REALIZZARE UN SITO WEB

II° PARTE

Introduzione.

Con l'arrivo massiccio dei dispositivi mobile i siti web devono essere responsitive, ovvero ad essere in grado di adattarsi a diversi schermi.
In realtà ci sono due diversi tipi di pensiero:

- quelli che creano una serie di pagine ottimizzate SOLO per dispositivi mobili
- quelli che creano le pagine che si "adattano" in base alla risoluzione

NOTA IMPORTANTE: Questo tutorial rappresenta la continuazione e l'approfondimento di quello precedente https://www.filoweb.it/appunti/lhtml.pdf .

Si richiede una conoscenza di base di HTML e delle strutture gerarchiche di un sito web.

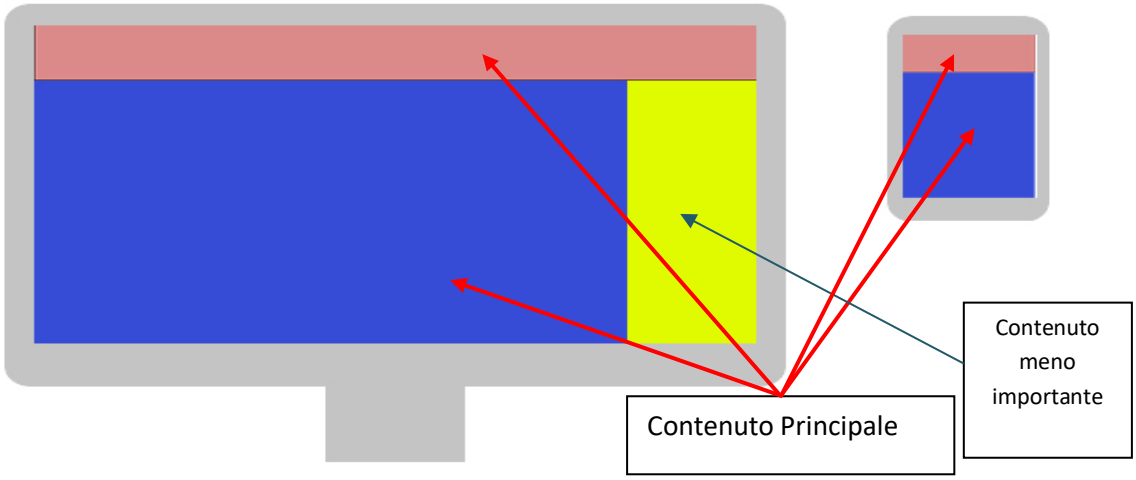
Il primo metodo crea maggior lavoro nella fase di sviluppo ma minore nella fase di progettazione (ogni pagina deve essere creata 2 volte, indipendentemente che sia dinamico o meno il sito) e, visto che si usano script per determinare dove reindirizzare il sito non sempre danno i risultati desiderati. Un esempio di questo è avvenuto con l'introduzione sul mercato del Ipad mini, dove molti script non erano in grado di riconoscerlo.
Se invece si creano siti che si adattano (tramite l'uso di css e script) il lavoro risulta minore nella fase di creazione ma richiederà una maggior attenzione nella fase di progettazione (in questo caso parliamo di *siti reattivi*).
Dato che abbiamo sempre detto che ogni sito web richiede "attenzione" soprattutto nella fase preliminare per questa seconda parte del tutorial useremo il metodo responsitive che si adatta alle diverse risoluzioni.

Alcune chiarificazioni prima di iniziare.

Quando si parla di dispositivi mobili per la visualizzazione di un sito web il 90% delle volte ci riferiamo ad uno smartphone (più che ad un tablet) quindi si tende a vedere lo schermo in orizzontale anziché in verticale. In un monitor, invece, la visualizzazione è orizzontale (con un rapporto che va dai 4:3 ai 16:9).
Quindi come primo punto il nostro sito deve essere visibile sia in orizzontale che in verticale senza perdere l'usabilità, quindi la prima cosa da **evitare** è **lo scrolling orizzontale**!

Immaginando la mia pagina devo iniziare vedendola con gli occhi della mente (e disegnandola sul foglio) partendo dalla versione più piccola ovvero quella che compare nei dispositivi mobili per poi "crescere" creando tutte le altre visualizzazioni; devo quindi avere chiare le idee di:

- ✓ quelli che sono i contenuti principali (che dovranno essere subito visibili),
- ✓ quelli meno importanti che avranno una visibilità secondaria ed eventualmente non mostrarli.



Il peso della pagina è un'altra delle caratteristiche da tener presente: certo video e foto in alta risoluzione sono belle da vedere ma appesantiscono il caricamento, così come un uso eccessivo di plug-in e file esterni. È quindi doveroso prestare molta attenzione anche a quello che viene caricato dalla pagina.
Per quanto riguarda i fonts e l'aspetto tipografico la scelta dei font è molto delicata: mentre nei PC la scelta dei font è molto ampia negli smartphone è limitata ed alcuni hanno solamente un font per ogni famiglia generica; in questo caso, tra le molteplici soluzioni, tre si prestano a risolvere il problema:

Per 'contenuti' non si intendono qui semplicemente i testi, le immagini e gli altri elementi multimediali che costituiscono la base informativa del sito. Sono 'contenuti' anche il logo, il menu di navigazione principale, le sezioni secondarie, i moduli per la ricerca sul sito, etc.

1. Caricare i font (magari tramite usare un servizio di font esterni come nel caso di google font);
2. Usare font che sicuramente sono installati su tutti i dispositivi
3. Creare differenti font a seconda delle dimensioni dello schermo (uno specifico per computer ed uno per i dispositivi mobili)

L'unità di misura da adottare dovrebbe essere quella relativa, quindi espressa in em (tranne le tag body dove il font-size sarà al 100%).
Evitare font troppo complessi o di difficile lettura (come ad esempio gli handwrite) e non caricare troppi font in un sito (per evitare di rallentarne il caricamento)
Per finire bisogna porre moltissima attenzione all'usabilità evitando menù e link troppo vicini gli uni agli altri o soluzioni per navigare che confondano gli utenti: tutto deve essere ben visibile e a "portata di dito".
The last but not the least i menu: se ci si accorge che sono troppi, scomodi, non facili da usare spostarli o usare il cossi detto "hamburger menu" (≡).

Ricapitolando prima di iniziare poniamoci queste domande:

- ✓ quali contenuti inseriamo sulla pagina?
- ✓ per i dispositivi mobili inseriamo gli stessi contenuti e le stesse funzionalità che usiamo per il desktop?
- ✓ se non inseriamo gli stessi contenuti, quali manteniamo?
- ✓ in che ordine li collochiamo?
- ✓ Quale elemento va 'sotto' o 'sopra'?
- ✓ sui dispositivi i contenuti 'superflui' li nascondiamo o li compattiamo?

Una volta risposto a queste domande possiamo iniziare a creare la nostra pagina…

I breakpoint

I breakpoint (o punti di rottura) rappresentano le “dimensioni” dove il comportamento della nostra pagina cambia. La definizione di questi punti di cambiamento è uno dei primi punti da considerare quando si vuole creare una pagina in quanto vengono definiti con valori numerici e unità di misura nel contesto delle media queries inserite nei nostri fogli di stili CSS e quindi riportati su tutte le pagine.
Pur non esistendo uno standard per i breakpoint solitamente si tende a considerare quelli per le risoluzioni più comuni che sono:

0 px < larghezza < 320 px	Smartphone in verticale
321 px < Larghezza < 480 px	Smartphone in orizzontale
481 px < Larghezza < 768 px	Tablet in verticale
769 px < Larghezza < 1024 px	Tablet in Orizzontale o monitor 4:3
1025 px < Larghezza < 1200 px	Monitor 16:9

Nel nostro CSS questi diventeranno:

```
/* risoluzione normale */
body {

}

/* primo breakpoint */
@media screen and (min-width: 480px) {
 body {
  background: #FF0;
 }
}

/* secondo breakpoint */
@media screen and (min-width: 768px) {
 body {
  background: #006400;
 }
}

/* Terzo breakpoint */
@media screen and (min-width: 1024px) {
 body {
  background: #0000C1;
 }
}

/* Quarto breakpoint */
@media screen and (min-width: 1200px) {
 body {
  background: #BBB;
 }
}
```

Nulla vieta di creare più o meno breakpoint a seconda dei propri gusti o intenzioni volendo anche specificando per la maggior parte dei dispositivi, online se ne trovano moltissimi già fatti, anche se una soluzione del genere non sempre risulta comoda come si può vedere dall'esempio qua sotto dove le stesse impostazioni diventeranno:

```
/* Smartphones (portrait and landscape) */
@media only screen and (min-device-width : 320px) and (max-device-width : 480px) { body{background: #XXXXXX;}}

/* Smartphones (landscape) */
@media only screen and (min-width : 321px) {body{background: #XXXXXX;}}

/* Smartphones (portrait) */
@media only screen and (max-width : 320px) { body{background: #XXXXXX;}}

/* iPads (portrait and landscape) */
@media only screen and (min-device-width : 768px) and (max-device-width : 1024px) {body{background: #XXXXXX;}}

/* iPads (landscape) */
@media only screen and (min-device-width : 768px) and (max-device-width : 1024px) and (orientation : landscape)
{body{background: #XXXXXX;}}

/* iPads (portrait) */
@media only screen and (min-device-width : 768px) and (max-device-width : 1024px) and (orientation : portrait) {
body{background: #XXXXXX;} }
@media only screen and (min-device-width : 768px) and (max-device-width : 1024px) and (orientation : landscape) and (-webkit-
min-device-pixel-ratio : 2) { body{background: #XXXXXX;} }
@media only screen and (min-device-width : 768px) and (max-device-width : 1024px) and (orientation : portrait) and (-webkit-min-
device-pixel-ratio : 2) { body{background: #XXXXXX;} }

/* Desktops and laptops */
@media only screen and (min-width : 1224px) { body{background: #XXXXXX;}}

/* Large screens */
@media only screen and (min-width : 1824px) { body{background: #XXXXXX;}}

/* iPhone 4 */
@media only screen and (min-device-width : 320px) and (max-device-width : 480px) and (orientation : landscape) and (-webkit-
min-device-pixel-ratio : 2) { body{background: #XXXXXX;}}
@media only screen and (min-device-width : 320px) and (max-device-width : 480px) and (orientation : portrait) and (-webkit-min-
device-pixel-ratio : 2) { body{background: #XXXXXX;} }

/* iPhone 5 */
@media only screen and (min-device-width: 320px) and (max-device-height: 568px) and (orientation : landscape) and (-webkit-
device-pixel-ratio: 2){ body{background: #XXXXXX;} }
@media only screen and (min-device-width: 320px) and (max-device-height: 568px) and (orientation : portrait) and (-webkit-
device-pixel-ratio: 2){ body{background: #XXXXXX;} }

/* iPhone 6 */
@media only screen and (min-device-width: 375px) and (max-device-height: 667px) and (orientation : landscape) and (-webkit-
device-pixel-ratio: 2){ body{background: #XXXXXX;} }
@media only screen and (min-device-width: 375px) and (max-device-height: 667px) and (orientation : portrait) and (-webkit-
device-pixel-ratio: 2){ body{background: #XXXXXX;}}

…. Ecc.
```

In questo caso per ogni breakpoint devo creare le regole css, quindi meglio cercare di rimanere sulle impostazioni generiche e usare un layout più fluido possibile.

Vediamo adesso una lista dei possibili elementi che ci paiono essenziali che andranno ad essere i nostri contenuti principali:

- ✓ Logo e titolo
- ✓ Navigazione principale
- ✓ Sezione contenuti in evidenza
- ✓ Ultimi contenuti inseriti
- ✓ Navigazione secondaria
- ✓ Box di ricerca
- ✓ Social Share
- ✓ Info di contatto e supporto
- ✓ Altri contenuti
- ✓ Contenuti in evidenza
- ✓ Tags

Fatto questo dobbiamo decidere come suddividere i contenuti in base alla loro importanza e quello che vogliamo che compaia sui diversi dispositivi.
Creiamo due macro gruppi: uno per i dispositivi mobili e tablet ed uno per i Computer e decidiamo cosa mettere nei vari gruppi e dove

PC	Smartphone e trablet
Logo e titolo Navigazione principale Sezione contenuti in evidenza Ultimi contenuti inseriti Navigazione secondaria Box di ricerca Social Share Info di contatto e supporto Altri contenuti Contenuti in evidenza Tags	Logo e titolo Navigazione principale Sezione contenuti in evidenza ~~Ultimi contenuti inseriti~~ Navigazione secondaria Box di ricerca Social Share Info di contatto e supporto ~~Altri contenuti~~ ~~Contenuti in evidenza~~ ~~Tags~~

In questo caso il menu secondario, i social e il motore di ricerca sono stati spostati a fondo pagina, raggiungibili tramite il tasto menu nel menu principale.

