



HTML, CSS & JavaScript

HTML



HTML

- HTML (HyperText Markup Language) è il linguaggio base per creare pagine web
- Definisce la **struttura** e il **contenuto** di un pagina web
 - ad es. titoli, paragrafi, immagini, link

Tag HTML

- HTML usa dei **tag** (come `<p>` , `<h1>` , `<body>`) per strutturare il contenuto
- I tag spesso appaiono in coppia: tag di apertura e tag di chiusura
- Ogni tag può contenere altri tag figli: **struttura ad albero**
- Ogni pagina HTML è una struttura base con `<html>` , `<head>` e `<body>`
 - `<head>` contiene informazioni sulla pagina, ad es. il titolo
 - `<body>` contiene il contenuto effettivo che vedrà l'utente

```
<!DOCTYPE html>
<html>
  <head>
    <title>Titolo della pagina</title>
  </head>
  <body>
    <h1>Benvenut&#601;! Questo &egrave; un titolo</h1>
    <p>
      Questo &egrave; un paragrafo di testo.
      Lorem ipsum dolor sit amet...
    </p>

    <div>
      <input type="text" />
      <button>Pulsante</button>
    </div>

    <a href="https://www.wikipedia.org/">
      Link a Wikipedia
    </a>

    <div>
      
  </body>
</html>
```



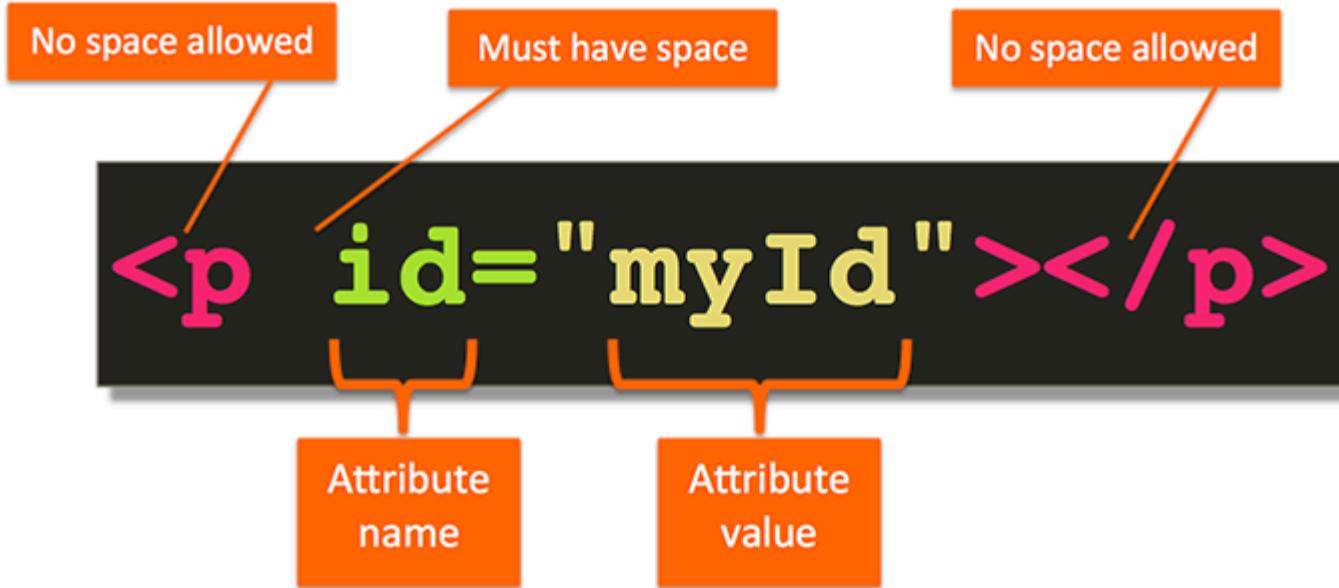
Tag principali

- `<h1>` , `<h2>` ... `<h6>` : Titoli di livello
- `<p>` : Paragrafo
- `<div>` : Contenitore generico. Usato anche per layout di stile
- `` : Testo inline
- `` : Link ad un'altra pagina web
- `` : Immagine
- `` : Lista non ordinata
- `` : Lista ordinata
- `` : Elemento della lista
- `<form>` : Modulo per compilare e inviare dati
- `<input>` : Campo di input
- `<button>` : Pulsante

Attributi HTML

- Gli elementi HTML posso avere **attributi**
- Gli attributi forniscono informazioni aggiuntive agli elementi
- Vanno specificati nel tag iniziale dell'elemento
- Solitamente ogni attributo è costituito da una coppia nome/valore, es. `name="value"`

