**Metatag**

Principali metatag:

Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente

Quello a riga 12 definisce ogni quanti secondi la pagina deve aggiornarsi in maniera automatica.

**Entità HTML**

Entità html che ancora oggi hanno un senso d’utilizzo:

Immagine che contiene testo, schermata, Carattere, ricevuta

Descrizione generata automaticamente

**Commenti HTML**

Inserire commenti nel codice:

Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente

Immagine che contiene testo, Carattere, schermata, software

Descrizione generata automaticamente

**Struttura tipica di un sito html-css**

Immagine che contiene testo, schermata, Carattere, software

Descrizione generata automaticamente

**Funzioni per i colori**

Definizione dei colori attraverso la funzione css rgb (red green blue):

Immagine che contiene testo, schermata, software, Carattere

Descrizione generata automaticamente

Definizione dei colori con la funzione rgba. L’ultimo valore indica l’opacità ed è un numero compreso tra 0 e 1:

Immagine che contiene testo, schermata, Carattere, software

Descrizione generata automaticamente

Definizione dei colori con codici esadecimale:

Immagine che contiene testo, schermata, Carattere, software

Descrizione generata automaticamente

Definizione dei colori con le funzioni hsl e hsla:

Immagine che contiene testo, schermata, Carattere, software

Descrizione generata automaticamente

Si consiglia di utilizzare i cosiddetti ‘web self color’, ossia quei colori la cui rappresentazione esadecimale è formata da coppie con singoli valori uguali (esempio #FF0099). Questi colori hanno la caratteristica di essere visibili allo stesso modo su qualunque monitor.

**Dimensionamento degli elementi: proprietà width, height e similari**

Definizione della larghezza di un elemento utilizzando come unità di misura i pixel:

Immagine che contiene testo, schermata, Carattere, software

Descrizione generata automaticamente

Possiamo utilizzare anche delle dimensioni relative all’elemento genitore con le percentuali:

Immagine che contiene testo, schermata, Carattere, software

Descrizione generata automaticamente

Le dimensioni relative sono preferibili perché rendono l’elemento più adatto al responsive, caratteristica fondamentale nel web.

Nell’esempio specifico il div avrà una larghezza pari alla metà dell’elemento padre, che in questo caso è il body. Con la proprietà ‘height’ possiamo modificare l’altezza di un elemento ma il funzionamento è diverso. Infatti, le dimensioni relative avranno effetto solo se l’elemento padre avrà un’altezza specificata. In caso contrario, non verranno prese in considerazione. Per quanto riguarda le dimensioni in pixel, il funzionamento è analogo alla larghezza.

In generale, quando l’elemento contiene del testo, si preferisce non utilizzare la proprietà ‘height’ per evitare l’effetto overflow. In questo modo l’altezza dell’elemento si adatterà automaticamente al contenuto.

Immagine che contiene testo, schermata, Carattere, software

Descrizione generata automaticamente

La proprietà ‘max-width’ permette di impostare la larghezza massima di un elemento. L’esempio sopra riportato definisce alla riga 5 una larghezza massima di 800 pixel per i div. Questo sta a significare che se la larghezza del viewport sarà superiore a tale valore, il div avrà una larghezza fissa di 800 pixel. In caso contrario, il div si adatterà automaticamente alla dimensione del viewport.

La proprietà ‘min-width’ funziona in maniera analoga alla precedente ma, in questo caso, viene stabilita una larghezza minima dell’elemento:

Immagine che contiene testo, schermata, software, Carattere

Descrizione generata automaticamente

Nell’esempio sopra riportato, il div ha una larghezza minima di 600 pixel, pertanto, se il viewport avrà una larghezza inferiore, all’interno del browser apparirà una barra di scorrimento orizzontale per consentirci di visualizzare l’elemento nella sua interezza. In caso contrario, le dimensioni del div si adatteranno automaticamente allo schermo.

È possibile utilizzare contemporaneamente le proprietà ‘width’ e ‘min-width’:

Immagine che contiene testo, schermata, software, schermo

Descrizione generata automaticamente

Nell’esempio sopra indicato, la larghezza del div sarà sempre di 600 pixel per una larghezza del viewport <= 1200 pixel. Al crescere di quest’ultima, la larghezza del div sarà sempre la metà di quella del viewport.

Le proprietà ‘min-height’ e ‘max-height’ funzionano analogamente alle precedenti solo che si applicano all’altezza dell’elemento.

Tutte le proprietà che riguardano la larghezza e l’altezza possono essere applicate solo a determinati tipi di elementi, ossia a quei tag di tipo block o inline-block. Per gli altri tipi di tag, le impostazioni dell’altezza e della larghezza saranno ignorate.

**Proprietà margin e padding**

La proprietà ‘margin’ serve per creare dello spazio tra elementi diversi e può essere assegnata in vari modi:

* Specificando un solo valore;
* Specificando due valori;
* Specificando tre valori;
* Specificando quattro valori.

Nel primo caso, il valore assegnato sarà applicato a tutti i lati dell’elemento che pertanto avrà un margine costante in ogni direzione.

Quando si specificano due valori, invece, il primo fa riferimento al margine superiore e inferiore, mentre il secondo a quello destro e sinistro.

Nel terzo caso, il primo valore fa riferimento al margine superiore, il secondo a quello destro e sinistro, il terzo al margine inferiore.

Quando si specificano quattro valori per la proprietà ‘margin’, il primo fa riferimento al margine superiore, il secondo a quello destro, il terzo al margine inferiore mentre il quarto a quello sinistro. In pratica i valori vanno interpretati in senso orario, a partire dal margine superiore.

Qualora volessimo specificare separatamente i valori dei margini, possiamo fare riferimento alle seguenti proprietà:

* margin-top;
* margin-right;
* margin-bottom;
* margin-left.

I valori dei margini possono essere assegnati sia in pixel che in percentuale. Nel secondo caso, il valore sarà calcolato in base alla larghezza dell’elemento genitore.

Per quanto riguarda i margini laterali (destro e sinistro) è possibile utilizzare anche un valore particolare: auto. Questo sta ad indicare che il valore del margine verrà adattato automaticamente dal browser.

Allo stato pratico tale valore è utile quando l’elemento in questione ha una larghezza inferiore allo spazio disponibile e l’effetto che si vuole ottenere è quello di centrarlo orizzontalmente rispetto al viewport:

Immagine che contiene testo, schermata, Carattere, software

Descrizione generata automaticamente

I discorsi fatti per la proprietà ‘margin’ valgono anche per il ‘padding’. Questa proprietà si occupa della spaziatura interna all’elemento, ossia tra il contenuto e il bordo. Anche in questo caso si possono specificare uno o più valori, con le regole viste in precedenza, oppure assegnare valori alle singole proprietà per personalizzare il padding sui vari lati dell’elemento.

