# Fondamenti di programmazione

Adriano Venturini

Project Manager

Software Solutions Developer

# Presentazione

- Lezione del 12 novembre 2024
  - *Introduzione alla Programmazione e JavaScript (quarta parte)*
  - *Esercizi*

# Introduzione alla programmazione

**(quarta parte)**

**Contenuti:**

- Fondamenti di CSS
  - CSS e Selettori di Base
  - Box Model e Layout di Base
  - Posizionamento e Stili Avanzati
  - Flexbox e Responsive Design
- Fondamenti di JavaScript
  - Il teorema di Bohom-Jacopini applicato a JavaScript
  - Pseudocodice
  - Che cos’è un blocco di codice in JavaScript
  - Condizioni e Strutture di Controllo
  - Cicli e Iterazioni
- Esercizi

## Introduzione ai Selettori CSS

Un selettore in CSS è una parte di codice che identifica gli elementi HTML ai quali applicare una regola di stile.

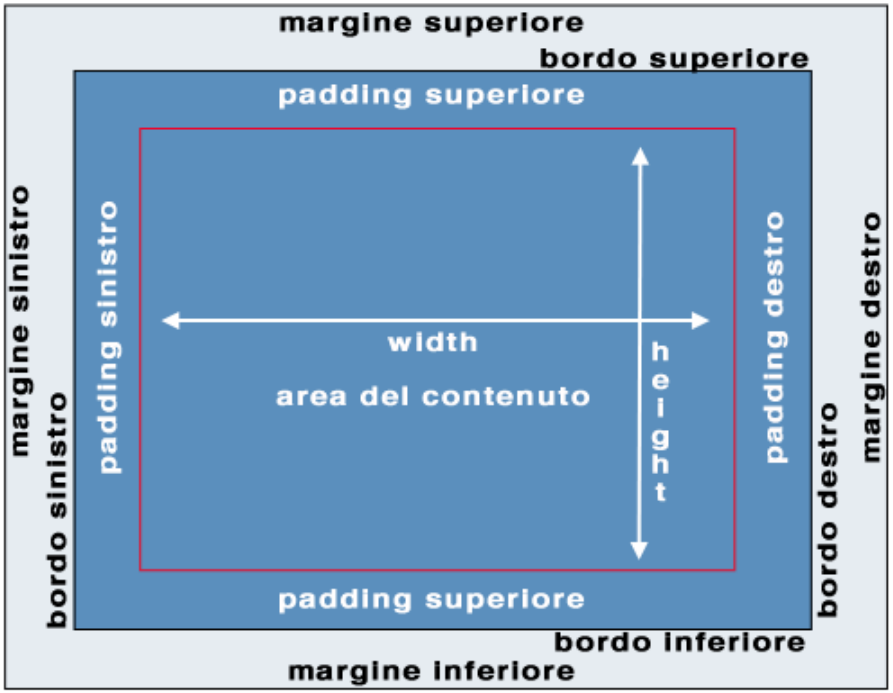
- Selettori di Tipo
- Selettori di Classe
- Selettori ID
- Selettori Universali
- Selettori Attributo
- Selettori Discendenti
- Selettori Figlio Diretto
- Selettori di Gruppo

**NOTA:** il testo completo e gli esercizi si trovano nel file: CSS_selettori_di_base.pdf

## Box Model e Layout di Base

In HTML e CSS, il **Box Model** e i **Layout di Base** sono due concetti fondamentali per la costruzione delle interfacce.

Il Box Model in CSS:



**Box Model e Layout di Base**

- Proprietà del Box Model
- I Layout di Base
  - Layout con display: block e display: inline
  - Layout Flexbox
  - Layout Grid

**NOTA:** la spiegazione ed esempi si trovano nel file:
**Box_Model_Base_01.pdf**

**Posizionamento Avanzato in CSS3**

- Posizionamento Statico, Relativo, Assoluto e Fisso
- CSS Grid
- CSS Flexbox

**Stili Avanzati in CSS3**

- Gradienti
- Ombre (Box Shadow e Text Shadow)
- Transizioni
- Animazioni
- Pseudo-Elementi e Pseudo-Classi
- Variabili CSS

