Daniele78

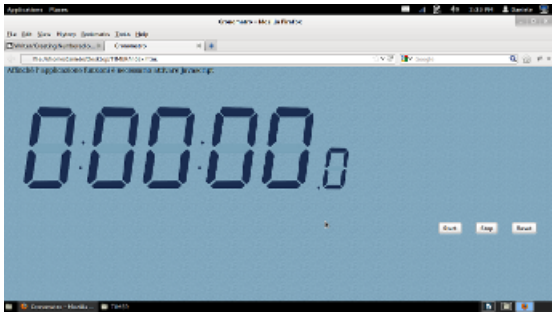
OROLOGIO + CRONOMETRO DIGITALE SU PC 2A PARTE

16 July 2012

In questa seconda parte mi occuperò di descrivere la parte dinamica dell' applicazione. Spiegherò il tutto dando per scontato che chi legge abbia già esperienze di programmazione anche se non per il web. Chi conosce ad esempio il linguaggio C non dovrebbe avere problemi.

Nella prima parte ho spiegato il codice del file index.html per la visualizzazione del display, insieme al foglio di stile associato ad esso e chiamato jqClock.css

Utilizzando solo questi 2 file, il risultato che otteniamo è il seguente:



Screenshot 2.png

Adesso parlerò degli script che vanno a modificare dinamicamente i contenuti della pagina.

In pratica si tratta di rimpiazzare la stringa in alto che chiede l' attivazione di Javascript con del codice html mostrante data e ora correnti.

Un' altra modifica che lo script dovrà effettuare, è la modifica delle immagini del display in modo da visualizzare il conteggio esattamente come avverrebbe con un cronometro reale. Per la realizzazione, ho utilizzato JQUERY. Si tratta di un framework che converte runtime del codice scritto in maniera molto elegante, in JS. Per poter utilizzare Jquery occorre linkare un particolare file inserendo nel blocco <head></head> della nostra pagina index.html il seguente link:

```
<script type="text/javascript" src="jquery-1.7.2.min.js"></script>
```

src="jquery-1.7.2.min.js" proprio perchè il file query-1.7.2.min.js si trova nella stessa directory di index.html

Nell' applicazione per visualizzare data e ora correnti è stato utilizzato un plugin JQuery. Questo è il link dal quale ho scaricato il file jqClock.js: https//github.com/Lwangaman/jQuery-Clock-Plugin

Nella stessa pagina è presente anche la descrizione di come utilizzarlo.

Dopo aver aggiunto il file jqClock.js nel blocco head, ho aggiunto anche il link al file che ho creato io che ho chiamato myfunction.js.

Esaminiamolo subito: al suo interno è presente il seguente blocco:

```
$(document).ready(function(){

});
```

Serve a dire: terminato il caricamento della pagina esegui tutto quello che c' è all' interno di questo blocco.

L' intero codice che ho scritto l' ho infatti inserito al suo interno.

Ho iniziato con lo scrivere:

```
var reset_html = $("#chronometer").html();
```

Le variabili in JS si dichiarano semplicemente con var. Non si specifica il tipo di variabile come avviene in altri linguaggi di programmazione. Dichiaro quindi la variabile reset_html inizializzandola con la stringa restituita da :$("#chronometer").html(); che sta a significare: restituisci tutto il codice html racchiuso dal blocco con id = "chronometer" .

Il # indica che chronometer è un id.

Il modo di referenziare dei tag usando JQuery, è identico a quello usato nei CSS.

Si scrive $("") ... inserendo tra i doppi apici ciò che si scriverebbe su un foglio di stile.

In pratica, salvo il contenuto del div contenente il codice html del cronometro in una variabile.Perchè faccio ciò?Abbiamo detto che il codice nella pagina cambierà dinamicamente e a noi interessa, alla pressione del pulsante reset, ripristinare le condizioni iniziali.Da notare che è la prima istruzione che viene eseguita terminato il caricamento della pagina (proprio quando i contenuti non sono stati ancora alterati). In JS tutte le variabili dichiarate all' esterno di blocchi funzione sono visibili dal codice di tutte le funzioni successivamente implementate.