**Definizione di un bordo**

Il bordo può essere definito specificando tre proprietà: la sua larghezza, il colore e lo stile.

Tali proprietà possono essere specificate contemporaneamente:

Immagine che contiene testo, schermata, software, schermo

Descrizione generata automaticamente

Oppure separatamente:

Immagine che contiene testo, schermata, Carattere, software

Descrizione generata automaticamente

Se non si specifica lo stile, nessun bordo viene applicato.

Inoltre, il bordo può essere costruito anche su uno specifico lato dell’elemento:

Immagine che contiene testo, schermata, software, Carattere

Descrizione generata automaticamente

Immagine che contiene testo, schermata, Carattere, software

Descrizione generata automaticamente

Inoltre, si possono definire anche bordi diversi su diversi lati dell’elemento:

Immagine che contiene testo, schermata, Carattere, software

Descrizione generata automaticamente

**Proprietà box-sizing**

Quando si utilizzano le proprietà ‘width’ ed ‘height’, le impostazioni di default del css prevedono che tali proprietà si riferiscano soltanto alla cosiddetta ‘content area’, ossia alla porzione del box-model dedicata al contenuto dell’elemento. Pertanto, le dimensioni totali, se consideriamo il margine come uno spazio esterno, saranno:

* larghezza = width + padding-right + padding-left + border-right + border-left;
* altezza = height + padding-top + padding-bottom + border-top + border-bottom.

Possiamo variare questo comportamento di default attraverso la proprietà ‘box-sizing’, assegnando a quest’ultima il valore ‘border-box’.

Con questa impostazione, i valori assegnati alle proprietà ‘width’ ed ‘height’ saranno considerati complessivi di padding e bordi. Pertanto, eventuali valori assegnati alle proprietà ‘padding’ e ‘border’ verranno sottratti alla ‘content-area’ e l’elemento sarà effettivamente largo di una quantità pari al valore di ‘width’ e alto di una quantità pari al valore di ‘height’.

Questo è il motivo per il quale molto spesso si utilizza una regola css di questo tipo:

Immagine che contiene testo, schermata, software, Software multimediale

Descrizione generata automaticamente

L’asterisco utilizzato alla riga 1 indica che quella regola sarà applicata a qualunque tag.

**Tag per il testo, proprietà display è visibilità di un elemento**

Dal punto di vista semantico non è del tutto corretto inserire del testo direttamente dentro un div, sarebbe più opportuno utilizzare un paragrafo (tag p):

Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente

Immagine che contiene testo, schermata, software, Software multimediale

Descrizione generata automaticamente

La proprietà ‘text-indent’ definita alla riga 13 specifica uno spazio iniziale in corrispondenza della prima riga presente all’interno del paragrafo.

Un altro elemento destinato a contenere del testo è lo span che, a differenza del tag p, è di tipo inline. A tal proposito, non sarà possibile assegnare a questo elemento alcune proprietà css, quali width, height, margin ed altre. Un elemento inline occupa solo lo spazio destinato al suo contenuto. Pertanto, è possibile affiancare più elementi inline sulla stessa riga.

Gli elementi span vengono utilizzati spesso all’interno dei paragrafi, allo scopo di personalizzare la formattazione di specifiche righe di testo.

Il fatto che un elemento sia di tipo blocco o inline dipende da una proprietà css che si chiama ‘display’.

Modificando il valore di questa proprietà possiamo cambiare il comportamento di un elemento. I valori assegnabili sono molteplici, tra i quali troviamo anche ‘inline-block’. Un elemento di tipo ‘inline-block’ presenta tutte le caratteristiche di un elemento di tipo blocco riguardanti il box model ma, allo stesso tempo, può essere affiancato da un altro elemento inline o inline-block.

La proprietà display può essere utilizzata anche per eliminare un elemento dalla pagina html. Infatti, valorizzando tale proprietà a ‘none’ si ottiene proprio questo comportamento.

La visibilità di un elemento all’interno di una pagina html può essere controllata attraverso altre due proprietà css: visibility e opacity.

La proprietà ‘visibility’ può assumere due valori, ‘visible’ o ‘hidden’, i cui significati sono intuitivi.

La differenza sostanziale tra la regola ‘display: none’ e ‘visibility: hidden’ è la seguente: nel primo caso l’elemento viene effettivamente rimosso dalla pagina mentre nel secondo non risulterà visibile ma continuerà ad occupare lo spazio ad esso assegnato.

La proprietà ‘opacity’, invece, ha lo stesso comportamento del canale alfa visto nelle funzioni css ‘rgba’ e ‘hsla’. A tale proprietà è possibile assegnare un qualunque valore numerico compreso tra 0 e 1. Più tale valore si avvicina ad 1 e più l’elemento risulterà visibile.

**I tag headers**

Immagine che contiene testo, schermata, software, Software multimediale

Descrizione generata automaticamente

I tag headers sono di tipo blocco e vengono utilizzati per specificare dei titoli alle varie sezioni di una pagina. Si differenziano tra loro per la grandezza del font che è maggiore nel tag h1, per poi decrescere fino ad arrivare al tag h6 che è definito con il font più piccolo. In una pagina hatml ci dovrebbe essere un solo elemento h1, ad indicarne il titolo principale, ma ci possono essere più headers appartenenti agli altri cinque tipi.

**Le liste**

Di seguito un esempio di lista non ordinata costituita da quattro item:

Immagine che contiene testo, schermata, schermo, software

Descrizione generata automaticamente

In una lista non ordinata, di default gli item vengono visualizzati con un pallino a fianco:

Immagine che contiene testo, Carattere, schermata, Elementi grafici

Descrizione generata automaticamente

Possiamo modificare questo aspetto agendo sulla proprietà ‘list-style-type’ del tag <li>:

Immagine che contiene testo, Carattere, linea, schermata

Descrizione generata automaticamente

Sono stati evidenziati in verde i principali possibili valori assegnabili alla proprietà in questione (‘disc’ è quello di default).

