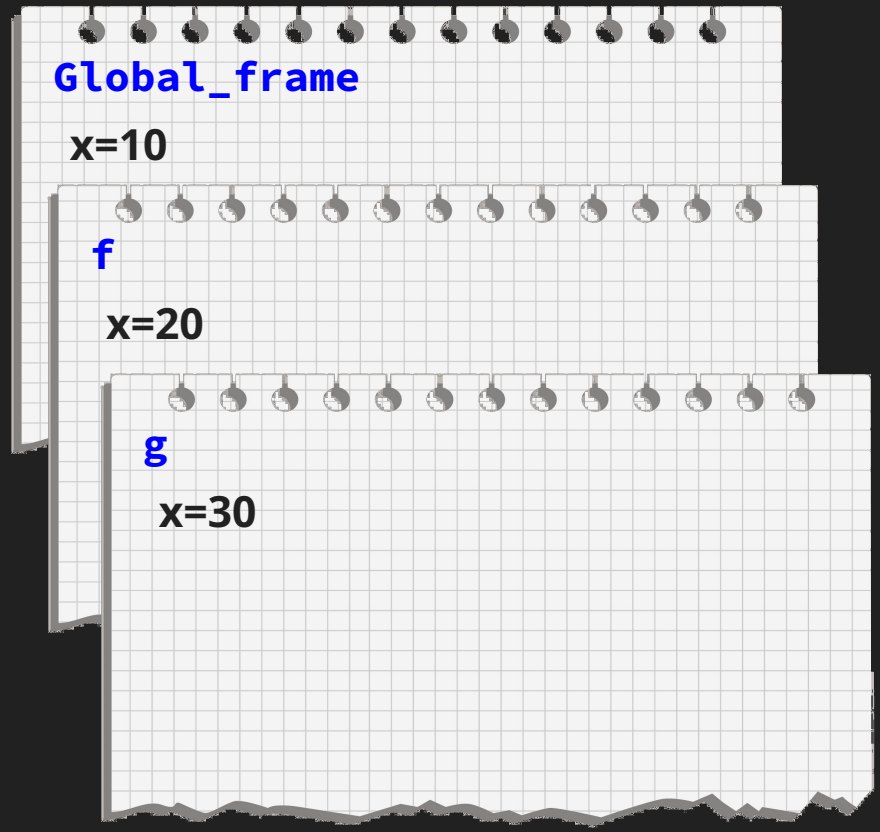
Ambienti come fogli di carta sovrapposti

```
1  let x = 10;
2
3  function f(){
4    let x = 20;
5    console.log("In f x vale", x);
6  }
7
8  function g(){
9    console.log("In g x vale", x);
10 }
11
12 f();
13 g();
```



Blocchi annidati

```
1  let x = 10;
2  function f(){
3    let x = 20;
4
5    function g(){
6      let x = 30;
7    }
8  }
```

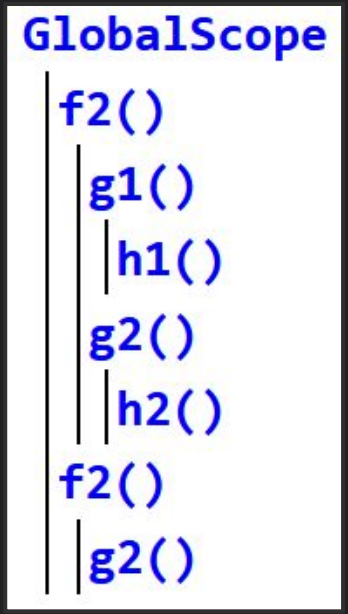


Come si impilano i fogli?

