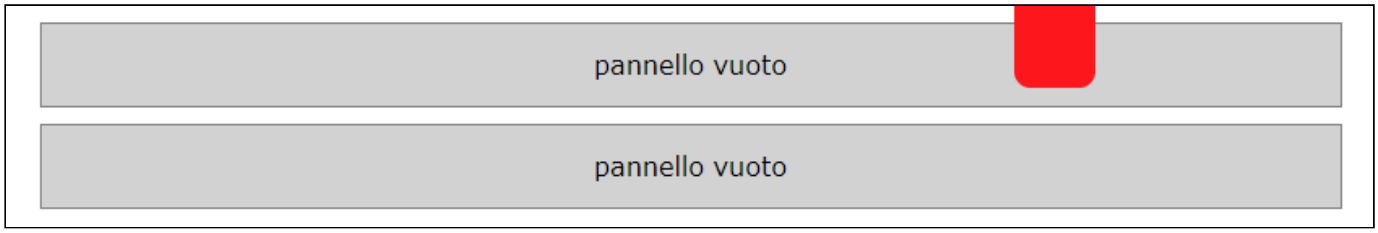
Stop and Go /1

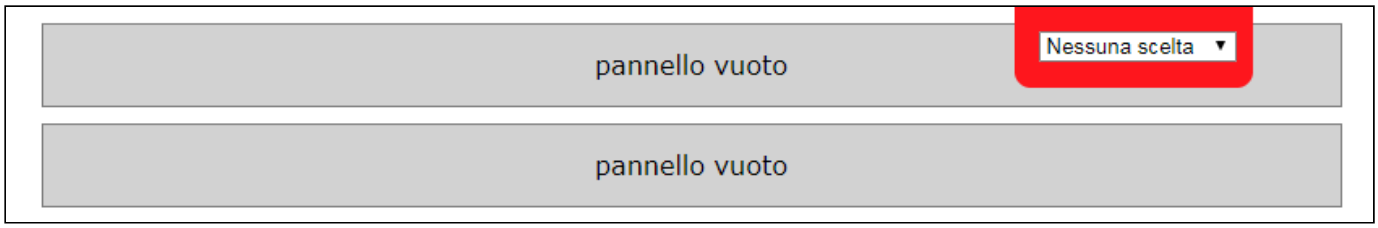
10

1) La mia città

Vogliamo realizzare un progetto che quando parte si presenta così: due pannelli (dichiaratamente) vuoti e una linguetta rossa.



Quando il mouse arriva su questa linguetta, essa si estende (piano piano) fino a diventare un piccolo form con un menu a tendina.



Quando l'utente seleziona una delle tre voci significative si passa alla seconda fase del progetto.

In questa fase il primo pannello si riempie con una immagine e il secondo con dei testi.
Inoltre il menu a tendina perde la sua prima voce ("nessuna scelta")

Stop and Go /2 10

Da qui in avanti ogni scelta operata sul menu a tendina modificherà il contenuto dei due pannelli in accordo con la scelta operata dall'utente.

Cominciamo dunque!
Creaiamo la cartella **36-caserta** con al suo interno le cartelle **img**, **css** e **script**.
Copieremo poi il solito file **reset.css** in **css** e le immagini del file **www.bbuio.it/varie/36-caserta.zip** nella cartella **img**. Teniamo i testi per dopo.

Passiamo quindi alla realizzazione del file **index.html**: puoi fare da solo la sezione **head** e puoi copiare il codice qui sotto per la sezione **body**.

index.html

```
10.  <body>
11.      <article>
12.          <div class="vuoto">
13.              pannello vuoto
14.          </div>
15.          <div class="vuoto">
16.              pannello vuoto
17.          </div>
18.      </article>
19.  </body>
```

Per far avere ai due **div** l'aspetto mostrato nelle immagini in apertura dovrai impostare due regole css nell'apposito file.

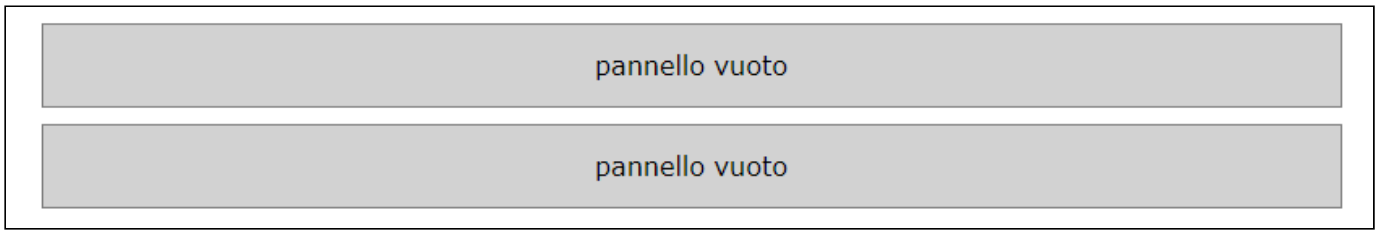
Con la prima indicherai che il tuo body:

- avrà dei testi scritti in **Verdana**
- sarà centrato orizzontalmente
- avrà un'ampiezza di 800 pixel
- avrà un padding verticale di 10 pixel

Con la seconda invece indicherai che tutti gli elementi della classe **"vuoto"**

- avranno un'ampiezza del 100% e un'altezza di 50 pixel
- ospiteranno testi centrati sia verticalmente che orizzontalmente
- saranno di colore grigio chiaro con un bordo grigio (sottile 1 pixel)
- avranno tutti un **margin-bottom** pari a 10 pixel

Stop and Go /3 10



So che ormai sei un programmatore provetto ma devo chiedertelo: hai verificato che il tuo browser presenti esattamente quello che appare in questa immagine?

Adesso occupiamoci di quella linguetta rossa.

index.html

```
19.      <div id="menu">
20.      </div>
```

Aggiungi queste due righe di codice al **body** dopo l'**article** che contiene i due pannelli vuoti e poi aggiungi questa regola al foglio di stile.

stili.css

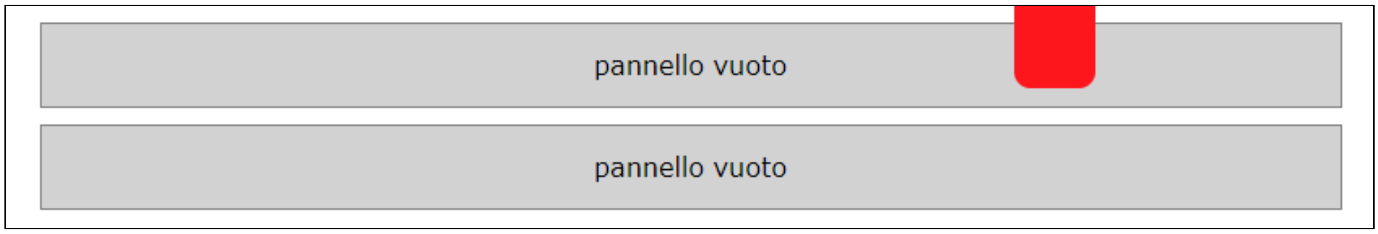
```
18.  div#menu {
19.      position: absolute;
20.      top: 0px;
21.      left: 600px;
22.      width: 50px;
23.      height: 50px;
24.      background: red;
25.      border-radius: 0px 0px 10px 10px;
26.  }
```

Con le prime tre dichiarazioni stiamo indicando al browser che l'elemento indicato deve uscire dal normale flusso di rendering e deve essere posizionato in maniera definitiva attaccato al bordo superiore e a 600 pixel dal bordo sinistro.

Abbiamo visto però che questa dichiarazione da sola non basta: per far si che la nostra linguetta resti ferma ridmensionando la finestra dobbiamo aggiungere la seguente dichiarazione alla regola per il suo contenitore (ovvero per **body**).

stili.css

```
06.  position: relative;
```



Dopo aver verificato che il tuo browser reagisce come ci aspettiamo dedichiamo la nostra attenzione all'animazione della linguetta.

Stop and Go /4 10

Vogliamo che quando il mouse passa sopra la linguetta questa incrementi la sua dimensione.

stili.css

```
29.  div#menu:hover {
30.      width: 150px;
31.  }
```

Per fare in modo che la transizione dal valore inziale attribuito alla proprietà **width** al valore indicato nell'ultima dichiarazione non sia immediata dobbiamo aggiungere un'ultima dichiarazione alla regola per il **div** con identificativo **menu**

stili.css

```
19.  div#menu {
20.      position: absolute;
21.      top: 0px;
22.      left: 600px;
23.      width: 50px;
24.      height: 50px;
25.      background: red;
26.      border-radius: 0px 0px 10px 10px;
27.      transition: width 2s;
28.  }
```

Andiamo quindi a riempire questa linguetta.

index.html

```
19.      <div id="menu">
20.          <form id="modulo">
21.              <select>
22.                  <option>Nessuna scelta</option>
23.                  <option>Atteone e Diana</option>
24.                  <option>Pompei</option>
25.                  <option>La sedia volante</option>
26.              </select>
27.          </form>
28.      </div>
```

Lancia adesso il tuo lavoro: se non interveniamo subito il nostro menu a tendina resterà lì per sempre.

stili.css

```
34.  div#menu form#modulo {
35.      position: relative;
36.      top: 15px;
37.      left: 15px;
38.  }
```

Che ne dici? Ti piace così?

No, sul serio: ma l'hai visto?