**NOTA:** la spiegazione completa e gli esempi si trovano nel file: Stili_Avanzati_CSS3.pdf

**Flexbox** e il **Responsive Design**

- **Flexbox, o Flexible Box Layout**, è un sistema di layout CSS che facilita l'allineamento e la distribuzione degli elementi all'interno di un contenitore, anche quando le dimensioni degli elementi non sono note o variano. Flexbox è pensato per gestire layout unidimensionali.
- Il **Responsive Design** consiste nella creazione di layout che si adattano automaticamente alle dimensioni dello schermo su cui vengono visualizzati

**NOTA:** la spiegazione completa e gli esempi si trovano nel file: Flexbox_Responsive_Design.pdf

## Il Teorema di Böhm-Jacopini

### Enunciato del Teorema di Böhm-Jacopini

Il teorema afferma che:

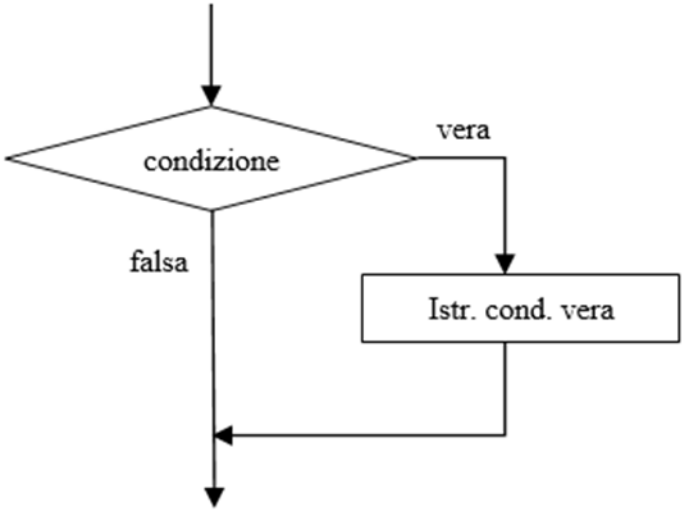
- Qualsiasi algoritmo può essere espresso senza l'uso di **goto** e salti arbitrari, utilizzando esclusivamente tre strutture di controllo fondamentali: **sequenza**, **selezione** e **iterazione**.

**Strutture di sequenza**

La sequenza è la struttura più semplice ed indica una sequenza di operazioni.

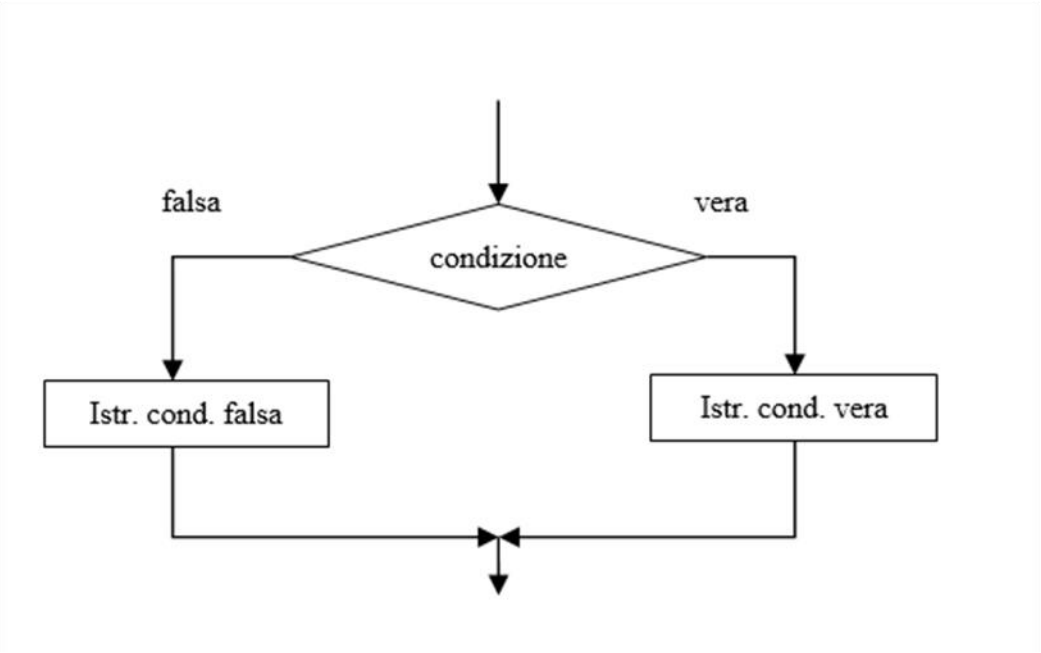
## Strutture di selezione

La selezione è una struttura di controllo che indica all'elaboratore quale fra due sequenze eseguire.