Seguiamo un set di regole

Scope statico

- Partiamo da un ambiente globale
- Definiamo una catena statica
- Quando una funzione viene chiamata il suo ambiente viene sovrapposto all’ambiente del genitore nella catena statica

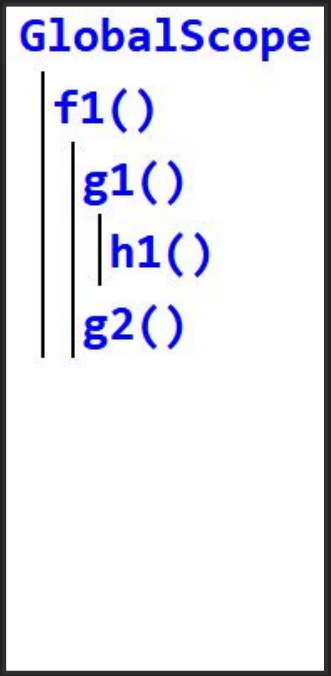


La catena statica

Per definire la catena statica osserviamo l'annidamento delle funzioni nel codice sorgente.

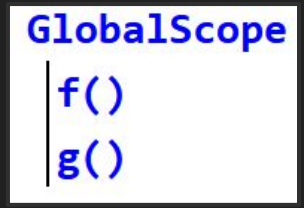
Questa viene definita a tempo di compilazione, per questo si chiama statica.

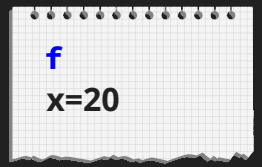
```
1  function f1(){
2      function g1(){
3          function h1(){
4          }
5      }
6      function g2(){
7      }
8  }
9
```



Esempio di scope statico

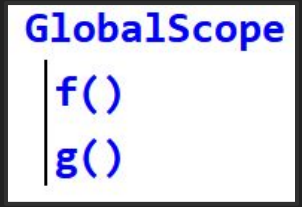
```
1  let x = 10;
2
3  function f(){
4    let x = 20;
5    console.log("In f x vale", x);
6    return;
7  }
8
9  function g(){
10   console.log("In g x vale", x);
11   return;
12 }
13
14 f();
15 g();
```

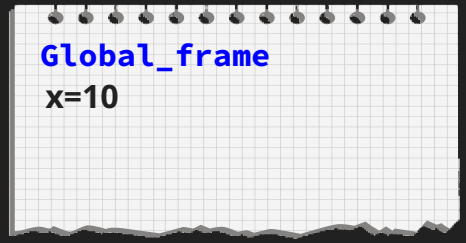




Esempio di scope statico

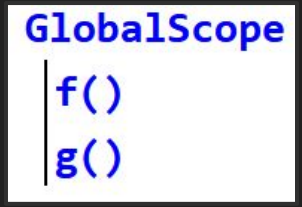
```
1  let x = 10;
2
3  function f(){
4    let x = 20;
5    console.log("In f x vale", x);
6    return;
7  }
8
9  function g(){
10   console.log("In g x vale", x);
11   return;
12 }
13
14 f();
15 g();
```

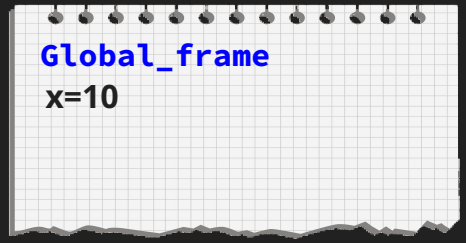




Esempio di scope statico

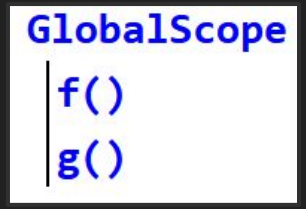
```
1  let x = 10;
2
3  function f(){
4    let x = 20;
5    console.log("In f x vale", x);
6    return;
7  }
8
9  function g(){
10   console.log("In g x vale", x);
11   return;
12 }
13
14 f();
15 g();
```

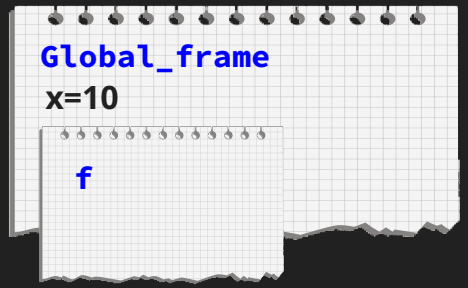




Esempio di scope statico

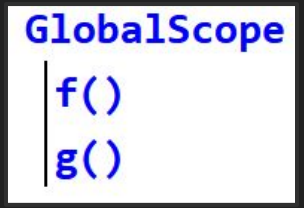
```
1  let x = 10;
2
3  function f(){
4    let x = 20;
5    console.log("In f x vale", x);
6    return;
7  }
8
9  function g(){
10   console.log("In g x vale", x);
11   return;
12 }
13
14 f();
15 g();
```

