Esercitazione 1 - 28/09

Argomenti:

- editor e interprete
- input e output (`console.log` e `prompt`)
- tipi
- variabili e costanti
- operatori aritmetici

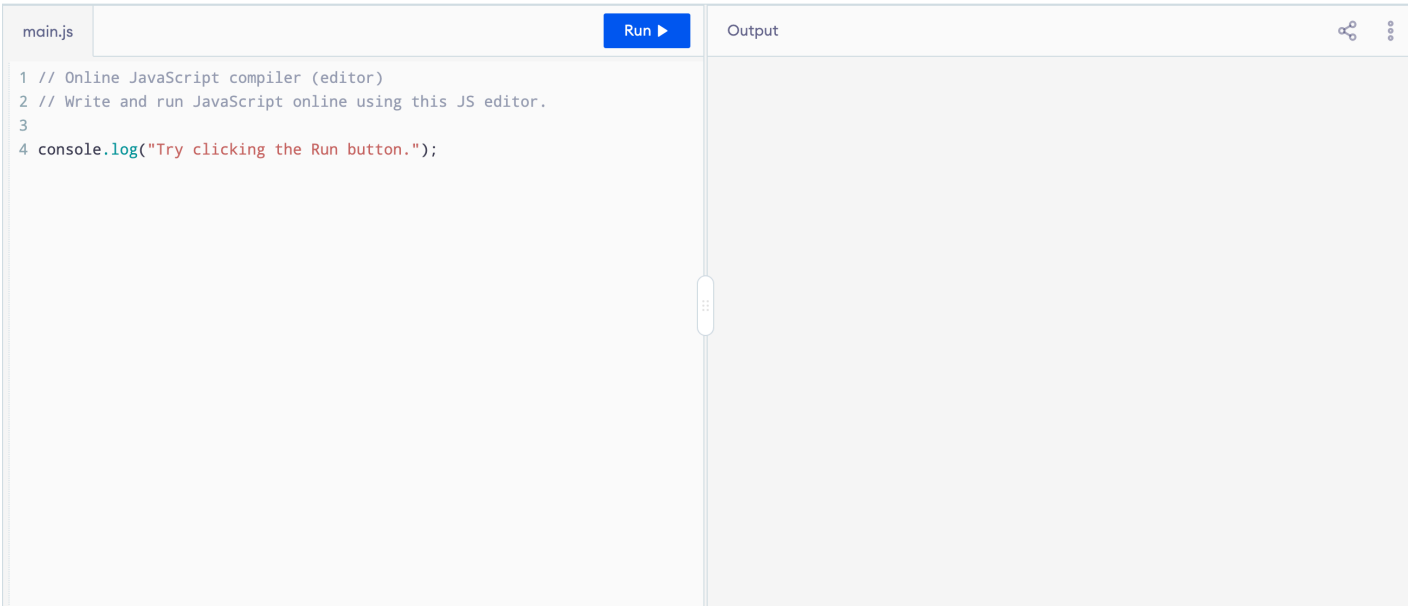
Indice degli esercizi

Esercizio	Argomento
Esercizio 1	Tipi di dato
Esercizio 2.1	Input e output
Esercizio 2.2	Stringhe
Esercizio 3	Variabili
Esercizio 4	Operatori aritmetici
Esercizio 5	Operatori aritmetici
Esercizio 6	Casting
Esercizio 7	Operatori aritmetici
Esercizio 8	Operatori aritmetici
Esercizio 9	Casting
Esercizio 10	Funzioni matematiche
Esercizio 11	Operatori aritmetici

Regole anche per le prossime volte:

- **provare** prima di rinunciare
- **commenti** dove il codice non parla da solo
- prima di gridare vittoria, eseguite **sempre** il programma con più valori, compresi i casi "strani" (0, numeri negativi, decimali).