Sintassi di tag HTML e attributi



Attributi comuni

- `class` : usato per assegnare una classe CSS ad un elemento HTML e applicargli uno stile.

```
<p class="important-text">C'era una volta...</p>
```

- `id` : usato per assegnare un identificativo univoco ad un elemento HTML. Utile per identificare un elemento in JavaScript.

```
<input type="text" id="myTextInput" />
```

- `href` : attributo del tag `<a>` per specificare l'indirizzo di destinazione (URL).

```
<a href="https://www.wikipedia.org/">Go to Wikipedia</a>
```

Attributi comuni (2)

- `src` : attributo del tag `` per specificare l'indirizzo dell'immagine da visualizzare.

```

```

- `alt` : usato per specificare il testo alternativo di un'immagine. Il testo alternativo viene mostrato al posto dell'immagine se l'immagine non è disponibile, ad es.:
 - errore nell'attributo `src`
 - connessione lenta
 - l'utente usa uno screen reader

```

```

CSS 

CSS

- CSS (Cascading Style Sheets) è il linguaggio base per decorare una pagina web
- Definisce lo **stile**, l'apparenza e il posizionamento di ogni elemento della pagina, ad es.:
 - Colore
 - Dimensione
 - Allineamento
 - Spaziatura rispetto ad altri elementi

Regole CSS

- Una regola CSS è formata da un selettore e da un blocco di dichiarazioni
- In caso di regole in conflitto, vince la regola più specifica

```
h1 {    /* selettore: applica la regola agli elementi "h1" (titoli) */
    color: green;    /* dichiarazione: colora il testo di verde */
    text-align: center;    /* dichiarazione: allinea il testo al centro */
    margin-bottom: 2rem;    /* dichiarazione: margine inferiore di 32 pixel */
}

.important-text {    /* applica la regola agli elementi con classe "important-text" */
    text-decoration: underline;    /* sottolineatura */
    font-weight: bold;    /* grassetto */
    color: blue;
}

p.important-text {    /* applica la regola SOLO ai paragrafi con classe "important-text" */
    color: red;
}

#myTextInput {    /* applica la regola all'elemento con ID "myTextInput" */
    background-color: yellow;
}
```

Regole CSS (2)

- Il selettore usato più comunemente è il selettore di classe CSS

```
<p class="important-text">C'era una volta...</p>
```

```
.important-text {  
    text-decoration: underline;  
    font-weight: bold;  
    color: blue;  
}
```

Colori

In CSS, i colori vengono utilizzati per definire l'aspetto visivo degli elementi (es. testo, sfondi, bordi). Possono essere specificati in diversi modi:

- Colori per nome: `red` , `blue` , `green` , `black` , `white` , ecc.
- Colori RGB e RGBA: specifica i valori dei canali rosso, verde e blu da 0 a 255.

```
color: rgb(255, 0, 0); /* Rosso */  
color: rgb(0, 255, 0); /* Verde */  
color: rgb(0, 0, 255); /* Blu */  
color: rgb(255, 255, 0); /* Giallo */  
color: rgb(128, 128, 128); /* Grigio */
```

- Colori esadecimale: colori composti da tre coppie: #RRGGBB (red, green, blue). Ogni coppia esadecimale va da `00` (più scuro) a `ff` (più chiaro)

```
color: #ff0000; /* Rosso */  
color: #00ff00; /* Verde */  
color: #0000ff; /* Blu */  
color: #ffff00; /* Giallo */  
color: #333333; /* Grigio scuro */  
color: #dddddd; /* Grigio scuro */
```

Unità di misura

Le unità di misura vengono utilizzate per definire dimensioni (come larghezze, altezze, margini, dimensione del testo, ecc.).

- `px` (pixel): Misura fissa e precisa. `1px` è un punto sullo schermo
 - Es. `font-size: 16px;`
- `rem` : Misura migliore per creare layout scalabili. Solitamente `1rem` equivale a `16px`
 - Es. `margin-right: 2rem;`
- `%` : Relativa all'elemento HTML genitore nella struttura ad albero
 - Es: `width: 50%;` è metà della larghezza dell'elemento genitore.