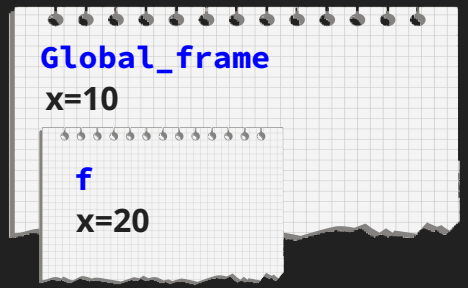




Esempio di scope statico

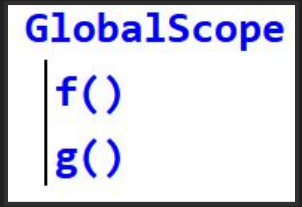
```
1  let x = 10;
2
3  function f(){
4    let x = 20;
5    console.log("In f x vale", x);
6    return;
7  }
8
9  function g(){
10   console.log("In g x vale", x);
11   return;
12 }
13
14 f();
15 g();
```

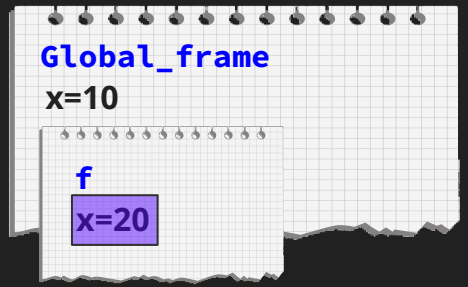




Esempio di scope statico

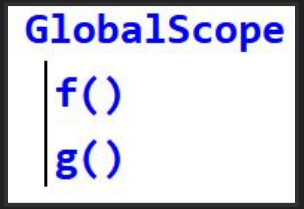
```
1  let x = 10;
2
3  function f(){
4    let x = 20;
5    console.log("In f x vale", x);
6    return;
7  }
8
9  function g(){
10   console.log("In g x vale", x);
11   return;
12 }
13
14 f();
15 g();
```

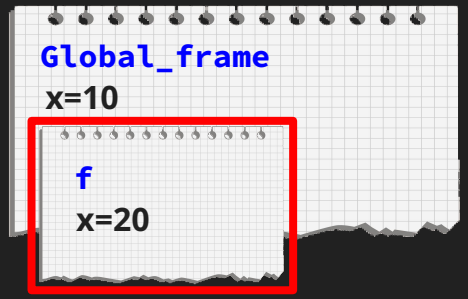




Esempio di scope statico

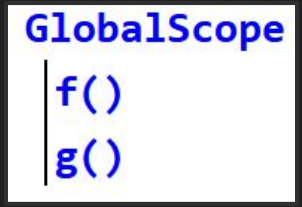
```
1  let x = 10;
2
3  function f(){
4    let x = 20;
5    console.log("In f x vale", x);
6    return;
7  }
8
9  function g(){
10   console.log("In g x vale", x);
11   return;
12 }
13
14 f();
15 g();
```

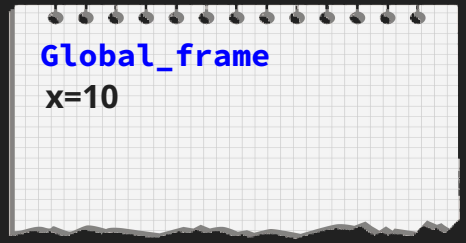




Esempio di scope statico

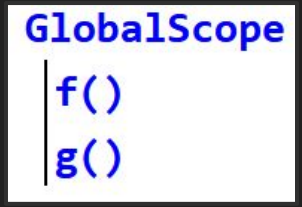
```
1  let x = 10;
2
3  function f(){
4    let x = 20;
5    console.log("In f x vale", x);
6    return;
7  }
8
9  function g(){
10   console.log("In g x vale", x);
11   return;
12 }
13
14 f();
15 g();
```

