Metodologie per la Programmazione per il Web - MF0437
Esercizi - *Programmazione client-side con Javascript*

Docente

Giancarlo **Ruffo**[giancarlo.ruffo@uniupo.it]

Informazioni, materiale e risorse su:

moodle [https://www.dir.uniupo.it/course/view.php?id=16455]

- Esercizi JS
- Caso di studio: Hangman
- Caso di studio: Colored Squares

- **Esercizi JS**
- Caso di studio: Hangman
- Caso di studio: Colored Squares

W3Schools

- https://www.w3schools.com/js/exercise_js.asp?filename=exercise_js_dom_html1
- Potete farli tutti, se volete

Debugging

Evitate "echo printing" sulla pagina per controllare i valori delle varie espressioni

`console.log()`: consente di stampare nella console del browser (vedi: strumenti dello sviluppatore) valori di espressioni Javascript.

Fermare l'esecuzione dello script:

- Tramite istruzione "`debugger`" dentro al codice
- Tramite inserimento di breakpoint manualmente durante l'esecuzione
- In entrambi i casi devi aprire l'ambiente di debugging del browser che stai usando

https://www.w3schools.com/js/js_debugging.asp

Strumenti di debugging dei browser

- Usare la chiave F12 per attivare il debugging, quindi selezionare "Console" menù del debugger
- Oppure
  - Chrome
    - "More tools" > "Developer tools" > "Console".
  - Firefox, Edge
    - "Web Developer" > "Web Console".
  - Opera
    - "Developer" > "Developer tools" > "Console".
  - Safari
    - Safari > Preferences > Advanced in the main menu
    - Seleziona "Enable Show Develop menu in menu bar
    - Quando l'opzione "Develop" appare nel menù principale:
      - "Show Javascript Console"

Esercizio 1

- Creare una pagina con un campo testo ed un pulsante. La pagina deve anche avere un elemento span che deve contenere un testo a piacere. Se l'utente clicca sul pulsante, ciò che è scritto all'interno del campo testo deve sostituire il contenuto precedente dell'elemento span
  - Suggerimento: usare il tag `<button>`*testo*`</button>` di html per il pulsante, dando un valore all'attributo `id` per potere usare il metodo `getElementById()` di js:
    ```
    <button type="button" id="mioPulsante">Click Me!</button>
    ```
- Usare il debugger del browser per seguire passo passo quello che succede sulla pagina man mano che le singole istruzioni vengono eseguite

Esercizio 2

- Creare un pulsante con la scritta "Trova gli anni bisestili". Quando il pulsante viene premuto, viene chiamata una funzione javascript che calcola i 10 anni bisestili successivi ad un anno selezionato dall'utente per mezzo di un campo testo.
  - il campo testo per inserire l'input corrisponde al tag `<input>`, con attributo `type = text`, es:
    `<input type="text" id="anno">`
  - il valore inserito dall'utente dentro il campo testo è accessibile da javascript tramite l'attributo `value`
- Ricordare che un anno è bisestile se:
  - è multiplo di 400
  - oppure, quando non è multiplo di 100, deve essere almeno multiplo di 4
- La sequenza di questi anni deve essere mostrata in uno span inizialmente vuoto

Esempio: shuffler

- Guardare negli esempi forniti (anche) su moodle
- File: shuffle*
- Provare ad usare il debugger incluso nel browser (chrome, firefox, etc.)

Esempio: never gonna give you up

- File: rickastley*

Esercizio 3

- Crea una pagina web con un'immagine di Homer Simpson al centro. Scrivi uno script che stampi un messaggio: "Duh, you are hovering!!" ogni volta che il mouse passa sopra l'immagine

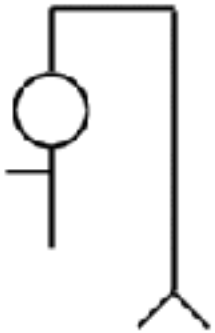
- Aggiungi 5 pulsanti a questa pagina chiamati: red, yellow, green, black, e silver. Ogni volta che clicchi su uno di questi pulsanti, il colore dello sfondo cambia di conseguenza assumendo il colore corrispondente.