Per quanto riguarda le liste ordinate, l’impostazione di default prevede la visualizzazione dei singoli item accompagnati da un numero intero crescente:

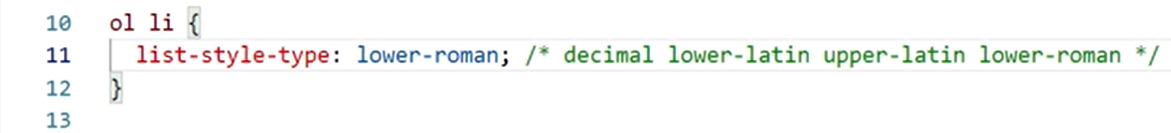
Immagine che contiene testo, schermata, software, schermo

Descrizione generata automaticamente

Immagine che contiene testo, Carattere, Elementi grafici, Turchese

Descrizione generata automaticamente

Come visto in precedenza, possiamo modificare questo comportamento attraverso la proprietà ‘list-style-type’ che per le liste ordinate prevede un altro set di valori possibili (decimal è quello di default):



È possibile far iniziare la lista ordinata da un numero qualsiasi utilizzando l’attributo ‘start’ dentro il tag <ol>:

Immagine che contiene testo, schermata, software, schermo

Descrizione generata automaticamente

Immagine che contiene testo, Carattere, Elementi grafici, schermata

Descrizione generata automaticamente

Un’ulteriore personalizzazione la si ottiene attraverso l’attributo ‘value’ applicato all’interno del tag <li>. Questo attributo ci permette di modificare in modo puntuale il valore numerico da visualizzare accanto al singolo item:

Immagine che contiene testo, schermata, software, schermo

Descrizione generata automaticamente

Immagine che contiene testo, Carattere, schermata, Elementi grafici

Descrizione generata automaticamente

Qualora volessimo inserire un’immagine al posto dei numeri che affiancano gli elementi della lista, possiamo avvalerci della proprietà css ‘list-style-image’:

Immagine che contiene testo, schermata, software, Software multimediale

Descrizione generata automaticamente

Immagine che contiene testo, schermata, Carattere, logo

Descrizione generata automaticamente

Un’altra tipologia di liste che si può utilizzare è la lista di definizione, utile soprattutto quando si vogliono descrivere dei termini specifici con del testo di lunghezza non trascurabile:

Immagine che contiene testo, schermata, software, Software multimediale

Descrizione generata automaticamente

La lista di distribuzione è costituita dal tag <dl>, al cui interno troviamo altri due tag: <dt> specifica il termine da descrivere mentre <dd> contiene la sua descrizione. Nell’esempio sopra indicato la lista contiene la descrizione di due piatti culinari:

Immagine che contiene testo, Carattere, schermata, aqua

Descrizione generata automaticamente

**Tag per la formattazione del testo, font**

Di seguito alcuni tag utili per la formattazione del testo con l’indicazione del loro utilizzo:

Immagine che contiene testo, schermata, software, Software multimediale

Descrizione generata automaticamente

Immagine che contiene testo, schermata, software, Software multimediale

Descrizione generata automaticamente

Immagine che contiene testo, schermata, software, schermo

Descrizione generata automaticamente

Immagine che contiene testo, elettronica, schermata, schermo

Descrizione generata automaticamente

Immagine che contiene testo, schermata, software

Descrizione generata automaticamente

Immagine che contiene testo, schermata, software, Software multimediale

Descrizione generata automaticamente

Immagine che contiene testo, schermata, software, schermo

Descrizione generata automaticamente

Per quanto riguarda la formattazione del testo, una proprietà css molto importante è la ‘font-family’ che determina la categoria di font utilizzata. È consigliabile specificare più valori per questa proprietà, questo perché non sappiamo a priori quale siano le famiglie di font effettivamente disponibili sul dispositivo che stiamo utilizzando per visualizzare la pagina web:

Immagine che contiene testo, schermata, software, Software multimediale

Descrizione generata automaticamente

Generalmente la famiglia di font da utilizzare viene impostata per l’intera pagina quindi la proprietà ‘font-family’ viene specificata per l’elemento body, come nell’ultimo esempio. Questo accade perché tale proprietà viene ereditata dagli elementi figli. Ciò non toglie la possibilità di utilizzare un font diverso per il testo contenuto in specifici elementi.

La grandezza del font è gestibile attraverso la proprietà css ‘font-size’, che viene anche questa ereditata. Pertanto, se definita nell’elemento body, tutti i tag figli utilizzeranno la stessa grandezza del font.

Generalmente la proprietà ‘font-size’ viene espressa attraverso un valore numerico, ma c’è anche la possibilità di utilizzare dei valori testuali: ‘xx-small’, ‘x-small’. ‘medium’, ‘large’, ‘x-large’, ‘xx-large’, ‘xxx-large’.

Quest’ultima tecnica, comunque, non è molto usata.

La proprietà ‘font-style’, invece, ci permette di impostare del testo in corsivo utilizzando il valore ‘italic’.

**Affiancare elementi con display: inline-block**

Le tecniche maggiormente utilizzate per la selezione degli elementi html nel css fanno uso degli attributi ‘id’ e ‘class’.

L’attributo ‘id’ identifica univocamente un elemento all’interno di una pagina html. Pertanto, non si dovrebbe assegnare lo stesso identificativo a più elementi. Ecco un esempio di utilizzo dell’attributo ‘id’;

Immagine che contiene testo, schermata, software, Software multimediale

Descrizione generata automaticamente

L’esempio sopra riportato mostra la tecnica di affiancamento di elementi che utilizza la proprietà ‘display’ con valore ‘inline-block’. Nel caso specifico, abbiamo impostato una larghezza della card pari al 25% dell’elemento genitore, in questo caso il body. Questo ci farebbe pensare che, definendo quattro card nel file html, dovremmo poterle visualizzare tutte su una riga, mentre il risultato effettivamente ottenuto è il seguente:

Immagine che contiene testo, schermata, linea, Carattere

Descrizione generata automaticamente

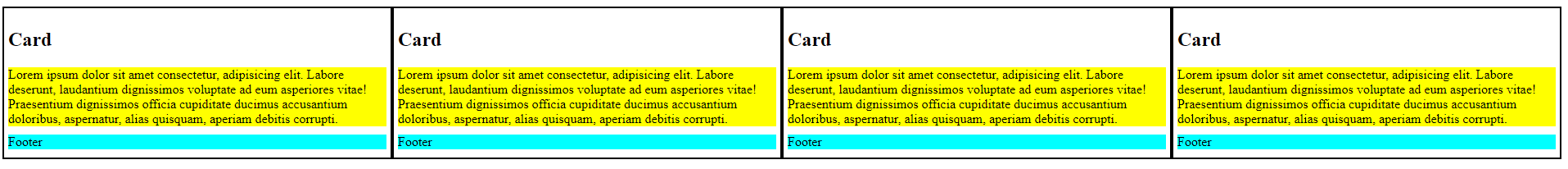
Come possiamo notare, la quarta card viene inserita sulla seconda riga. Questo accade a causa dello spazio presente tra le card.

