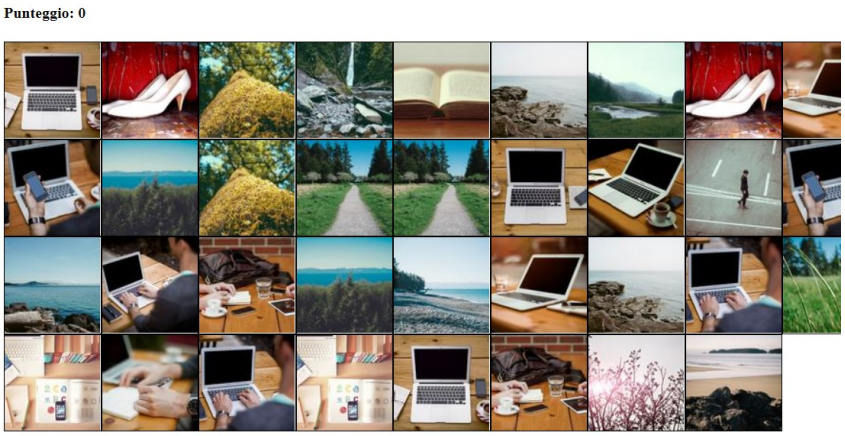
Laboratorio 9 – Martedì 01/12/2020

Gioco Memory con AJAX

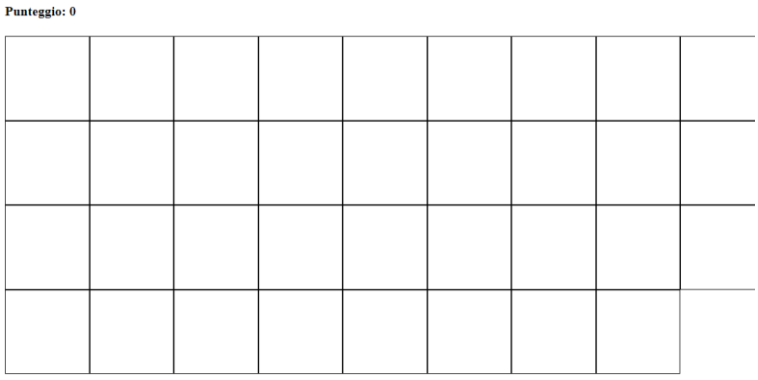
- **Regole gioco**:
  - Il gioco memory consiste nell'osservare, per un periodo limitato, le carte disposte su un tavolo. Le carte disposte sul tavolo consistono in coppie di carte uguali.
  - A un certo punto le carte vengono girate. Il giocatore deve ricordarsi la disposizione delle carte e trovare il maggior numero di coppie possibili.
- **Lorem ipsum e Lorem Picsum**:
  - *Lorem ipsum* consiste in testo in lingua pseudo-latina con cui possiamo riempire i paragrafi del nostro sito. Possiamo generare testo lorem ipsum attraverso il sito https://www.lipsum.com/
  - *Lorem Picsum* è un sito (https://picsum.photos/) con un grande numero di immagini randomiche. Possiamo:
    - Impostare lunghezza e larghezza delle immagini
    - Scegliere se averle a colori o in bianco e nero
    - Scegliere se averle sfocate o meno

  Utilizzeremo le immagini di *Lorem Picsum* per ricreare il gioco di Memory.

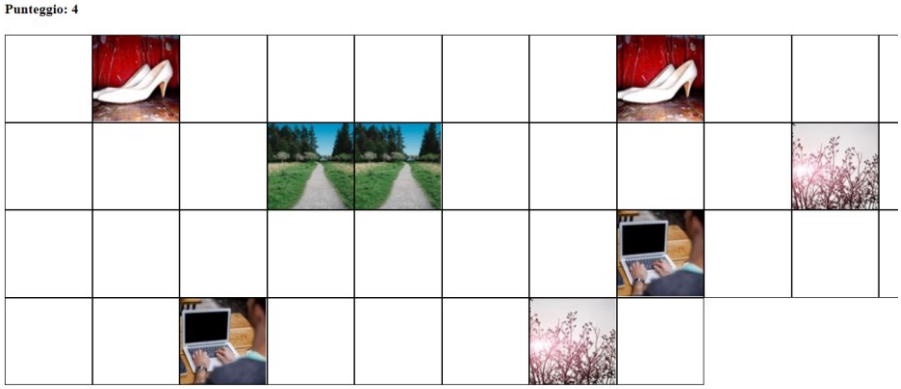
- **Anteprima del gioco**:
  - All'inizio