Proprietà `display`

Specifica il modo in cui sarà visualizzato l'elemento. Può assumere svariati valori, i principali sono:

- `inline` : l'elemento è posizionato sulla stessa linea degli altri elementi (come ``), occupando solo lo spazio necessario. Eventuali proprietà `width` o `height` saranno ignorate
- `block` : mostra l'elemento su una nuova linea e di default occupa tutta la larghezza, come `<p>` e `<div>`
- `inline-block` : mostra l'elemento sulla stessa linea come `inline`, ma è possibile impostare `width` e `height`
- `none` : l'elemento non appare e non sarà presente nell'albero HTML
- `flex` : mostra l'elemento come un contenitore flex di tipo block (ne parleremo nelle prossime lezioni)
- `inline-flex` : mostra l'elemento come un contenitore flex di tipo inline (ne parleremo nelle prossime lezioni)

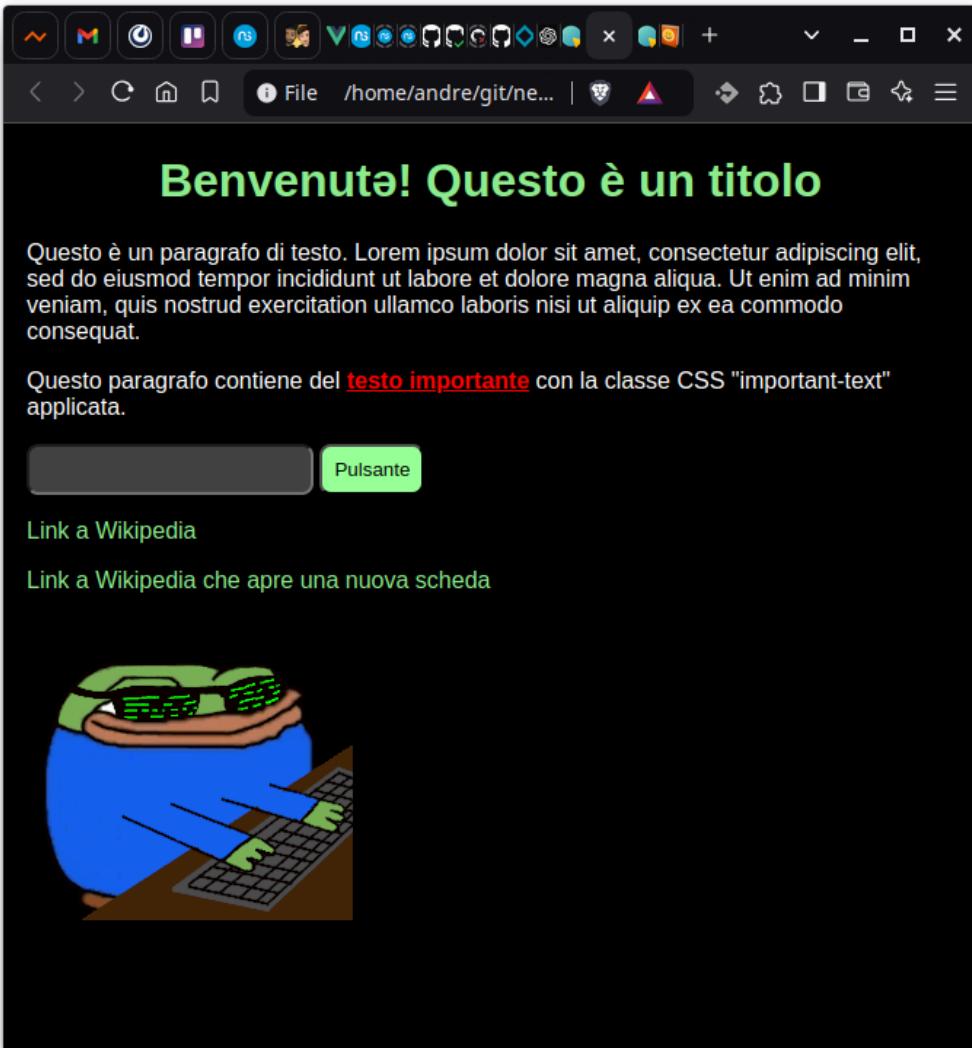
Usare le regole CSS in un pagina HTML

- Definire le regole CSS in un file `.css`
- Includere il file `.css` all'interno del tag `<head>` del codice HTML
 - Es. `<link rel="stylesheet" href="style.css" />`
- Assegnare agli elementi elementi HTML le classi CSS per applicargli stile

Usare le regole CSS in un pagina HTML (2)

```
/* style.css */  
.important-text {  
    text-decoration: underline;  
    font-weight: bold;  
}  
...
```

```
<!-- index.html -->  
<!DOCTYPE html>  
<html>  
    <head>  
        <title>Titolo della pagina</title>  
        <link rel="stylesheet" href="style.css" />  
    </head>  
    <body>  
        ...  
        <p class="important-text">C'era una volta...</p>  
    </body>  
</html>
```



Layout gerarchico

- Organizzare gerarchicamente i container in un layout di una pagina web è fondamentale per garantire una struttura logica e mantenibile
- Pensare ad ogni sezione della pagina come un contenitore
- Iniziare dalla struttura generale della pagina (header, contenuto principale, footer) e suddividere ciascun contenitore in sezioni più piccole
- Ogni contenitore potrà essere stilizzato con regole CSS

```
<div class="container">  
  ...  
  <div class="chat">  
    ...  
  </div>  
</div>
```

Chatbot HTML + CSS

Ciao chatbot!