Programiz: i tre componenti



La pagina è fatta di **tre componenti**, ognuno con un compito preciso:

Componente	Dove	Cosa fa
Editor	pannello di sinistra	qui **scrivete** il programma (testo, come in un blocco note)
Interprete	si attiva col pulsante **Run**	**legge** il programma e lo **esegue** istruzione per istruzione
Console	pannello di destra	qui **compaiono** i risultati: output ed errori

Il programma non è "fatto" di nessuno dei tre: è solo **testo**.

L'editor serve a scrivere il codice sorgente, l'interprete lo fa vivere, la console mostra cosa è successo.

Programiz: cosa succede premendo Run?

1. **scrivete** il codice nell'editor;
2. **premete Run**: è la "chiamata" all'interprete, che riceve tutto il testo dell'editor;
3. l'interprete esegue le istruzioni **dall'alto verso il basso**, una alla volta;
4. tutto ciò che il programma **stampa** compare nella console; se c'è un errore, l'interprete **si ferma** e lo scrive lì.

Ogni **Run** riparte da zero: nulla si conserva da una esecuzione all'altra.

Programiz: il primo programma

Scrivete questo nell'**editor** e premete **Run**:

```
console.log("Ciao, mondo!");
```

Nella **console** compare:

```
Ciao, mondo!
```

Attenzione al nome: `console.log(...)` è l'istruzione con cui il **programma** dice all'interprete "scrivi questo nella console". Non è la console in sé, ma il modo di usarla.

Provate ora a **rompere** il programma: togliete una parentesi e premete Run. Cosa compare nella console? L'interprete non riesce a leggere il testo e ve lo dice: leggete sempre il messaggio.

```
console.log("Ciao, mondo!";      // manca la parentesi chiusa
```

Nella console compare un messaggio di errore, simile a questo:

```
console.log("Ciao, mondo!";
            ^^^^^^^^^^^^^^

SyntaxError: missing ) after argument list
```

Come leggerlo:

- `SyntaxError` è il **tipo** di errore: l'interprete non ha capito come è scritto il programma (errore di *sintassi*);
- `missing ) after argument list` è la **spiegazione**: manca una `)` dopo gli argomenti di `console.log`;

- la riga col simbolo `^^^^` indica **dove** l'interprete si è accorto del problema (spesso è il punto *dopo* quello da correggere);
- in caso di errore di sintassi il programma **non parte** nemmeno: non compare nessun output.

Recap 1 · Tipi e valori

Ogni valore che Javascript può manipolare ha un **tipo**.

I quattro tipi fondamentali :

Tipo	Esempi	Note
`number`	`42` , `3.14` , `-7`	numeri interi e decimali
`string`	`"ciao"` , `"42"` , `""`	testo tra apici
`boolean`	`true` , `false`	valori che rappresentano condizioni di verità
`undefined`	`undefined`	"nessun valore assegnato"

```
console.log(42);       // number
console.log("42");     // string: le virgolette cambiano il tipo!
console.log(true);     // boolean
```

Recap 2 · Operatori aritmetici

Aritmetici: operano su valori di tipo `number` :

Operatore	Significato	Esempio
`+` `-`	somma, differenza	`7 - 2` → `5`
`*` `/`	prodotto, divisione **reale**	`7 / 2` → `3.5`
`%`	resto	`7 % 2` → `1`
`**`	potenza	`2 ** 3` → `8`

Precedenza: prima `**` , poi `*` `/` `%` , poi `+` `-` . Le parentesi comandano sempre (come a scuola).

Recap 2 · Operatori su `string`

Operatore	Significato	Esempio
`+`	concatenazione	`"cia"+"o"` → `"ciao"`

Attenzione! Il tipo decide **cosa fa l'operatore** +: `"3" + "4"` è `"34"`, ma `3 + 4` è `7`.

Possiamo convertire un tipo in un altro; Ad esempio, per convertire una stringa in numero: `Number("42")`. Se non è convertibile il risultato è `NaN` (*Not a Number*).