Ah, ok! Bene, per fare in modo che il menu a tendina sia visibile solo per la parte della linguetta e non vada al di fuori dei suoi confini, dobbiamo aggiungere la seguente dichiarazione al **div** della linguetta.

stili.css

```
28.      overflow: hidden;
```

Stop and Go /5 10

A questo punto nascondiamo facciamo in modo che il menu a tendina compaia e scompaia piano piano.
Per ottenere questo effetto usiamo la proprietà **opacity**: ponendola a zero abbiamo che l'elemento è invisibile, ponendola a uno diventa visibile e con la proprietà **transition** indichiamo al browser che il passaggio tra questi due valori deve avvenire piano piano.
Cominciamo intervenendo nel foglio di stile.

stili.css

```
34.  div#menu form#modulo {
36.      position: relative;
37.      top: 15px;
38.      left: 15px;
39.      opacity: 0;
40.      transition: opacity 2s;
41.  }
```

E adesso andiamo a fare in modo che quando l'utente arriva con il mouse sul pannellino rosso la proprietà **opacity** diventi 1.

index.html

```
19.  <div id="menu" onmouseover="azionaMenu()" >
```

script.js

```
01.  function azionaMenu() {
02.      document.getElementById("modulo").style.opacity = "1";
03.  }
```

Ed ora chiudiamo il cerchio con la gestione della riscomparsa del nostro menu a tendina.

index.html

```
19.  <div id="menu" onmouseover="azionaMenu()" onmouseout="chiudiMenu()" >
```

script.js

```
05.  function chiudiMenu() {
06.      document.getElementById("modulo").style.opacity = "0";
07.  }
```

E' tutto decisamente molto bello, ma si può migliorare ancora un pochino: possiamo rendere il menu a tendina visibile solo all'interno del suo contenitore, e non anche fuori, come succede adesso.

stili.css

```
19.  div#menu {
20.      position: absolute;
21.      top: 0px;
22.      left: 600px;
23.      width: 50px;
24.      height: 50px;
25.      background: red;
26.      border-radius: 0px 0px 10px 10px;
27.      transition: width 2s;
28.      overflow: hidden;
29.  }
```

E con questo abbiamo finito la prima fase.

Stop and Go /6 10

La seconda fase comincia con la scelta da parte dell'utente di una voce di menu diversa da quella iniziale.

Per gestire questo evento usiamo la **onchange** sull'elemento **select** al quale diamo anche un identificativo univoco in modo tale da potercisi riferire nella funzione javascript.

index.html

```
21.  <select id="tendina" onchange="nuovaScelta()">
```

Adesso andiamo a creare la funzione javascript **nuovaScelta()**.
Innanzitutto mediante l'attributo **selectedIndex** di **select** andiamo ad ottenere l'indice dell'elemento selezionato e successivamente useremo questo valore come indice sul vettore **options** dell'elemento **select**.
Infine metteremo a video il valore dell'attributo **text** dell'elemento **option** selezionato.

script.js

```
09.  function nuovaScelta() {
10.      var elemento = document.getElementById("tendina");
11.      var indice = elemento.selectedIndex;
12.      var voceSelezionata = elemento.options[indice].text;
13.
14.      alert(voceSelezionata);
15.
16.  }
```

L'hai provato? Funziona? Bene! Adesso puoi togliere quell'**alert** inutile.

Adesso dobbiamo gestire il passaggio dalla prima alla seconda fase.
Per far questo creiamo una var globale **primaFase** inizializzata a **true** per poi metterla a **false** la prima volta viene invocata la funzione **nuovaScelta()**.

Successivamente a ciò dovremo fare tutta un'altra serie di attività.

script.js

```
09.  // var globale per la gestione della prima fase
10.  var primaFase = true;
11.
12.  function nuovaScelta() {
13.      var elemento = document.getElementById("tendina");
14.      var indice = elemento.selectedIndex;
15.      var voceSelezionata = elemento.options[indice].text;
16.
17.      if (primaFase) {
18.          // chiudo definitivamente la prima fase
19.          primaFase = false;
20.
21.          // tolgo la voce "Nessuna scelta" dal menu
22.
23.          // elimino dai due div il testo
24.
25.          // annullo per i due div tutte le impostazioni
26.
27.          // inserisco nel div per l'immagine un elemento img
28.
29.          // inserisco nel div per il testo un elemento h1 e un elemento paragrafo
30.      }
31.  }
```

Stop and Go /7 10

Andiamo adesso a riempire le varie sezioni del nostro script.