Ciao utente! Come posso esserti utile oggi?

Ho bisogno di aiuto con un argomento di fisica

Certo! Dimmi cosa ti serve e farò del mio meglio per aiutarti. 🤖📚

Parlami della forza di attrito

La forza di attrito è una forza che si oppone al movimento relativo tra due superfici che sono a contatto. È una forza fondamentale nella nostra esperienza quotidiana, responsabile di fenomeni come il poter camminare senza scivolare, frenare un'auto o trattenere oggetti su una superficie inclinata.

Come si calcola la forza di attrito di un corpo su un piano inclinato di 45 gradi?

Che domanda difficile, fammici pensare... 🧐⌚

Scrivi un messaggio...

Invia

container

title

chat

message-bar

Approfondimenti su HTML e CSS

- W3Schools: <https://www.w3schools.com/>
- Tutorial HTML: https://www.w3schools.com/html/html_intro.asp
- Esempi HTML: https://www.w3schools.com/html/html_examples.asp
- Tutorial CSS: https://www.w3schools.com/css/css_intro.asp
- Esempi CSS: https://www.w3schools.com/css/css_examples.asp

JavaScript



Cos'è JavaScript?

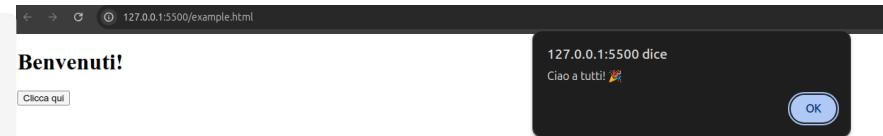
- JavaScript (JS) è un linguaggio di programmazione per rendere le pagine web interattive.
- Se l'HTML è lo scheletro di una pagina e il CSS il vestito, JavaScript è il cervello!
- Con JavaScript possiamo fare cose come:
 - Mostrare messaggi quando si clicca un pulsante
 - Nascondere o cambiare il contenuto di una pagina
 - Creare giochi, animazioni e tanto altro!

Come funziona JavaScript?

- Il browser web (ad esempio Chrome o Firefox) è in grado di leggere ed eseguire JavaScript.
- Possiamo scrivere il nostro codice JavaScript:
 - Direttamente in una pagina HTML
 - Oppure in un file separato con estensione .js.

```
<!DOCTYPE html>
<html>
  <head>
    <title>La mia prima pagina JS</title>
  </head>
  <body>
    <h1>Benvenuti!</h1>
    <button onclick="saluta()">Clicca qui</button>

    <script>
      function saluta() {
        alert('Ciao a tutti! 🎉');
      }
    </script>
  </body>
</html>
```



Cosa fa questo codice?

1. Crea un pulsante sulla pagina.
2. Quando clicchiamo il pulsante, JavaScript mostra un messaggio con la scritta "Ciao a tutti!" in una finestra pop-up.

Variabili

- Le variabili sono "scatole" in cui possiamo memorizzare informazioni, come numeri o parole.
- Possiamo crearle con le parole chiave let o const.

```
let nome = 'Mario'; // Una variabile che contiene il nome
let eta = 16;      // Una variabile che contiene un numero
const scuola = 'Liceo'; // Una variabile che non può essere cambiata
```

Tipi di dati

- In JavaScript, le variabili possono contenere diversi tipi di informazioni:
- Stringhe (testo): Racchiuse tra virgolette, ad esempio "Ciao!".
- Numeri: Ad esempio 42 o 3.14.
- Booleani (vero/falso): true o false.
- Array: Una lista ordinata di valori.
- Oggetti: Una struttura che memorizza dati e metodi correlati.

```
let messaggio = 'Buongiorno!'; // Stringa
let temperatura = 25;           // Numero
let isStudent = true;           // Booleano
let numeri = [ 10, 20, 30 ];    // Array
let persona =
{
  nome: 'Mario',
  eta: 30,
  lavoro: 'Programmatore'
};
```

Gli oggetti

- Gli oggetti in JavaScript servono per memorizzare dati più complessi.
- Un oggetto è composto da coppie chiave-valore. Le chiavi sono i nomi delle proprietà, e i valori possono essere stringhe, numeri, funzioni o altri oggetti.

```
let persona =  
{  
  nome: 'Mario',  
  eta: 30,  
  lavoro: 'Programmatore',  
};
```

Come accedere alle proprietà di un oggetto?

- Possiamo accedere alle proprietà di un oggetto in due modi:

- Con il punto (.): `persona.nome`
- Con le parentesi quadre ([]): `persona['nome']`

```
console.log(persona.nome); // Mostra: "Mario"
console.log(persona['eta']); // Mostra: 30
```

- Per vedere il risultato della `console.log`, apri la console degli sviluppatori *premendo F12*.