Per modificare questo comportamento basta impostare una font-size nulla all’elemento contenitore delle card, in questo caso il body. Poiché la proprietà font-size viene ereditata, per continuare a vedere il testo inserito all’interno delle card dobbiamo specificare una font-size diversa da 0 per gli elementi div:

Immagine che contiene testo, schermata, software, Software multimediale

Descrizione generata automaticamente

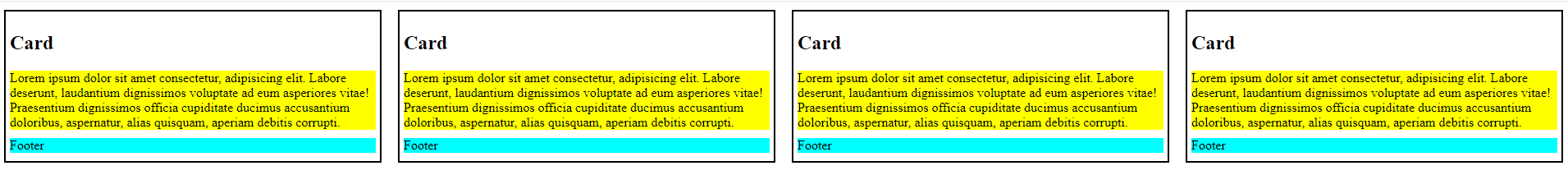
Con l’aggiunta di queste istruzioni otteniamo il risultato desiderato:



Qualora volessimo distanziare le quattro card, dovremmo impostare un margine per l’elemento ‘card-container’. Il problema è che, poiché il margine non è contenuto nel box-sizing, torneremo ad avere il problema precedente, ossia con una card visualizzata sulla seconda riga. Per evitare questo, dobbiamo sottrarre il margine che impostiamo alla larghezza della card. A tal proposito, risulta molto utile la funzione css ‘calc’:

Immagine che contiene testo, schermata, software, Software multimediale

Descrizione generata automaticamente



L’esempio sopra specificato in realtà è formalmente scorretto dal punto di vista html, poiché utilizziamo lo stesso identificativo per diversi elementi. Possiamo risolvere questo problema utilizzando l’attributo ‘class’ al posto di ‘id’:

Immagine che contiene testo, schermata, software

Descrizione generata automaticamente

**Proprietà per la formattazione del testo**

La proprietà ‘font-weight’ permette di gestire il peso del font. Può essere valorizzata utilizzando parole chiave, come ad esempio ‘bold’ per il grassetto. In alternativa possono essere utilizzati valori interi compresi tra 100 e 900, dove 900 corrisponde a ‘bold’.

La proprietà ‘text-decoration’ gestisce la sottolineatura del testo. Il valore più utilizzato per questa proprietà è ‘underline’ che imposta la sottolineatura classica. Specificando il valore ‘none’, invece, la sottolineatura viene disattivata. È possibile modificare lo stile o il colore della sottolineatura attraverso le proprietà ‘text-decoration-style’ e text-decoration-color’.

La proprietà ‘text-shadow’ consente di impostare un’ombra al testo. Per questa proprietà vanno specificati quattro valori: il primo indica lo scostamento orizzontale, il secondo quello verticale, il terzo indica la sfocatura, mentre il quarto indica il colore da assegnare all’ombra.

Per quanto riguarda il primo valore, se è positivo lo scostamento sarà verso destra, altrimenti sarà verso sinistra. Il secondo valore funziona in modo analogo: se è positivo, lo scostamento sarà verso il basso, altrimenti sarà verso l’alto. Per quanto riguarda la sfocatura, più il valore è alto è più l’ombra sarà sfocata. Generalmente i primi tre parametri vengono espressi in pixel.

La proprietà ‘text-shadow’ può essere anche multipla, ossia può specificare più quadruple contemporaneamente separate dalla virgola. Questa tecnica ci permette di applicare l’ombra in varie direzioni.

La proprietà ‘text-transform’ ci permette di applicare alcune trasformazioni ad una stringa di testo. I valori possibili sono ‘uppercase’, ‘lowercase’, ‘capitalize’ e ‘none’. I primi due valori sono intuitivi, il terzo imposta la prima lettera di ogni parola in maiuscolo mentre l’ultimo viene utilizzato per annullare tale proprietà.

La proprietà ‘font-variant’ viene spesso utilizzata per ottenere l’effetto ‘maiuscoletto’, ossia impostare tutte le lettere di una stringa in maiuscolo ma con la dimensione del font delle lettere minuscole. Questo effetto si ottiene utilizzando il valore ‘small-caps’.

Un’altra proprietà importante è ‘text-align’ che serve per allineare del testo. I valori possibili sono: ‘left’ (default), ‘right’, ‘center’ e ‘justify’. Quest’ultimo è poco utilizzato nel web perché crea degli spazi interni alla stringa non sempre piacevoli.

La proprietà ‘text-align’ può anche essere utilizzata per allineare orizzontalmente specifici elementi, in particolare per centrarli. Tali elementi devono essere più piccoli del loro contenitore e, inoltre, devono essere anche di tipo ‘inline-block’ o ‘inline’. La proprietà ‘text-align’ va applicata al contenitore dell’elemento da allineare:

Immagine che contiene testo, schermata, software, schermo

Descrizione generata automaticamente

Immagine che contiene testo, schermata, software, Software multimediale

Descrizione generata automaticamente

Immagine che contiene Rettangolo, verde, bandiera

Descrizione generata automaticamente

La proprietà ‘line-height’ gestisce l’interlinea e viene utilizzata per aumentare o diminuire lo spazio presente tra righe di testo. Tale proprietà viene valorizzata con numeri reali puri.

La proprietà ‘white-space’ permette di gestire gli spazi all’interno di una stringa di testo. Utilizzando il valore ‘nowrap’ forziamo il testo a non andare a capo. Questa impostazione potrebbe generare problemi di overflow, soprattutto quando il contenitore del testo ha una dimensione fissa.

