Laboratorio 8 – Mercoledì 24/11/2020

Libretto universitario

- Riprendiamo l'idea del libretto universitario realizzata con Javascript dal professore Tanganelli.
- Per i concetti di deviazione standard e mediana vedere il Laboratorio 3.
- Realizziamo la stessa cosa in PHP utilizzando le variabili `$_SESSION`:

Sono evidenziate in **grassetto** funzioni PHP molto utili da tenere in mente.

```
<!DOCTYPE html>
<html>
   <head>
      <meta charset="utf-8">
      <title>LIBRETTO UNIVERSITARIO</title>
      <style>
      form > span {
           display: block;
           float: left;
           width: 70px;
      }
      </style>
   </head>
   <body>
      <?php
```

Funzione per calcolare la deviazione standard

```
function Stand_Deviation($arr) {
  $num_of_elements = count($arr);

  $variance = 0.0;

  // calculating mean using array_sum() method
  $average = array_sum($arr)/$num_of_elements;

  foreach($arr as $i) {
      // sum of squares of differences between all numbers and means.
      $variance += pow(($i - $average), 2);
  }

  return (float)sqrt($variance/$num_of_elements);
}
```

Funzione per calcolare la mediana

```
function median($numbers=array()){
     rsort($numbers);
     $mid = (count($numbers) / 2);
return ($mid % 2 != 0) ? $numbers{$mid-1} : (($numbers{$mid-1}) + $numbers{$mid}) / 2;
}
```

```
session_start();
```

Inserimento di un voto solo se vengono indicati i due parametri necessari.

```
if (isset($_GET['materia']) && isset($_GET['voto']))  {
   $voto = intval($_GET['voto']);
   if ($voto >= 18 && $voto<=33)
      $_SESSION['voti'][$_GET['materia']] = $voto;
   else
      print "Inserisci un num tra 18 e 33";
};
```

Converto il valore in ingresso in un intero e verifico la consistenza del voto

Aggiungo un nuovo elemento all'array avente come indice il nome della materia e come voto `$voto`

```
if (!isset($_SESSION['voti']) || isset($_GET['avvia']))  {
   $_SESSION['voti'] = array();
};
?>
```

Inizializzo la variabile come array vuoto quando visitiamo la pagina per la prima volta (e la variabile non è inizializzata) o quando lo richiediamo noi (vedere codice form)

```
        <h3>Libretto</h3>
        <form action="index.php" methods="GET">
           <span>Materia:</span>
           <input name="materia" type="text" autofocus></input><br>

           <span>Voto:</span>
           <input name="voto" type="text" ></input><br>

           <input type="submit" value="Inserisci"></input>
           <button name="avvia" value="true">Riavvia</button>
        </form>
        <br>
```

Focus sull'input al caricamento della pagina

Bottone per re-inizializzare l'array e quindi il registro.
Ricordarsi che il controllo viene validato solo se premuto.

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

Stampiamo in una tabella la lista dei voti inseriti

```
        <br>

        <?php if (count($_SESSION['voti'])>0) : ?>
        <table border="1">
           <tr>
              <th>Statistica</th>
              <th>valore</th>
           </tr>
           <tr>
              <td>Media</td>
              <td>
              <?php
              print round(array_sum($_SESSION['voti']) / count($_SESSION['voti']), 2);
              ?>
              </td>
           </tr>
           <tr>
              <td>Mediana</td>
              <td>
                 <?php
                 print median($_SESSION['voti']);
                 ?>
              </td>
           </tr>
           <tr>
              <td>Deviazione Standard</td>
              <td>
                 <?php
                 print Stand_Deviation($_SESSION['voti']);
                 ?>
              </td>
           </tr>
        </table>
        <?php endif; ?>
     </body>
  </html>
```