  - Dopo la scomparsa delle carte



  - Dopo aver indovinato delle carte



In questa versione possiamo continuare a indovinare coppie liberamente: il punteggio viene incrementato in caso di coppia individuata e lasciato inalterato in caso di errore (il gioco non si conclude in caso di errore). Se la coppia scelta è errata le immagini verranno nuovamente nascoste!

- Il gioco presenta i seguenti file:
  - `index.php`, la pagina che visitiamo col browser per giocare
  - `get_id.php`, file con cui interagiremo via AJAX.
  - Un file nella cartella js
    - `ajax_jquery.js`, scriveremo il codice in jQuery.

- index.php

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>TEST AJAX</title>
  <style type="text/css">
      .square {
            width: 100px;
            height: 100px;
            float: left;
            border: 1px solid black;
            text-align: center;
      }
  </style>
```

Definisco la grafica del "quadratino":
- Lunghezza e larghezza di 100px
- `float:left` per creare righe di box disposti accanto
- Un bordo di colore nero, continuo (*solid*), avente spessore di 1px
- Il testo è centrato

Includo i file JS da utilizzare (il file col mio codice jQuery e la libreria jQuery)

```
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
  <script type="text/javascript" src="./js/ajax_jquery.js"></script>
</head>

      <body>
```

Stampa di 50 box, ciascuno identificato da un numero. Attenzione: il numero qua presente è diverso dal numero identificativo delle immagini su LoremPicsum. I numeri qua considerati sono relativi agli indici degli elementi dell'array `$_SESSION['random']`

```
            <h4>Punteggio: 0</h4>
            <div id="container">
            <?php
            foreach (range(0,49) as $key => $value) {
                  print "<div class='square' id='".$value."'>".$value."</div>";
            }
            ?>
            </div>
      </body>
</html>
```

Identifico ogni elemento HTML col numero corrispondente

- get_id.php

Se visitiamo la pagina per la prima volta dobbiamo inizializzare l'array contenente gli identificativi numerici delle immagini di *LoremPicsum*. Creiamo un array unico a partire da due array generati con `range` (li fondiamo con `array_merge`). Gli indici dell'array unico vanno da 0 a 49.

```
<?php
session_start();

if (!isset($_SESSION['random'])) {
      $array = array_merge(range(1, 25), range(1,25));
      shuffle($array);
      $_SESSION['random'] = $array;
}
```

Alteriamo l'ordine degli array in modo casuale

Salviamo l'array generato in una variabile $_SESSION

```
$res = array();
$res['id'] = $_GET['id'];
$res['value'] = $_SESSION['random'][$_GET['id']];
print json_encode($res);
?>
```

Generiamo l'oggetto da stampare in JSON. L'id consiste nell'identificativo dell'elemento *square* (cioè nell'indice di un elemento dell'array), value nell'identificativo dell'immagine di LoremPicsum

- ajax jquery.js

```
var tentative_id = -1;
var tentative_value = -1;
var score = 0;
```

Identificativo del secondo elemento della coppia

```
function reset(id) {
      // Nascondo le immagini della coppia
      $("#"+tentative_id).empty();
      $("#"+id).empty();

      // Re-inizializzo le variabili
      tentative_id = -1;
      tentative_value = -1;
}
```

Inizializziamo le variabili che ci serviranno per gestire il controllo delle coppie scelte dall'utente.

- Con `tentative_id` salvo l'identificativo del primo elemento della coppia che l'utente ha selezionato.
- Con `tentative_value` salvo l'identificativo dell'immagine di Lorem Picsum.
- Con `score` memorizzo il punteggio dell'utente

Se entrambe le variabili hanno come valore -1 significa che non abbiamo ancora selezionato il primo elemento della coppia.

Inizializzo la funzione `reset`() quando la coppia scelta non è corretta

L'immagine viene stampata con elemento img all'interno dell'elemento square.
Per nascondere l'immagine mi basta cancellare il contenuto dell'elemento square.