Esercizio 4

- Aggiungi un link con il testo: “CLICK ME!”. Scrivi una funzione JS che scelga a caso uno dei seguenti siti e lo collega al testo del link:
  - http://slashdot.org/
  - http://www.thinkgeek.com/
  - http://despair.com/
  - http://www.redbubble.com/
  - http://googleresearch.blogspot.com/

- Esercizi JS
- **Caso di studio: Hangman**
- Caso di studio: Colored Squares

Caso di studio: hangman



_ e _ e n e _ t _ a t e

Guess

New Game

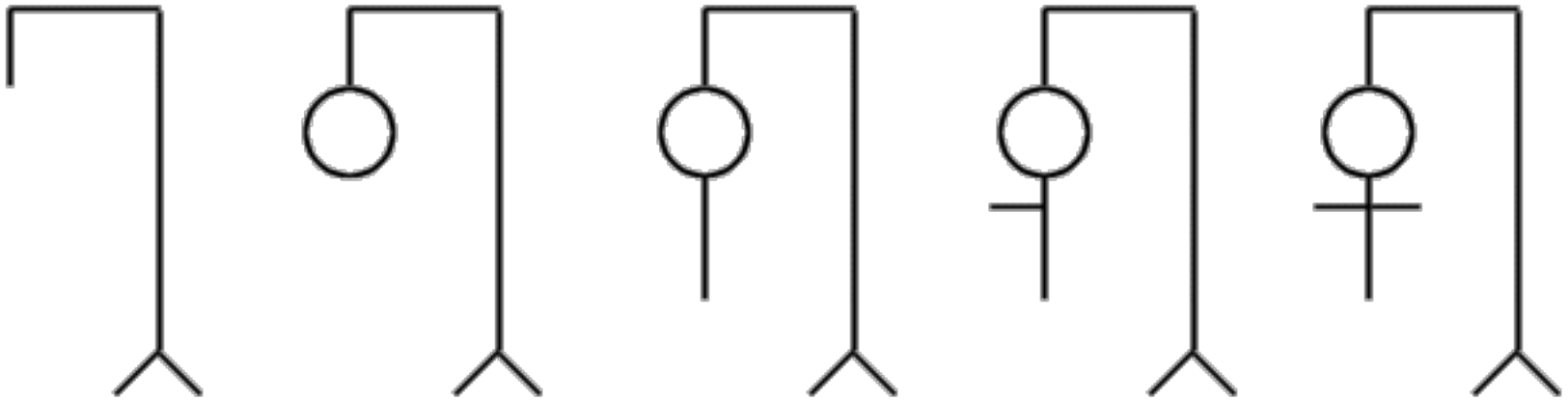
Guesses: tqaencl

Passi

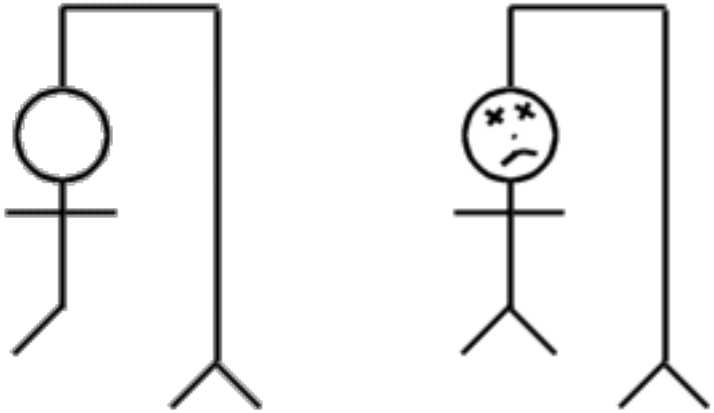
1. Inizia con il codice HTML e CSS di base
2. Gestire la pressione del pulsante “New Game” in JS, cosí che dopo averlo premuto appaia sulla pagina una parola fatta di caratteri blank _
3. Gestire i tentativi dell’utente: consentire di provare una lettera e mostrare i diversi tentativi sulla pagina
4. Verificare se una determinata lettera è presente nella parola oppure no e gestire le due diverse situazioni
5. Pulire il codice aggiungendo la possibilità di giocare più partite, evitare che l’utente faccia lo stesso tentativo due volte, etc.

1. Codice HMLT/CSS di base

- Abbiamo bisogno di 7 diverse immagini, che devono essere mostrate nelle diverse fasi del gioco



`hangman6.gif`



`hangman0.gif`

1. HTML


```
<!DOCTYPE html>
<html>
  <head>
    <title>Hangman</title>
  </head>

  <body>
    <div>
      <img src="hangman6.gif" alt="hangman">
    </div>
    <div id="clue">Press New Game to play!</div>

    <div>
      <input id="guess" type="text" size="1" maxlength="1">
      <button>Guess</button>
    </div>

    <div id="newgamearea">
                  <button>New Game</button>
    </div>
  </body>
</html>
```

HTML

1. CSS

```
<link href="hangman.css" type="text/css" rel="stylesheet" >
```

HTML


```
body {
  text-align: center;
}

#newgamearea {
  margin-top: 2em;
}

#clue, #guesses {
  font-family: monospace;
  font-size: 2em;
  padding: 1em;
}
```