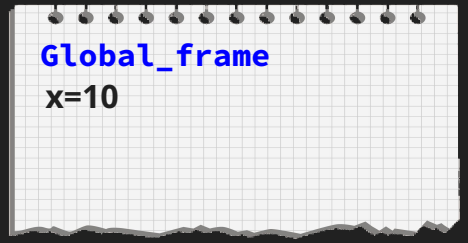




Esempio di scope statico

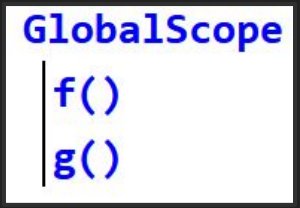
```
1  let x = 10;
2
3  function f(){
4    let x = 20;
5    console.log("In f x vale", x);
6    return;
7  }
8
9  function g(){
10   console.log("In g x vale", x);
11   return;
12 }
13
14 f();
15 g();
```

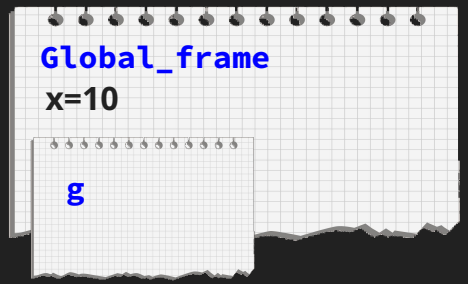




Esempio di scope statico

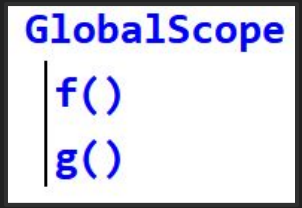
```
1  let x = 10;
2
3  function f(){
4    let x = 20;
5    console.log("In f x vale", x);
6    return;
7  }
8
9  function g(){
10   console.log("In g x vale", x);
11   return;
12 }
13
14 f();
15 g();
```

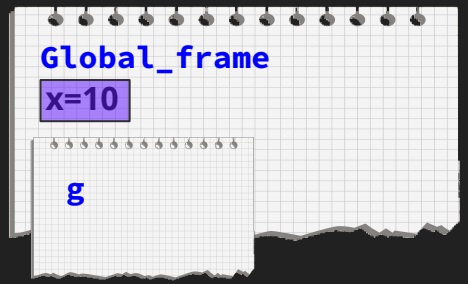




Esempio di scope statico

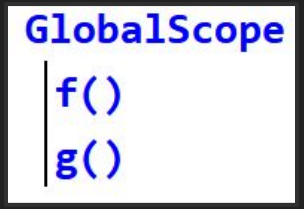
```
1  let x = 10;
2
3  function f(){
4    let x = 20;
5    console.log("In f x vale", x);
6    return;
7  }
8
9  function g(){
10   console.log("In g x vale", x);
11   return;
12 }
13
14 f();
15 g();
```

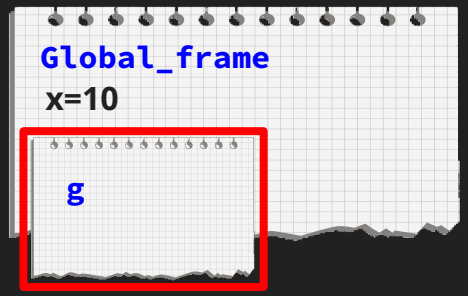




Esempio di scope statico

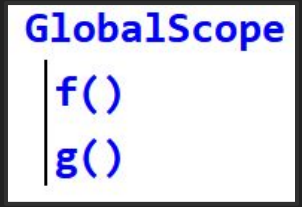
```
1  let x = 10;
2
3  function f(){
4    let x = 20;
5    console.log("In f x vale", x);
6    return;
7  }
8
9  function g(){
10   console.log("In g x vale", x);
11   return;
12 }
13
14 f();
15 g();
```