Si tratta dunque di variabili globali.

```
var arr_chronometer={
                  tenths:0,
                  seconds:0,
                  minutes:0,
                  hours:0
                };
```

Ho dichiarato un array associativo. Si tratta di un array in cui gli indici non sono valori numerici ma delle stringhe. Il vantaggio è quello di generare minor confusione in chi scrive il codice o lo sta leggendo. In pratica andrò a creare un contatore incrementando periodicamente questi valori. Perchè ho utilizzato un array e non delle semplici variabili? Non c' è nessun motivo dietro questa scelta.

Ecco la prima funzione:

```
function reset_arr_chronometer(){
        for(index in arr_chronometer) {
            arr_chronometer[index]=0;
        }
    }
```

essa si occupa semplicemente di azzerrare il contenuto di arr_chronometer. La sintassi del ciclo for è particolare proprio perchè si tratta di un array associativo. A cosa serve? Quando viene effettuato un reset abbiamo detto che viene ripristinato l' html iniziale ma occorre anche resettare le variabili di conteggio altrimenti il conteggio non riparte da zero.

Dichiaro altre due variabili globali per le funzioni che seguono:

```
var timer_id=null; var isstarted=false;
```

Adesso si passa alle implementazioni delle 3 funzioni cronometro: start(), stop() e reset()

```
function start(){
        if(!isstarted){
            isstarted=true;
            var start_date = $(".clocktime").html();
            $("#start_time").html("Start ore: "+ start_date);
            timer_id=setInterval(function(){
                increment_tenth();
```

```
            }, 100);
        }
    }//close function start()
```

Ha senso avviare il cronometro, se questo non è già avviato; da quì la necessità di utilizzo della variabile globale isstarted che indica se è già avviato oppure no.

L' applicazione prevede la visualizzazione dell' ora in cui è stato fornito lo start. Tale orario viene semplicemente letto dalla stringa visualizzata dal plugin che si occupa della visualizzazione della data e dell' ora correnti non ancora esaminato. In pratica la stringa contenente la data, il plugin, la inserisce in un div che ha l' attributo class = "clocktime" che inserisce lui stesso nella pagina.Noi andiamo a leggerla usando $(".clocktime").html()Da notare che viene usato il . Per indicare che clocktime è un attributo class.

A questo punto copiamo tale stringa all' interno del blocco con id = "start_time" presente nella nostra pagina html.Da notare che nel blocco inseriamo una stringa che è la concatenazione della stringa:

```
"Start ore: "  e della stringa contenuta nella variabile  start_date.
```

Per concatenare delle stringhe in JS si usa il + .

A questo punto settiamo isstarted e inizializziamo il timer JS

```
timer_id=setInterval(function(){
            increment_tenth();
        },100);
```

La funzione setInterval è una funzione JS che non fa altro che eseguire il codice presente all' internodel blocco

```
function(){
            ...
        }
```

ad intervalli regolari.

Il valore 100 indica che tale intervallo dovrà essere di 100ms, ovvero increment_tenth(); dovrà essere chiamata ogni decimo di secondo.

Questa funzione si occuperà dell' incremento dei decimi di secondo.La funzione setInterval() restituisce un timerID che assegnamo alla variabile globale timer_id e che come vedremo tra poco, servirà successivamente per fermare il conteggio.

```
function stop(){
        if(isstarted){
            isstarted = false;
            clearInterval(timer_id);
        }
    }//close function stop()
```

Questa funzione ferma il conteggio eseguendo un clearInterval sul valore del timerID che era stato precedentemente ottenuto con setInterval() ed assegnato alla variabile globale timer_id. Non ha senso fermare il conteggio se questo è gia fermo, per cui viene prima eseguito un controllo sulla variabile isstarted.

```
function reset(){
        stop();
        $("#chronometer").html(reset_html);
        $("#start_time").html(" ");
        reset_arr_chronometer();
    }
```

La funzione reset() ha I seguenti compiti:

- fermare il conteggio mediante una chiamata alla funzione stop()
- Ripristinare il codice html del cronometro;

```
$("#chronometer").html(reset_html); non fa altro che sostituire il contenuto del div c
l' html salvato all' inizio.
```

- Cancellare il contenuto del div mostrante l' ora in cui è stato dato lo start.