CSS

2. Scegliere la parola con JS

```
<script src="hangman.js" type="text/javascript"></script>
```
HTML

```
<button id="btn_new">New Game</button>
```
HTML

```
// lista di parole tra le quali scegliere
const POSSIBLE_WORDS = ["obdurate", "verisimilitude", "defenestrate",
                        "obsequious", "dissonant", "toady","idempotent"];
```

```
window.onload = function () {
   document.getElementById("btn_new").onclick=newGame;
}
```

2. Abbozziamo la funzione newGame

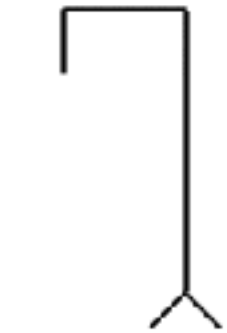
```JS
// scegliamo una parola a caso
let randomIndex = parseInt(Math.random() * POSSIBLE_WORDS.length);
let word = POSSIBLE_WORDS[randomIndex];
```

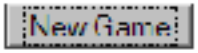
2. Scegliere la parola con JS

Per ogni lettera della parola, stampiamo a video un carattere blank _

```JS
// Chooses a new random word and displays its clue on the page.
function newGame() {
  // choose a random word
  const randomIndex = parseInt(Math.random() * POSSIBLE_WORDS.length);
  word = POSSIBLE_WORDS[randomIndex];

  // show initial word clue - all underscores
  let clueString = "";
  for (let i = 0; i < word.length; i++) {  // ugly version with for...

    clueString += "_ ";
  }
  const clue = document.getElementById("clue");
  clue.innerHTML = clueString;
}
```

2. output





3. Gestire i tentativi

```
<div>
     <input id="guess" type="text" size="1" maxlength="1" />
     <button id="btn_guess">Guess</button>
</div>
...
<div id="guesses"></div>
```

HTML

```
window.onload = function () {
   // ...
   document.getElementById("btn_guess").onclick=guessLetter;
}
// Chooses a new random word and displays its clue on the page.
function newGame() {
   // choose a random word
   const randomIndex = parseInt(Math.random() * POSSIBLE_WORDS.length);
   word = POSSIBLE_WORDS[randomIndex];
   updatePage();   // show initial word clue - all underscores
}
```

JS

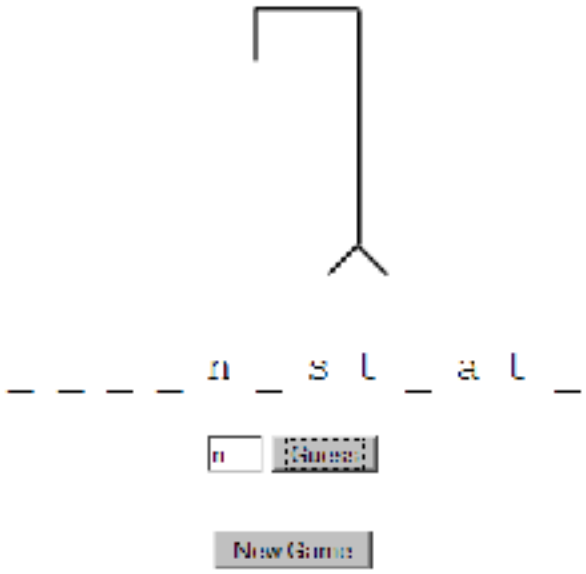
3. updatePage

```JS
// Updates the word clue, guesses, etc. to the current game state.
function updatePage() {
  // update clue string such as "h _ l l _ "
  let clueString = "";
  for (let i = 0; i < word.length; i++) {  // still ugly...
    const letter = word.charAt(i);
    if (guesses.indexOf(letter) >= 0) {   // letter has been guessed
      clueString += letter + " ";
    } else {                              // not guessed
      clueString += "_ ";
    }
  }
  const clue = document.getElementById("clue");
  clue.innerHTML = clueString;

  // show guesses made by player
  const guessArea = document.getElementById("guesses");
  guessArea.innerHTML = "Guesses: " + guesses;
}
```

3. guessLetter

```
// Guesses a letter.  Called when the user presses the Guess button.
function guessLetter() {
  var input = document.getElementById("guess");
  var letter = input.value;
  guesses += letter;
  updatePage();   // rebuild word clue
}
```

JS



4. Gestire l'esito dei tentativi

L'utente può sbagliare al massimo 6 volte. Dobbiamo inoltre fare un "conto alla rovescia" per ogni tentativo fallito

```
const MAX_GUESSES = 6;            // number of total guesses per game
let guessCount = MAX_GUESSES;     // number of guesses player has left
```

```
// global variables
let word = "?";                   // random word user is trying to guess
let guesses = "";                 // letters the player has guessed
let guessCount = MAX_GUESSES;     // number of guesses player has left
```

```
function newGame() {
const randomIndex = parseInt(Math.random() * POSSIBLE_WORDS.length);
word = POSSIBLE_WORDS[randomIndex];
guessCount = MAX_GUESSES;
guesses = "";
...
}
```