Prima di tutto togliamo la voce "Nessuna scelta" dal menu a tendina. Per far ciò useremo la funzione **remove** dei vettori sulla cella zero del vettore **options** dell'elemento **menu a tendina** .

script.js

```
21.         // tolgo la voce "Nessuna scelta" dal menu
22.         elemento.options.remove(0)
```

Adesso dobbiamo togliere il testo dai due cosiddetti "pannelli vuoti".
Per far ciò dobbiamo dare un nome a questi **div**.

index.html

```
11.         <article>
12.             <div class="vuoto" id="divImg">
13.                 pannello vuoto
14.             </div>
15.             <div class="vuoto" id="divTxt">
16.                 pannello vuoto
17.             </div>
18.         </article>
```

Ecco, adesso ci basta cancellare il valore di **innerHTML** per entrambi.

script.js

```
24.         // elimino dai due div il testo
25.         document.getElementById("divImg").innerHTML = "";
26.         document.getElementById("divTxt").innerHTML = "";
```

Adesso, per annullare tutte le impostazioni css per questi due **div**, dobbiamo eliminare l'attributo **class**.

script.js

```
28.         // annullo per i due div tutte le impostazioni
29.         document.getElementById("divImg").removeAttribute("class");
30.         document.getElementById("divTxt").removeAttribute("class");
```

Ed eccoci alla parte più interessante.

Per inserire nel **div** per l'immagine un elemento **img** e per inserire nell'altro **div** un elemento **h1** ed un elemento **p** dobbiamo fare delle modifiche al **DOM**.

Nello specifico dobbiamo aggiungere elementi.

Dovremo pertanto:

- creare un elemento
- creare tutti gli attributi che ci serviranno
- riempire ciascun attributo
- aggiungere l'elemento appena creato come figlio di un elemento preesistente

Stop and Go /8 10

script.js

```
32.         // inserisco nel div per l'immagine un elemento img
33.         var divImg = document.getElementById("divImg");
34.         var img = document.createElement("img");
35.         var attributo = document.createAttribute("src");
36.         img.setAttributeNode(attributo);
37.         attributo = document.createAttribute("alt");
38.         img.setAttributeNode(attributo);
39.         attributo = document.createAttribute("id");
40.         attributo.value = "immagine";
41.         img.setAttributeNode(attributo);
42.         divImg.appendChild(img);
43.
44.         // inserisco nel div per il testo un elemento h1 e un elemento paragrafo
45.         var divTxt = document.getElementById("divTxt");
46.         var h1 = document.createElement("h1");
47.         attributo = document.createAttribute("id");
48.         attributo.value = "titolo";
49.         h1.setAttributeNode(attributo);
50.         divTxt.appendChild(h1);
51.         var p = document.createElement("p");
52.         attributo = document.createAttribute("id");
53.         attributo.value = "testo";
54.         p.setAttributeNode(attributo);
55.         divTxt.appendChild(p);
```

Adesso abbiamo nel nostro documento altri tre elementi: un **img** con identificativo **immagine**, un **h1** con identificativo **titolo** e un elemento **p** con identificativo **testo**.

La funzione, lo script ed il progetto in generale si completano andando a riempire questi tre elementi in accordo con la scelta fatta dall'utente.

script.js

```
57.     if (voceSelezionata=="Atteone e Diana") {
58.         // imposto l'immagine, il testo del titolo ed il testo del paragrafo
59.     }
60.
61.     if (voceSelezionata=="Pompei") {
62.         // imposto l'immagine, il testo del titolo ed il testo del paragrafo
63.     }
64.
65.     if (voceSelezionata=="La sedia volante") {
66.         // imposto l'immagine, il testo del titolo ed il testo del paragrafo
67.     }
```

Sono sicuro che puoi andare avanti in autonomia: ricorda che

- per l'elemento **img** dovrai andare a impostare il valore di **src**
- per gli elementi **h1** e **p** dovrai andare a impostare il valore di **innerHTML**

Ricordati inoltre che

- puoi recuperare i testi dallo zip che hai scaricato all'inizio
- puoi recuperare i titoli dalle immagini proposte in apertura

- puoi individuare quando andare a capo nei testi o guardando le immagini o analizzando i testi forniti.

Stop and Go /9

10

Chiudiamo l'esercizio andando a completare il foglio di stile.

stili.css

```
42.  h1#titolo {
43.      width: 100%;
44.      font-size: 35px;
45.      text-align: center;
46.  }
47.
48.  div#divImg, div#divTxt, {
49.      margin: 0px 0px 10px 0px;
50.  }
```

Stop and Go /10 10

2) Un po' di colore

Vogliamo realizzare adesso una piccola webapp che colori un testo lettera per lettera secondo le indicazioni dell'utente.

Cominciamo con il file **index.html** che definisce la struttura del pannello principale.

Immaginando che tu ormai sia un asso nel preparare la sezione **head**, spero non ti offenda se ti propongo qui la sezione **body**

index.html

```
10.  <body>
11.      <section>
12.          <h1>COLORAZIONE</h1>
13.          <form>
14.              <div id="bloccoDiSinistra">
15.              </div>
16.              <div id="bloccoDiDestra">
17.              </div>
18.          </form>
19.      </section>
20.  </body>
```

Stop and Go /11 10

Proseguiamo riempiendo le due colonne.