Aggiungere o modificare proprietà

- È possibile aggiungere o aggiornare una proprietà di un oggetto in modo dinamico:

```
persona.indirizzo = 'Via Roma, 10'; // Aggiunge una nuova proprietà  
persona.eta = 31; // Modifica una proprietà esistente  
  
console.log(persona);
```

Gli array

- Gli array servono per memorizzare più valori in una sola variabile.
- Ogni valore in un array ha un indice che parte da 0 (il primo elemento si trova in posizione 0, il secondo in posizione 1, e così via).

```
let numeri = [ 10, 20, 30 ]; // Un array con 3 numeri
let colori = [ 'rosso', 'verde', 'blu' ]; // Un array con 3 stringhe
```

Come accedere ai valori di un array?

- Possiamo usare il numero dell'indice tra parentesi quadre per accedere a un valore.

```
let colori = [ 'rosso', 'verde', 'blu' ];
console.log(colori[0]); // Mostra: "rosso"
console.log( numeri[2] ); // Mostra: 30
```

Aggiungere elementi a un array

- Possiamo aggiungere nuovi elementi a un array con il metodo `.push()`.

```
let colori = [ 'rosso', 'verde', 'blu' ];
colori.push('giallo'); // Ora l'array è ['rosso', 'verde', 'blu', 'giallo']
console.log(colori);
```

Operatori

Gli operatori ci permettono di fare operazioni con i dati.

Operatore	Significato	Esempio	Risultato
+	Addizione	$5 + 2$	7
-	Sottrazione	$5 - 2$	3
*	Moltiplicazione	$5 * 2$	10
/	Divisione	$6 / 2$	3
==	Uguaglianza	$5 == 5$	true
>	Maggiore	$6 > 2$	true

Condizioni if/else

- Le condizioni ci permettono di eseguire del codice solo se una certa regola è vera.

```
let ora = 10;

if (ora < 12) {
  console.log('Buongiorno!'); // Mostra "Buongiorno!" se l'ora è prima di mezzogiorno
} else {
  console.log('Buon pomeriggio!');
}
```

Cicli (loops)

- I cicli servono per ripetere un'azione più volte.
- Ad esempio, stampare i numeri da 1 a 5:

```
for (let i = 1; i <= 5; i++) {  
    console.log(i); // Mostra: 1, 2, 3, 4, 5  
}
```

Funzioni

- Le funzioni sono gruppi di istruzioni che possiamo riutilizzare.
- Possiamo "chiamare" una funzione per far eseguire il codice al suo interno.

```
function saluta(nome) {  
    console.log('Ciao ' + nome + '!');  
}  
  
saluta('Luigi'); // Mostra: "Ciao Luigi!"  
saluta('Anna'); // Mostra: "Ciao Anna!"
```

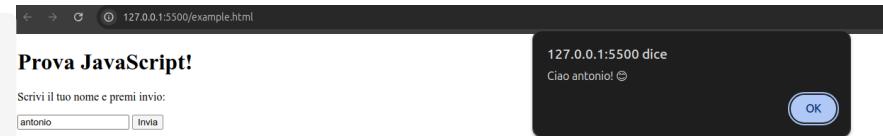
Eventi

- JavaScript può "ascoltare" quello che succede sulla pagina web, come cliccare un pulsante o muovere il mouse.

```
document.getElementById('mioBottone').addEventListener('click', function() {
  alert('Hai cliccato il bottone!');
});
```

```
<!DOCTYPE html>
<html>
  <head>
    <title>Esempio completo</title>
  </head>
  <body>
    <h1>Prova JavaScript!</h1>
    <p>Scrivi il tuo nome e premi invio:</p>
    <input type="text" id="nomeInput" />
    <button onclick="saluta()">Invia</button>

    <script>
      function saluta() {
        let nome = document.getElementById('nomeInput').value
        alert('Ciao ' + nome + '! 😊');
      }
    </script>
  </body>
</html>
```



Esercizi HTML & CSS



Esercizio 0: Prepariamo l'ambiente di lavoro



- Scarica [Visual Studio Code](#)
- Installa Visual Studio Code
- Scarica il codice che contiene gli esercizi
 - Vai su <https://github.com/nethesis/nethKaris>
 - Clicca il pulsante verde **Code**
 - Clicca **Download ZIP**
 - Estrai il file .zip dove preferisci, ad esempio sul Desktop
- Apri Visual Studio Code
- Clicca su **File > Open Folder...**
- Entra nella cartella in cui hai estratto il file .zip, seleziona la cartella `code/frontend`, ad es.
`Desktop/nethKaris-main/code/frontend` e clicca **OK**