JS

4. Gestire l'esito dei tentativi

```JS
// Guesses a letter.  Called when the user presses the Guess button.
function guessLetter() {
  const input = document.getElementById("guess");
  const letter = input.value;

  if (word.indexOf(letter) < 0) {
    guessCount--;       // an incorrect guess
  }

  guesses += letter;
  updatePage();
}
```

4. Cambiare immagine dopo un fallimento

```
<div><img id="hangmanpic" src="hangman6.gif" alt="hangman" /></div>
```
HTML

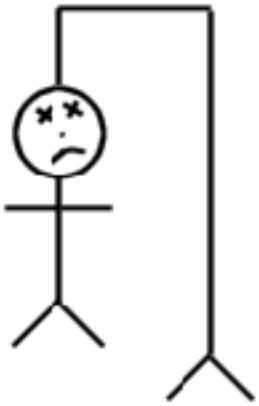
Cambia immagine

```
function updatePage() {
...
 // update hangman image
  const image = document.getElementById("hangmanpic");
  image.src = "hangman" + guessCount + ".gif";
}
```
JS

Mostra se alla fine l'utente ha vinto o perso

```
// show the guesses the player has made
  const guessArea = document.getElementById("guesses");
  if (guessCount == 0) {
    guessArea.innerHTML = "You lose.";       // game over (loss)
  } else if (clueString.indexOf("_") < 0) {
    guessArea.innerHTML = "You win!!!";      // game over (win)
  } else {
    guessArea.innerHTML = "Guesses: " + guesses;
  }
```
JS

4. Game over (due possibilità)

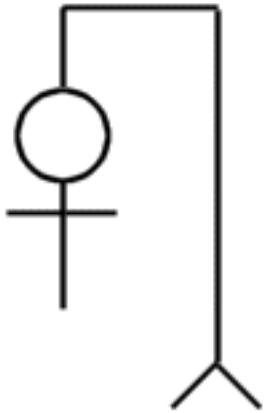


o _ d _ _ a _ e

f Guess

New Game

You lose.



t o a d y

d Guess

New Game

You win!!!

4. Bloccare tutto a gioco finito

Bug: l'utente può continuare a fare dei tentativi fino alla fine.
Soluzione:

```
function guessLetter() {
  const input = document.getElementById("guess");
  const clue = document.getElementById("clue");
  const letter = input.value;
  if (guessCount == 0 || clue.innerHTML.indexOf("_") < 0 ||
      guesses.indexOf(letter) >= 0) {
    return;   // game is over, or already guessed this letter
  }
  ...
}
```

JS

5. usare higher-order function invece di for

- Alternative al for: forEach o anche map (in questo caso)

```
let clueString = "";
for (let i = 0; i < word.length; i++) {
  const letter = word.charAt(i);
  if (guesses.indexOf(letter) >= 0) {
      clueString += letter + " ";
  } else {
      clueString += "_ ";
  }
}
```

- Prova e poi… vedi codice con soluzione del docente!