```
function showImg(id, test) {
      $.getJSON("get_id.php?id="+id, function(result){
            var img = "<img src='https://picsum.photos/id/"+result.value+"/100' />";
            $("#"+result.id).html(img);

            if (test) { Primo elemento della coppia
                  if (tentative_id == -1) {
                        // sono alla prima carta da scoprire
                        tentative_id = result.id;
                        tentative_value = result.value;
                  }
                  Secondo elemento della coppia – coppia giusta
                  else if (tentative_value == result.value) {
                        // sono alla seconda carta ed ha lo stesso valore
                        tentative_id = -1;
                        tentative_value = -1;
                        score++;
                        $('h4').html("Punteggio: "+ score);
                  }
                  else { Secondo elemento della coppia – coppia errata
                        //sono alla seconda carta e non ho indovinato
                        setTimeout("reset("+result.id+")", 3000);
                  }
            }
      });
}
```

Resetto i valori di tentative_id e tentative_value, incremento score e aggiorno il punteggio nell'HTML.

La funzione `showImg` viene eseguita in due occasioni:

- Quando carichiamo la pagina index.php e dobbiamo mostrare le immagini (parametro `test = false`).
- Quando selezioniamo gli elementi delle coppie (parametro `test = true`)

```
$(document).ready(function(){
      console.log("START");

      for (var i=0;i<50;i++) {
            showImg(i, false);
      }
      setTimeout(function(){
            $(".square").empty();
      }, 5000);

      $(".square").click(function(event){
            console.log("CLICK", event.target.id);
            showImg(event.target.id, true);
      });
});
```

Stampa delle immagini quando carichiamo la pagina

Nascondiamo le immagini dopo 5000 millisecondi (5 secondi)

Associamo l'esecuzione della funzione `showImg` al click di un qualunque elemento `square`.

Libretto universitario con accesso a database ed AJAX

- Riprendiamo il solito esercizio e modifichiamolo: invece di salvare attraverso variabili `$_SESSION` utilizziamo un database MySQL. Inseriamo gli elementi sfruttando AJAX e jQuery[1]
- Ometto nella spiegazione il file *sql* di creazione del database (lo trovate nello zip indicato a inizio nella dispensa). Tesconi ha creato le tabelle utilizzando phpMyAdmin: vi sconsiglio di farlo nella creazione del progetto (non dimenticate tutte le regole di buon senso imparate a *Basi di dati* per avere un database consistente e privo di ridondanze).
- Altro aspetto problematico di questo codice è la scarsa modularità dello stesso: Tesconi ripete in ogni pagina il codice per connettersi al database (in particolare, se modifichiamo i dati di accesso al database siamo costretti a modificare ogni singolo file). Fare una cosa del genere nel progetto comporta perdere punti relativamente all'organizzazione e alla modularità.

- Il sistema si articola su più file:
  - `index.html`, pagina di login (presente form che rimanda ad `authenticate.php`)
  - `authenticate.php`, pagina visitata dopo la sottomissione della form per fare login
  - `libretto.php`, pagina accessibile solo a coloro che hanno fatto login (contenuto del libretto)
  - `insert.php`, pagina che AJAX utilizza per inserire nuove valutazioni nel libretto.
  - `logout.php`, pagina per fare logout
  - `stats.php`, pagina contenente le funzioni utilizzate nello scorso esercizio sul libretto universitario (non le riscrivo, sono la fotocopia di quelle già viste)
  - Una cartella js con un file:
    - `ajax_jquery.js`, codice Javascript (in jQuery) per inserire nuovi elementi nel libretto. Ribadisco che il codice è incompleto.

- `index.html`

C'è poco da dire su questa pagina.