Esercizio 1: Sostituiamo un'immagine



Cambia il percorso di un'immagine in una pagina web

- Su Visual Studio Code apri il file `html-1-change-image/change-image.html`
- Apri lo stesso file sul browser (Chrome, Edge, Firefox...)
- Cerca un'immagine sul web (es. su [Google Immagini](#)) e scaricala sul computer
- Sposta il file dell'immagine dentro la cartella `html-1-change-image/`
- Modifica l'immagine contenuta in `change-image.html` modificando l'attributo `src` del tag `img`, ad esempio ``. I caratteri `. /` indicano che il file `nomeImmagine.jpg` si trova nella stessa cartella del file attuale (`test.html`)
- Ricarica la pagina `change-image.html` sul browser per vedere le tue modifiche (F5 sulla tastiera)

Esercizio 2: Cambiamo lo stile



Modifica lo stile di una pagina web

- Su Visual Studio Code apri il file `html-2-change-style/change-style.html`
- Apri il file `html-2-change-style/change-style.html` sul browser (Chrome, Edge, Firefox...)
- Su Visual Studio Code, esamina la struttura HTML di `change-style.html` e le classi CSS utilizzate dai vari tag
- Su Visual Studio Code apri anche il file `html-2-change-style/style.css`
- Cambia lo stile della pagina secondo i tuoi gusti, modificando il file `style.css` per cambiare:
 - i colori del testo (`color`)
 - i colori di sfondo (`background-color`)
 - l'allineamento del testo (`text-align`)
 - l'arrotondamento del pulsante
 - margini e padding
- Ricarica la pagina `change-style.html` sul browser per vedere le tue modifiche (F5 sulla tastiera)

Esercizio 3: I miei preferiti



Crea una pagina web in cui mostrare una classifica: ad esempio i tuoi 3 film preferiti, oppure le tue 3 canzoni preferite, oppure i tuoi 3 libri preferiti ecc.

- Su Visual Studio Code apri il file `html-3-favorites/favorites.html`
- Apri lo stesso file anche sul browser (Chrome, Edge, Firefox...)
- Aggiungi un titolo nella pagina web inserendo un tag `<h1>` all'interno del tag `<body>`, ad es. "I miei 3 film preferiti"
- Ricarica la pagina web sul browser per vedere le tue modifiche (F5 sulla tastiera)
- Aggiungi un titolo alla pagina inserendo un tag `<title>...</title>` all'interno del tag `<head>`. Il titolo dovrà apparire nella scheda del browser
- Aggiungi un titolo `<h2>` sotto il titolo H1 e scrivi il titolo del primo classificato (es. il nome del tuo film preferito)
- Aggiungi un paragrafo `<p>` sotto il titolo H2 e descrivi brevemente perché ti è piaciuto
- (continua nella prossima pagina)

Esercizio 3: I miei preferiti (2)

- Aggiungi un link `` sotto il titolo H2 che porta alla pagina Wikipedia del film. Per aprire il link in una nuova scheda del browser, usare l'attributo `target="_blank"`.
 - Ad esempio: `Apri su Wikipedia`
- Aggiungi un titolo `<h2>` sotto il link precedente per il secondo classificato
- Aggiungi un paragrafo `<p>` per descrivere il secondo classificato
- Aggiungi un link a Wikipedia per il secondo classificato
- Aggiungi titolo, paragrafo e link anche per il terzo classificato
- (continua nella prossima pagina)

Esercizio 3: I miei preferiti (3)

- Aggiungi stile alla pagina
 - Aggiungi l'attributo `class="nomeClasse"` agli elementi a cui vuoi applicare qualche stile
 - Ad es: `<h1 class="mainTitle">I miei film preferiti</h1>`
 - Crea un file `style.css` nella stessa cartella del file HTML (`html-3-favorites`)
 - Includi il file `.css` all'interno del tag `<head>` del codice HTML per caricare lo stile nella pagina
 - Es: `<link rel="stylesheet" href="style.css" />`
 - Scrivi nel file `style.css` le regole di stile, ad es:

```
.mainTitle {  
    text-align: center;  
    color: light;  
}  
  
.movieDescription {  
    ...  
}
```

Esercizio 4: Chatbot

Aggiungi stile ad una conversazione con un chatbot

- Su Visual Studio Code apri il file `html-4-chatbot/chatbot.html`
- Apri lo stesso file anche sul browser (Chrome, Edge, Firefox...)
- Crea un file `style.css` nella stessa cartella del file HTML (`html-4-chatbot`)
- Includi il file `.css` all'interno del tag `<head>` del codice HTML
 - Es: `<link rel="stylesheet" href="style.css" />`
- Scrivi nel file `style.css` le regole di stile per ottenere uno stile simile a quello della pagina successiva

Chatbot HTML + CSS

Ciao chatbot!