I CSS o Fogli di Stile

Nella prima parte di questi tutorials (https://filoweb.it/appunti/lhtml.pdf) abbiamo parlato dei CSS e di come si usano. Facciamo un rapido riassunto:
I CSS o fogli di stile sono uno strumento di formattazione delle pagine web che consente di personalizzare l'aspetto della pagina senza agire direttamente sul codice HTML.
Solitamente un linguaggio di markup (come l'HTML) è utilizzato per descrivere le informazioni sul contenuto di un documento, non il suo stile. I CSS, al contrario, definiscono lo stile, non il contenuto.
Anche se l' HTML fornisce qualche sistema per specificare lo stile (ad esempio il tag <b> per indicare il grassetto), l'uso dei CSS evita di "sporcare" il codice di markup, concentrando tutte le informazioni sullo stile in un file separato che verrà caricato nella pagina.

Solitamente si è usi creare un file con estensione CSS che verrà incorporato nella pagina tramite il tag:
```
<link rel="stylesheet" type="text/css" href="fogliodistile.css" />
```

Allo stesso modo si possono importare altri fogli di stile all'interno di un foglio tramite l'uso di @import:
```
@import url("foglio2.css");
```

Un foglio di stile non è altro (come una pagina html) che un file di testo contenente:

- regole
- commenti
- direttive (come ad esempio i breakpoint accennati in precedenza)

Le regole sono forse la parte principale di un foglio di stile che permette di definire gli stili di ogni oggetto presente nella pagina.
La tipica struttura di una regola CSS è composta da due parti:

- Il selettore
- Il blocco delle dichiarazioni (racchiuso tra le parentesi graffe)

Ecco un esempio di una regola CSS per formattare i paragrafi ( )

```
p { color: #FF2F97; }
```

In questa regola abbiamo un selettore P, che serve a definire la parte del documento cui verrà applicata la regola, un blocco di regole che sono racchiude tra le parentesi {}, e le regole racchiuse dentro di esse e separate dal ; .
In un blocco possono essere presenti più regole:

```
p {
        color:# FF2F97;
        font-family:Arial, Helvetica, sans-serif;
        font-weight:700;
  }
```

I commenti sono molto importanti quando si crea una pagina o un foglio di stile in quanto mi permette (soprattutto a distanza di tempo) di sapere a cosa serve quella regola e cosa mi modifica.
Nei fogli di stile i commenti sono racchiusi all'interno dei caratteri: /* e */.

```
/* modifico i paragrafi questo è un commento */
p {
        color:# FF2F97;
        font-family:Arial, Helvetica, sans-serif;
        font-weight:700;
  }
```

Le ridettive **(chiamate anche @-rules)** sono tipi particolari di costrutti che hanno una caratteristica comune: sono tutti introdotti dal simbolo della chiocciola**; rappresentano vie alternative (** più flessibili e potenti**)**, per realizzare cose attuabili in altri modi**, servono ad importare altri fogli di stile, font, ecc., a definire i breakpoint, definire le dimensioni per la stampa e molto altro.**

@-rules	Funzione
@page {size: 210mm 297mm; margin: 30mm;}	Imposta la pagina per la stampa
@import url('https://fonts.googleapis.com/css?family=Roboto');	Importa un font
@import url("stile.css");	Importa un foglio di stile
@media only screen and (max-width: 1001px) { }	Imposta un Breckpoint

I metatag

L'ultima cosa che bisogna sapere sono quei particolari tag HTML attraverso i quali è possibile specificare informazioni: i metatag.
L'elenco completo dei meta-tag è disponibile nella documentazione ufficiale W3C, ma possiamo dire che servono a specificare delle informazioni che saranno visibili al motore di ricerca, al browser, la condivisione sui social network, ecc. e vanno inseriti all'interno del tag <head> </head>
I più importanti sono:

- ✓ **<meta name="viewport" content="width=device-width, initial-scale=1.0">:** Obbligatorio se voglio un sito responsitive;
- ✓ **<meta http-equiv=" ….." >:** Che contiene informazioni utilizzate nella comunicazione tra server e browser, come ad esempio la lingua, il charset (utf-8, iso-8859-1, ecc,)
- ✓ **<meta name="title" content="titolo pagina" />:** contiene il titolo della pagina
- ✓ **<meta name="description" content="…">:** serve per specificare una descrizione della pagina

In ambito SEO sono molto importanti per il posizionamento sui motori di ricerca.

NOTE: In questa sede non andremo ad analizzare i singoli stili da usare nelle regole ma come si usano.
Per chi volesse approfondire tutte le regole CSS consiglio di guardare il link:
https://www.w3schools.com/css/ dove è presente una esaustiva lista delle regole con tanto di esempi.

Iniziamo con un esempio

Per creare un sito web non servono software particolari o costosi, certo aiutano ma è sufficiente un normale editor di testo. Se proprio si vuole un po' di aiuto consiglio di usare notepad++ un editor di testo gratuito.
Per quanto riguarda le regole prima di iniziare rimando alla prima parte del tutorial sempre sul sito: http://filoweb.it/ nella sezione appunti.

Attenzione: non usare editor di testo come word, worpad, openword, ecc., che creano formattazioni ma editor txt normali

Creiamo il nostro foglio CSS

Apriamo il nostro editor preferito e creiamo la prima parte, il foglio css.
Nell'editor definiamo le direttive regole principali definendo il tipo di carattere, le dimensioni dei fonts, i colori, i link scrivendo:

```
@charset "utf-8"; /* definisco il char set */

html{
        font-size: 100%; /* dimensione del font per tutto il documento */
}
body{
        font-size =1em; /* dimensione del font per tutto il body */
        line-height:1.125em; /* dimensione interlinea  */
        font-family:Verdana, Geneva, sans-serif; /* tipo di carattere */
        text-align: center; /* allinemamento */
        color: #444; /*colore del font */
        background:#000; /* Colore sfondo */
        padding: 0; /* Padding del body */
        margin: 0; /* Margine del body */
}
/* definisco i link (tag a href) */
a:link {
        text-decoration: none; /* nessuna decorazion */
        color: #444; /*colore uguale a quello del body */
}
a:visited {
        text-decoration: none;
        color: #444;
}
a:hover {
        text-decoration: none;
        color:# #FF0D0D; /*colore rosso se passo sopra */
}
a:active {
        text-decoration: none;
        color: #444;
}
```

Salviamo con il nome style.css nella directory dove metto il mio progetto in una sottodirectory chiamata CSS (uso una sottodirectory solo per questione di ordine, posso salvarlo nella directory principale)

Analizziamo le parti principali.

Il nostro foglio di stile inizia con la regola che mi imposta il charset. Non è obbligatorio inserirlo nel foglio CSS, posso anche inserirlo nella pagina con il metatag http-equiv="Content-Type" (html4) o meta charset="utf-8" (html5).
La codifica dei caratteri o "charset" è l'associazione fra un insieme di codici numerici e i rispettivi caratteri caratteri di un determinato linguaggio.
UTF-8 o "Unicode Transformation Format, 8 bit" permette di rappresentare pressoché tutti i caratteri necessari al linguaggio umano scritto, comprese le lettere accentate (quindi per noi italiani, francesi, spagnoli, ecc è non consigliabile ma obbligatorio usarlo!)

Note: il charset utf-8 non supporta le lettere maiuscole accentate o altri caratteri particolari per i quali è necessario utilizzare le "entità HTML"

Nelle regole dei fogli di stile è importante ricordare che sono ereditarie, quindi si riportano sui selettori presenti all'interno del selettore padre.
In questo caso impostando il font-size a 100% nel tag HTML (che contiene tutta la pagina) questa dimensione di base si riporta su tutto quello che è compreso in <html> </html>.
Impostare il font-size a 100% nel tag body vuol dire impostarlo nelle dimensioni di default del browser (solitamente 16px)

Vediamo adesso che nel selettore body abbiamo impostato le dimensioni a 1em, ma cosa vuol dire?
All'inizio di questo tutorial abbiamo detto che l' unità di misura da adottare per i fonts dovrebbe essere quella relativa.
L'em è un'unità di misura relativa molto comoda da usare: calcola la dimensione del testo in base ad una misura fissa di base (in questo caso il 100% del html che è il contenitore padre).
1em, quindi, corrisponde alla dimensione di base assegnata al documento o all'elemento, ovvero le dimensioni di default del browser.
Quindi se consideriamo 1em paria 16PX (testo) avremo:
1em = 16px.
Da qui **1px = 1 ÷ 16 = 0.0625em**.
Quindi moltiplichiamo per N per ottenere gli em necessari alle dimensioni del nostro documento:
dimensioni = **0.0625em** × **N**
Vediamo allora che sempre nel body l'interlinea è impostata a 1.125em (*usare il . e non la virgola per il separatore decimale*) ovvero 18.016px.

Vediamo adesso le opzioni per il selettore a (quello dei link link
Essendo ereditarie le caratteristiche non dobbiamo preoccuparci delle dimensioni o del tipo di fonts utilizzati.
Invece diverso il comportamento del colore e della sottolineatura: se non lo specifichiamo prenderà quello predefinito del browser. Ecco quindi che decidiamo di impostare un comportamento diverso a seconda dell'azione che compie l'utente sul link.

a:link sono i link ancora da visitare (da cliccare)
a:visited sono invece i links visitati (già cliccati)
a:hover si riferisce all'evento del mouse nel momento preciso in cui viene posizionato sopra il link.
a:active è invece riguarda un link quando è attivato (cliccato)

NOTA: :link, :visited, :hover e :active sono pseudo-elementi e come tali vanno sempre collegati ad altri selettori (selettore:pseudo-elemento {dichiarazioni;}) es. ***p:first-letter {color: red; font-weight: bold;}*** imposta la prima lettera del paragrafo in rosso e grassetto.

Non confondere i *pseudo selettori* con i *pseudo elementi*!

Il resto del codice risulta ben commentato e non necessita spiegazioni, chi volesse maggiori informazioni riguardo le regole css può leggere: (https://www.w3schools.com/css/default.asp).

Creiamo adesso la nostra pagina web.

Sempre con il nostro editor creiamo adesso un nuovo file nel quale scriviamo:

```
<html>
    <head>
        <title>La mia pagina</title>

        <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
        <meta name="viewport" content="width=device-width, initial-scale=1.0" />
        <meta name="title" content="La mia Pagina" />
        <meta name="description" content="La mia prima pagina web" />

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

    </head>

    <body>
        <p>Ciao questa è la mia prima pagina</p>
        <p><a href="https://www.google.it">questo è un link a google</a></p>
    </body>

</html>
```

Vediamo subito che nella sezione head sono presenti i metatag che definiscono il charset (non necessario se impostato anche nel CSS), il viewport per i dispositivi mobili, il title e la description.

Segue il riferimento al foglio di stile che mi definisce la pagina.
Salviamo con il nome pagina.html e apriamola nel nostro browser.
Ecco il risultato, quando passo sopra la scritta questo è un link quest'ultima diventerà rossa.