Nel blocco di sinistra il primo elemento che compare è quello che memorizza il testo da colorare.

index.html

```
15.                 <label for="testo">Testo</label>
16.                 <input id="testo" type="text" placeholder="Inserisci qui un testo"/>
```

L'elemento **label** con il suo attributo **for** posto al valore dellidentificativo dell'elemento **input** successivo lega i due elementi anche logicamente: cliccando sulla label si attiva il controllo per l'immissione dell'input.

Con l'attributo **placeholder** faccio comparire un'indicazione in grigetto nel controllo: testo che scomparirà non appena l'utente comincia a scirvere.

Le prossime due righe di codice invece presentano un controllo di tipo **number**: l'input sarà quindi necessariamente un valore numerico che parte da **value** e con le due freccettine si sposta fino a un minimo di **min** e fino a un massimo di **max**. Ad ogni click il valore viene decrementato o incrementato di **step** unità.

index.html

```
17.                 <label for="dimensione">Dimensione del carattere (in px)</label>
18.                 <input id="dimensione" type="number" value="25" min="15" max="50" step="2"/>
```

Adesso tocca al menu a tendina.

Abbiamo già ampiamente giocato con questo controllo nell'esercizio preceente.

index.html

```
19.                 <label for="fontface">Tipo di carattere</label>
20.                 <select id="fontface">
21.                     <option>Verdana</option>
22.                     <option>Arial</option>
23.                     <option>Times New Roman</option>
24.                     <option>Courier New</option>
25.                 </select>
```

Guardando l'immagine alla pagina precedente ci accorgiamo che è il turno di un altro **input** di tipo **number**

index.html

```
26.                 <label for="spaziatura">Spazio tra le lettere (in px)</label>
27.                 <input id="spaziatura" type="number" value="0" min="0" max="10"/>
```

L'ultimo controllo di questa prima colonna è il **radio buttun** nel **fieldset**.

Stop and Go /12 10

Per inserire un'intestazione al **fieldset** useremo un elemento **legend**.

Per fare in modo che l'utente possa cliccare tanto sul pallino (relativo al radio buttun) quanto sul testo inseriamo entrambi in un elemento **label**

index.html

```
28.  <fieldset>
29.      <legend> Grassetto </legend>
30.      <label>
31.          <input type="radio" name="setGrassetto" id="grassettoSi" checked="checked"/>Si
32.      </label>
33.      <label>
34.          <input type="radio" name="setGrassetto" id="grassettoNo" />No
35.      </label>
36.  </fieldset>
```

Il carattere speciale che abbiamo messo davanti e dietro il testo che deve apparire come intestazione del **fieldset** ci serve per fare in modo che venga lasciato uno spazio e che il testo non sia addossato alle linee che delimitano il **fieldset**.

Passiamo alla colonna di destra.

Il codice da inserire nel blocco di destra sarà quello necessario a far comparire i controlli per inserire il tasto che sceglie un colore.

La prima coppia di istruzioni sarà la seguente.

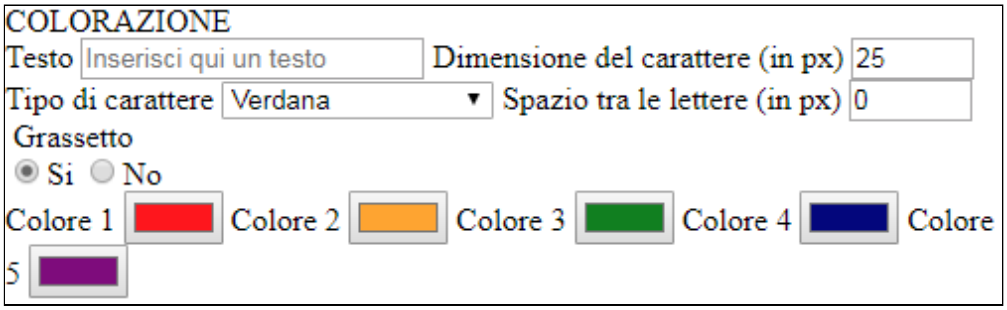
index.html

```
39.                 <label for="colore1">Colore 1</label>
40.                 <input id="colore1" type="color" value="#FF0000" />
```

Queste due righe di codice mettono a video un controllo per scegliere un colore: nel nostro esempio il colore di partenza è il rosso (#FF0000).

Scrivi da solo/a le codice per gli altri 4 colori sapendo che partono da: FFA500, 008000, 000080, 800080.

Ecco abbiamo finito di sistemare gli elementi html, prova a vedere cosa dice il tuo browser.



Uhm.. forse è meglio occuparsi anche del css.

Stop and Go /13 10

Cominciamo con lo sfondo. E' evidentemente uno sfondo con gradiente. Prova cosa accade con le tre dichiarazioni che seguono in modo da capire l'utilizzo del **no-repeat** e di **fixed**

stili.css

```
01.  body {
02.      background: radial-gradient(#7070FF, 12FFFF);
03.  }
```

stili.css

```
01.  body {
02.      background: radial-gradient(#7070FF, 12FFFF) no-repeat;
03.  }
```

stili.css

```
01.  body {
02.      background: radial-gradient(#7070FF, 12FFFF) no-repeat fixed;
03.  }
```

Aggiungi adesso alla regola per **body** un paio di dichiarazioni semplici semplici.