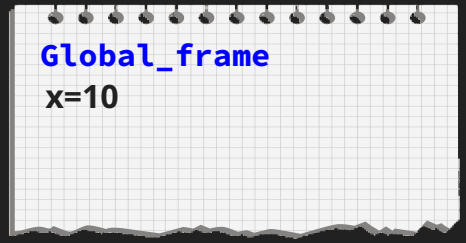




Esempio di scope statico

```
1  let x = 10;
2
3  function f(){
4    let x = 20;
5    console.log("In f x vale", x);
6    return;
7  }
8
9  function g(){
10   console.log("In g x vale", x);
11   return;
12 }
13
14 f();
15 g();
```





Ora tocca a voi!

```
let saldo = 1000;
let tassoInteresse = 0.02;

function calcolaInteressi() {
    let saldo = 500;
    let mesi = 12;
    console.log("A) Saldo in calcolaInteressi:", saldo);
    console.log("B) Tasso in calcolaInteressi:", tassoInteresse);

    function applicaBonus() {
        let bonus = 50;
        let tassoInteresse = 0.05;
        console.log("C) Saldo in applicaBonus:", saldo);
        console.log("D) Tasso in applicaBonus:", tassoInteresse);
        console.log("E) Calcolo interesse:", saldo * tassoInteresse);
        return saldo + bonus;
    }

    let risultato = applicaBonus();
    console.log("F) Risultato bonus:", risultato);
    console.log("G) Saldo dopo bonus:", saldo);
    return;
}
```

```
function verificaCredito() {
    let commissioni = 10;
    console.log("H) Saldo in verificaCredito:", saldo);
    console.log("I) Saldo - commissioni:", saldo - commissioni);

    function controllaLimite() {
        let limite = 100;
        console.log("J) Controllo saldo > limite:", saldo > limite);
        return;
    }

    controllaLimite();
    return;
}

console.log("1) Saldo iniziale:", saldo);
calcolaInteressi();
console.log("2) Saldo dopo calcolaInteressi:", saldo);
verificaCredito();
```

- Definite la catena statica
- Create i fogli per ogni funzione
- Provate ad eseguire il codice
- Cosa stampa?

Verifichiamo i risultati

```
let saldo = 1000;
let tassoInteresse = 0.02;

function calcolaInteressi() {
    let saldo = 500;
    let mesi = 12;
    console.log("A) Saldo in calcolaInteressi:", saldo);
    console.log("B) Tasso in calcolaInteressi:", tassoInteresse);

    function applicaBonus() {
        let bonus = 50;
        let tassoInteresse = 0.05;
        console.log("C) Saldo in applicaBonus:", saldo);
        console.log("D) Tasso in applicaBonus:", tassoInteresse);
        console.log("E) Calcolo interesse:", saldo * tassoInteresse);
        return saldo + bonus;
    }

    let risultato = applicaBonus();
    console.log("F) Risultato bonus:", risultato);
    console.log("G) Saldo dopo bonus:", saldo);
    return;
}
```

```
function verificaCredito() {
    let commissioni = 10;
    console.log("H) Saldo in verificaCredito:", saldo);
    console.log("I) Saldo - commissioni:", saldo - commissioni);

    function controllaLimite() {
        let limite = 100;
        console.log("J) Controllo saldo > limite:", saldo > limite);
        return;
    }

    controllaLimite();
    return;
}

console.log("1) Saldo iniziale:", saldo);
calcolaInteressi();
console.log("2) Saldo dopo calcolaInteressi:", saldo);
verificaCredito();
```

https://sij82.github.io/didattica/lezione_1/esempio04/

Prossima lezione

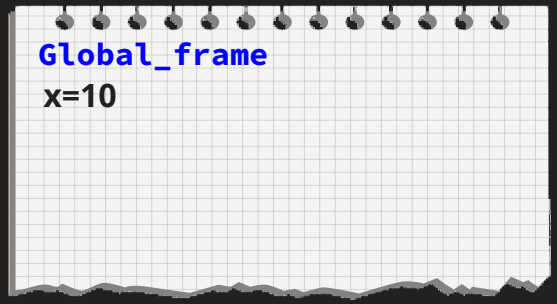
scope dinamico

Scope dinamico

- Partiamo da un ambiente globale
- Quando una funzione viene chiamata il suo ambiente viene sovrapposto all’ambiente della funzione chiamante