Proviamo a cambiare nel foglio CSS nel selettore body la regola background con il valore #FFF; (il colore di sfondo) e vediamo che diventerà bianco lo sfondo.

```
body{
  ..
  background:#FFF; /* Colore sfondo */
..
}
```

Iniziamo a dare una forma alla nostra pagina creando una testata.
Nel nostro foglio di stile aggiungiamo:

```
#testata {
        width:100%;
        margin-bottom:2px;
        border-bottom:#000 3px solid;
        background-color:#E0E0E0;
}
```

Mentre nella nostra pagina, sotto l'apertura del tag body inseriamo:

```
<div id="testata">
        Qua la testata
</div>
```

HTML	CSS

<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
 	
<title>Documento senza titolo</title>
<link href="css/style.css" rel="stylesheet" type="text/css">
</head>
 	
<body>
 	
Qua la testata
 	
 Ciao questa è la mia prima pagina
 questo è un link a google
</body>
</html>
	@charset "utf-8"; /* definisco il char set */
	
	html{
	font-size: 100%;
	}
	
	body{
	font-size =1em; /* dimensione del font per tutto il body */
	line-height:1.125em; /* dimensione interlinea */
	font-family:Verdana, Geneva, sans-serif; /* tipo di carattere */
	text-align: center; /* allinemamento */
	color: #444; /*colore del font */
	background:#FFF; /* Colore sfondo */
	padding: 0; /* Padding del body */
	margin: 0; /* Margine del body */
	}
	
	/* definisco i link (tag a href) */
	a:link {
	text-decoration: none; /* nessuna decorazion */
	color: #444; /*colore uguale a quello del body */
	}
	a:visited {
	text-decoration: none;
	color: #444;
	}
	a:hover {
	text-decoration: none;
	color:#FF0D0D; /*colore rosso se passo sopra */
	}
	a:active {
	text-decoration: none;
	color: #444;
	}
	#testata {
	width:100%;
	margin-bottom:0.5em;
	border-bottom:#000 3px solid;
	background-color:#E0E0E0;
	}

Abbiamo creato una testata per la nostra pagina

Adesso aggiungiamo nel nostro file css la classe .titolo

```
.titolo {
        text-transform:uppercase;
        font-size:1.5em;
        color:#F00;
}
```

E nella pagina html scriviamo al posto di “Qua la testata” il codice:

```
<h1 class="titolo"> Il mio primo sito web </h1>
```

Salviamo entrambi i file e vediamo che abbiamo creato una testata.

Differenza tra ID e Classe nei CSS

Nella prosecuzione della nostra pagina abbiamo inserito due nuovi selettori chiamati classe e ID (#).
Questi sono due attributi fondamentali applicabili a tutti gli elementi ed hanno la stessa funzione, con la differenza che gli ID si possono richiamare, all’interno della pagina, una sola volta, invece le classi possono essere richiamate sempre in tutta la pagina.
Per definire una classe si usa far precedere il nome da un semplice punto e quindi si definisce l'insieme di regole . A questo punto se voglio applicare questo stile ad un elemento della pagina basta richiamarlo con l’attributo class=”NOME DELLA CLASSE”.
Gli ID differenziano dalle classi per la caratteristica di poter essere utilizzati sono una volta ll’interno del documento. Questa caratteristica deli ID li rendono particolarmente adatti nell’utilizzo di elementi che riguardano l’architettura del sito.
L’uso degli ID solitamente è indicato per i DIV.

Paragrafi Div e Span

Ci sono alcuni tag che permettono di contenere del testo. Questi sono:

- ✓ ** ** che identifica i paragrafi
- ✓ ** ** che identifica un contenitore generico
- ✓ **** che identifica un tipo di contenitore inline (segue il flusso della linea)

NOTE: Un div può contenere sia ID che class insieme o singolarmente

< div class="titolo">

Span solitamente si usa all’interno di altri tag. Ad esempio:

```
<p>un paragrafo <span class="titolo"> SPAN!</span></p>
```

<P> e <DIV> sono invece tag di tipo block che vengono utilizzati per determinare appunto dei blocchi di informazione nella pagina, come nel nostro esempio che abbiamo definito il blocco testata.
Le principali differenze sono:

- ✓ Un elemento block può contenere altri elementi block e anche elementi inline, mentre un elemento inline può contenere solo altri elementi inline
- ✓ Un elemento block può avere delle dimensioni, un elemento inline no.
- ✓ Ad un elemento block non possiamo attribuire una proprietà vertical-align.
- ✓ Un elemento inline non interrompe il normale flusso della pagina causando la creazione di una nuova riga
- ✓ Con float si può trasformare un elemento inline in un elemento block-level

I blocchi

Aggiungiamo adesso una seziona con i contenuti principali ed una con i contenuti secondari.
Nella sezione css aggiungiamo i selettori:

```
#corpo {
        width:50%;
        min-width:300px;
        max-width:1200px;
        display:inline-block;
        border:#CCC 1px solid;
        vertical-align:top;
}
#dx {
        width:30%;
        min-width:150px;
        max-width:300px;
        display:inline-block;
        border:#CCC 1px solid;
        vertical-align:top;
}
```

E nella file html aggiungiamo sotto il div testata:

```
<div id="corpo">
        <p>Ciao questa è la mia prima pagina</p>
        <p><a href="https://www.google.it">questo è un link a google</a></p>
</div>

<div id="dx">
        qua a dx
</div>
```

Ecco il risultato

Non bellissimo ma funzionale a capire quello che abbiamo fatto
Notiamo che se ridimensioniamo la finestra del nostro browser i blocchi corpo e dx si ridimensionano e spostano. Questo perché abbiamo definito, per ognuno, delle dimensioni massime e minime nel nostro CSS.
La proprieta display:inline-block; e da preferire mi permette di allineare dei blocchi che si dispongano al centro della pagina automaticamente per creare una griglia di blocchi.

Il primo breakpoint

Vediamo di definire il nostro primo breakpoint in modo tale che, quando le dimensioni della larghezza della pagina scendono sotto i 480px, il div DX venda nascosto ed il div corpo occupi il 98% della pagina.
Per fare questo sarà sufficiente modificare il nostro foglio di stile, senza dover toccare la pagina.
Aggiungiamo nel file CSS le seguenti righe:

```
@media screen and (max-width: 480px) {
    #dx {
        display:none; /* nascondo il div DV */
        }

     #corpo {
        width:98%; /* aumento la larghezza del div corpo */
            }
}
```

Come abbiamo accennato all'inizio di questo tutorial abbiamo utilizzato la direttiva @media impostando un comportamento quando le dimensioni massime dello schermo (screen) sono inferiori a 479px;

Si possono anche creare breakpoint specifici per determinate dimensioni. Ad esempio se scrivessi:

```
@media only screen and (min-device-width: 320px) and (max-device-width: 480px) { }
```

Le regole dei selettori contenuti in questo breakpoint sarebbero valide solamente per una risoluzione che va da 321px fino a 479px.

Delle regole per la stampa

Allo stesso modo di come impostiamo delle direttive per le dimensioni dello schermo possiamo impostare delle direttive da usare nel caso si stampi la pagina.
Per fare questo non è sufficiente creare la direttiva: @media print{ } che contiene le regole da usare ma occorre anche impostare la pagina e i font (oltre a quello che si vuole o non vuole che venga stampato)

Vediamo un esempio

```
@media print{
        @page {size: 210mm 297mm; margin: 10mm;} /* imposto le dimensioni della pagina stampa A4 */

body {
  background: white!important;
  font-size: 10pt!important;
  color: black!important;
  font-family: "Times New Roman", Times, serif!important;

        }
#testata { background: white!important; }
#dx { display:none; }
#corpo {width:98%; }
}
```

NOTE: La keyword **!important** serve per far prevalere una regola CSS rispetto alle altre con cui è eventualmente in concorrenza indipendentemente se si trovi prima o dopo l'altra regola.

Per prima cosa si definisce il selettore print al cui interno si definisce il selettore che imposta le dimensioni della pagina (A4 in questo caso)

Poi la sezione body: impostiamo il colore della pagina bianco e le dimensioni dei font a 10pt (in questo caso usiamo una misura tipografica e non relativa dove **1pt= 0.352777… mm**), quindi il colore dell'inchiostro (black) e la famiglia di font per la stampa.
Allo stesso modo impostiamo lo sfondo della testata da grigio a bianco in modo da non stamparlo a colori. E impostiamo le dimensioni del corpo al 98% (il 98% di 210mm ovvero 205mm) e nascondiamo il div dx.

Creiamo i menù

Normalmente i menù di un sito sono situati sopra e scompaiono alle risoluzioni minori per lasciar spazio ad un hamburger menu.
A questo punto creeremo i nostri menu con una barra che posizioneremo sopra il #testata.
Nella nostra sezione principale del foglio di stile, prima del breakpoint inseriamo un nuovo id chiamato menu:

```
#menu {
        width:100%;
        text-align:right;
        background-color:#000;
        line-height:2em;
        font-size:1.2em;
        color:#FFF;
}
#hamburgher {
        display:none;
}
```

Adesso vogliamo che quando le dimensioni dello schermo diventano troppo piccole il menù sparisca, quindi nel nostro breakpoint inseriremo la regola

```
#menu {
        display:none;
}
#hamburgher {
        display:block;
        float:right;
        margin-right:1em;
}
.titolo {
        text-transform:uppercase;
        font-size:0.9em;
        color:#F00;
        text-align:left;
}
```

In questo modo abbiamo creato una barra che compare nelle grandi dimensioni e scompare nelle piccole, lasciando al suo posto un hamburger menu (che definiremo più avanti).
Abbiamo anche inserito una regola nel breakpoint dove il titolo viene rimpicciolito per far spazio all'hamburgher menu.
Adesso nel nostro html tra il tad body e id div testata inseriamo:

```
<div id="hamburgher">  &#x2630; </div>
<div id="menu">  menu normale </div>
```

☰ è il codice unicode utf-8 che genera il simbolo ☰.

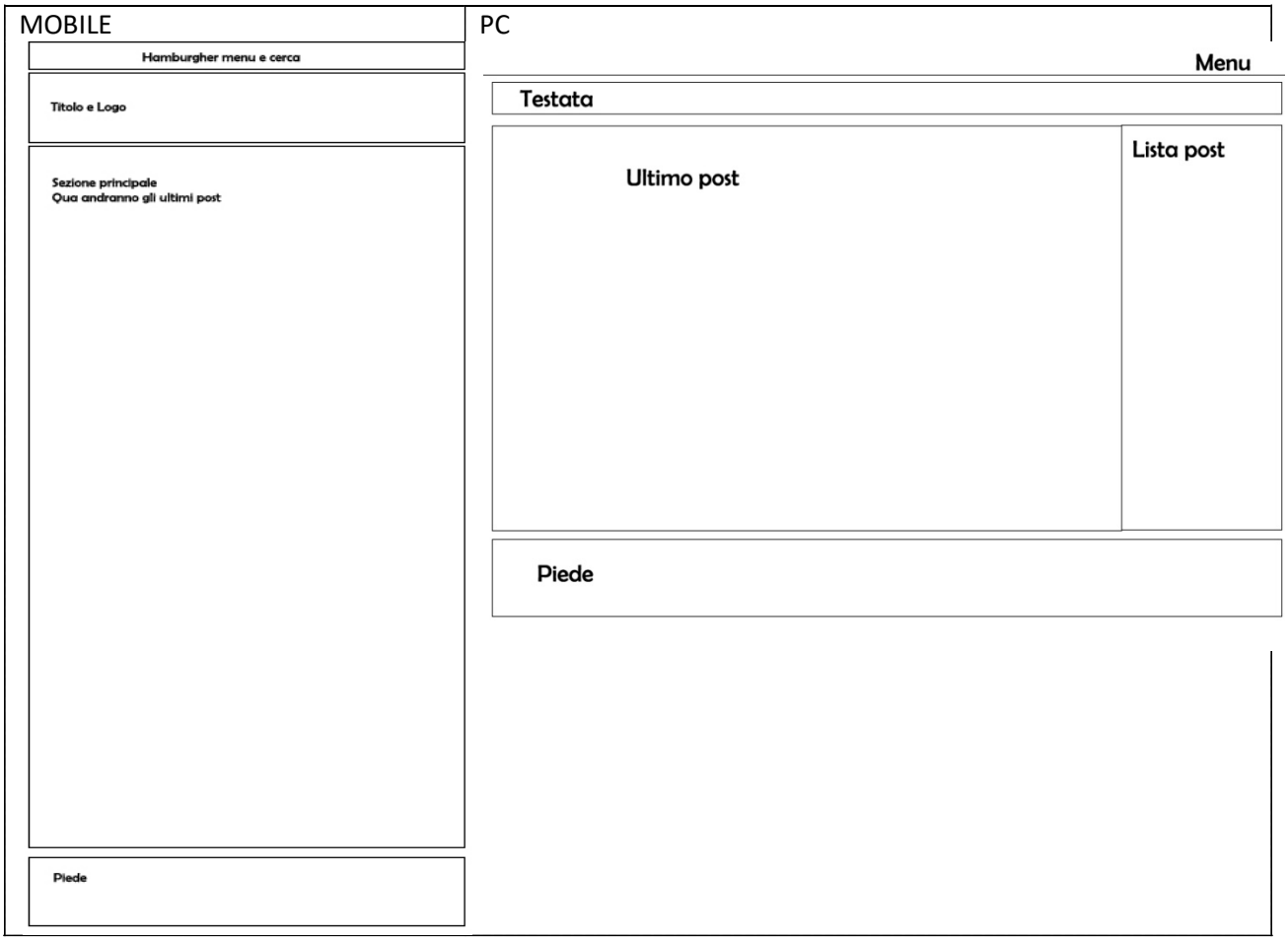
Facciamo le cose seriamente

Ok! Adesso che ci siamo scaldati e abbiamo iniziato a capire come funziona il meccanismo possiamo iniziare a lavorare seriamente su un progetto.
Come ho sempre ripetuto prima di iniziare a scrivere un codice bisogna aver bene in mente quello che si vuole creare, mettersi seduti a scrivere quali saranno i contenuti del sito in ordine di importanza, qual è il target che voglio raggiungere, decidere una veste grafica, colori, ecc. e solo all'ultimo si inizia a scrivere il codice.
Vediamo di fare un sito web per un piccolo blog personale.
Precisiamo subito che in questa sede ci concentreremo sul design ed i fattori di seo di base più che sulla programmazione e l'integrazione con un database.