- Imposta la dimensione del testo a 14 pixel.
- Imposta il tipo di carattere a **Verdana**

Nell'**html** abbiamo posto un elemento **section** dentro il **body**: vogliamo questo elemento centrato sia orizzontalmente che verticalmente.

In realtà - per dirla tutta - verticalmente vogliamo porre il pannello un po' più in alto in modo da lasciare un po' di posto anche per la barra con l'output.

stili.css

```
07.  section {
08.      background: white;
09.      width: 500px;
10.      height: 470px;
11.
12.      position: absolute;
13.      top: 50%;
14.      left: 50%;
15.      margin-left: -250px;
16.      margin-top: -300px;
17.  }
```

Le prime tre sono dichiarazioni banali.

Le successive invece servono per tenere questo elemento sempre centrato alla finestra.
Il trucchetto sta nell'assegnare il valore **50%** sia a **top** che a **left** e ad assegnare la metà - al negativo - dei valori **width** ed **height** a **margin-left** e **margin-top**.

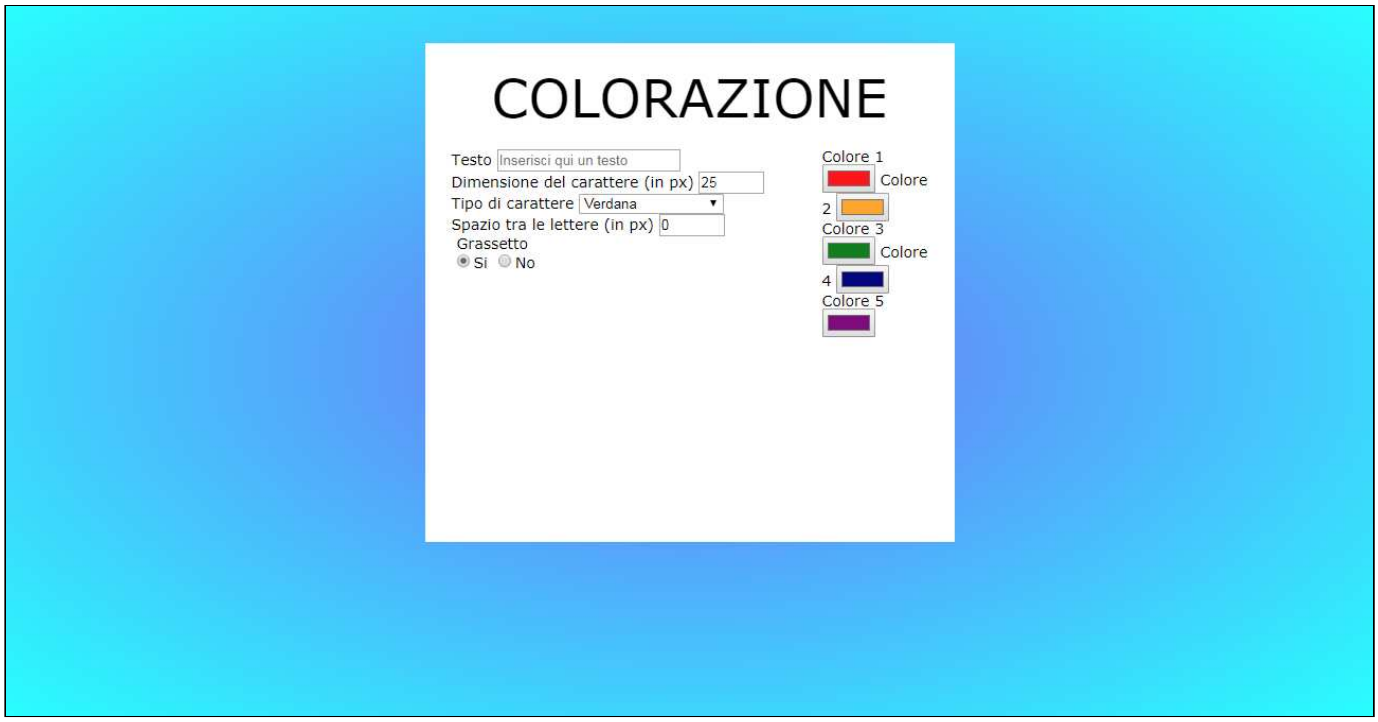
Adesso andiamo ad occuparci del titolo.

Crea una regola per **h1** con la quale imposti il testo centrato e dimensionato a 50 pixel oltre che un **padding-top** pari a 25 pixel.