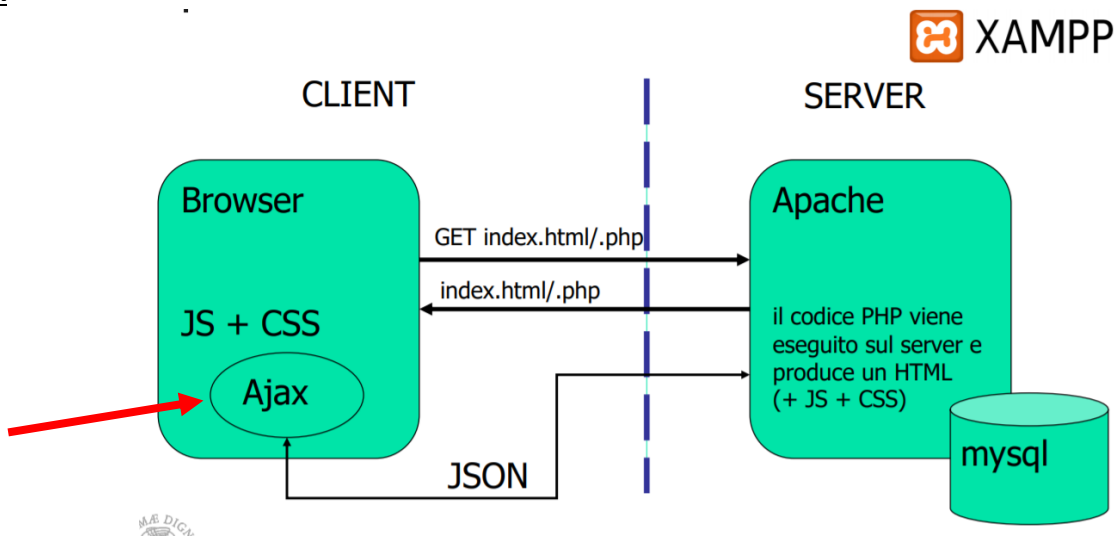
Perché nascondiamo la tabella?

- Utilizziamo funzioni che richiedono array non vuoti (se le eseguiamo vengono stampati errori PHP).
- Ha poco senso mostrare la tabella se non sono state inserite valutazioni.

Chiamata delle funzioni `median` e `Stand_Deviation` introdotte all'inizio del codice

- **Funzioni**:
  - `intval($val)`: restituisce il contenuto della variabile convertito in intero.
  - `count($array)`: restituisce il numero di elementi presenti nell'array
  - `array_sum($array)`: restituisce la somma degli elementi nell'array
  - `pow($base, $exp)`: restituisce il risultato di $\$base^{\$exp}$
  - `sqrt($arg)`: esegue la radice quadrata di `$arg`
  - `rsort($array[, $flag])` : funzione che permette di ordinare un array in ordine decrescente (dal valore più grande al più piccolo). PHP offre una grandissima varietà di funzioni per ordinare array (con possibilità di ordinare sia in base ai valori sia in base agli indici degli elementi)
  - `round($val[, $precision])`: arrotondamento di un valore (per eccesso o per difetto in base al numero), con una certa precisione.

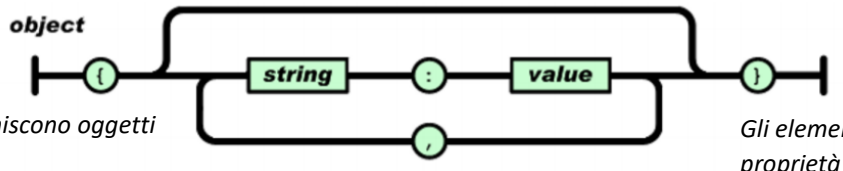
AJAX



- Riprendiamo l'immagine vista nel laboratorio precedente sull'architettura Client/Server e introduciamo uno step in più.
- AJAX consiste in codice Javascript con cui possiamo svolgere richieste in background.
- **Vincolo**: non posso fare richieste in background a server diversi da quello che ospita la pagina richiedente (per ragioni di sicurezza).

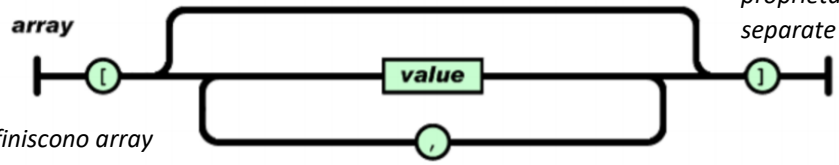
- Le funzioni con cui svolgiamo queste richieste in background restituiscono la risposta. La risposta può essere di tipo HTML, plain text, XML o JSON…. Molti anni fa XML (acronimo di *eXtensible Markup Language*) era la via maestra per gestire queste risposte: nel tempo è stato soppiantato dal JSON.
- **JSON** è acronimo di *JavaScript Object Notation* e presenta una notazione molto semplice:

Le proprietà di un oggetto sono definite da una coppia (nome-valore)