Le sezioni
Iniziamo con il decidere l'argomento del blog, quindi pensiamo a quali sezioni ci interessano.
Vediamo che avremo:

- Una testata
- Un menu di navigazione
- La sezione con il contenuto principale
- La sezione con i post
- Un piede

La struttura
Come abbiamo detto ad inizio del tutorial bisogna iniziare disegnando la sezione per mobile per poi crescere. Quindi armiamoci di carta e penna e disegniamo prima la parte mobile e poi quella desk



I colori

Pensiamo adesso ai colori che vogliamo utilizzare (non è importante i colori esatti ma solo le tonalità e quello dominante)

Chiaramente la scelta dei colori dipende dal tipo di contenuti che il blog conterrà e del messaggio che voglio trasmettere.

Prendiamo ad esempio il colore nero: sembra il colore più sofisticato, ma risulta anche il più impersonale, mentre il color borgogna viene è espressione di ricchezza e raffinatezza; il color oro può venir associato ad idea di pacchiano. Se volessi fare un blog che parla di cibo sicuramente devo considerare il colore rosso, se invece mi occupo di rimedi naturali o cucina bio meglio il verde.

Non sapendo come fare noi rimarremo sulle sfumature di bianco e nero con qualche macchia qua e la di azzurro o rosso.

I font

Quello della scelta dei fonts e l'aspetto tipografico è uno dei problemi tipici nello sviluppo del sito.

La scelta del font giusto influenza non solo la leggibilità del sito ma anche i tempi di caricamento (che approfondiremo dopo), e l'estetica generale. Ricordate che più è comprensibile quello che scrivo, più aumenta il tempo di permanenza del lettore sul sito.

POCHI COLORI, POCHI FONT: Non esagerare con i colori, scegline pochi (tre, al massimo quattro) e adeguati. Con i font, anche meno. Un sito internet deve essere di facile lettura e gradevole, pertanto devi scegliere colori e font con attenzione.

Per scegliere il font giusto è bene seguire delle semplici regole:

1. **La leggibilità prima di tutto**. Se un sito risulta difficile da leggere, non importa quando sia bello verrà dimenticato presto. Per grandi blocchi di testo (come ad esempio un articolo) è sempre meglio optare per font serif in quanto garantiscono una buona leggibilità del contenuto;
2. **Non lasciamoci influenzare dai gusti personali**: Sebbene un font ci piaccia non è detto che vada bene in quel contesto e soprattutto che sia una buona idea usarlo. Attenzione ad evitare (o quantomeno a limitarne l'uso) dei font calligrafici se li si vogliono usare limitarsi a loghi slogan;
3. **Focalizzarsi sul messaggio che trasmette il sito**. È inutile utilizzare un font gotico o complesso per un prodotto farmaceutico o high tech;
4. **Attenzione alla spaziatura**. Molti fonts creano una fastidiosa spaziatura tra i caratteri, quindi è meglio prima di usarlo provarlo;
5. **Non sia troppo pesante**. Un font che per essere caricato impiega qualche secondo rallenta tutte le prestazioni del sito;
6. **Omogeneità**. Evitate di mischiare, all'interno della stessa pagina, font stilisticamente molto diversi tra loro;

Un servizio molto comodo per la scelta del font può essere quello di google font, tra i quali suggeriamo di usare (se sono interessato alla leggibilità) Roboto, Open Sans, Ubuntu, Pt Sans, Lato.

Lo stile.

Lo stile da utilizzare in un sito è un altro fattore da considerare: come nella moda è importante non che un sito non utilizzi uno stile desueto; la scelta dello stile influenza anche la navigabilità del sito e il target destinatario.

Negli ultimi anni lo stile più utilizzato è quello del material design, anche se molti utilizzano ancora il flat design (per le differenze rimandiamo ai vari articoli disponibili in rete).

Da evitare assolutamente:

- ✓ **Gif animate:** quando in una home page tutto si muove e lampeggia, l'effetto finale risulta veramente fastidioso, inoltre il danno alla credibilità del sito diventa enorme con l'uso indiscriminato di animazioni, tra più diffuse da evitare le animazioni dei lavori in corso e le caselle di posta animate.
- ✓ **Background con texture ripetute:** evitare come la peste gli sfondi psichedelici e ripetuti
- ✓ **Bottoni:** evitare i bottoni fatti con le immagini tramite i filtri di photoshop, gimp, ecc., portati al massimo

Gli script

I siti web moderni fanno largo uso di script, bisogna stare attenti a non esagerare e cercare di ottimizzarli il più possibile, cercando di creare i nostri senza usare quelli di terze parti che a volte hanno molte funzioni inutili che appesantiscono il caricamento.

Posizionare gli elementi da caricare.

Le pagine web vengono caricare partendo "dall'alto", se ad inizio pagina posiziono uno script finché non sarà caricato la pagina non prosegue il caricamento. Devo quindi prestare molta attenzione a dove posiziono gli elementi nella mia pagina per dare la precedenza a quelli importanti nel caricamento.

Ottimizziamo per tutti i browser

Non tutti i browser visualizzano le informazioni allo stesso modo, quindi è molto importante evitare le differenze (per quanto possibile) cross-browser. Per fare questo conviene resettare gli stili CSS.
Tra le varie soluzioni preparate una delle migliori è affidarsi ad un file che si chiama normalize.css che è il frutto di oltre 100 ore di ricerche dettagliate fra le differenze di stile nei browser.
In alternativa si possono resettare tutti gli stili a mano.
Le principali differenze tra un reset o l'utilizzo di normalize.css è che un reset solitamente porta ad un azzeramento totale di tutte le proprietà degli elementi, mentre normalize lascia inalterata la formattazione di base di molti elementi e modificare solo ciò che potrebbe creare diversità nel layout.
L'uso di un metodo piuttosto che l'altro è legata al tipo di progetto e ai gusti personali

Attenzione al SEO

Ultima cosa ma forse una delle prime da considerare è iniziare già in fase di sviluppo a prestare attenzione a tutti quegli aspetti che riguardano l'ottimizzazione del sito web per i motori di ricerca, come ad esempio l'uso dei tag H1, H2,H3, l'inserimento dei social, title sul tag a, attributo alt sulle immagini, coerenza titolo h1, ecc.

Non aver fretta

Infine è bene ricordarsi che la fretta è sempre una cattiva consigliera.
Sembra che Albert Einstein una volta disse: "*Se avessi un'ora soltanto per salvare il mondo, utilizzerei 55 minuti per definire il problema e solo 5 minuti per trovare la soluzione*".
In altre parole, se non vuoi perdere troppo tempo a pianificare e preparare il lavoro meglio che inizi a usare Wordpress, joomla, o qualunque altro dei tools disponibili ma non pretendere che questo sia saper fare un sito web!

Iniziamo a scrivere

Apriamo il nostro editor e iniziamo a scrivere il nostro foglio di stile

Preparazione del foglio di stile
Per prima cosa resettiamo tutte le proprietà scrivendo:

```
html, body, div, span, applet, object, iframe, h1, h2, h3, h4, h5, h6, p, blockquote, pre, a, abbr, acronym, address,
big, cite, code, del, dfn, em, img, ins, kbd, q, s, samp, small, strike, strong, sub, sup, tt, var, b, u, i, center, dl, dt, dd,
ol, ul, li, fieldset, form, label, legend, table, caption, tbody, tfoot, thead, tr, th, td, article, aside, canvas, details,
embed, figure, figcaption, footer, header, hgroup, menu, nav, output, ruby, section, summary, time, mark, audio,
video { margin: 0; padding: 0; border: 0; font-size: 100%; font: inherit; vertical-align: baseline; }
article, aside, details, figcaption, figure, footer, header, hgroup, menu, nav, section { display: block; }
body { line-height: 1; }
ol, ul { list-style: none; }
blockquote, q {    quotes: none; }
blockquote:before, blockquote:after, q:before, q:after { content: ''; content: none; }
table { border-collapse: collapse; border-spacing: 0; }
```

Adesso che abbiamo fatto tabula rasa possiamo iniziare a scrivere il codice come ci piace.
Il font che abbiamo deciso di utilizzare sarà Robot da google font. Provvediamo a caricarlo nel foglio di stile aggiungendo: @import url('https://fonts.googleapis.com/css?family=Roboto').
Impostiamo adesso la base della nostra pagina con sfondo bianco, dimensioni del testo 1em, interlinea di 1.125em, font Roboto e allineamento centrato.
Impostiamo poi le proprietà per i link con colore nero e nessuna sottolineatura

```
html{
            font-size: 100%;
}
body {
            background-color:#FFF;
            font-size =1em;
            line-height:1.125em;
            font-family: 'Roboto', sans-serif;
            text-align: center;

}
a {color:#000; }
a:link {text-decoration: none; }
a:visited {text-decoration: none; color: #000; }
a:hover {text-decoration: none;     color: #000; }
a:active {text-decoration: none;     color: #000; }
```

NOTE: Sebbene si potesse raggruppare in un'unica linea (a:link, a:visited, a:hover, ecc.) le proprietà dei link preferiamo lasciarli separati per poter più agevolmente cambiarne le impostazioni sulle singole proprietà.

Creiamo adesso il nostro file HTML chiamandolo index.html

```
<html >
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>La mia Pagina</title>
</head>


</body>
</html>
```

Ed importiamo il nostro file CSS all'interno della pagina inserendo all'interno del prima della chiusura del tag </head> il comando:

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

Dove all'interno del href inserirò il percorso del mio file CSS.
Ecco come comparirà il mio file html

```
<html >
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>La mia Pagina</title>
<link href="css/stile.css" rel="stylesheet" type="text/css">
</head>

</body>
</html>
```

Creiamo la struttura principale

Torniamo al nostro file CSS e creiamo il nostro breakpoint in modo da iniziare rendere il nostro sito responsitive ad una risoluzione sotto gli 800px scrivendo @media screen and (max-width: 800px) { } alla fine del nostro CSS.