Adesso crea due regole per i due div **bloccoDiSinistra** e **bloccoDiDestra**. IL primo sarà flottante a sinistra ed il secondo flottante a destra. Entrambi avranno un padding del 5%. Infine **bloccoDiSinistra** sarà largo il 60% dello spazio disponibile e **bloccoDiDestra** il 20%.

Stop and Go /14 10

Dovresti avere ottenuto una cosa del genere.



La prima cosa che notiamo è che poichè sia gli elementi di tipo **input** che gli elementi di tipo **label** sono di tipo inline, si dispongono tutti uno di sequito all'altro.

Provvediamo a rendere di tipo blocco le **label**.

stili.css

```
38.  label {
39.      display:block;
40.  }
```

Dai un'occhiata al browser. C'è ancora un po' di lavoro da fare su questi controlli.

Dimensioniamo tutti alla stessa ampiezza e distanziamo gli uni gli altri con un **margin-bottom**.

stili.css

```
42.  input, select, fieldset {
43.      width: 250px;
44.      margin-bottom: 15px;
45.  }
```

Poi dimensioniamo a 50 pixel i controlli dei colori e i **radio button**

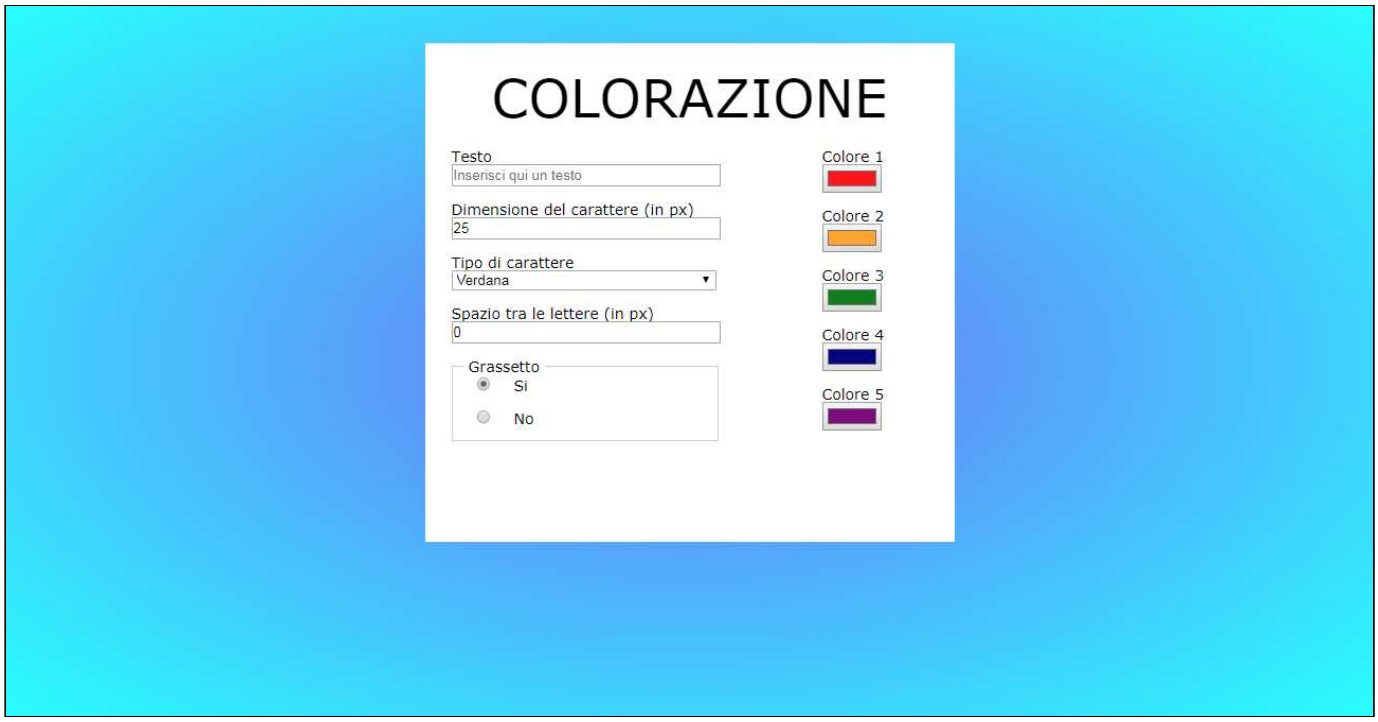
stili.css

```
47.  input[type='color'], input[type='radio'] {
48.      width: 50px;
49.  }
```

Ed infine occupiamoci del **fieldset**. Crea tu stesso una regola per attribuirgli un **border** grigio chiaro di un pixel e poi una regola per spostare il testo del suo elemento **legend** di 10 pixel in avanti.