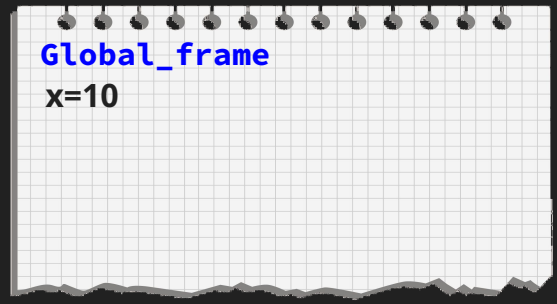
Esempio di scope dinamico

```
1  let x = 10;
2  function f(){
3    console.log("Scope di f");
4    let x = 20;
5    g();
6    return;
7  }
8
9  function g(){
10   console.log("In g x vale", x);
11   return;
12 }
13
14 console.log("Main scope");
15 g();
16 f();
```



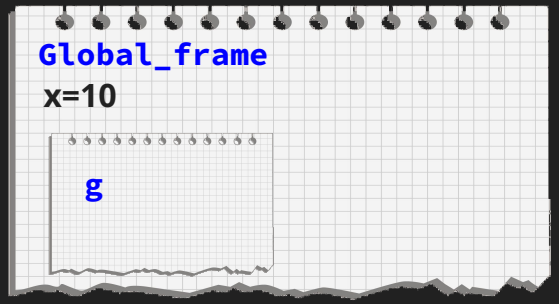
Esempio di scope dinamico

```
1  let x = 10;
2  function f(){
3    console.log("Scope di f");
4    let x = 20;
5    g();
6    return;
7  }
8
9  function g(){
10   console.log("In g x vale", x);
11   return;
12 }
13
14 console.log("Main scope");
15 g();
16 f();
```



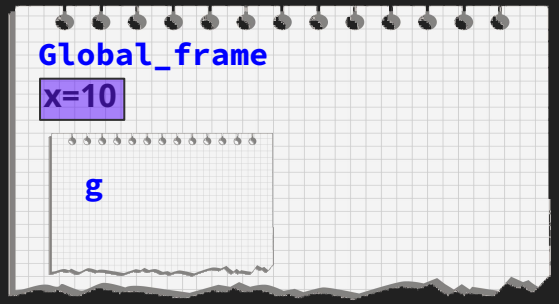
Esempio di scope dinamico

```
1  let x = 10;
2  function f(){
3    console.log("Scope di f");
4    let x = 20;
5    g();
6    return;
7  }
8
9  function g(){
10   console.log("In g x vale", x);
11   return;
12 }
13
14 console.log("Main scope");
15 g();
16 f();
```



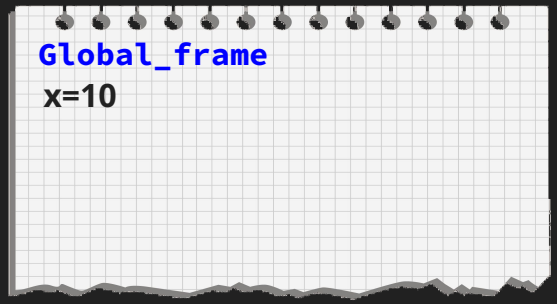
Esempio di scope dinamico

```
1  let x = 10;
2  function f(){
3    console.log("Scope di f");
4    let x = 20;
5    g();
6    return;
7  }
8
9  function g(){
10   console.log("In g x vale", x);
11   return;
12 }
13
14 console.log("Main scope");
15 g();
16 f();
```