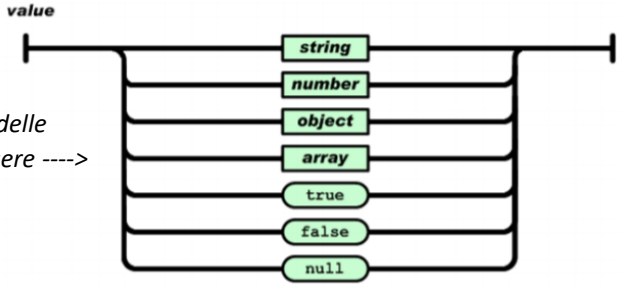


Le parentesi graffe definiscono oggetti

Gli elementi di un array o le proprietà di un oggetto sono separate da virgola

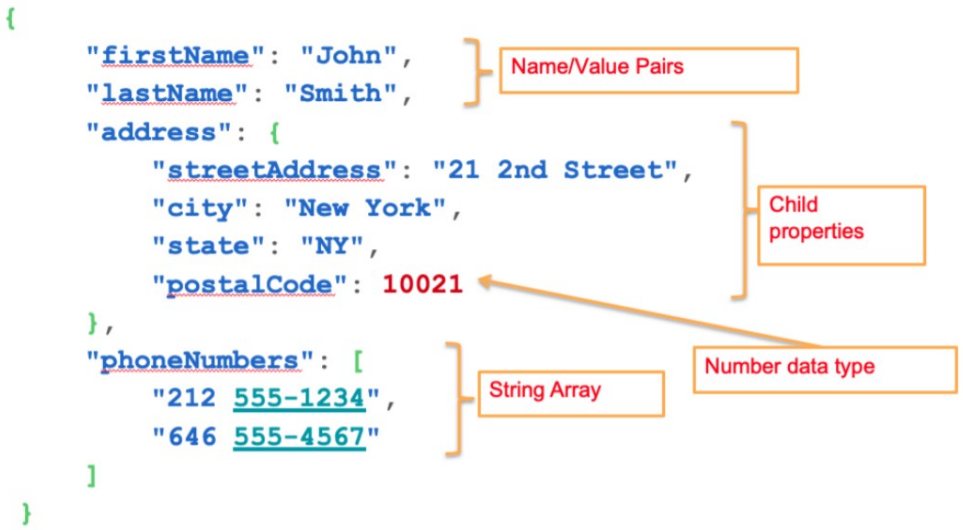


Le parentesi quadre definiscono array



I valori degli elementi di un array o delle proprietà di un oggetto possono essere ---->

- **Esempio JSON**:



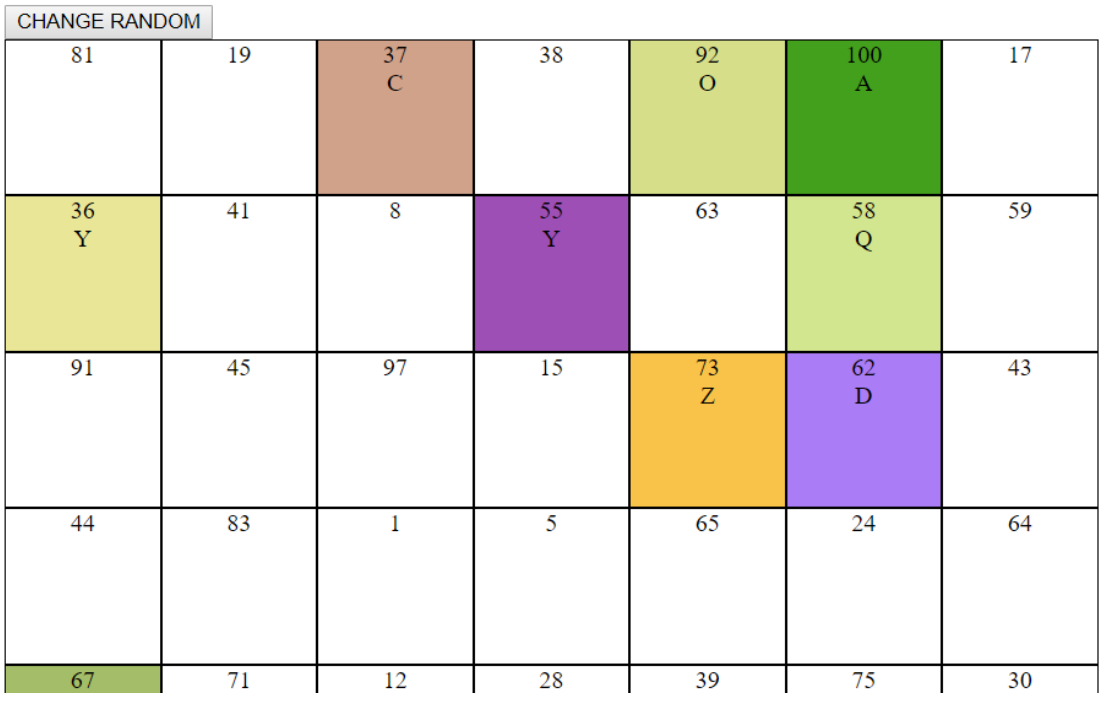
Suggerimento

- Nel caso in cui si lavori con una grande quantità di dati (quindi un JSON di dimensioni modeste) conviene utilizzare un validatore online.
- Scrivete *JSON Validator* su Google e selezionate quello che più vi aggrada.

Esercizio AJAX

- Implementare una pagina PHP che crea 100 div quadrati (100px) con ID da 1 a 100 in modo randomico.
- Fare una chiamata Ajax che restituisca in modo casuale:
  - Un numero da 1 a 100
  - Un colore da #000000 a #FFFFFF
  - Una lettera da A-Z
- La callback (cioè la funzione eseguita quando si riceve una risposta dal server) usa il numero per selezionare l'ID (cioè il div dove intervenire), cambia il colore di sfondo e inserisce la lettera nel quadrato.

- Anteprima del codice dopo aver premuto il bottone in alto svariate volte:



- Le esercizio si articola in una serie di file:
  - Il file `index.php` che contiene la tabella generata da PHP con numeri random
  - Il file `get_random.php` a cui rivolgiamo le nostre richieste con AJAX. È realizzato in modo tale da restituire una risposta in formato JSON.
  - Due file nella cartella `js`:

- `ajax.js` (versione Javascript classica della parte js dell'esercizio)
- `ajax_jquery.js` (versione jQuery della parte js dell'esercizio)

Perché questi due file? Il docente ha utilizzato questo esercizio per introdurci *jQuery*, una libreria Javascript nata con l'idea di semplificare selezione, manipolazione e gestione degli eventi. jQuery, contrariamente al classico Javascript, gestisce la compatibilità tra browsers (in Javascript classico dobbiamo scrivere noi la gestione della compatibilità).

- 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 (uno tra i due introdotti prima 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.js"></script>
  <!--script type="text/javascript" src="./js/ajax_jquery.js"></script-->
</head>
      <body>
```

Bottone associato alla chiamata AJAX (vedere codice js)

```
<button id="get_random">CHANGE RANDOM</button>
```

Stampa dei 100 box randomici. Con `range` creo un array di elementi aventi valori da 1 a 100 (in ordine). Con `square` scombino l'ordine dei numeri.

```
<div id="container">
<?php
$array = range(1, 100);
shuffle($array);
foreach ($array as $key => $value) {
     print "<div class='square' id='".$value."'>".$value."</div>";
}
?>
</div>
```

Identifico ogni elemento HTML col numero corrispondente

```
      </body>

</html>
```

- get_random.php

```
<?php
function random_color_part() {
   return str_pad(dechex(rand(0,255)),2,'0',STR_PAD_LEFT);
}
```

Il colore si indica con notazione RGB caratterizzata da cancelletto e una serie di numeri esadecimali. La funzione `random_color_part` (copiata e incollata da Tesconi) restituisce due cifre esadecimali.

Prendo un numero randomico tra 0 e 255 e lo converto in esadecimale (si va da 0 a FF). Con `str_pad` obbligo il valore ad essere lungo due cifre (la funzione aggiunge uno zero se il valore restituito da `dechex` è compreso tra 0 e 9).

```
function random_color() {
   return "#".random_color_part() . random_color_part() .
random_color_part();
}
```

Restituisco una concatenazione tra cancelletto e i risultati di tre chiamate di `random_color_part()`. Otterremo un numero esadecimale di sei cifre!