Partiamo dall'alto e iniziamo a scrivere le proprietà dei due menù: vediamo che abbiamo 2 linee che contengono i menu: una per la risoluzione pc che contiene tutti i link, l'altra del mobile che contiene solo l'icona che apre i menu. Quando sono in risoluzione sopra gli 800px mi compare la linea per il pc e non compare quella per tablet e viceversa.
Scriviamo quindi prima del breakpoint:

```
#menupc {
        width:100%;
        text-align:right;
        font-size:1.1em;
        line-height:1.5em;
        height:1.8em;
        background-color:#000;
        color:#FFF;
        display:inline-block;
}
#menumobile {
        width:98%;
        text-align:center;
        border-bottom:#000 1px solid;
        font-size:1.2em;
        margin-bottom:0.5em;
        display:none;
}
```

Mentre all'interno del nostro breackpoin scriviamo:

```
#menupc {
        display:none; }
#menumobile {
        display:inline-block; }
```

Soffermiamoci un attimo ad analizzare questo passaggio: abbiamo definito nella sezione principale le proprietà dei due id specificando quale sarà visibile (display:inline-block;) e quale no (display:none;)

Nel nostro breakpoint sarà sufficiente invertire le due proprietà per far comparire il secondo e scomparire il primo senza cambiarne le proprieà.
Nel nostro file HTML scriviamo all'interno dei tag body:

```
<div id="menumobile">
qua menu mobile
</div>

<div id="menupc">
qua menu pc
</div>
```

Salviamo e apriamo la nostra pagina. Ora proviamo a ridimensionare il browser e vediamo che cambia la linea.
Adesso proseguiremo in questo modo per tutte le altre proprietà della pagina.
Segue quindi la testata.
Per la testata decidiamo di usare un nuovo font. Importiamolo quindi da google font:

```
@import url('https://fonts.googleapis.com/css?family=Rozha+One').
```

sotto l'importazione del font roboto.
Anche per la testata avremo delle dimensioni differenti a seconda della risoluzione dello schermo.
Scriviamo quindi nella sezione principale:

```
#testata {
        width:100%;
        font-size:2em;
        line-height:2em;
        height:2em;
        font-family: 'Rozha One', serif;
        color:#004080;
        font-weight:700;
        text-align:left;

}
```

Mentre nel breakpoing:

```
#testata {
        width:98%;
        font-size:1.3em;
        line-height:1.3em;
        height:1.3em;

}
```

Mentre nel file HTML scriveremo sotto i menù:

```
<div id="testata">
Il mio Blog
</div>
```

Riassumiamo un attimo i nostri due file (nel file CSS è stata rimossa la sezione relativa al reset per questioni di spazio)

HTML

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Documento senza titolo</title>

<link href="css/stile.css" rel="stylesheet" type="text/css">
</head>

<body>
<div id="menumobile">
qua menu mobile
</div>

<div id="menupc">
qua menu pc
</div>

<div id="testata">
Il mio Blog
</div>

</body>
</html>
```

CSS

```
@charset "utf-8";
/* sezione relativa al reset dello stile */
/* CSS Document */

/* inizio foglio */
@import url('https://fonts.googleapis.com/css?family=Roboto');
@import url('https://fonts.googleapis.com/css?family=Rozha+One');

html{
 font-size: 100%;
}
body {
background-color:#FFF;
font-size =1em;
line-height:1.125em;
font-family: 'Roboto', sans-serif;
text-align: center;
}
a {color:#000; }
a:link {text-decoration: none; }
a:visited {text-decoration: none; color: #000; }
a:hover {text-decoration: none; color: #000; }
a:active {text-decoration: none; color: #000; }
#menupc {
width:100%;
text-align:right;
font-size:1.1em;
line-height:1.5em;
height:1.8em;
background-color:#000;
color:#FFF;
display:inline-block;
}
#menumobile {
width:98%;
text-align:center;
border-bottom:#000 1px solid;
font-size:1.2em;
margin-bottom:0.5em;
display:none;
}
#testata {
width:100%;
font-size:2em;
line-height:2em;
height:2em;
font-family: 'Rozha One', serif;
color:#004080;
font-weight:700;
text-align:left;
}

@media screen and (max-width: 800px) {
#menupc {
display:none;
}
#menumobile {
display:inline-block;
}
#testata {
width:98%;
font-size:1.3em;
line-height:1.3em;
height:1.3em;
}

}
```

Passiamo adesso a considerare la sezione contenuti che è un po' più complessa.
Mentre nella visione mobile la sezione contenuti ha una dimensione che copre tutta la pagina nella visione per PC sarà solamente al 70% della pagina perché al 30% vi sarà una barra laterale.

Scriviamo quindi nel nostro fil CSS nella sezione principale:

```
#contenuto {
        width:70%;
        max-width:900px;
        min-width:500px;
        display:inline-block;
        vertical-align:top;
        min-height:80%;
}
#barradx {
        width:28%;
        max-width:300px;
        min-width:150px;
        display:inline-block;
        text-align:right;
        vertical-align:top;
        min-height:80%;
}
```

Mentre nel breakpoin scriveremo:

```
#contenuto {
        width:98%;
        min-width:280px;
}

#barradx {
        display:none;
}
```

Creiamo ora il piede che non cambierà per nessuna risoluzione e che quindi andrà scritto solamente nella sezione principale:

```
#piede {
        width:100%;
        text-align:center;
        font-size:0.8em;
        display:inline-block;
        margin-top:1em;
}
```

Mentre nel codice HTML saà sufficiente aggiungere le righe sotto il div #testata:

```
<div id="contenuto"> Contenuto  </div>

<div id="barradx"> barra dx </div>

<footer id="piede"> Qua il piede </footer>
```

Adesso che abbiamo creato la struttura principale pensiamo a "rifinirla" tramite l'uso delle classi e degli script.

Rifiniamo tramite le classi

Adesso che abbiamo creato l'ossatura del sito possiamo proseguire con le rifiniture.
Sempre partendo dall'alto iniziamo a creare i menù.

I menu
Prima quello mobile: abbiamo detto che nella sezione mobile ci sarà un hamburger menu e una barra di ricerca.
Per creare un hamburger menu ci sono molti metodi: posso creare un immagine, posso usare i css, posso usare dei fonts o usare il codice ☰ .
Siccome a noi non interessa molto come fare decidiamo (per velocià) di usare il codice ☰.
Per chi volesse usare soluzioni già pronte o studiarle suggerisco il link: http://freefrontend.com/css-hamburger-menu-icons/

Creiamo il CSS inserendo sotto l'ultimo id della sezione principale il codice:

```
.hmenu {
        float:left;
        margin: 0.5em 0.5em 0.5em 0.5em;
        font-weight:800;
        cursor:pointer;
        background-color:#000;
        color:#FFF;
        font-size:1.1em;
}
```

Mentre all'interno dell'ID della sezione mobile nel nostro file HTML scriveremo:

☰

```
<div id="menumobile">
  <span class="hmenu">&amp;#9776;</span>
</div>
```

Non è molto bello ma per adesso ci basta; più tardi quando creeremo gli script vedremo come far aprire un menu laterale.

Creiamo adesso il form di ricerca: sempre prima nel CSS e poi nel file HTML. Per fare questo useremo un modulo tramite il tag FORM di HTML che permette di inviare dati da una pagina all'altra.
Il tag form è essenzialmente un contenitore che contiene vari tipi di altri tag, tra questi quello che ci interessa a noi è il tag INPUT che può contenere sia un testo (input type="text") che un bottone per inviare il testo (input type="submit").

Creiamo prima le classi per il form e gli oggetti che contiene:

```
.form {
        margin: 0.5em 0.5em 0.5em 0.5em;
        float:right;
}
.input {
      border:none;
      border-bottom:#000 1px solid;
      font-size:0.8em;
      width:150px;
```

```
}
.bottone {
        background-color:#FFF;
        color:#000;
        -webkit-border-radius: 80px;
        -moz-border-radius: 80px;
        border-radius: 80px;
}
```

Vediamo in questo passaggio che abbiamo inserito una nuova proprietà: webkit.
WebKit è un motore di rendering di browser Web supportato inizialmente solamente da Safari e Chrome è adesso supportato anche a molti altri browser (IE ancora non lo supporta ma chi usa IE oggigiorno?).
Un esempio delle potenzialità di web kit si può vedere su: http://filoweb.it/tutorials/pianeti/sistema.html
È quindi importante considerare nell'uso di web kit che non tutti i browser potrebbero supportarlo ma visto che semplifica di molto la vita e crea dei risultati molto interessanti è un peccato non usarlo.
In questo esempio usiamo web kit per creare un bordo rotondo.

Nel nostro HTML scriviamo (sempre all'interno dell'ID della sezione mobile, dopo l'hamburgher menu) il form con i suoi componenti:

```
<div id="menumobile">
  <span class="hmenu">&#9776;</span>
  <form name="form1" method="post" action="cerca.aspx" class="form">
       <input type="text" name="cercatxt" id="cercatxt" class="input">
      <input type="submit" name="cercabt" id="cercabt" value="?" class="bottone">
  </form>
</div>
```

Alla pressione del bottone chiamato CERCABT il contenuto del testo CERCATXT verrà inviato alla paigna CERCA.ASPX.

Creiamo adesso i menu per la sezione PC. Qua la cosa è più facile in quanto abbiamo più spazio.
Creiamo quindi la classe per i menu:

```
.pcmenu {
            font-size:0.8em;
            margin-right:0.5em;
            color:#FFF;
}
```

E la classe per i menu quando si passa sopra con il mouse:

```
.pcmenu:hover {
            color:#004080;
}
```

Quindi inseriamo le nostre voci di menu all'interno dell'ID menupc

```
<div id="menupc">
  <a href="pg1.html" class="pcmenu" title="pagina1"> link1 </a>
   <a href="pg2.html" class="pcmenu" title="pagina2"> link2 </a>
   <a href="pg3.html" class="pcmenu" title="pagina3"> link3 </a>
    ...
</div>
```

Il titolo

Pensiamo adesso alla sezione del titolo: qua l'unica cosa da cambiare è aggiungere il tag H1 nell' ID del titolo: <h1>Il mio Blog</h1>. Vedremo poi nella sezione dedicata al SEO perché è importante usare i tag H.

I contenuti

Passiamo adesso alla struttura del contenuto principale.
Tutti noi abbiamo usato almeno una volta nella vita Twitter, Facebook o siamo andati a dare un'occhiata al sito news di google e ci siamo accorti che i contenuti sono divisi in "schede".
Per il nostro blog vogliamo creare qualcosa di simile.
Creiamo quindi un contenitore per le schede nella sezione principale:

```
.scheda {
width:99%;
text-align:justify;
min-height:90px;
padding: 0.5em 0.5em 0.5em 0.5em;
background: #fff;
border-radius: 2px;
display: inline-block;
margin-bottom:0.3em;
margin-top: 0.3em;
box-shadow: 0 1px 3px rgba(0,0,0,0.12), 0 1px 2px rgba(0,0,0,0.24);
transition: all 0.3s cubic-bezier(.25,.8,.25,1);
}
```

Nella sezione del breakpoint inseriamo invece solamente un piccolo cambiamento dove faremo le dimensioni della scheda al 95% e il pudding leggermente minore in modo da non creare barre di scorrimento laterali.