La proprietà ‘overflow’ viene in nostro soccorso per gestire, appunto, l’overflow del testo all’interno di un contenitore. Utilizzando il valore ‘hidden’ nascondiamo il testo che eccede la dimensione del contenitore. Generalmente questo valore viene usato contemporaneamente alla proprietà ‘text-overflow: ellipsis’ grazie alla quale vengono visualizzati i tre puntini di sospensione alla fine del testo, in modo da far capire la sua continuazione. Con il valore ‘scroll’ per la proprietà ‘overflow’, invece, forziamo il browser a visualizzare una barra di scorrimento, ma questo è un effetto che non dovrebbe mai essere utilizzato perché non è molto piacevole.

L’overflow può essere gestito anche in una solo direzione, orizzontale o verticale, attraverso le proprietà ‘overflow-x’ e ‘overflow-y’.

Tornando alla proprietà ‘white-space’, un altro valore possibile è ‘pre’, con il quale vengono mantenuti tutti gli spazi impostati nel file html. In sostanza abbiamo lo stesso comportamento che si avrebbe con l’utilizzo del tag <pre>. Con il valore ‘pre-line, gli spazi vengono compattati, il testo non va mai in overflow ma gli a capo specificati nel file html vengono rispettati. Con il valore ‘pre-wrap’, gli spazi e gli a capo vengono rispettati, il testo non va mai in overflow e quindi si potrebbero avere ulteriori a capo per evitare quest’ultimo effetto.

La proprietà ‘overflow-wrap’ viene utilizzata per gestire parole molto lunghe all’interno di un testo che potrebbero eccedere il contenitore. Attraverso il valore ‘break-word’, tali parole vengono spezzate per evitare l’overflow. Utilizzando contemporaneamente la proprietà ‘hyphens: auto’ otteniamo la sillabazione di queste parole, con l’inserimento del trattino per andare a capo. Qualora si volesse personalizzare il carattere utilizzato per andare a capo, si può usare la proprietà ‘hyphenate-character’ per specificare il nuovo simbolo.

La proprietà ‘font-stretch’ ci permette di modificare lo spazio tra i caratteri:

Immagine che contiene testo, schermata, Carattere, numero

Descrizione generata automaticamente

La proprietà ‘letter-spacing’ è molto simile alla precedente ma consente di specificare valori in pixel.

Esiste anche la proprietà ‘word-spacing’ per regolare lo spazio tra parole.

La proprietà ‘font’ viene detta ‘shorthand property’ perché consente di specificare in una sola riga di codice più proprietà css riguardanti il font. Alcune proprietà sono obbligatorie, altre facoltative:

Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente

**Collegare il codice css al file HTML, conflitti css**

Ci sono diversi modi per collegare del codice css ad una pagina html. La modalità più utilizzata è quella vista fino ad ora, ossia utilizzando un file css esterno a cui la pagina html fa riferimento attraverso il tag <link> inserito all’interno del tag <head>.

Esistono altri due modi per effettuare questa connessione css-html. Il primo richiede l’utilizzo del tag <style> all’interno del tag <head>:

Immagine che contiene testo, schermata, software, schermo

Descrizione generata automaticamente

Il secondo metodo viene detto ‘inline’ perché il codice css viene specificato all’interno del tag html a cui si vuole assegnare quel determinato stile. Questo è possibile attraverso l’utilizzo dell’attributo style:



Il metodo preferibile, comunque, rimane quello del file css esterno.

Si ha un conflitto tra regole css quando le stesse, selezionando il medesimo o i medesimi elementi, valorizzano in maniera diversa una determinata proprietà.

In questo caso, per capire quale sia la regola che verrà effettivamente applicata, va considerato il criterio della specificità. In particolare, valgono le seguenti considerazioni:

* una regola css che utilizza come selettore l’attributo id risulta essere più specifica di una regola che, invece, utilizza l’attributo class;
* una regola che utilizza come selettore l’attributo class risulta essere più specifica di una regola che utilizza il nome del tag;
* le regole css inline sono più specifiche di qualsiasi altra regola css.

A parità di specificità, conta l’ordine di scrittura delle regole css. In questo caso, viene applicata sempre la regola che viene scritta per ultima, poiché il codice css viene interpretato dall’alto verso il basso.

**Selettori css**

Dato il seguente codice html:

Immagine che contiene testo, schermata, software, Software multimediale

Descrizione generata automaticamente

Di seguito alcuni esempi di selettori di parentela o gerarchici:

Immagine che contiene testo, schermata, software, Carattere

Descrizione generata automaticamente

La regola css alle righe 2-4 seleziona tutti gli elementi <p> contenuti all’interno di un elemento <section>. Facendo riferimento al codice html sopra riportato, tale regola viene applicata sia al paragrafo a riga 11 che a quello a riga 13.

La regola css alle righe 6.8, invece, viene applicata solo al paragrafo definito a riga 11, poiché seleziona tutti gli elementi <p> che sono figli diretti di un elemento <section>. Pertanto, i discendenti non diretti vengono esclusi dalla selezione. Questo è il motivo per cui la regola non viene applicata anche al paragrafo di riga 13, in quanto tale elemento si trova all’interno di un <div> e, di conseguenza, non è un discendente diretto dell’elemento <section>.

Per selezionare elementi che si trovano allo stesso livello nel codice html si utilizzano i cosiddetti selettori di fratellanza. L’esempio sotto riportato mostra alcuni tag <p>, <span> e <div> fratelli tra loro:

Immagine che contiene testo, schermata, software, Software multimediale

Descrizione generata automaticamente

In una situazione di questo tipo possiamo distingue fratelli destri e sinistri. I fratelli sinistri di un dato elemento sono tutti i tag che si trovano allo stesso livello dell’elemento in questione ma al di sopra di esso. Nell’esempio sopra riportato, il tag <p> a riga 10 è un fratello sinistro di tutti gli altri tag.

Detto questo, è facile intuire quali siano i fratelli destri di un data elemento. Si parla, invece, di fratelli adiacenti quando gli elementi sono attaccati tra loro. Tornando al codice html di esempio, il tag <div> a riga 13 e il tag <p> a riga 14 sono fratelli adiacenti.

Di seguito un file css che fa uso di selettori di fratellanza. La selezione riguarda sempre i fratelli desti e mai quelli sinistri:

Immagine che contiene testo, schermata, software, Carattere

Descrizione generata automaticamente

La regola alle righe 2-4, che fa uso del simbolo ‘+‘, seleziona un elemento <span> che è fratello adiacente di un elemento <p>. Nel codice html di esempio, tale regola viene applicata solo allo <span> di riga 11.