Ciao utente! Come posso esserti utile oggi?

Ho bisogno di aiuto con un argomento di fisica

Certo! Dimmi cosa ti serve e farò del mio meglio per aiutarti. 🤖📚

Parlami della forza di attrito

La forza di attrito è una forza che si oppone al movimento relativo tra due superfici che sono a contatto. È una forza fondamentale nella nostra esperienza quotidiana, responsabile di fenomeni come il poter camminare senza scivolare, frenare un'auto o trattenere oggetti su una superficie inclinata.

Come si calcola la forza di attrito di un corpo su un piano inclinato di 45 gradi?

Che domanda difficile, fammici pensare... 🧐⏳

Scrivi un messaggio...

Invia

Chatbot HTML + CSS

Ciao chatbot!

Ciao utente! Come posso esserti utile oggi?

Ho bisogno di aiuto con un argomento di fisica

Certo! Dimmi cosa ti serve e farò del mio meglio per aiutarti. 🤖📚

Parlami della forza di attrito

La forza di attrito è una forza che si oppone al movimento relativo tra due superfici che sono a contatto. È una forza fondamentale nella nostra esperienza quotidiana, responsabile di fenomeni come il poter camminare senza scivolare, frenare un'auto o trattenere oggetti su una superficie inclinata.

Come si calcola la forza di attrito di un corpo su un piano inclinato di 45 gradi?

Che domanda difficile, fammici pensare... 🧐⌚

Scrivi un messaggio...

Invia

container

title

chat

message-bar

Esercizio 4: Chatbot (4)

Suggerimenti per il file di stile

```
body {  
  ...  
}  
  
.container {  
  max-width: 40rem;  
  margin: auto; /* centra orizzontalmente il contenitore */  
}  
  
.user-message {  
  display: inline-block;  
  ...  
}  
  
.message-input {  
  width: 79%;  
  font-size: 1rem;  
  border: none;  
  ...  
}  
...
```

Esercizi JavaScript 💪

Esercizio 1: Variabili e operazioni base



1. Apri il file `javascript-1-variables/index.html` sul browser.

2. Modifica il file `javascript-1-variables/script.js` e:

- Crea una variabile per il nome di un animale:

```
let animale = "cane";
```

- Crea una variabile per il numero di zampe:

```
let zampe = 4;
```

- Crea una variabile per il numero di animali totali:

```
let numeroAnimali = 2;
```

Esercizio 1: Variabili e operazioni base (2)

- Calcola la somma del totale delle zampe di due cani:

```
let totaleZampe = zampe * numeroAnimali;  
console.log("Totale zampe: " + totaleZampe);
```

- Ricarica la pagina web e verifica il risultato nella console del browser (premi F12 sulla tastiera).

Esercizio 2: Interazione con l'utente



1. Apri il file `javascript-2-prompt-alert/index.html` sul browser.

2. Modifica il file `javascript-2-prompt-alert/script.js` :

- Chiedi all'utente il suo nome usando `prompt` :

```
let nome = prompt("Come ti chiami?");
```

- Saluta l'utente con un messaggio personalizzato:

```
alert("Ciao, " + nome + "!");
```

Esercizio 3: Condizioni (if/else)



1. Apri il file `javascript-3-conditions/index.html` sul browser.

2. Modifica il file `javascript-3-conditions/script.js` :

- Chiedi all'utente la sua età:

```
let eta = prompt("Quanti anni hai?");
```

- Usa una condizione per verificare se è maggiorenne:

```
if (eta >= 18) {  
    alert("Sei maggiorenne!");  
} else {  
    alert("Sei minorenne!");  
}
```

Esercizio 4: Cicli e array



1. Apri il file `javascript-4-loops/index.html` sul browser.

2. Modifica il file `javascript-4-loops/index.html`.

- Crea un array con i tuoi 3 colori preferiti:

```
let colori = ["rosso", "verde", "blu"];
```

- Usa un ciclo for per stampare ogni colore nella console del browser (premi F12 sulla tastiera):

```
for (let i = 0; i < colori.length; i++) {  
    console.log(colori[i]);  
}
```

Esercizio 5: Gestione di una lista di nomi



1. Apri il file `javascript-5-user-list/index.html` sul browser.

- Aggiungi un text input alla tua pagina:

```
<input type="text" id="esempio" placeholder="inputFieldEsempio">
```

- Aggiungi un pulsante che permetterà l'aggiunta del nome tramite il richiamo della funzione `aggiungiNome()`:

```
<button onclick="aggiungiNome()">Aggiungi</button>
```