```
.scheda {
        width:95%;
        padding: 0.2em 0.2em 0.2em 0.2em;
       font-size:0.95em;
}
```

Creiamo anche un titolo per le singole schede

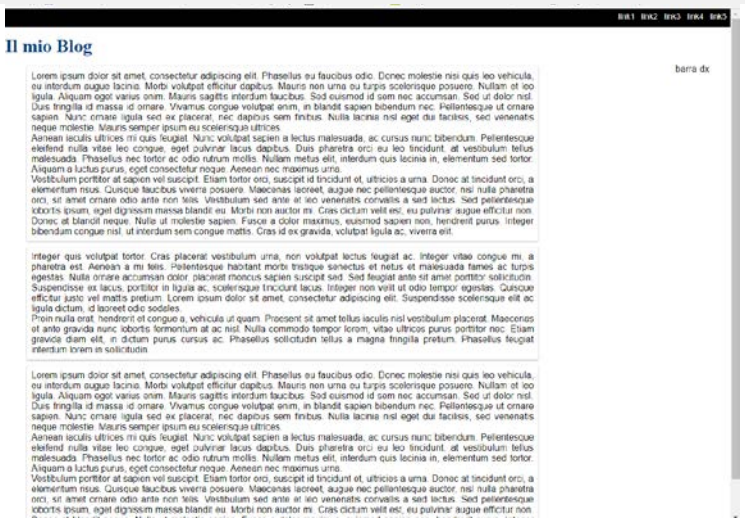
```
.titolino {
      font-family: 'Rozha One', serif;
      color:#004080;
      font-size:1.5em;
      font-weight:700;
      margin-bottom:0.5em;
}
```

Nella sezione HTML all'interno dell'ID contenuto inseriamo un paio di schede

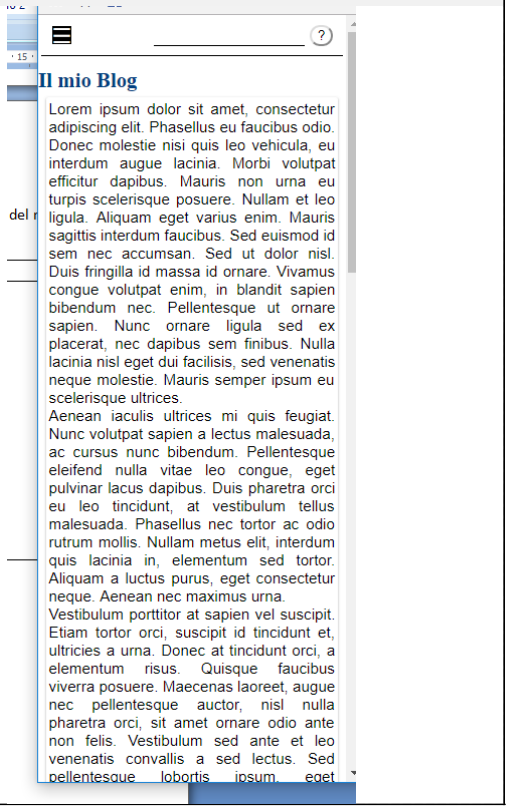
```
<article class="scheda">
        <header><h2 class="titolino"> Tito blog</h2> </header>
          sk1
</article>
<article class="scheda">
         <header><h2 class="titolino"> Tito blog</h2></header>
        sk2
</article>
 <article class="scheda">
       <header> <h2 class="titolino"> Tito blog</h2></header>
        sk3
</article>
```

Possiamo riempire il contenuto delle schede tramite un generatore di testi per avere un idea del risultato.

Versione PC



Versione Mobile



<u>Il tag Article</u>

Il tag <article> (tipico dell'HTML5) rappresenta una sezione autonoma in una pagina o sito potenzialmente ridistribuibile.
Serve per identificare il post di un forum, l'articolo di un blog, un articolo di una rivista o di un giornale, ecc.
Il tag Article può far uso del tah headr e time per specificare il titolo e l'ora della pubblicazione; può anche contenere un tag footer (piede) dove viene specificato ad esempio l'autore o altro.
Per specificare la data di pubblicazione lelemento <time> viene usato con l'attributo pubdate (pubdate datetime="2011-10-09T14:28-08:00").

Inseriamo delle immagini

Sebbene i contenuti siano molto importanti inserire anche qualche immagine non è male. Creiamo dunque le classi per poter inserire immagini sia a destra che a sinistra del testo scrivendo nella sezione principale del CSS le righe:

```
.imgsx {
        float:left;
        width:25%;
        min-width:100px;
        height:auto;
        margin: 0.5em 0.5em 0.5em 0.5em;
        border:#293343 1px solid;
}
.imgdx {
        float: right;
        width:25%;
        min-width:100px;
        height:auto;
        margin: 0.5em 0.5em 0.5em 0.5em;
        border:#293343 1px solid;
}
```

Impostando **height:auto;** nelle proprietà dell'immagine diciamo al browser di calcolare automaticamente l'altezza in base alla larghezza dell'immagine (che è pari al 25% del contenitore che la contiene).

Per inserire un immagine sarà sufficiente posizionarsi nel punto interessato e inserire il tag img con la classe imgdx o imgsx a seconda che la voglia a sinistra o a destra del testo.

```
<article class="scheda">
  <header><h2 class="titolino"> Tito blog</h2></header>
        <p>
   <img src="https://filoweb.it/galleria2015/foto/XJY96QJ95AMROZ5QE1BDA0NY9IBN2E.jpg" class="imgsx"
alt="foto del primo blog" title="Foto del primo blog">Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Phasellus eu faucibus odio. Donec molestie nisi quis leo vehicula, eu interdum augue lacinia. Morbi volutpat
efficitur dapibus. Mauris non urna eu turpis scelerisque posuere. Nullam et leo ligula. Aliquam eget varius enim.
Mauris sagittis interdum faucibus. Sed euismod id sem nec accumsan. Sed ut dolor nisl. Duis fringilla id massa id
ornare. Vivamus congue volutpat enim, in blandit sapien bibendum nec. Pellentesque ut ornare sapien. Nunc
ornare ligula sed ex placerat, nec dapibus sem finibus. Nulla lacinia nisl eget dui facilisis, sed venenatis neque
molestie. Mauris semper ipsum eu scelerisque ultrices.</p>
</article>
<article class="scheda">
  <header><h2 class="titolino"> Tito blog2</h2></header>
        <p>
<img src="https://filoweb.it/galleria2015/foto/7FZ23YSPFQKBUZMZTLFW4LVEG7ETFW.jpg" class="imgdx" alt="foto
del secondo blog" title="Foto del secondo blog">Integer quis volutpat tortor. Cras placerat vestibulum urna, non
volutpat lectus feugiat ac. Integer vitae congue mi, a pharetra est. Aenean a mi felis. Pellentesque habitant morbi
tristique senectus et netus et malesuada fames ac turpis egestas. Nulla ornare accumsan dolor, placerat rhoncus
sapien suscipit sed. Sed feugiat ante sit amet porttitor sollicitudin. Suspendisse ex lacus, porttitor in ligula ac,
scelerisque tincidunt lacus. Integer non velit ut odio tempor egestas. Quisque efficitur justo vel mattis pretium.
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Suspendisse scelerisque elit ac ligula dictum, id laoreet odio
sodales.</p>
<p>Proin nulla erat, hendrerit et congue a, vehicula ut quam. Praesent sit amet tellus iaculis nisl vestibulum
placerat. Maecenas et ante gravida nunc lobortis fermentum at ac nisl. Nulla commodo tempor lorem, vitae ultrices
purus porttitor nec. Etiam gravida diam elit, in dictum purus cursus ac. Phasellus sollicitudin tellus a magna fringilla
pretium. Phasellus feugiat interdum lorem in sollicitudin.</p>
</article>
```

Per le immagini sarebbe meglio usare il tag **<picture> </picture>** di HTML5 ma per il nostro anche il semplice img va più che bene

La barra laterale

Passiamo adesso alla sezione di destra della versione PC.
Per creare questa sezione useremo delle liste, e per fare questo usiamo il costrutto css list-style UL e LI
Creiamo la sezione css solamente nella sezione principale ed inseriamo una proprietà per i titolo delle liste.

```
ul {
        margin-bottom:1em;
}
.titlista {
        color:#004080;
        font-size:1em;
        text-decoration:overline underline;
        font-weight:700;
}
```

Mentre nella nostra pagina html scriviamo il codice all'interno dell'ID barradx:

```
<div id="barradx">

    <ul id="bllist" >
        <li>
          <h4 class="titlista">2018</h4>
            <p>
              <a href="blog1.html" title="blog11"> titolo blog </a><br/>
              <a href="blog2.html" title="blog2"> titolo blog </a><br/>
           </p>
        </li>
    </ul>

    <ul id="bllist" >
        <li>
          <h4 class="titlista">2017</h4>
            <p>
              <a href="blog3.html" title="blog3"> titolo blog </a><br/>
              <a href="blog14.html" title="blog14"> titolo blog </a><br/>
            </p>
        </li>
    </ul>

</div>
```

Abbiamo diviso i diversi anni in blocchi ed ogni blocco è racchiuso tra un paragrafo, il perché di questa soluzione sarà chiara più avanti quando creeremo gli script.

Il piede pagina.

Questa ultima sezione la possiamo personalizzare come vogliamo, solitamente vi si inseriscono i dati di chi fa il blog o i contatti o i ©.

```
<footer id="piede">
Sito web by <a href="https://filoweb.it" target="_blank" title="Filo web">www.filoweb.it</a><br/>
(c) 2018
</footer>
```

Gli script

Abbiamo lasciato per ultimo la sezione relativa agli script perché (come abbiamo detto molte volte) si inizia dal piccolo e si cresce.
Per fare i nostri script useremo jscript e la libreria jquery; gli script sono l'ultima cosa che inseriremo per avere la nostra pagina pronta ed è anche l'ultima cosa (quando possibile) da inserire sono gli script.
Un'altra cosa che consigliamo è di creare un file a parte per i java script che verrà importato nella pagina.
Per approfondire i Jquery consiglio il tutorial: https://www.filoweb.it/appunti/jQuery.pdf

Lo script menu

Il primo script che creiamo è quello per aprire il menu nella versione mobile.
Per prima cosa bisogna richiamare la libreria jquery all'interno della nostra pagina scrivendo nel file html prima della chiusura del tag body:

```
<script type="text/javascript" src="https://ajax.googleapis.com/ajax/libs/jquery/1.6/jquery.min.js" async></script>
```

Adesso torniamo nella sezione CSS; per il nostro esempio creiamo una semplicissimo DIV che copre tutto lo schermo e si sovrappone a tutto quando premo l'hamburgher (chiamato slashmenu) menu e si chiude quando premo il tasto X (che avrà la proprietà chiamata menu).
Nel nostro file css scriviamo quindi nella sezione principale:

```
#slashmenu {
        width:100%;
        height:100%;
        min-height:100%;
        z-index:1001;
        position:fixed;
        top:0px;
        left:0px;
        background-color:#000;
        display:none;
}
.cmenu {
        font-weight:800;
        cursor:pointer;
        background-color:#000;
        color:#FFF;
        font-size:1.1em;
}
```

Che andremo a posizionare come primo div nel nostro file HTML subito dopo il tag di aperture <body> e che conterrà le voci del menu.

```
<div id="slashmenu">
    <h2> Il mio Blog</h2>
      <ul>
          <br/>
          <li> <a href="pg1.html" class="pcmenu" title="pagina1"> link1 </a> </li>
          <li> <a href="pg2.html" class="pcmenu" title="pagina2"> > link2 </a> </li>
          <li> <a href="pg3.html" class="pcmenu" title="pagina3"> > link3 </a> </li>
```

```
            <li> <a href="pg4.html" class="pcmenu" title="pagina4"> > link4 </a> </li>
        </ul>
</div>
```

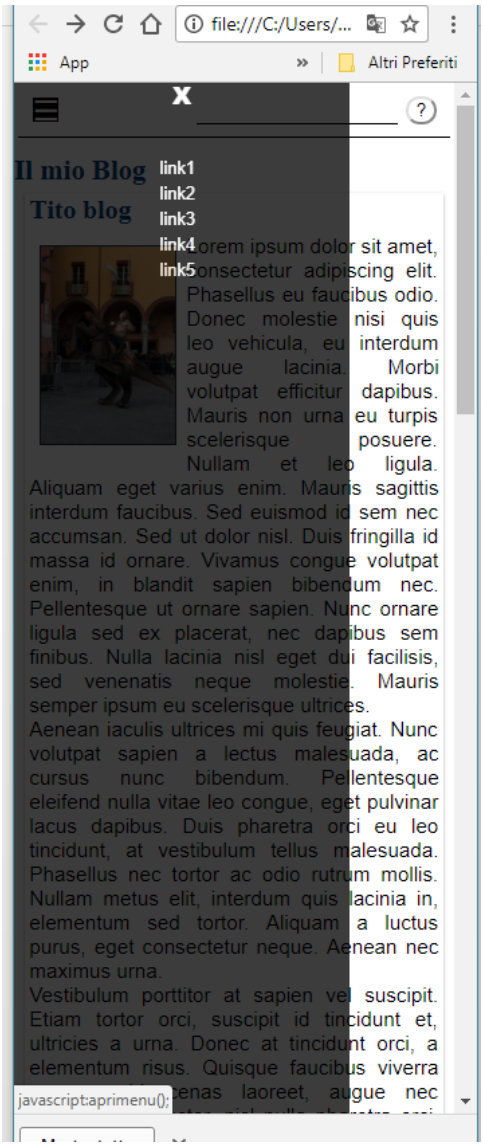
Chiaramente avendo impostato la proprietà display:none; non vedrò ancora nulla.
Creiamo adesso un nuovo file con il nostro editor che chiameremo script.js e scriviamo due funzioni che verranno richiamate dal click su hmenu per aprire il menu e cmenu per chiuderlo.