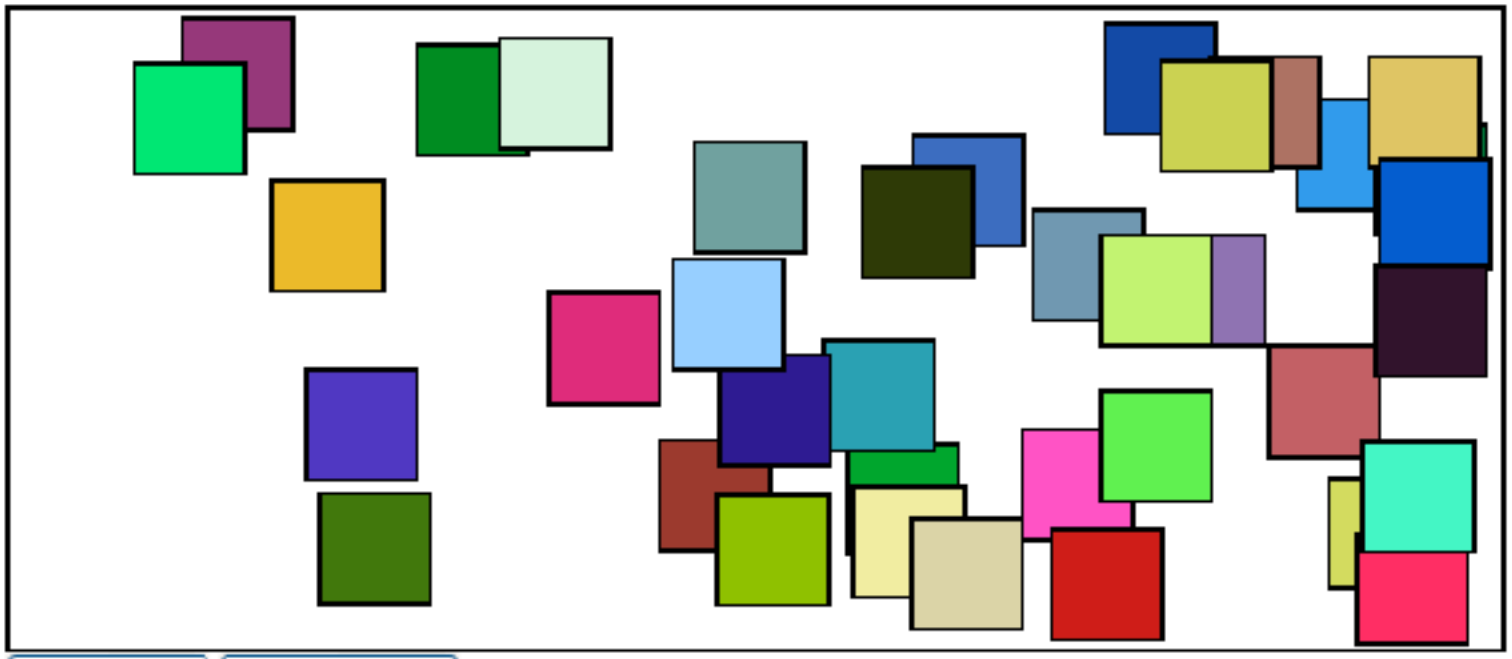
File definitivi

- Tutti i file sono in `cs03-js`
  - `hangman.js`
  - `hangman.html`
  - `hangman.css`
  - `hangman0.gif, ..., hangman6.gif`

- Esercizi JS
- Caso di studio: Hangman
- **Caso di studio: Colored Squares**

Caso di studio: colored squares

- Pagina web che visualizza una serie di quadrati colorati e posizionati a caso sulla pagina
- Possono sovrapporsi tra di loro
- Quando l'utente clicca su un quadrato, esso passa in primo piano
- Quando l'utente fa doppio click su un quadrato parzialmente nascosto (o click singolo su un quadrato già in primo piano), allora il quadrato viene cancellato
- Ci sono anche dei pulsanti che consentono all'utente di aggiungere un nuovo quadrato o cambiare colore a quelli esistenti



Click a square to move it to the front. Click again to delete it.

Passi

- Creare il codice HTML/CSS di base
- Usare codice JS per creare dei quadrati con posizioni e colori random
- Aggiungere i pulsanti per aggiungere i quadrati e cambiare i colori ai quadrati esistenti
- Gestire il click ed il doppio click sui quadrati per portarli in primo piano o cancellarli

Codice HTML/CSS

```
<div id="squarearea"></div>

<div>
  <button id="add">Add Square</button>
  <button id="colors">Change Colors</button>
</div>

<p>Click a square to move it to the front. Click again to delete it.</p>
```

```
.square {
  width: 50px;
  height: 50px;
  border: 2px solid black;
  position: absolute;
}

#squarearea {
  width: 700px;
  height: 300px;
  border: 2px solid black;
  position: relative;
}
```


Add Square
Change Colors
Click a square to move it to the front. Click again to delete it.

Creare i quadrati

Quanti quadrati creaiamo?

```
const squareCount = parseInt(Math.random() * 21) + 30;
```

Creiamo un quadrato (un div cui applicheremo un dato stile)