Prima di procedere facciamo una piccola pausa cambiando un po' argomento. Dicevamo che per quanto riguarda la visualizzazione dell' ora attuale è stato utilizzato utilizzato un plugin. Questo va inizializzato nel seguente modo:

```
$("#clock").clock({
        "langSet":"it"
    });
```

"#clock" indica che vogliamo la visualizzazione dell' orologio all' interno del div che ha id = "clock" che come ricordate è quello che mostra la richiesta di attivare Javascript. Il plugin poteva essere integrato anche così:

```
$("#clock").clock();
```

In questo modo però l' orologio è in inglese. Per averlo in italiano è stato necessario settare l' opzione langSet. Chiusa questa breve parentesi ritorniamo al cronometro.

Catturiamo gli eventi click sui tre pulsanti con id rispettivamente Start, Stop e Reset ed eseguiamo per ciascun evento la funzione appropriata.

```
$("#Start").click(function(){
        start();
    });
$("#Stop").click(function(){
        stop();
    });$("#Reset").click(function(){
        reset();
    });
```

Analizzerò adesso le funzioni di incremento dei contatori.La prima è increment_tenth() che viene chiamata direttamente dal timer JS ogni decimo di secondo. Vediamo come funziona:

```
function increment_tenth(){
   if(arr_chronometer['tenths']==9){
            arr_chronometer['tenths']=0;
            increment_second();
        }else
           arr_chronometer['tenths']++;
    view_tenths();
}
```

La visualizzazione dei decimi di secondo va da 0 a 9 quindi se il contatore dei decimi di secondo è giunto a 9, lo riporta a zero e chiama la funzione di incremento dei secondi. In caso contrario incrementa semplicemente il contatore dei decimi. In tutti I casi effettua alla fine l' aggiornamento della visualizzazione su display mediante la chiamata a view_tenths()L' operazione di aggiornamento di visualizzazione su display non fa altro che visualizzare, usando le immagini dei digit a 7 segmenti, il contenuto di arr_chronometer['tenths'].Le funzioni di incremento dei secondi e dei minuti, funzionano in modo analogo a parte il fatto che il conteggio non va da 0 a 9 ma da 0 a 59.Per quanto riguarda l' incremento delle ore il conteggio va ancora da 0 a 9 ma giungendo a 9 viene effettuato semplicemente l' arresto del conteggio.Anche le funzioni:

```
view_tenths();
view_seconds();
view_minutes();
view_hours();
```

sono tra loro simili e si occupano rispettivamente dell' aggiornamento a display dei decimi di secondo, dei secondi, dei minuti e delle ore.Si definiscono altre due variabili globali:

```
var HDigit;
var LDigit;
```

si tratta di variabili che ho utilizzato per memorizzare temporaneamente 2 cifre numeriche che possono rappresentare decimi, secondi, minuti oppure ore.Immaginiamo che ciascuno dei valori da visualizzare sia a due cifre. Quella che rappresenta le unità è Ldigit mentre quella che rappresenta le decine è Hdigit. Per quanto riguarda i decimi di secondo e le ore, poichè sono rappresentati da una sola cifra, verrà utilizzata solo la variabile Ldigit. Hdigit in questo caso sarà sempre 0.

```
function view_tenths(){
    setDigit(arr_chronometer['tenths']);
    $("#D0 img").attr("src", "digits/mini/"+LDigit + ".png");
    }
```

Viene effettuata dapprima una chiamata alla funzione setDigit che riceve in input il valore del conteggio. La funzione si occupa semplicemente di spezzare il valore numerico ricevuto in decine ed unità ed assegnarlo rispettivamente alle variabili globali Hdigit ed Ldigit.Esaminiamo attentamente l' istruzione successiva:

```
$("#D0 img").attr("src", "digits/mini/"+LDigit + ".png");
```

Significa accedi al tag img (da notare che img non ha davanti a se ne . e ne # proprio per indicare che ci stiamo riferendo al nome di un tag) contenuto all' interno di un blocco con id = *D0* e assegna all' attributo src il valore: "digits/mini/"+LDigit + ".png".