```
function aprimenu () { $("#slashmenu").show("slow"); };
function chiudimenu () { $("#slashmenu").hide("slow"); };
```

Adesso non ci resta che richiamare questo script all'interno del nostro file HTML allo stesso modo di come abbiamo richiamato la libreria Jquery, mettendolo dopo quest'ultima:

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

Ecco il risultato dell'hamburger menu in apertura:



Lo script archivio

Quando abbiamo creato la barra laterale abbiamo detto che la struttura in blocchi dove ogni blocco è racchiuso tra un paragrafo sarebbe stata più chiara, vediamo di chiarire il perché si è voluto creare un sistema simile invece che limitarsi a mettere una semplice lista.
All'apertura del sito vogliamo che le liste dei blog negli anni vengano chiuse e rimangano solo gli anni che si aprono al click. Scriviamo quindi dentro il nostro file script.js

```
$(document).ready(function(){
        $("#bllist li > *:not(h4)").hide();
        $("#bllist h4").css("cursor","pointer").click(function (){
                var  anno = $(this).siblings();
                anno.slideToggle("slow");
         });
  });
```

Ecco il risultato della nostra pagina per PC (chiaramente nella versione mobile la barra laterale non è presente).

Condivisone

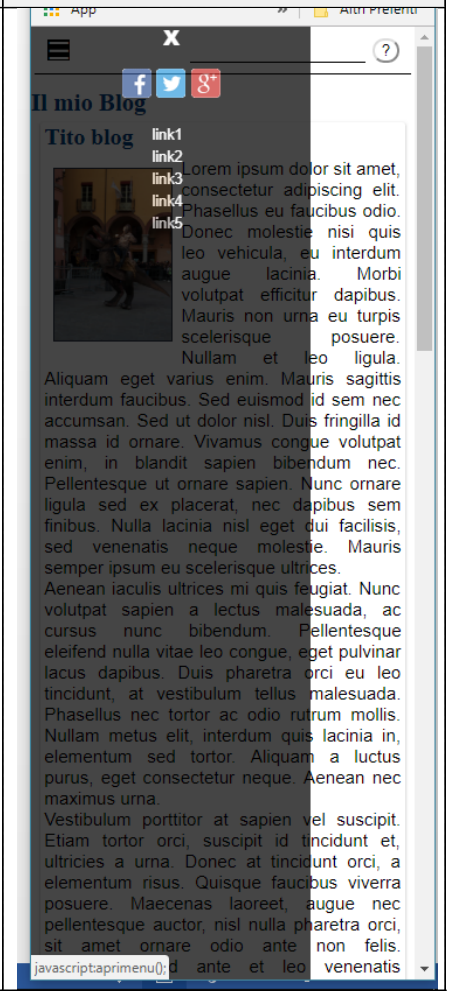
Condividere sui social è diventato ormai un obbligo e per fortuna i maggiori social ci mettono a disposizione gli strumenti per farlo con poche righe da inserire nel nostro file HTML nel punto dove voglio che compaiano. Nel nostro caro li metteremo per la versione PC sopra gli anni sulla barra di destra

```
<a href="https://www.facebook.com/sharer.php?u=https://wwwilmioblog/index.html&title=Il mio blog"
target="_blank" title="Facebook"  >
<img src="https://www.filoweb.it/fb.png" alt="FB" title="Facebook" width="25" height="25"  /></a>
<a href="https://twitter.com/share?u=ttps://wwwilmioblog/index.html&title=Il mio blog" target="_blank"
title="Twitter">
<img src="https://www.filoweb.it/twitter.png" alt="TW" title="twitter" width="25" height="25" /></a>
<a href="https://wwwilmioblog/index.html&title=Il mio blog" target="_blank" title="G+">
<img src="https://www.filoweb.it/gplus.png" alt="G+" title="G+" width="25" height="25" /></a>
```

mentre per la versione mobile sopra i menu nel div slashmenu sotto il tag H2

```
<h2> Il mio Blog</h2>
<a href="https://www.facebook.com/sharer.php?u=https://wwwilmioblog/index.html&title=Il mio blog"
target="_blank" title="Facebook" >
<img src="https://www.filoweb.it/fb.png" alt="FB" title="Facebook" width="25" height="25" /></a>
<a href="https://twitter.com/share?u=ttps://wwwilmioblog/index.html&title=Il mio blog" target="_blank"
title="twitter">
<img src="https://www.filoweb.it/twitter.png" alt="TW" title="twitter" width="25" height="25" /></a>
<a href="https://wwwilmioblog/index.html&title=Il mio blog"  title="G+" target="_blank">
<img src="https://www.filoweb.it/gplus.png" alt="G+" title="G+" width="25" height="25" /></a>
```

Ecco I due risultati:

Versione PC	Versione Mobile
	

Un po' di SEO

Fare SEO (ovvero ottimizzare un sito web per i motori di ricerca) non è una cosa semplice e non basta focalizzarsi solo sulle parole chiave che sono all'interno del nostro sito, ma bisogna anche considerare la maggior parte delle oltre 200 caratteristiche considerate da Google.
Vediamo adesso le caratteristiche principali.

Il Titolo

Prima di tutto, ogni pagina deve avere un titolo che si inserisce tramite il tag <title> </title>
Il titolo deve essere chiaro e facile, deve essere riportato nel meta tag title e deve essere lo stesso del tag <H1> </H1>; il tag title è una delle prime cose e più importanti che i motori di ricerca cercano per la determinazione del contenuto e della pertinenza del sito, deve quindi essere esplicativo del contenuto della pagina senza superare il 60 caratteri.
Nella nostra pagina HTML scriviamo tra i tag <head> </head>:

```
<title>Il mio Blog</title>
<meta name="title" content="Il mio Blog" />
```

Che è lo stesso che è scritto nel tag <h1></h1>

Description

Il meta tag description è l'attributo HTML che fornisce le spiegazioni concise dei contenuti della pagina web, Viene comunemente usato dai motori di ricerca nelle pagine dei risultati per visualizzare frammenti di anteprima di una determinata pagina. La lunghezza ottimale è di circa 155 caratteri, spazi compresi.
Google ha annunciato nel 2009 che le description non costituiscono un fattore negli algoritmi di ranking, ma sono importanti per permettere una buona anteprima sui motori di ricerca, inoltre viene richiamata nella descrizione della pagina quando la si condivide sui social network nel caso non siano presenti gli Open Graph (che vedremo più avanti).
Sempre nell'<head> della pagina, sotto il meta name, scriviamo:

```
<meta name="description" content="Come creare un sito web II° Parte. In questa seconda parte parleremo di blog, script, seo e CSS con esempi facili" />
```

La Lingua Usata

È importante specificare la lingua usate nella pagina. Per fare questo si usa il meta Content-language che va inserito sempre nell'<head>

```
<meta http-equiv="content-language" content="it-it">
```

I Robots

Per fornire istruzioni dirette agli spider dei motori di ricerca sul comportamento da tenere scansionando la pagina, possiamo utilizzare i meta tag robots. Questo meta tag offre diverse possibilità:

- ✓ index - si richiede di indicizzare la pagina (di inserirla cioè nell'archivio del motore di ricerca);
- ✓ noindex - si richiede di non includere la pagina negli archivi del motore di ricerca;
- ✓ follow - si richiede che tutti i link presenti nella pagina vengano seguiti; questo consente anche il passaggio di valore da una pagina all'altra, come avremo modo di vedere in modo molto ampio;
- ✓ nofollow - si richiede di non seguire i link che da quella pagina puntano verso altre pagine.

Scriviamo dopo il meta del linguaggio:

```
<meta name="robots" content="INDEX,FOLLOW" />
```

Opengraph

Open Graph è un protocollo introdotto nel 2010 che permette la condivisione di contenuti, website e applicazioni di terze parti con il proprio network di amici.
Open Graph è un protocollo aperto che fa uso di alcuni tag <meta> all'interno della porzione <head> del documento; attraverso Open Graph è possibile definire le informazioni che saranno condivise dai social nel momento in cui la nostra pagina verrà condivisa da qualcuno.
Facebook e Twitter poi hanno degli open graph specifici.
I principali tag sono:

- ✓ og:url - serve per indicare la URL canonica della specifica risorsa web che viene condivisa (la URL canonica è una specie di URL "ufficiale", priva cioè di tutti quegli elementi facoltativi come querystring che impostano sessioni, filtri o quant'altro di non indispensabile).
- ✓ og:title - serve per definire il titolo della specifica risorsa web che viene condivisa (bisogna considerare che Facebook tronca il titolo dopo 88 caratteri, quindi non esageriamo con titoli troppo lunghi).
- ✓ og:image - serve per indicare la URL assoluta dell'immagine che andrà a rappresentare lo specifico oggetto che verrà creato nel Social Graph (la dimensione ideale è di 1200 x 630 pixel, mentre la dimensione minima è di 600 x 315 pixel; se l'immagine è inferiore a 600 x 315 pixel continuerà a essere visibile nel post ma con dimensioni notevolmente minori).

Mentre per gli opengraph specifici di facebook iniziano con fb: e quelli di tweitter con twitter: .
In particolare sono importanti per facebook

```
<meta property="fb:app_id" content="123456789"/>
<meta property="fb:pages" content="https://www.facebook.com/XXXXX"/>
```

Che identificano l'utente facebook proprietario della pagina
Mentre gli stessi open graph in twitter diventano:

```
<meta name="twitter:site" content="@filob2013"/>
<meta name="twitter:creator" content="@filob2013"/>
```

Un esempio di open graph potrebbe essere

Meta Open Graph	Significato
<meta property="fb:app_id" content="123456789"/>	Id Facebook autore
<meta property="fb:pages" content="https://www.facebook.com/xx"/	Pagina facebook autore
<meta property="og:site_name" content="https:/filoweb.it"/>	Indirizzo sito web
<meta property="og:url" content="https://filoweb.it/pagina.html"/>	Indirizzo della pagina
<meta property="og:type" content="article"/>	Tipo di contenuto
<meta property="og:locale" content="it_IT"/>	Lingua e nazione
<meta property="og:title" content="TITOLO"/>	Titolo della condivisione
<meta property="og:image" content="https://filoweb.it/sfondo.jpg"/>	Immagine che viene condivisa
<meta property="og:description" content="Il mio primo Blog"/>	Descrizione
<meta name="twitter:card" content="summary"/>	Open Graph di twitter
<meta name="twitter:title" content="Il mio blog" />	Titolo
<meta name="twitter:description" content=" Il mio primo Blog " />	Descrizione
<meta name="twitter:site" content="@xxxxxx"/>	Indirizzo pagina twitter
<meta name="twitter:creator" content="@filob2013"/>	Autore (nome twitter)
<meta name="twitter:image" content="https://filoweb.it/sfondo.jpg"/>	Immagine

Il nostro <head></head> sarà

```
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />

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

<title>Il mio Blog</title>
<meta name="title" content="Il mio Blog" />
<meta name="description" content="Come creare un sito web II° Parte. In questa seconda parte parleremo di blog, script, seo e CSS con esempi facili" />
<meta http-equiv="content-language" content="it-it">
<meta name="robots" content="INDEX,FOLLOW" />