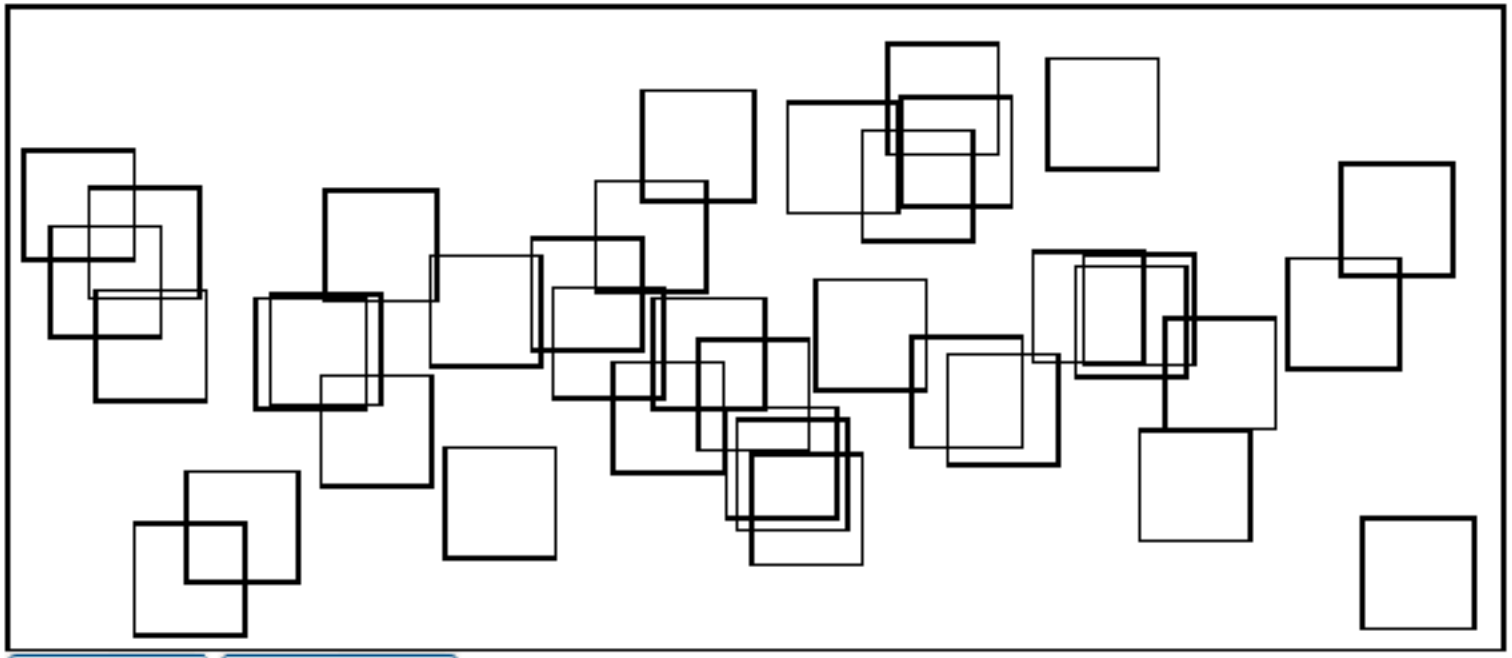
```
const square = document.createElement("div");
square.className = "square";
```

Per posizionare il quadrato, dobbiamo riferirci all'area 700x300 che deve contenerli tutti. Ogni quadrato ha una dimensione di 50px: possiamo spostarci dal lato sx dell'area da 0 a 650 px e dall'alto da 0 a 250

```
square.style.left = parseInt(Math.random() * 650) + "px";
square.style.top = parseInt(Math.random() * 250) + "px";
```

Gestione del window.onload

```
window.onload = function() {
  const squareArea = document.getElementById("squarearea");
  const squareCount = parseInt(Math.random() * 21) + 30;

  for (let i = 0; i < squareCount; i++) {
    const square = document.createElement("div");
    square.className = "square";
    square.style.left = parseInt(Math.random() * 650) + "px";
    square.style.top = parseInt(Math.random() * 250) + "px";
    squareArea.appendChild(square);
  }
};
```



Click a square to move it to the front. Click again to delete it.

Generare un colore random

Questa funzione restuisce stringhe casuali del tipo: "#f08a7c"