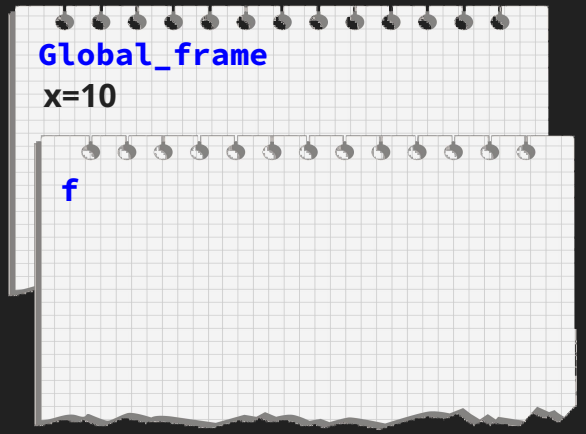
Esempio di scope dinamico

```
1  let x = 10;
2  function f(){
3    console.log("Scope di f");
4    let x = 20;
5    g();
6    return;
7  }
8
9  function g(){
10   console.log("In g x vale", x);
11   return;
12 }
13
14 console.log("Main scope");
15 g();
16 f();
```



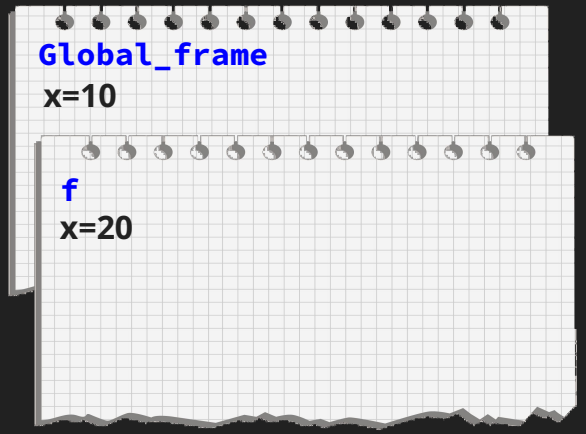
Esempio di scope dinamico

```
1  let x = 10;
2  function f(){
3    console.log("Scope di f");
4    let x = 20;
5    g();
6    return;
7  }
8
9  function g(){
10   console.log("In g x vale", x);
11   return;
12 }
13
14 console.log("Main scope");
15 g();
16 f();
```



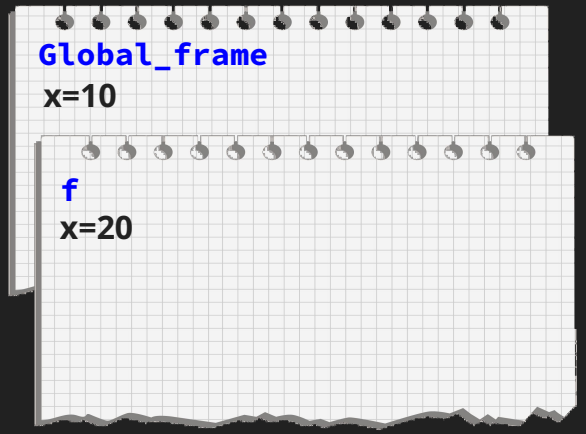
Esempio di scope dinamico

```
1  let x = 10;
2  function f(){
3    console.log("Scope di f");
4    let x = 20;
5    g();
6    return;
7  }
8
9  function g(){
10   console.log("In g x vale", x);
11   return;
12 }
13
14 console.log("Main scope");
15 g();
16 f();
```



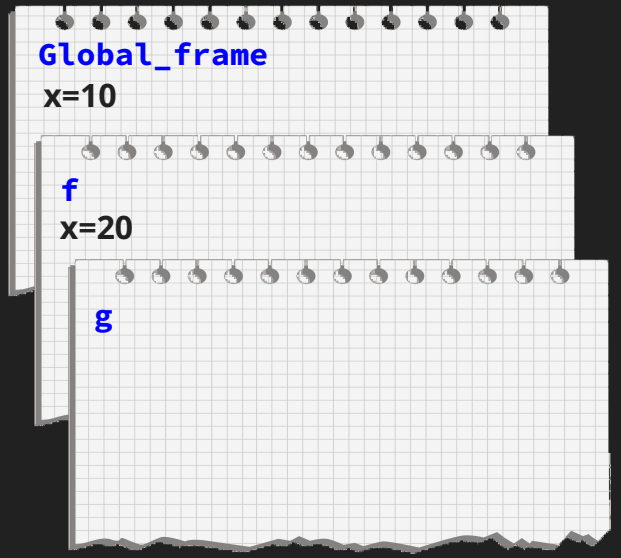
Esempio di scope dinamico

```
1  let x = 10;
2  function f(){
3    console.log("Scope di f");
4    let x = 20;
5    g();
6    return;
7  }
8
9  function g(){
10   console.log("In g x vale", x);
11   return;
12 }
13
14 console.log("Main scope");
15 g();
16 f();
```