Per capire bene occorre aver davanti il codice html su cui agirà la funzione JS:

```
<div id="chronometer">
<ul>
  <li id="D5"><img class="digit" src ="digits/0.png" width="134" height="206" alt="" />
  <li><img class="dp2" src ="digits/2p.png" width="14" height="70" alt="" /></li>
  <li id="D4"><img class="digit" src ="digits/0.png" width="134" height="206" alt="" />
  <li id="D3"><img class="digit" src ="digits/0.png" width="134" height="206" alt="" />
  <li><img class="dp2" src ="digits/2p.png" width="14" height="70" alt="" /></li>
  <li id="D2"><img class="digit" src ="digits/0.png" width="134" height="206" alt="" />
  <li id="D1"><img class="digit" src ="digits/0.png" width="134" height="206" alt="" />
  <li><img class="dp1" src ="digits/1p.png" width="7" height="7" alt="" /></li>
  <li id="D0"><img class="digit" src ="digits/mini/0.png" width="68" height="104" alt="
```

```
</ul>
</div>
```

Abbiamo detto che accede ad un blocco con id = "D0". Si tratta di un blocco li

```
<li id="D0"></li>
```

Questo blocco al suo interno ha un blocco img il quale ha un attributo chiamato src che rappresenta il link all' immagine da visualizzare. Noi in pratica andiamo a rimpiazzare dinamicamente tale link. Le immagini dei digit ho scelto di chiamarle in base alla cifra numerica che rappresentano. In pratica avremo 0.png per la cifra 0, 1.png per la cifra 1 e così via fino alla cifra 9.png che rappresenta il digit 9.

Stiamo dicendo rimpiazza il valore assegnato a src con quest' altro:"digits/mini/" + LDigit + ".png" E' una concatenazione di stringhe (le stringhe in JS si concatenano usando il +)

"digits/mini/" rappresenta il path dell' immagine.LDigit è la variabile contenente il valore numerico da visualizzare.LDigit non è racchiuso tra apici proprio per indicare che non dobbiamo concatenare la stringa LDigit ma il valore numero contenuto nella variabile LDigit.JS quando si cerca di concatenare un valore numerico con una stringa, considera quel valore numerico di tipo stringa. Si dice che effettua un cast implicito. Per completare aggiungiamo l' ultimo pezzo: ".png" Se per esempio LDigit contiene il valore numerico 5, ciò che ne verrebbe fuori sarà:

"digits/mini/5.png".

Completo passando direttamente ad esaminare la funzione:

```
function setDigit(val){
        if(val < 10){
            HDigit = 0;
            LDigit = val;
        }else{
            val=val+"";     //convert in string format
            HDigit = val.substring(0,1);
            LDigit = val.substring(1,2);
        }
    }
```

Come dicevo, questa funzione si occupa semplicemente di spezzare il valore val, ricevuto in cifra delle decine e cifra delle unità e assegnare tali valori alle 2 variabili globali HDigit e LDigit. Se il valore contenuto nella variabile val è minore di 10, la cifra delle decina sarà 0 mentre il valore delle unità sarà esattamente quello di val.

In caso contrario si spezza il contenuto di val. Per spiegare come funziona faccio un esempio: supponiamo che val contenga la cifra numerica 15.

Se converto tale cifra in stringa, posso dire che il primo carattere sarà 1 e che il secondo carattere sarà 5Per convertire in stringa, ho semplicemente concatenato a val una stringa vuota.In JS il tipo di una variabile è dato dal valore in esso contenuto. Faccio un esempio:

var a = 15 intero var b = '15' stringa

var c = 6 intero a+b = '1515' stringa

a+c = 21 interoLa somma tra due interi tratta entrambi come interi, mentre la somma di un intero con una stringa tratta entrambi come stringhe e perciò le concatena.A questo punto 15 + *da* '*15*' . A me interessa avere delle stringhe in modo da poter utilizzare la funzione substring di JS

Come potete notare JQUERY e JS sono perfettamente compatibili tra loro.Sto tranquillamente mischiando codice JQUERY e codice JS senza nessun problema.In substr il primo parametro rappresenta la posizione di inizio della sottostringa da prelevare. Si inizia a contare da 0 ovvero il primo carattere di una stringa è quello in posizione 0, il secondo quello in posizione 1 e così via. Il secondo parametro di substr indica il numero di caratteri da prelevare. In pratica con

```
val.substring(0,1);
```

otteniamo il primo carattere della stringa val che rappresenta ledecine mentre con

```
val.substring(1,1);
```

otteniamo il secondo carattere della stringa val che rappresenta le unità.

A questo punto ho terminato.

Se ci sono delle domande o anche suggerimenti di come migliorare l' applicazione scrivetemi nei commenti.

Grazie!

Estratto da "http://www.electroyou.it/mediawiki/index.php?title=UsersPages:Daniele78:orologio-cronometro-digitale-su-pc-2a-parte"