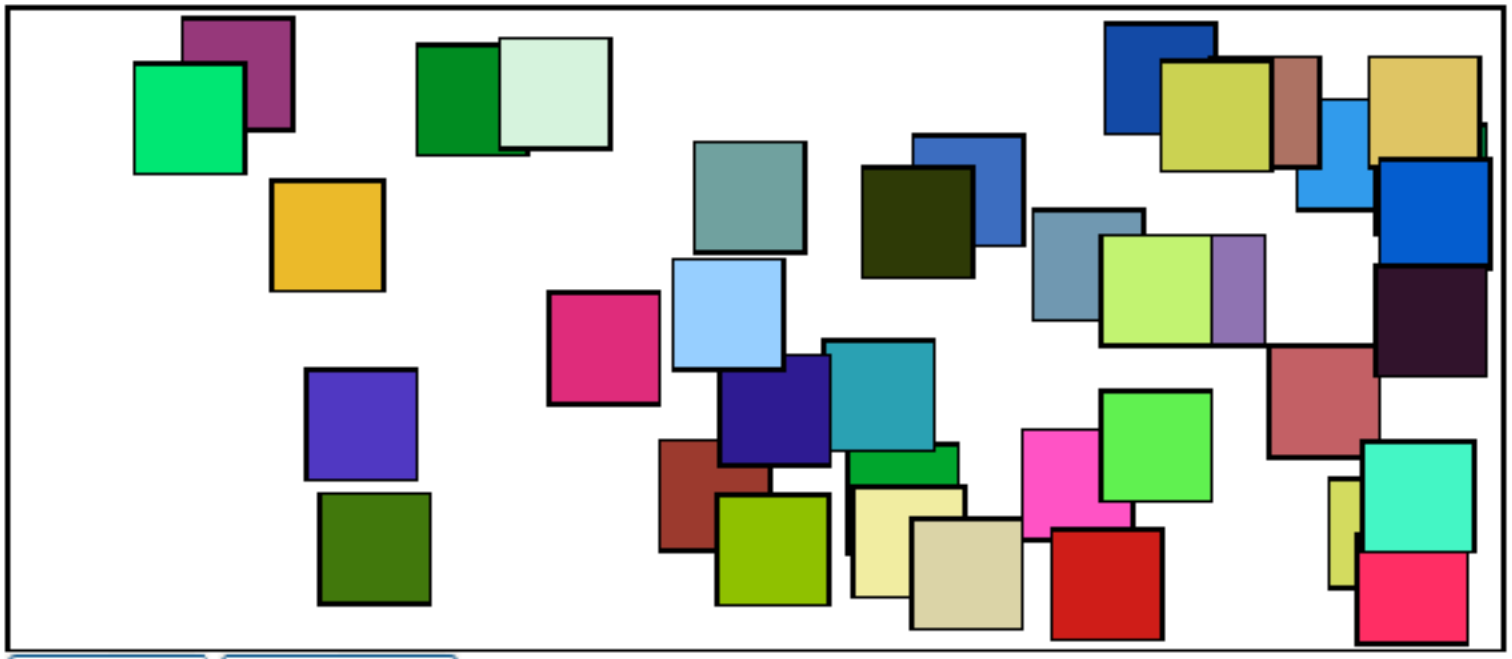
```
function getRandomColor() {
  var letters = "0123456789abcdef";
  var result = "#";
  for (var i = 0; i < 6; i++) {
    result += letters.charAt(parseInt(Math.random() * letters.length));
  }
  return result;
}
```

```
square.style.backgroundColor = getRandomColor();
```

Nota: Possiamo fare a meno del for ed usare higher-order function

usare higher-order function quando possibile!

```
function getRandomColor() {
   const letters = "0123456789abcdef";
   const result = Array
     .from({ length: 6 },
       () => letters.charAt(parseInt(Math.random() * letters.length)))
     .join("");
   return `#${result}`;
 }
```



Click a square to move it to the front. Click again to delete it.

Aggiungiamo quadrati

Il codice per aggiungere un quadrato è al momento all'interno del gestore del window.onload – dobbiamo scrivere una funzione a parte

```
function addSquare() {
  const square = document.createElement("div");
  square.className = "square";
  square.style.left = parseInt(Math.random() * 650) + "px";
  square.style.top = parseInt(Math.random() * 250) + "px";
  square.style.backgroundColor = getRandomColor();

  var squareArea = document.getElementById("squarearea");
  squareArea.appendChild(square);
};
```

Cambiamo colore

Lo stesso dicasi per la gestione del cambio del colore

```
function changeColors() {
  const squareArea = document.getElementById("squarearea");
  const squares = squareArea.getElementsByTagName("div");
  for (let i = 0; i < squares.length; i++) {
    squares[i].style.backgroundColor = getRandomColor();
  }
}
```

Oppure (grazie ad HTML5 e forEach):

```
function changeColors() {
  const squares = document.querySelectorAll("#squarearea div");
  squares.forEach((square)=>{
      square.style.backgroundColor = getRandomColor();
    });
}
```

Modifichiamo il comportamento su window.onload

```
window.onload = function() {
  const add = document.getElementById("add");
  add.onclick = addSquare;
  const colors = document.getElementById("colors");
  colors.onclick = changeColors;

  // create several randomly positioned squares
  const squareCount = parseInt(Math.random() * 21) + 30;
  for (let i = 0; i < squareCount; i++) {
    addSquare();
  }
};
```

Quadrati interattivi

Adesso, portiamo i quadrati in primo piano al singolo click. Per prima cosa, aggiungiamo un comportamento al singolo click

```
function addSquare() {
  const square = document.createElement("div");

  ...

  square.onclick = squareClick;

  const squareArea = document.getElementById("squarearea");
  squareArea.appendChild(square);
};
```

Quadrati interattivi (2)