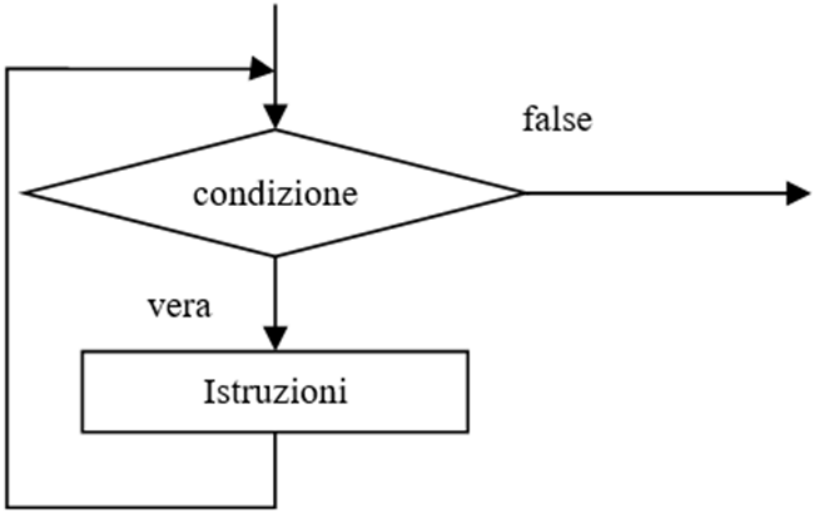
## Struttura selezione a due vie

Se la condizione è vera esegue il blocco di istruzioni *Istr. cond. vera*
se è falsa esegue il blocco di istruzioni *Istr. cond. falsa*.



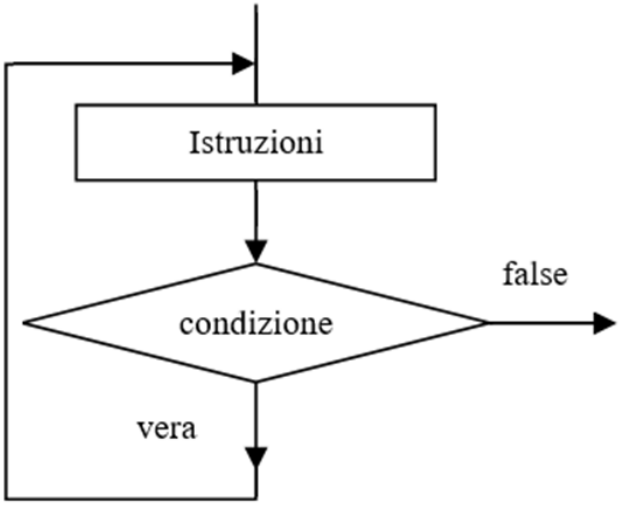
## Struttura di iterazione

Il blocco di istruzione è ripetuto fino a quanto la condizione è vera,
il ciclo si interrompe quando la condizione è falsa.
La forma **pre-condizionale** prevede la condizione in testa, prima del blocco di istruzioni:



## Struttura di iterazione

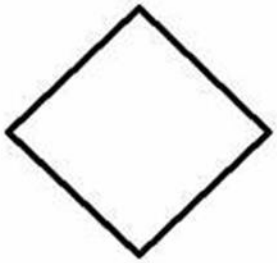
Il blocco di istruzione è ripetuto fino a quanto la condizione è vera,
il ciclo si interrompe quando la condizione è falsa.
La struttura di iterazione **post-condizionale** prevede la condizione dopo il blocco di istruzioni:



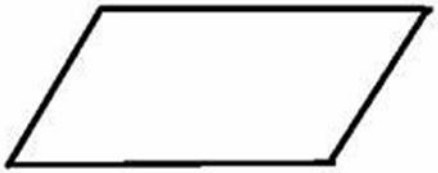
# Introduzione alla programmazione

Le forme OVALI indicano l'inizio e la fine del diagramma.