Recap 3 · Variabili

Una **variabile** è un nome a cui è associato un valore.

```
let eta = 23;            // dichiara e assegna
eta = eta + 1;
const PI_GRECO = 3.14;   // const: non si può riassegnare
```

- `let` per valori che cambiano, `const` per valori che restano fissi;
- `=` **non** significa "uguale a": significa "valuta l'espressione a destra, assegnala al nome a sinistra";

Recap 4 · Input e output

Output: `console.log(...)` stampa i suoi argomenti separati da uno spazio.

```
console.log("Somma:", 3 + 4);   // Somma: 7
```

Input: `prompt(<string>)` mostra una stringa e restituisce **sempre una stringa**.

```
let eta_input = prompt("Inserisci la tua età:");    // "20"  (stringa!)
let eta = Number(eta_input);                            // 20    (number)
console.log(eta);
```

- `prompt(...)` **ferma** l'esecuzione e chiede un valore all'utente;
- il valore digitato torna al programma come **stringa** e viene assegnato a `eta_input`;
- il valore di `eta_input` viene trasformato in intero e assegnato ad una seconda variabile `eta`
- `console.log(...)` lo stampa nella console.

In generale è bene tenere a mente uno schema generale: **leggo l'input → elaboro e calcolo → stampo**.

Esercizio 1

Scrivere un programma con tre istruzioni. La prima che visualizzi un numero, la seconda che visualizzi una stringa, la terza che visualizzi un valore booleano.

Note:

- Le istruzioni vanno terminate con ;
- Si aggiungano al programma tutti i commenti ritenuti necessari

Esercizio 2.1 · Ciao, X!

Scrivete un programma che legge il vostro nome dall'input e poi stampa la stringa `Ciao <nome>!` in output.

Ad esempio, se il nome è Giulia, stamperà la stringa `"Ciao Giulia!"`

Esercizio 2.2 · Ciao, X!

Modificare l'esercizio precedente leggendo anche il **cognome** e stampate `Ciao, Nome Cognome!`.

Esempio:

```
Nome: Ada
Cognome: Lovelace
Ciao, Ada Lovelace!
```

Esercizio 3 · Dati anagrafici

Scrivere un programma che visualizzi su 4 righe separate i propri dati anagrafici: nome, cognome, luogo di nascita e data di nascita.

L'output deve essere di questo tipo (con i vostri dati):

```
Nome: Mario
Cognome: Rossi
Luogo di nascita: Bologna
Data di nascita: 01/01/1990
```

Esercizio 4

Scrivere un programma che visualizzi il cubo del numero reale inserito in input dall'utente. Si aggiungano tutti i commenti ritenuti necessari.

Esercizio 5

Scrivere un programma che calcoli e visualizzi il resto che si ottiene dividendo il proprio numero di matricola per 2. Si aggiungano poi tutti i commenti ritenuti necessari.

Esercizio 6

Creare un programma JS che legge l'**età** dell'utente e stampa due righe:

```
Età: 20
Tra un anno avrai 21 anni
Un anno fa avevi 19 anni
```

Esercizio 7

Scrivere un programma che, leggendo il valore del raggio di un cerchio in input, calcoli e visualizzi il valore del perimetro e dell'area del cerchio. Si aggiungano al programma tutti i commenti ritenuti necessari.

Promemoria: dato il raggio `r`, perimetro = `2 · π · r`, area = `π · r²`.

Esercizio 8

Leggete una temperatura in gradi **Celsius** e stampate l'equivalente in **Fahrenheit** e in **Kelvin**.

Formule: `F = C · 9/5 + 32` e `K = C + 273.15`.

Esempio con `23`:

```
23 °C = 73.4 °F = 296.15 K
```

Esercizio 9

Leggete un **nome** e un **anno di nascita** dall'input, poi stampate una frase come:

```
Ciao Anna, potresti avere 20 o 21 anni
```

(Non conosciamo la data di nascita completa, quindi la differenza tra anni vale solo se il compleanno è già passato; altrimenti l'età è di uno in meno.)

Esercizio 10

Creare un programma JS che estragga 3 numeri naturali casuali compresi tra 1 e 100 e li mostri a schermo.