La regola css alle righe 6-8, che fa uso del simbolo della tilde, seleziona tutti gli elementi <p> che sono fratelli destri di un elemento <span>. Questa regola viene applicata ai paragrafi di riga 14 e 15.

Analogamente, la regola css alle righe 11-13 viene applicata ai <div> presenti alle righe 13 e 15 del codice html.

Quando si desidera applicare lo stesso stile a più selettori, si può utilizzare il simbolo ‘,’ per separare i selettori e poi specificare le regole css una sola volta, con evidente risparmio di codice. Di seguito un esempio:

Immagine che contiene testo, schermata, Carattere, software

Descrizione generata automaticamente

Il codice sopra può essere ulteriormente semplificato con l’utilizzo della pseudo-classe ‘:is’:

Immagine che contiene testo, schermata, Carattere, software

Descrizione generata automaticamente

Il selettore a riga due seleziona tutti gli elementi <p> all’interno di un <div>, tutti gli elementi <span> che si trovano all’interno di un <div> e tutti gli elementi <ul> che si trovano dentro un <div, indipendentemente che siano figli diretti o indiretti.

È possibile realizzare anche selettori di attributo. Di seguito alcuni esempi:

Immagine che contiene testo, schermata, software, schermo

Descrizione generata automaticamente

Di seguito alcuni esempi di selettori che fanno uso delle pseudo-classi ‘nth-child’, ‘first-child’ e ‘last-child’, utili per selezionare i discendenti diretti di un determinato elemento:

Immagine che contiene testo, schermata, software, Software multimediale

Descrizione generata automaticamente

Nell’esempio sopra riportato i selettori vengono utilizzati per selezionare i figli di tipo <li> all’interno di un elemento <ul>, quindi di una lista non ordinata.

Per selezionare i figli pari e/o dispari si può utilizzare il seguente codice che fa uso della variabile ‘n’ e delle parole chiave ‘even’ e ‘odd’:

Immagine che contiene testo, schermata, software, Carattere

Descrizione generata automaticamente

Sono molto usate anche le pseudo-classi ‘nth-of-type’, ‘first-of-type’ e ‘last-of-type’, simili alle precedenti ma che risultano utili soprattutto quando i figli di un determinato elemento sono di tipo diverso. Queste pseudo-classi, infatti, considerano solo i figli di un determinato tipo, escludendo tutti gli altri:

Immagine che contiene testo, schermata, software

Descrizione generata automaticamente

Immagine che contiene testo, schermata, software, Carattere

Descrizione generata automaticamente

Il selettore a riga 1 seleziona il terzo figlio di tipo paragrafo contenuto nella sezione. Facendo riferimento al codice html sopra riportato, l’elemento selezionato è in realtà il quinto figlio del tag <section>, ma è il terzo di tipo <p>. Gli altri selettori sono analoghi e il loro significato è molto intuitivo. Anche in questo caso è possibile utilizzare la variabile ‘n’ e le parole chiave ‘even’ e ‘odd’, il loro funzionamento è identico a quello visto in precedenza.

Altre pseudo-classi utili alla costruzione di selettori sono riportate nel seguente esempio:

Immagine che contiene testo, schermata, software

Descrizione generata automaticamente

Immagine che contiene testo, schermata, software, Carattere

Descrizione generata automaticamente

Il selettore a riga 1 seleziona tutti gli elementi <p> racchiusi in una <section> ad eccezione del primo paragrafo. Sostanzialmente la pseudo-classe ‘not’ ci permette di effettuare delle esclusioni.

Il selettore a riga 5, che fa uso della pseudo-classe ‘has’, seleziona tutti gli elementi <section> che abbiano al loro interno almeno un paragrafo.

Il selettore a riga 9, facente uso della pseudo-classe ‘only-of-type’, seleziona un elemento <span> contenuto in una <section> solo se si tratta dell’unico elemento di quel tipo, in questo caso dell’unico <span>. Facendo riferimento al codice html di esempio, questo selettore seleziona l’elemento <span> a riga 21, poiché è l’unico <span> figlio dell’elemento <section> definito alle righe 19-23. L’elemento <section> definito alle righe 10-18 non viene preso in considerazione perché contiene al suo interno più elementi <span>.

**Gestione del codice css non supportato dai browser**

Quando si utilizzano delle proprietà o funzioni css che non sono supportate in maniera completa da tutti i browser, è conveniente utilizzare il cosiddetto codice di backup. Tale tecnica consiste nell’impostare due volte la proprietà css coinvolta: nella prima istruzione dobbiamo inserire il codice supportato da tutti i browser, che sarà il nostro backup, mentre nella seconda inseriamo il codice con percentuale di supporto inferiore. Questo ci assicura la corretta valorizzazione della proprietà css, almeno attraverso il codice di backup, poiché l’interpretazione avviene dall’alto verso il basso. Infatti, se il browser utilizzato supporta solo il codice di backup, verrà applicata la prima istruzione e non la seconda. Se invece supporta entrambi i codici, verrà applicata l’ultima istruzione inserita.

Qualora non fosse possibile applicare la tecnica del codice di backup, oppure l’effetto ottenuto non fosse quello desiderato, una valida alternativa è costituita dall’istruzione ‘@supports’, che ci permette di specificare delle regole css da applicare quando una determinata proprietà css è supportata o non supportata. Di seguito un codice di esempio che mostra la sintassi da utilizzare:

Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente

**Proprietà border-radius e cursor**

La proprietà ‘border-radius’ viene utilizzata per arrotondare i bordi di un elemento. Di seguito del codice di esempio che ne mostra l’utilizzo:

Immagine che contiene testo, schermata, software, Software multimediale

Descrizione generata automaticamente

Immagine che contiene testo, schermata, software, schermo

Descrizione generata automaticamente

Nel file html abbiamo definito quattro <div> con altrettante classi.

La regola css a riga 9 imposta un ‘border-radius’ pari a 30 pixel per ogni angolo dell’elemento <div>. Il valore 30 pixel si riferisce sia al raggio orizzontale che a quello verticale.

La regola a riga 13, invece, specifica la proprietà in questione per ogni singolo angolo dell’elemento <div>, a partire da quello in alto a sinistra e procedendo in senso orario (top-left, top-right, bottom-right, bottom-left).

La regola a riga 17 è simile alla precedente ma specifica valori diversi per il raggio orizzontale e verticale dei quattro angoli. I valori alla sinistra del simbolo ‘/’ si riferiscono alla dimensione orizzontale, mentre quelli alla destra fanno riferimento alla dimensione verticale.