I RETTANGOLI contengono le azioni da compiere.



Nei ROMBI ci sono delle domande. La risposta può essere sì/no oppure vero/falso.



Talvolta ci sono anche dei PARALLELOGRAMMI. Essi contengono i dati che inseriamo (input) o che riceviamo (output).

**Strutture di controllo fondamentali del Teorema di Böhm-Jacopini**

**Sequenza**

- La **sequenza** le istruzioni vengono eseguite una dopo l'altra, in ordine, esattamente come sono scritte nel codice.

**Selezione** (o Struttura Condizionale)

- La **selezione** permette di prendere decisioni all'interno del programma, eseguendo un blocco di codice solo se una certa condizione risulta vera. Questo introduce la possibilità di percorsi alternativi all'interno del flusso di esecuzione.

**Iterazione** (o Ciclo)

- L'**iterazione** permette di eseguire una serie di istruzioni ripetutamente fino a quando una certa condizione non è più soddisfatta. Esistono diversi tipi di cicli:
  - **Ciclo while**: ripete un blocco di codice finché la condizione rimane vera.
  - **Ciclo for**: esegue un blocco di codice un numero predeterminato di volte.
  - **Ciclo do-while**: esegue il blocco almeno una volta, e poi verifica la condizione per determinare se continuare.

**Implicazioni del Teorema di Böhm-Jacopini**

- Prima della formulazione di questo teorema, la programmazione faceva largo uso del comando **goto**, che permetteva di saltare arbitrariamente da un punto all'altro del programma.
- Con l'introduzione del teorema di Böhm-Jacopini, si è dimostrato che l'uso di strutture come il **goto** non è necessario: qualsiasi programma può essere implementato utilizzando soltanto le tre strutture di controllo indicate.
- Questo ha portato alla nascita della **programmazione strutturata**, una metodologia che privilegia la scrittura di codice leggibile e organizzato.

**Applicazione del Teorema nella Programmazione**

- Nella pratica della programmazione moderna, il teorema di Böhm-Jacopini è il fondamento delle tecniche di **programmazione strutturata**.

**Eliminazione del comando Goto**

- Uno degli effetti più importanti del teorema è la **sostituzione del comando goto**, che permette salti arbitrari nel codice. Questo comando era molto comune nei primi linguaggi di programmazione, come l'Assembly e il Fortran, ma Böhm e Jacopini dimostrarono che il suo utilizzo poteva essere completamente evitato.
- L’uso di goto è fortemente scoraggiato, ma tecnicamente ancora possibile.

**Il Concetto di Blocco di Codice nel Teorema di Böhm-Jacopini**

Il **blocco di codice** è un concetto fondamentale nella programmazione strutturata.

- Un **blocco di codice** è una porzione di codice che raggruppa più istruzioni, trattandole come un'unità logica, spesso delimitata da simboli come le parentesi graffe {} (in linguaggi come C#, Java, ecc.).

**Punti di Ingresso e Uscita: Principio Fondamentale**

- Uno dei concetti chiave derivati dal teorema di Böhm-Jacopini è che **ogni blocco di codice deve avere un unico punto di ingresso e un unico punto di uscita**.
- Questo principio è alla base della **programmazione strutturata** ed è essenziale per garantire che il flusso di esecuzione del programma sia prevedibile, chiaro e facile da seguire.

**Perché un unico punto di ingresso e di uscita è cruciale?**

- Il principio di **un unico punto di ingresso e di uscita** si ricollega direttamente all'obiettivo di ridurre il caos e l'ambiguità nel flusso di controllo dei programmi, una delle problematiche che il teorema di Böhm-Jacopini si propone di risolvere.

**Un Esempio Pratico: Evitare Goto e Uso di Strutture di Controllo**

- In passato, i programmatori facevano largo uso del comando goto per saltare arbitrariamente da una parte del codice a un'altra. Il risultato era spesso un codice caotico con numerosi punti di ingresso e uscita all'interno di blocchi di codice, difficile da comprendere e mantenere.

Nota: per una descrizione completa ed esempi, vedi file:

teorema_di_bohom-jacopini.pdf

Grazie per l’attenzione

Adriano Venturini