<meta property="fb:app_id" content="123456789"/>
<meta property="fb:pages" content="https://www.facebook.com/xx"/
<meta property="og:site_name" content="https:/filoweb.it"/>
<meta property="og:url" content="https://filoweb.it/pagina.html"/>
<meta property="og:type" content="article"/>
<meta property="og:locale" content="it_IT"/>
<meta property="og:title" content="TITOLO"/>
<meta property="og:image" content="https://filoweb.it/sfondo.jpg"/>
<meta property="og:description" content="Il mio primo Blog"/>
<meta name="twitter:card" content="summary"/>
<meta name="twitter:title" content="Il mio blog" />
<meta name="twitter:description" content=" Il mio primo Blog " />
<meta name="twitter:site" content="@xxxxxx"/>
<meta name="twitter:creator" content="@filob2013"/>
<meta name="twitter:image" content="https://filoweb.it/sfondo.jpg"/>
```

Alt sulle immagini.

Quando inserisco un immagine all'interno di una pagina web è OBBLIGATORIO inserire l'attributo ALT.
L'attributo al non serve solamente a visualizzare una descrizione se non compare un immagine, ma viene anche usata dai motori di ricerca per indicizzare le immagini.

Il formato è:

Quindi, adesso torniano nella nostra pagina web e impostiamo l'attributo alt su tutte le immagini (comprese quelle dei social) se non lo abbiamo già fatto!

L'attributo title

L'attributo title è molto importante, e serve per specificare un testo esplicativo per l'elemento a cui l'attributo è riferito, questo favorisce l'accessibilità del sito anche ai disabili, alle persone per esempio che hanno disturbi alla vista. Se si lascia il cursore del mouse per qualche secondo su un collegamento dotato di title, si vedrà comparire una specie di etichetta con il testo specificato nel title, così come sulle immagini.
L'attributo title è anche utilissimo per migliorare la presenza nei motori di ricerca, che ne vanno a leggere il contenuto.
Quindi aggiungiamo l'attributo title nelle immagini e nei link nel caso non lo si sia già fatto.
Es.:
<img src="immagin.jpg" alt="titolo dell'immagine " **title="titolo dell'immagine"** class="immagine" /
www.filoweb.it

I microdata

I microdati sono una specificazione HTML5 che mette a disposizione una meta-sintassi per il markup dei dati strutturati.
I dati strutturati sono particolarmente importanti ai fini dell'ottimizzazione per i motori di ricerca, visto che le annotazioni semantiche agevolano l'indicizzazione di un sito e permettono di visualizzarlo nei risultati di ricerca con informazioni aggiuntive.
Per strutturare i dati, a seconda del tipo di contenuto consiglio di consultare il sito ufficiale Schema.org.
Per usare i dati strutturali posso usare dei markup specifici (itemscope itemtype), oppure tramite l'uso di JSON-LD, un linguaggio di marcatura delle pagine e che permette di comunicare direttamente con gli algoritmi robot del motore di ricerca e che è da preferire per l'indicizzazione su google e Bing.

Un esempio di json-ld per website è:

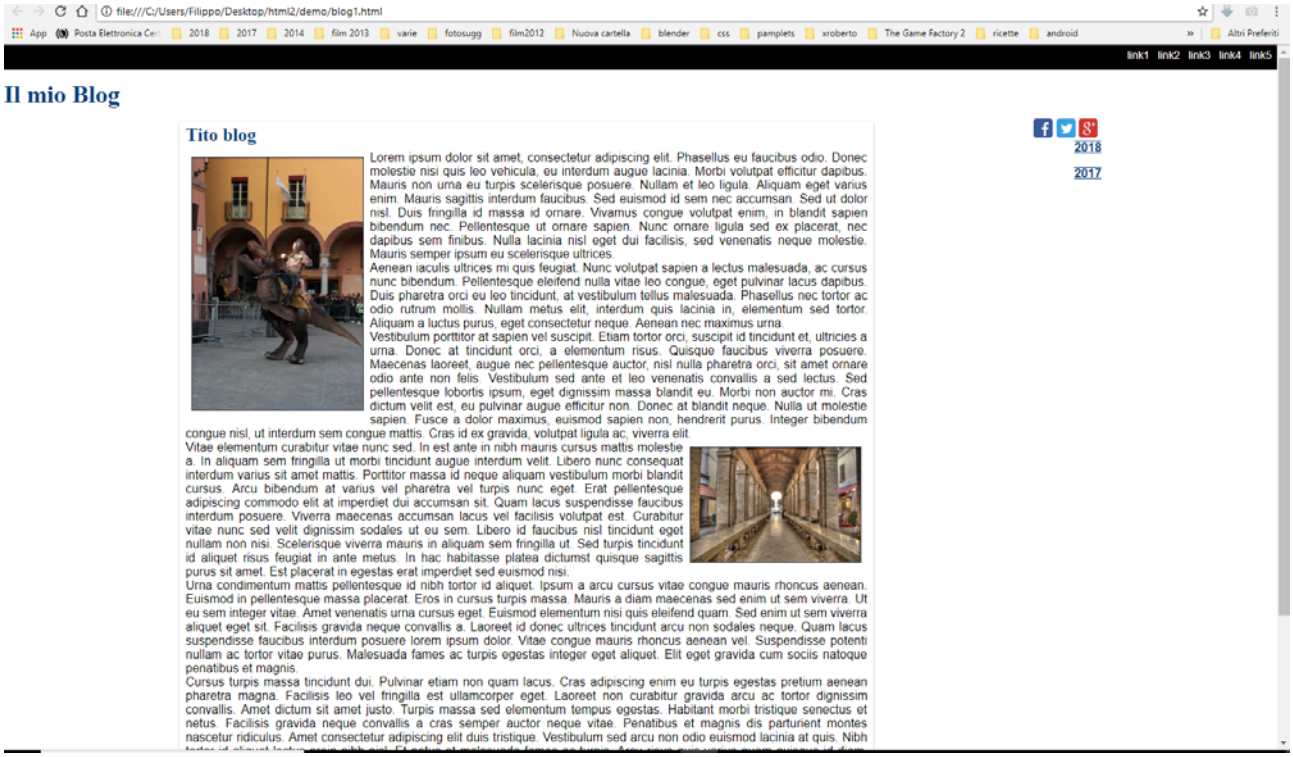
```
<script type='application/ld+json'>
{

"@context": "http://schema.org",
"@type": "NewsArticle",
"mainEntityOfPage": {
"@type": "WebPage",
"@id": "https://filoweb.it/giornale/"
                    },
"headline": "Qualcosa non ha funzionato",
"image": "https://www.filoweb.it/giornale/foto/73T2PFOZPNLP9W7.jpg",
"datePublished": "2018-2-28T08:00:00+08:00",
"dateModified": "2018-2-28T09:20:00+08:00",
"author": {
"@type": "Person",
"name": "Filippo Brunelli"
},
"publisher": {
"@type": "Organization",
"name": "Filoweb.it",
"logo": {
"@type": "ImageObject",
"url": "https://filoweb.it/sfondo.jpg"
}
},
"description": "Filoweb Magazin, articolo: Qualcosa non ha funzionato"
}
 </script>
```

Esistono moltissime strutture che possono essere trovate per ogni tipo di sito web all'indirizzo ufficiale: http://schema.org/

Creiamo le altre pagine

Bene adesso che abbiamo fatto la prima pagina (index.html) creiamo le alte.
Diciamo subito che le singole pagine non sono molto differenti dalla prima a parte che conterranno un solo post, quindi l'unica differenza sarà quella relativa al contenuto centrale con un solo <article> (nulla però ci vieta di metterne più di uno



Ricordiamoci di cambiare i metatag description e i link per la condivisione ed il seo.

Le regole per la stampa

Per finire definiamo le regole per la stampa.
Nella stampa ci interesserà che compaia il titolo del sito (#testata), la scheda (.scheda) con tutto il suo contenuto ed il piede. Il resto verrà nascosto.
Una delle cose da tener presente nella stampa delle pagine è l'interruzione della pagina per evitare che le parole siano metà su di una pagina e metà su di un'altra.
Per fare questo CSS ci mette a disposizione dei costrutti ad hoc che ci permettono di interrompere dopo un blocco, prima o impedire che un blocco sia troncato.
I principali costrutti sono:

- ✓ page-break-after: che inserisce un salto page-break dopo di questo elemento
- ✓ page-break-before: che inserisce un salto page-break prima di questo elemento

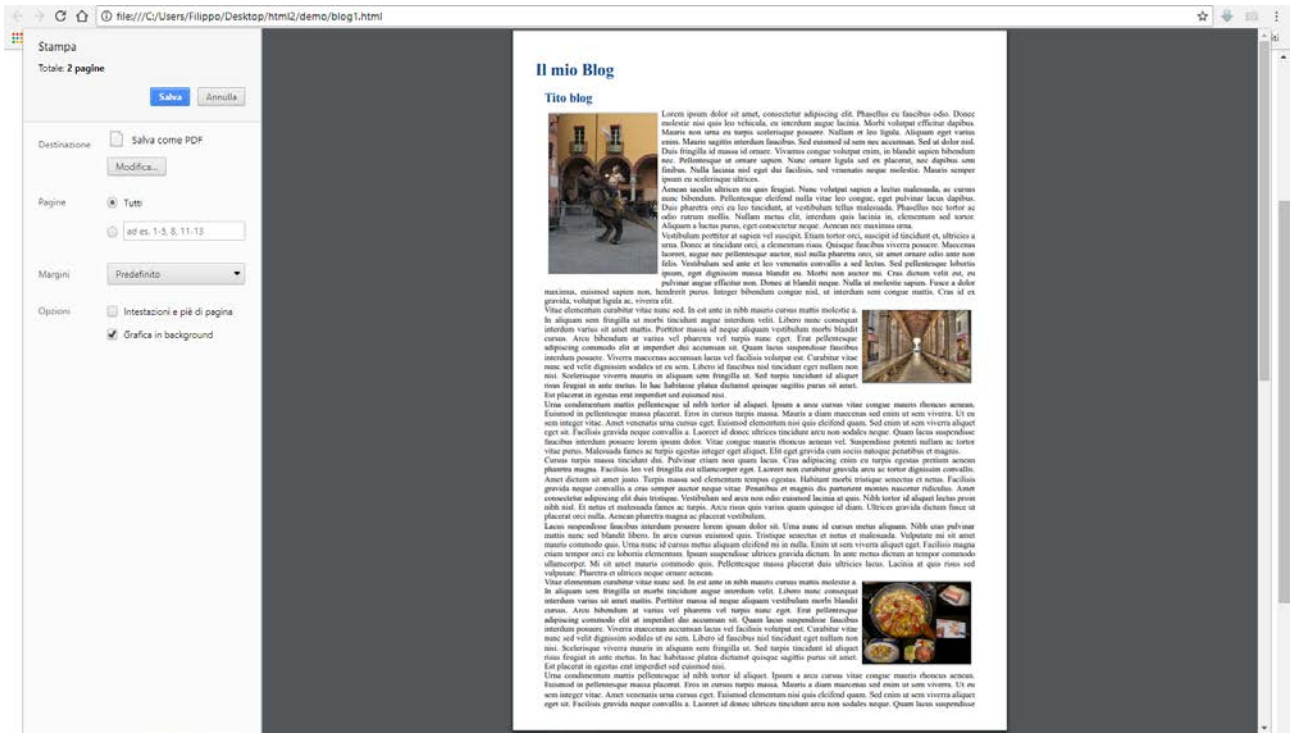
Quindi nella nostra direttiva @media print inseriremo nella classe .scheda la regola: page-break-before: always.
Ma non è tutto vogliamo che anche i paragrafi non siano tagliati, quindi inseriamo una nuovaproprietà per i paragrafi   dove inseriremo la stessa regola

```
@media print{

@page {size: 210mm 297mm; margin: 10mm;} /* imposto le dimensioni della pagina stampa A4 */

body {
   background: white!important;
   font-size: 10pt!important;
   color: black!important;
   font-family: "Times New Roman", Times, serif!important;
}
#testata { background: white!important; }
#contenuto { width:98%; }
.scheda {
            width:98%;
             border:none;
            page-break- after: always;
 }
 p { page-break- after: always;  }
 #barradx { display:none; }
#menupc { display:none; }
}
```

Ecco il risultato di una pagina stampata



Conclusioni

Questo tutorial non vuole essere una guida complete, ma un aiuto ad approfondire gli argomenti trattati nel precedente (https://www.filoweb.it/appunti/lhtml.pdf).
È richiesta una conoscenza di base di HTML e jquery (tutorial https://www.filoweb.it/appunti/jQuery.pdf).
Per approfondire gli argomenti trattati ci sono moltissimi altri tutorial e appunti, nonché i siti ufficiali W3C, Schema.org, ecc.
Non abbiamo trattato di programmazione PHP o .Net nonché di integrazione con un database (anche se sarebbe stato indispensabile usare un Database per u nblog) in quanto merita un trattamento particolare.
Ci siamo concentrati sul lato CSS, responsitive, script e Seo, in quanto la sezione HTML era stata precedentemente introdotta e spiegata.