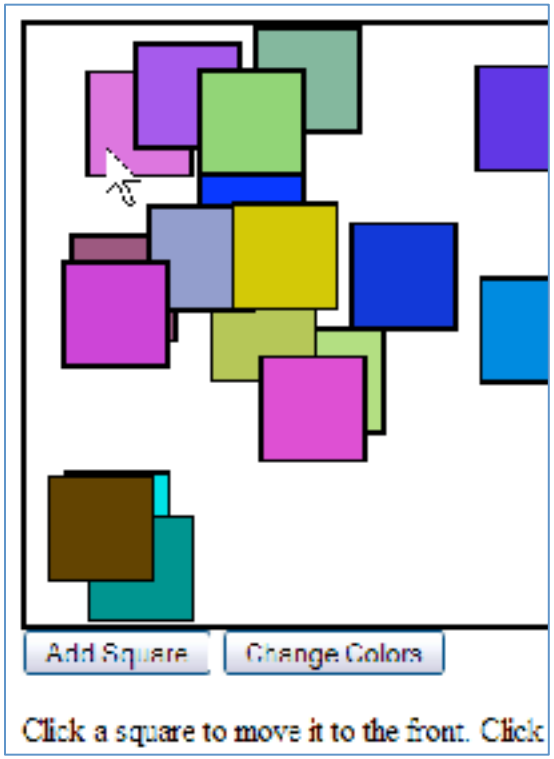
Usiamo la proprietà z-index del quadrato che è stato cliccato

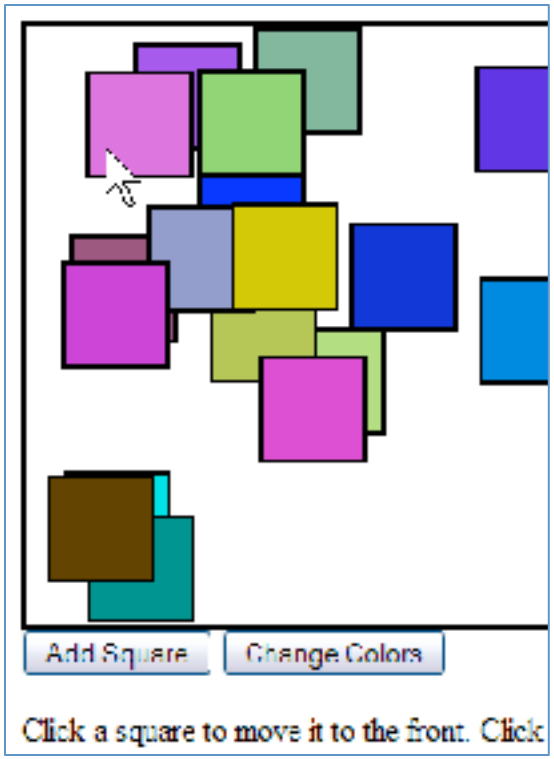
```
let maxZ = 1000; // z-index of rectangle that gets clicked


function squareClick() {
  maxZ++;
  this.style.zIndex = maxZ;
}
```

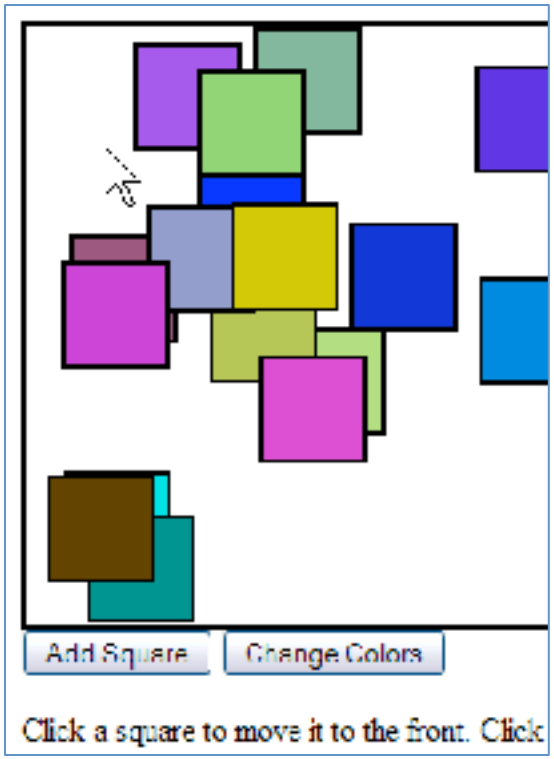
Quadrati interattivi (3)

A questo punto possiamo gestire anche la cancellazione al doppio click (o click singolo sul quadrato già in primo piano)

```
let maxZ = 1000; // z-index of rectangle that gets clicked


function squareClick() {
  const oldZ = parseInt(this.style.zIndex);
  if (oldZ == maxZ) {
        // square is on top; remove it
    this.parentNode.removeChild(this);
  } else {
    maxZ++;
    this.style.zIndex = maxZ;
  }
}
```


Add Square
Change Colors
Click a square to move it to the front. Click


Add Square
Change Colors
Click a square to move it to the front. Click


Add Square
Change Colors
Click a square to move it to the front. Click

File definitivi

- Tutti i file sono in `cs04_js2`
  - `coloredsquares.js`
  - `coloredsquares.html`
  - `coloredsquares.css`