È possibile specificare valori diversi per i singoli angoli anche attraverso le seguenti proprietà:

* border-top-left-radius;
* border-top-right-radius;
* border-bottom-left-radius;
* border-bottom-right-radius.

Le regole css alle righe 21-22 mostrano come realizzare un cerchio perfetto, ossia disegnando un quadrato per poi impostare la proprietà ‘border-radius’ al 50%.

Se, invece, vogliamo ottenere un ovale, basta utilizzare la tecnica appena descritta ma a partire da un rettangolo e non da un quadrato.

A partire da un rettangolo, inoltre, è possibile realizzare un’altra figura molto utilizzata detta ‘obrund’ che ha una forma simile alla seguente:

Immagine che contiene arancione, design

Descrizione generata automaticamente

Tale figura si ottiene impostando un ‘border-radius’ esageratamente grande, come nel seguente codice:

Immagine che contiene testo, schermata, Carattere, numero

Descrizione generata automaticamente

La proprietà ‘cursor’ viene utilizzata per personalizzare il puntatore del mouse. Se valorizzata a ‘pointer’, prende le sembianze di una piccola mano, effetto molto utilizzato quando un elemento dell’interfaccia grafica è cliccabile. Ecco un esempio di codice che utilizza tale proprietà insieme alla pseudo-classe ‘hover’:

Immagine che contiene testo, schermata, software, Software multimediale

Descrizione generata automaticamente

La pseudo-classe ‘hover’ si attiva quando il mouse passa sopra all’elemento a cui è applicata. Nell’esempio indicato, l’elemento in questione è uno <span> e quando il mouse ci finisce sopra vengono attivate le regole definite alle righe 37 e 38.

**Proprietà box-shadow**

La proprietà ‘box-shadow’ è molto utilizzata perché permette di realizzare diversi effetti grafici apprezzabili, in particolare il cosiddetto effetto ombra:

Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente

Tale proprietà costruisce un elemento identico a quello a cui viene applicata ma dietro di esso.

Il modo più semplice di utilizzare ‘box-shadow’ è specificando due valori che rappresentano rispettivamente lo spostamento orizzontale e verticale dell’elemento posteriore, detti ‘offset-x’ e ‘offset-y’. Per valori positivi degli offset, l’elemento posteriore si scosta da quello anteriore rispettivamente verso destra e verso il basso, mentre per valori negativi lo spostamento è verso sinistra e verso l’alto.

Qualora venga specificato un terzo valore, quest’ultimo viene interpretato come ‘blu-radius’, detto anche raggio di sfocatura. Maggiore sarà questo valore, più l’elemento posteriore sarà sfocato.

L’aggiunta di un quarto valore permette di specificare lo ‘spread-radius’, una quantità che ingrandisce l’elemento posteriore su tutti i lati.

Di default l’elemento posteriore ha un colore nero ma questo può essere personalizzato specificando un parametro opzionale, detto appunto ‘color’.

La parola chiave opzionale ‘inset’ permette di visualizzare il box shadow all’interno invece che all’esterno. Questo permette di realizzare degli effetti grafici particolari.

Di seguito un codice di esempio con il relativo layout:

Immagine che contiene testo, schermata, software, Software multimediale

Descrizione generata automaticamente

Immagine che contiene testo, schermata, software, Software multimediale

Descrizione generata automaticamente

Immagine che contiene Rettangolo, schermata, linea, quadrato

Descrizione generata automaticamente

La proprietà ‘box-shadow’ è utilizzabile anche in maniera multipla, specificando la lista degli shadows separati dalla virgola, realizzando così degli effetti ancora più particolari.

**Sfumature nei colori**

La funzione ‘linear-gradient’ permette di applicare dei colori con delle variazioni di sfumatura di tipo lineare:

Immagine che contiene testo, Software multimediale, software, Software per la grafica

Descrizione generata automaticamente

Nell’esempio sopra abbiamo applicato la proprietà ‘background-image’ ad un elemento <div> utilizzando la funzione ‘linear-gradient’ specificando due colori come valori di input. Di seguito l’effetto ottenuto:



Come possiamo notare, il colore dell’elemento presenta delle sfumature. Questo perché cambia in modo lineare dall’alto verso il basso. Il valore di partenza è quello specificato come primo parametro di input, mentre il secondo rappresenta il colore di arrivo.

Qualora volessimo cambiare la direzione di variazione del colore, basta specificare una parola chiave come primo parametro della funzione, in particolare:

* ‘to top’, se vogliamo che la variazione sia dal basso verso l’alto;
* ‘to right’ indica la variazione da sinistra verso destra;
* ‘to left’ indica la variazione da destra verso sinistra.

Il valore di default di questo parametro è ‘to bottom’.

Si possono creare delle direzioni in diagonale combinando le parole chiave sopra menzionate:

* ‘to top left’ significa che il gradiente varierà a partire dall’angolo in basso a destra verso l’angolo in alto a sinistra;
* ‘to bottom right’ implementa il funzionamento contrario al precedente;
* ‘to top right’ significa che il gradiente varierà a partire dall’angolo in basso a sinistra verso l’angolo in alto a destra;
* ‘to bottom left’ implementa il funzionamento contrario al precedente.

È possibile creare direzioni più fantasiose specificando il primo parametro in gradi, con valori compresi tra 0 e 360. Il valore 0 corrisponde alla direzione ‘to top’, 90 alla direzione ‘to right’, 180 a ‘to bottom’, 270 alla direzione ‘to left’ mentre 360 è identica al valore 0. Utilizzando valori intermedi a quelli elencati si possono ottenere delle sfumature particolari.

Inoltre, la funzione ‘linear-gradient’ è utilizzabile anche specificando più di due colori e questo consente di realizzare degli effetti grafici molto gradevoli:

Immagine che contiene testo, Software multimediale, software, schermata

Descrizione generata automaticamente



Nell’esempio sopra riportato abbiamo utilizzato una direzione di 45 gradi, che corrisponde a ‘to top right’, e tre colori: il primo è quello di partenza, il secondo è l’intermedio mentre il terzo è quello di arrivo.