```
<!DOCTYPE html>
<html>
<head>
   <meta charset="utf-8">
   <title>Login</title>
</head>
<body>
   <div class="login">
      <h1>Login</h1>
         <form action="authenticate.php" method="post">
         Username
         <input type="text" name="username" placeholder="Username" id="username" required>

         <br>

         Password
         <input type="password" name="password" placeholder="Password" id="password" required>

         <input type="submit" value="Login">
         </form>
      </div>
</body>
</html>
```

- `authenticate.php`

```
<?php
session_start();
// Change this to your connection info.
// code by https://codeshack.io/secure-login-system-php-mysql/

$DATABASE_HOST = 'localhost';
```

[1] Attenzione, la parte è incompleta. L'inserimento di un nuovo elemento avviene in background, ma la tabella contenente le valutazioni e quella contenente le statistiche non viene aggiornata (necessario il refresh della pagina)

```
$DATABASE_USER = 'root';
$DATABASE_PASS = '';
$DATABASE_NAME = 'libretto';

$con = mysqli_connect($DATABASE_HOST, $DATABASE_USER, $DATABASE_PASS, $DATABASE_NAME);
if ( mysqli_connect_errno() ) {
      // If there is an error with the connection, stop the script and display the error.
      exit('Failed to connect to MySQL: ' . mysqli_connect_error());
}
```

Connessione al database con gestione di eventuali errori

Verifica dell'inserimento delle credenziali di login. Se non sono state inserite mi fermo subito.

```
// Now we check if the data from the login form was submitted, isset() will check if the data exists.
if ( !isset($_POST['username'], $_POST['password']) ) {
      // Could not get the data that should have been sent.
      exit('Please fill both the username and password fields!');
}
```

Quanto segue è un metodo alternativo per eseguire una query filtrando i parametri (tratto l'username come una stringa, quindi lo sanitizzo impedendo SQL Injections.

```
// Prepare our SQL, preparing the SQL statement will prevent SQL injection.
if ($stmt = $con->prepare('SELECT id, password FROM accounts WHERE username = ?')) {
      // Bind parameters (s = string, i = int, b = blob, etc), in our case the username is a string so we use "s"
      $stmt->bind_param('s', $_POST['username']);
      $stmt->execute();

      // Store the result so we can check if the account exists in the database.
      $stmt->store_result();

      if ($stmt->num_rows > 0) {
            $stmt->bind_result($id, $password);
            $stmt->fetch();

            if (password_verify($_POST['password'], $password)) {
                  session_regenerate_id();
                  $_SESSION['loggedin'] = TRUE;
                  $_SESSION['username'] = $_POST['username'];
                  $_SESSION['account_id'] = $id;
                  header('Location: libretto.php');
            } else {
                  // L'utente esiste ma la password è errata
                  echo 'Incorrect username and/or password!';
            }
      } else {
            // Non esiste un utente con l'username indicato
            echo 'Incorrect username and/or password!';
      }
      $stmt->close();
}
?>
```

Redirect verso la pagina `libretto.php`

Non indicate mai all'utente il dato sbagliato!!!

La parte rimanente del codice è molto simile a quella già vista in un esercizio passato. Se esiste un utente con l'username indicato utilizzo la funzione `password_verify` per confrontare la password inserita con la password salvata nel database. Se c'è corrispondenza modifico le variabili `$_SESSION` per indicare il login avvenuto.

- libretto.php

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>LIBRETTO UNIVERSITARIO</title>
  <style>
      form > span {
```

```
            display: block;
            float: left;
            width: 70px;
        }
    </style>
```

Includo i file js necessari (il file col nostro codice jQuery e la libreria jQuery)

```
    <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
    <script type="text/javascript" src="js/ajax_jquery.js"></script>
</head>
<body>
      <?php
      include('stats.php');
      session_start();

      // Verifico se l'utente ha fatto login, se non lo ha fatto lo reindirizzo al login
      if (!isset($_SESSION['loggedin'])) {
            header('Location: index.html');
            exit;
      }

      // Connessione al database
      $mysqli = new mysqli("localhost", "root", "", "libretto");
      if (mysqli_connect_errno()) {
          printf("Connect failed: %s\n", mysqli_connect_error());
          exit();
      }
```