Stop and Go /15 10

Non che non mi fidi, ma per favore controlla che tu sia esattemente a questo punto qui:



Adesso vogliamo aggiungere il pulsante di conferma.
Poichè le ultime cose aggiunte prima di questo pulsante sono dei blocchi flottanti dobbiamo avvisare il broswer che abbiamo finito di adoperare questa metodologia di presentazione degli elementi.

index.html

```
46.             <div id="resetFloating"></div>
47.             <input type="submit" value="COLORA" />
```

stili.css

```
60. div#resetFloating {
61.     clear: both;
62. }
```

Guarda un po' se il pulsante è centrato, no, eh?

stili.css

```
64. input[type='submit'] {
65.     display: block;
66.     margin: auto;
67. }
```

Stop and Go /16 10

Aggiungiamo adesso il **div** (nascosto) per il risultato.

index.html

```
50.      <div id="risultato">
51.      </div>
```

stili.css

```
69.  div#risultato {
70.      background: white;
71.      width: 500px;
72.      height: 50px;
73.      line-height: 50px;
74.      text-align: center;
75.
76.      position: absolute;
77.      top: 50%;
78.      left: 50%;
79.      margin-left: -250px;
80.      margin-top: 190px;
81.
82.      visibility: hidden;
83.  }
```

Ed è finalmente arrivato il momento di aggiungere un po' di funzionalità al nostro progetto.

Stabiliamo che al click sul pulsante **submit** venga invocata una certa funzione **Colora()**

index.html

```
13.          <form onsubmit="colora()">
```

script.js

```
01.  function colora() {
02.      // imposto la configurazione indicata dall'utente al div
03.
04.      // creo un vettore dei colori
05.
06.      // creo il contenuto del tag
07.  }
```

Per prima cosa rendiamo visibile il pannello **risultato** e lo configuriamo con tutte le impostazioni indicate dall'utente: dimensione del font, spaziatura tra le lettere, tipo di font, grassetto.

script.js

```
02.      // imposto la configurazione indicata dall'utente al div
03.      var elemento = document.getElementById('risultato');
04.      elemento.style.visibility = "visible";
05.      elemento.style.fontSize = document.getElementById('dimensione').value + "px";
06.      elemento.style.letterSpacing = document.getElementById('spaziatura').value + "px";
07.      elemento.style.fontFamily = document.getElementById('fontface').value;
08.      if (document.getElementById('grassettoSi').checked) {
09.          elemento.style.fontWeight = "900";
10.      } else {
11.          elemento.style.fontWeight = "100";
12.      }
```

Stop and Go /17 10

Successivamente creiamo un vettore con i colori (anch'essi scelti dall'utente) sui quali successivamente ciclare per colorare le lettere.

script.js

```
14.     // creo un vettore dei colori
15.     var colori = [
16.         document.getElementById('colore1').value,
17.         document.getElementById('colore2').value,
18.         document.getElementById('colore3').value,
19.         document.getElementById('colore4').value,
20.         document.getElementById('colore5').value
21.     ]
```

Infine andiamo a creare il contenuto del **div risultato**.

script.js

```
23.     // creo il contenuto del tag
24.     var testo = document.getElementById('testo').value;
25.     elemento.innerHTML = dammiTestoColorato(testo, colori)
```

script.js

```
28. function dammiTestoColorato(testo, colori) {
29.     var stringa = '';
30.     for (var i=0; i<testo.length; i++) {
31.         stringa += dammiLetteraInSpan(testo.charAt(i), colori[i%5]);
32.     }
33.     return stringa;
34. }
```

script.js

```
28. function dammiLetteraInSpan(lettera, colore) {
29.     return "<span style='color:"+colore+"'>"+lettera+"</span>";
30. }
```

Ecco, dunque, il momento fatidico: prova a fare un test!

Funziona?

Come "no" ???

Bè, la verità è che funziona ma che poi dopo viene ricaricata la pagina inziale. Questo avviene perchè abbiamo utilizzato un pulsante **submit** che è tipico della gestione del **form**, ma di un **form** che però viene gestito lato server.

Stop and Go /18 10

In questo caso dovremo utilizzare un pulsante normale.

index.html

```
47.             <input type="button" value="COLORA"  onclick="colora()"/>
```

Evidentemente possiamo sganciare la funzione dall'evento **onsubmit**.

index.html

```
13.         <form>
```

E chiaramente dobbiamo modificare quella che era l'impostazione per l'**input** di tipo **submit**.

stili.css

```
64.  input[type='submit'] {
65.      display: block;
66.      margin: auto;
67.  }
```

E per concludere occupiamoci del titolone: giacchè abbiamo una funzione **dammiTestoColorato()** possiamo usare questa funzione per passargli il testo che abbiamo usato come intestazione e scrivere direttamente sul documento il codice html che colora questo testo.

Per far ciò ci basterà sostituire la riga di codice numero 12 con le seguenti:

index.html

```
12.         <h1>
13.             <script>
14.                         var colori = ['#FF0000','#FFA500','#080000','#000080','#800080'];
15.                 document.write(dammiTestoColorato("COLORAZIONE", colori));
16.             </script>
17.         </h1>
```