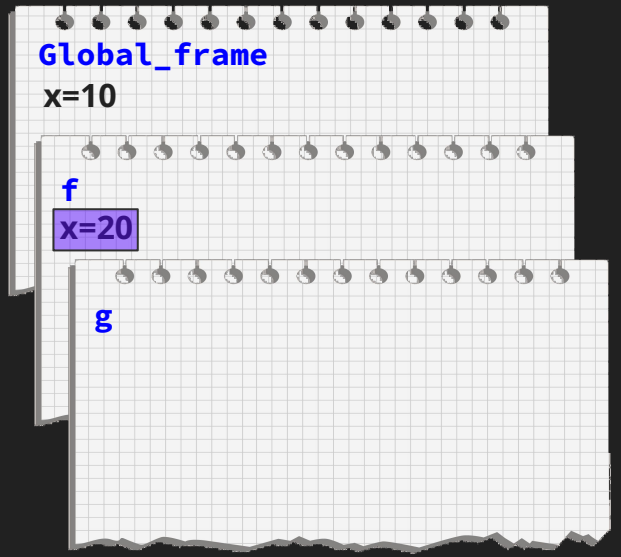
Esempio di scope dinamico

```
1  let x = 10;
2  function f(){
3    console.log("Scope di f");
4    let x = 20;
5    g();
6    return;
7  }
8
9  function g(){
10   console.log("In g x vale", x);
11   return;
12 }
13
14 console.log("Main scope");
15 g();
16 f();
```



Esempio di scope dinamico

```
1  let x = 10;
2  function f(){
3    console.log("Scope di f");
4    let x = 20;
5    g();
6    return;
7  }
8
9  function g(){
10   console.log("In g x vale", x);
11   return;
12 }
13
14 console.log("Main scope");
15 g();
16 f();
```



Ora tocca a voi!

```
let iva = 0.0;

function pagamento( prezzo ){
  let totale = prezzo + (prezzo * iva);
  return totale;
}

function italia( prezzo ){
  let iva = 0.22;
  return pagamento( prezzo );
}

function germania( prezzo ){
  let iva = 0.19;
  return pagamento( prezzo );
}

console.log( italia(100) );
console.log( germania(100) );
```

- Provate ad eseguire il codice
- Cosa stampa con scope statico?
- Cosa stampa con scope dinamico?

https://sij82.github.io/didattica/lezione_1/esempio06/

Confronto tra scope statico e scope dinamico

Scope statico

- Le associazioni sono note a tempo di compilazione
- Facile capire il contesto dal codice
- Errori rilevabili prima dell’esecuzione
- Più difficile da implementare

Scope dinamico

- Le associazioni sono derivate durante l’esecuzione del programma
- Più difficile capire il contesto dal codice
- Più flessibile, può adattarsi al contesto
- Più facile da implementare

Perché usare lo scope statico?

- Di uso più comune rispetto allo scope dinamico : C, Python, Java, …
- Codice prevedibile e leggibile
- Più sicuro e robusto (isolamento di funzioni)
- Rilevamento di errori prima dell’esecuzione
- Ottimizzazioni del compilatore

Ideale per : progetti con team numerosi, software mission-critical, librerie e framework riutilizzabili

Perché usare lo scope dinamico?

- Parametri derivati dall’ambiente : bash, Perl, Lisp, ...
- Flessibilità, le funzioni si adattano automaticamente al contesto senza bisogno di passaggio di parametri extra
- Meno codice boilerplate
- Debug interattivo, testing in contesti diversi senza dover modificare il codice

Ideale per : scripting e automazioni, linguaggi di configurazione, ambienti interattivi