```
$res = array();
$res['number'] = rand(1,100);
$res['color'] = random_color();
$res['char'] = chr(rand(65,90));

print json_encode($res);
?>
```

Genero un numero random tra 65 e 90. Con `chr` ottengo un carattere.

Vogliamo che la pagina stampi testo in notazione JSON. La pagina index.php visiterà questa pagina e riceverà come risposta quello che stiamo generando con questo codice! La stampa in notazione JSON avviene grazie alla funzione `json_encode`.

Sia oggetti che array in JSON sono generabili a partire da array PHP. La funzione riconosce come array gli array numerici e come oggetti gli array associativi.

- ajax.js

```
var XMLHttpFactories = [
    function () {return new XMLHttpRequest()},
    function () {return new ActiveXObject("Msxml3.XMLHTTP")},
    function () {return new ActiveXObject("Msxml2.XMLHTTP.6.0")},
    function () {return new ActiveXObject("Msxml2.XMLHTTP.3.0")},
    function () {return new ActiveXObject("Msxml2.XMLHTTP")},
    function () {return new ActiveXObject("Microsoft.XMLHTTP")}
];

function createXMLHTTPObject() {
    var xmlhttp = false;
    for (var i=0;i<XMLHttpFactories.length;i++) {
        try {
            xmlhttp = XMLHttpFactories[i]();
        }
        catch (e) {
            continue;
        }
        break;
    }
    return xmlhttp;
}
```

Codice copiato e incollato da Tesconi (roba presa da internet) per gestire la compatibilità.

La funzione sarà richiamata all'interno di `get_random()`

```
function get_random() {
     // Initialize the HTTP request.
     var xhr = createXMLHTTPObject(); //  new XMLHttpRequest();
     xhr.open('GET', 'get_random.php');

     // Track the state changes of the request.
     xhr.onreadystatechange = function () {
          var DONE = 4; // readyState 4 means the request is done.
          var OK = 200; // status 200 is a successful return.
          if (xhr.readyState === DONE) {
               if (xhr.status === OK) {
                    console.log(JSON.parse(xhr.responseText));
                    res = JSON.parse(xhr.responseText);
                    console.log(res.number);
                    var div = document.getElementById(res.number);
```

```
                        div.style.backgroundColor = res.color;
                        div.innerHTML = res.number + "<br>" + res.char;
                    } else {
                        console.log('Error: ' + xhr.status);
                    }
                }

        };

        // Send the request to send-ajax-data.php
        xhr.send(null);
}
```

Anche questo codice copiato e incollato. Con la funzione apriamo una richiesta verso `get_random.php`. Elementi interessanti nel codice:

- L'utilizzo di `console.log()` per fare debugging
- L'utilizzo di `JSON.parse` per gestire la *responseText* come JSON. L'elemento ottenuto può essere manipolato esattamente come un oggetto.

```
document.addEventListener('DOMContentLoaded', function() {
            document.getElementById("get_random").addEventListener("click",
get_random);
            var elements = document.getElementsByClassName("square");
            //console.log(elements);
            }

});
```

Associamo l'esecuzione della funzione get_random() al click del bottone `<button id="get_random">CHANGE RANDOM</button>`.

Facciamo questa cosa solo quando il DOM sarà caricato in modo completo (scelta che conviene soprattutto quando abbiamo documenti con un DOM di dimensione considerevole). Questo mi permette di evitare *giochi strani* da parte del codice.

- ajax_jquery.js

```
$(document).ready(function(){
      $("#get_image").click(function(){
            $.getJSON("get_random.php", function(result){
            $("#"+result.number).css("background-color", result.color);
            $("#"+result.number).text(result.number + "<br>" + result.char);
            });
      });
});
```

La differenza evidente è il numero decisamente minore di righe: il risultato è **LO STESSO**!!!

La sintassi è a mio parere piuttosto semplice da capire (in particolare si osservi che ogni funzioni jQuery è introdotta da un dollaro). Altra differenza sostanziale, non presente in questo codice, è la possibilità di cambiare proprietà del CSS a intere classi con una semplice riga (in Javascript dovrei utilizzare la `getElementsByClassName` e modificare ogni singolo elemento utilizzando un for).

Esempio: `$(".square").css("background-color","yellow");`

Raccomandazione conclusiva: Tesconi ha introdotto *jQuery*, ma Marcelloni non è molto d'accordo sul suo utilizzo. NON UTILIZZATELO NELLE PROVE PRATICHE. Nei progetti potete usarlo, ma solo per cose semplici (per Marcelloni *jQuery* ci semplifica troppo la vita, dobbiamo prima capire Javascript).