Esercizio 11

Scrivere un programma che stampi il valore delle unità, decine e centinaia che compongono un numero naturale a tre cifre (scelto a piacere in input). Si utilizzino gli operatori visti nel Capitolo 1. Si aggiungano al programma tutti i commenti ritenuti necessari.

Suggerimento:
Si parta da questa considerazione per poi procedere.
Se si considera ad esempio il numero 234, il numero di unità (cioè 4) si può ottenere dividendo il numero per 10 e considerando il resto (infatti, 234 diviso 10 fa 23 con resto 4).

Errori visti a lezione · 1. Chiamare o assegnare?

```
console.log=("risultato: ");      // SBAGLIATO
```

Quel `=` è un **assegnamento**: stiamo dicendo "`console.log` ora vale la stringa `"risultato: "`". Non stampa niente, e **non dà errore subito**.

Il problema si vede alla prossima istruzione che usa `console.log`:

```
TypeError: console.log is not a function
```

Per **chiamare** una funzione si scrivono le parentesi subito dopo il nome, senza `=`:

```
console.log("risultato: ");       // GIUSTO
```

Errori visti a lezione · 2. Cosa va dentro `prompt`?

```
anni = prompt("22");              // SBAGLIATO
```

La stringa dentro `prompt(...)` è il **messaggio mostrato all'utente**, non la risposta. Il valore (`22`) lo digita chi usa il programma, e `prompt` lo restituisce.

```
let anni_input = prompt("Quanti anni hai?");   // mostra la domanda, restituisce la risposta
(stringa)
let anni = Number(anni_input);                 // convertiamo: anni è un number
```

Notate anche `let` : senza, `anni` non è dichiarata. Dichiarate sempre le variabili.

Errori visti a lezione · 3. Convertito, ma usato quello sbagliato

```
anni = prompt("inserisci anni");
anni_num = Number(anni);
console.log("L'anno prossimo avrai", anni+1, "anni");     // SBAGLIATO
```

Con input `22` stampa:

```
L'anno prossimo avrai 221 anni
```

La conversione è stata fatta, ma nel calcolo c'è ancora `anni` , che è una **stringa**: `"22" + 1` è la concatenazione `"221"` . Convertire in `anni_num` non cambia `anni` : sono due variabili diverse.

```
let anni = prompt("inserisci anni");
let anni_num = Number(anni);
console.log("L'anno prossimo avrai", anni_num + 1, "anni");   // GIUSTO: 23
```

Domande dalla lezione · Gli spazi contano?

Non contano (tra un'istruzione e l'altra, attorno a `=` , `+` , `(` , a capo, rientri):

```
let   x=5 ;
let y   =   x+1;       // stesso significato di let y = x + 1;
```

L'indentazione serve solo a **leggere meglio** il codice.

Anche le **righe vuote** servono solo alla lettura: ad esempio per separare i blocchi logici del programma (input, elaborazione, output).

```
let anni_input = prompt("Quanti anni hai?");     // 1. prendo i dati in input
let anni = Number(anni_input);

let prossimo = anni + 1;                          // 2. li trasformo

console.log("L'anno prossimo avrai", prossimo);   // 3. stampo il risultato
```

Contano:

- **dentro le stringhe**: `"Ciao "` e `"Ciao"` sono diverse, e `"Ciao " + "mondo"` stampa `Ciao mondo`, mentre `"Ciao" + "mondo"` stampa `Ciaomondo`
- **nei nomi e nelle parole chiave**: `let mia variabile = 5;` dà `SyntaxError`; si scrive `mia_variabile`. Anche `let x` e `letx` sono cose diverse
- **nell'input**: `Number(" 22 ")` vale `22` (gli spazi ai lati si ignorano), ma `Number("2 2")` vale `NaN`, e `Number("")` (o solo spazi) vale `0`

`console.log("a", "b")` stampa `a b`: tra gli argomenti separati da virgola lo spazio lo mette lui.

 Ludovica Pannitto - Università di Pisa - CC BY-NC-SA 4.0