Recupero con una query i voti inseriti nel registro dell'utente. Riempio un array $voti

```
      $query = "SELECT Materia, Voto FROM voti WHERE account_id =".$_SESSION['account_id'];
      $result = $mysqli->query($query);
      $voti = array();
      while($row = $result->fetch_array()) {
         $voti[$row['Materia']] = $row['Voto'];
      }
      print("<h3>Benvenuto ".$_SESSION['username']."!</h3>")
      ?>

      <a href="logout.php">Logout</a>
```

Anchor alla pagina di logout.

```
      <h3>Libretto</h3>
      <form action="#">
```

<u>Form per l'aggiunta di nuovi voti</u>

```
            <span>Materia:</span>
            <input id="Materia" type="text" autofocus></input><br>
            <span>Voto:</span>
            <input id="Voto" type="text" ></input><br>
            <button id="Inserisci">Inserisci</button>
      </form>
      <br>
```

Stampo i voti recuperandoli dall'array $voti creato prima

```
      <table id="table_voti" border="1">
            <tr>
               <th>Materia</th>
               <th>Voto</th>
            </tr>
            <?php
            foreach ($voti as $key => $value) {
                  print "<tr><td>".$key."</td><td>".$value."</td></tr>";
            }
            ?>
      </table>

      <br>
```

```
      <?php if (count($voti)>0) : ?>

      // Parte omessa: fotocopia dell'esercizio precedente sul libretto universitario

      <?php
      endif;

      // Chiusura della connessione col database. Consigliato chiudere la connessione quando non più necessaria.
      $result->close();
      ?>
</body>
</html>
```

Stampo gli errori (o l'esito dell'inserimento) usando JSON. In questo modo Javascript può sapere se l'inserimento ha avuto successo o eventualmente stampare gli errori che si sono manifestati.

- insert.php

```
<?php
session_start();

if (!isset($_SESSION['loggedin'])) {
      $errore = array('errore' => 'utente non loggato');
      print json_encode($errore);
      exit;
}

$mysqli = new mysqli("localhost", "root", "", "libretto");
if (mysqli_connect_errno()) {
    printf("Connect failed: %s\n", mysqli_connect_error());
    exit();
}
```

Anche qui bisognerebbe stampare l'errore in JSON, ma va be…

```
if (isset($_GET['Materia'], $_GET['Voto']) ) {
      $Materia = $_GET['Materia'];
      $Voto = $_GET['Voto'];
      $account_id = $_SESSION['account_id'];

      $query = "INSERT INTO libretto.voti (`id`, `Materia`, `Voto`,
`account_id`) VALUES (NULL, '{$Materia}', '{$Voto}', '{$account_id}');";

      $query_result = $mysqli->query($query);

      // La funzione $mysqli->query restituisce TRUE se uno statement INSERT ha avuto successo, FALSE altrimenti.
      $result = array('result' => $query_result);
      print json_encode($result);
}
else {
      $errore = array('errore' => 'Parametri non corretti');
      print json_encode($errore);
}
?>
```

Valori chiaramente non sanitizzati. Possibili SQL Injections

- logout.php

```
<?php
session_start();
session_destroy();
header('Location: index.html');
?>
```

Con `session_destroy()` distruggiamo la sessione. Questo significa perdere tutte le variabili `$_SESSION` create. Dopo aver fatto ciò reindirizzo l'utente alla pagina di login.

- ajax jquery.js

```
$(document).ready(function(){
      $("#Inserisci").click(function(event){
            event.preventDefault();
```

Impedisco la sottomissione della form

```
            var Materia = $('#Materia').val();
            var Voto = $('#Voto').val();

            var param = {Materia: Materia, Voto: Voto};
            $.getJSON('insert.php', param, function(result){
                  console.log(result);
            });
      });
});
```

Recupero i valori dei controlli

Creo un oggetto avente come proprietà `Materia` e `Voto`. Lo utilizzo per comunicare i dati ad `insert.php` (richiesta in background di tipo GET).

Utilizzo `console.log` per fare debugging, come al solito.