- Aggiungi una lista sotto il pulsante contente l'array dei nomi:

```
<ul id="listaNomi"></ul>
```

Esercizio 5: Gestione di una lista di nomi (2)

2. Modifica il file `javascript-5-user-list/script.js` :

- Crea la funzione `aggiungiNome()` :

```
function aggiungiNome() {}
```

- Associa il nome al valore del text input:

```
let nome = document.getElementById("nameInput").value;
```

- Recupera la lista già esistente e aggiungi il nome:

```
let lista = document.getElementById("listaNomi");
lista.innerHTML += "<li>" + nome + "</li>";
```

- Svuota il campo di input dopo aver aggiunto il nome:

```
document.getElementById("nameInput").value = "";
```

Esercizio 6 : Lavoriamo con gli oggetti



1. Apri il file `javascript-6-object/index.html` sul browser.
2. Modifica il file `javascript-6-object/script.js` :
 - Dichiara un oggetto persona che contenga nome, eta, citta, e hobby:

```
let persona = {  
    nome: '',  
    eta: '',  
    citta: '',  
    hobby: [],  
};
```

- Aggiungi almeno un elemento all'array hobby:

```
persona.hobby.push(...);
```

Esercizio 6 : Lavoriamo con gli oggetti (2)

- Stampa in console il nome e la città della persona:

```
console.log('Nome: ' + ...);
```

- Stampa in console l'array hobby ciclando su tutti gli elementi:

```
for (let i = 0; i < persona.hobby.length; i++) {  
    ...  
}
```

- Stampa in console il primo elemento dell'array hobby:

```
console.log(persona.hobby[0]);
```

- Modifica il valore di "città" dell'oggetto persona

```
persona.citta = ...;
```

Esercizio HTML, CSS & JavaScript 💪

Esercizio finale: Lista della spesa



Crea una semplice webapp per la lista della spesa

- Su Visual Studio Code apri il file `javascript-7-grocery-list/grocery-list.html`
- Apri lo stesso file anche sul browser
- Aggiungi un titolo alla pagina inserendo un tag `<title>...</title>` all'interno del tag `<head>`. Il titolo dovrà apparire nella scheda del browser
- Aggiungi un `<div>` con classe CSS `container` all'interno del `<body>`
- (continua nella prossima pagina)

Esercizio finale: Lista della spesa (2)

- All'interno del `<div>` aggiungi:
 - un titolo `<h1>` "Grocery list" (lista della spesa)
 - un elenco `` con ID `grocery-list`
 - un campo di testo `<input type="text" />` con ID `new-item`
 - un campo di testo numerico con valore iniziale "1":

```
<input type="number" id="new-quantity" value="1" />
```
 - un pulsante con testo "Add"
- (continua nella prossima pagina)

Esercizio finale: Lista della spesa (3)

- Importa il file `script.js` come ultimo elemento del `<body>`, es:

```
...
<body>
  ...
  <script src="script.js"></script>
</body>
...
```

- Completa il codice nel file `script.js` :
 - L'utente può aggiungere elementi alla lista della spesa scrivendo descrizione (ad es. "Farina") e quantità (ad es. "3") nelle due caselle di input e cliccando il pulsante "Add"
 - (continua nella prossima pagina)

Esercizio finale: Lista della spesa



(4)

- La funzione `addItem` deve:
 - Recuperare gli elementi HTML della descrizione e della quantità digitate dall'utente, usando `getElementById`
 - Aggiungere all'array `groceryList` un oggetto con attributi `description` e `quantity`
 - Recuperare l'elemento HTML relativo all'elenco ``, usando `getElementById`
 - Svuotare il contenuto HTML dell'elenco, settando `innerHTML = ""`
 - Eseguire un ciclo `for` su tutti gli elementi dell'array `groceryList`, aggiungendo ogni volta all' `innerHTML` dell'elenco un elemento `` che riporti la descrizione e la quantità dell'elemento, es:

```
list.innerHTML += "<li>" + groceryList[i].description + " - " + groceryList[i].quantity + "</li>";
```

- Resetare le due caselle di testo, impostando una stringa vuota per la descrizione e "1" per la quantità
- (continua nella prossima pagina)

Esercizio finale: Lista della spesa



(5)

- Aggiungi stile alla pagina
 - Aggiungi l'attributo `class="nomeClasse"` agli elementi a cui vuoi applicare stile
 - Crea un file `style.css` nella stessa cartella del file HTML (`javascript-7-grocery-list`)
 - Includi il file `.css` all'interno del tag `<head>` del codice HTML per caricare lo stile nella pagina
 - Es: `<link rel="stylesheet" href="style.css" />`
 - Scrivi nel file `style.css` le regole di stile, usando colori, margini, padding e le altre proprietà CSS secondo i tuoi gusti