La funzione ‘linear-gradient’ può essere ulteriormente complicata utilizzando i cosiddetti ‘color stop’, che possono essere indicati in percentuale oppure in pixel:

Immagine che contiene testo, Software multimediale, Carattere, software

Descrizione generata automaticamente



Nell’esempio sopra riportato la direzione utilizzata è quella di default, ossia dall’alto verso il basso. Il primo ‘color-stop’ è pari al 30%, questo significa che il primo colore verrà raggiunto in corrispondenza del 30% dell’altezza dell’elemento, partendo dall’alto. Il secondo ‘color-stop’ è pari a 60%, pertanto a partire dal colore #faebd7 presente in corrispondenza del 30%, si raggiungerà il colore #f1ac52 in corrispondenza del 60% dell’altezza dell’elemento, sempre a partire dall’alto. Analogo discorso per il colore orangered, che sarà effettivamente raggiunto al 100% dell’altezza dell’elemento. Poiché la direzione è dall’alto verso il basso, questo accadrà in corrispondenza del lato inferiore del <div>. Di seguito un esempio con i ‘color-stop’ indicati in pixel:

Immagine che contiene testo, Software multimediale, Carattere, software

Descrizione generata automaticamente



È possibile anche specificare un terzo ‘color-stop’, qualora si volesse un colore a tinta unita per una porzione dell’elemento:

Immagine che contiene Software multimediale, Software per la grafica, software, schermata

Descrizione generata automaticamente



Nell’esempio di cui sopra, il colore orangered sarà raggiunto ad un’altezza di 120 pixel partendo dall’alto. Poiché l’elemento è alto 200 pixel, gli ultimi 80 pixel verso il basso avranno una tinta unita di colore orangered, come visibile dall’ultima figura.

Qualora si volesse ottenere la tinta unita per il colore intermedio invece che nell’ultimo, si potrebbe utilizzare del codice simile al seguente:

Immagine che contiene Software multimediale, testo, schermata

Descrizione generata automaticamente

Come è facilmente intuibile, da 80 pixel a 120 pixel il colore sarà sempre #f1ac52, per poi variare fino a raggiungere l’orangered.

La funzione ‘radial-gradient’ ha lo stesso scopo della precedente ma in questo caso la variazione del colore non avviene lungo una linea ma lungo una circonferenza o un’ellisse, comportamento che può essere stabilito attraverso le parole chiave ‘circle’ e ‘ellipse’.

Il gradiente di tipo circolare funziona nel modo seguente. Prima di tutto, bisogna stabilire se la circonferenza deve lambire, rispetto al suo centro:

* il lato più vicino;
* il lato più lontano;
* l’angolo più vicino;
* l’angolo più lontano.

La scelta si effettua utilizzando le seguenti parole chiave:

* closest-side;
* farthest-side;
* closest-corner;
* farthest-corner.

Alternativamente si può specificare il raggio della circonferenza in pixel.

Se non diversamente specificato, il comportamento di default prevede che il centro del cerchio corrisponda al centro dell’elemento a cui stiamo applicando la proprietà che utilizza la funzione ‘radial-gradient’.

Una volta definita la circonferenza, la funzione richiede di specificare la lista dei colori, con eventuali punti di stop come visto in precedenza, su cui articolare la variazione della tinta.

In base alle impostazioni sopra descritte, il browser disegnerà una circonferenza all’interno della quale avverrà la variazione dei colori scelti, a partire dal centro del cerchio. Qualora la circonferenza non includesse tutto l’elemento, lo spazio esterno ad essa sarò riempito con il colore finale.

Di seguito alcuni esempi esplicativi:

Immagine che contiene testo, Software multimediale, software, schermata

Descrizione generata automaticamente

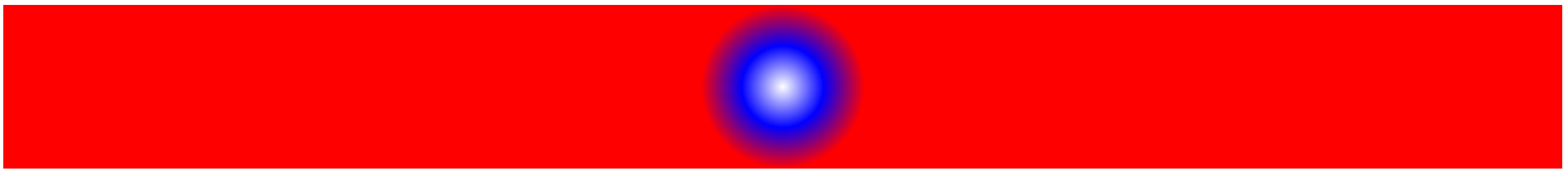


Immagine che contiene testo, Carattere, Software multimediale, schermata

Descrizione generata automaticamente

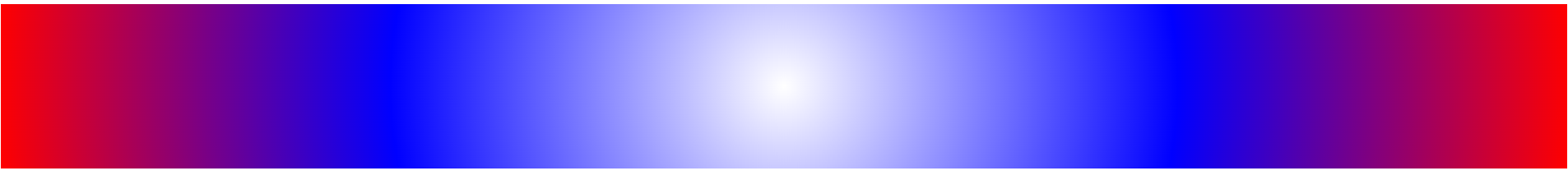


Immagine che contiene testo, Software multimediale, software, Software per la grafica

Descrizione generata automaticamente

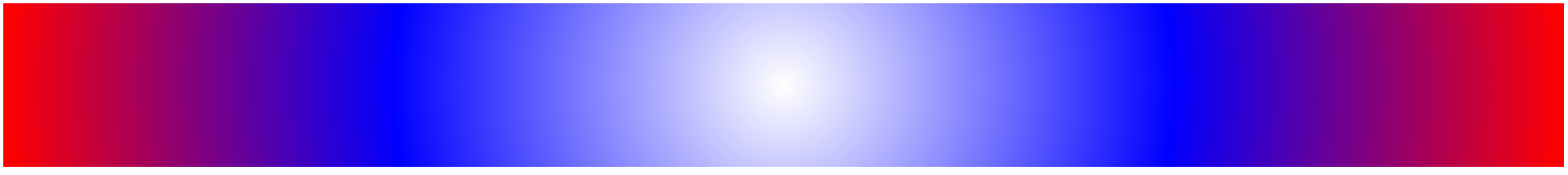


Immagine che contiene Software multimediale, software, Software per la grafica, testo

Descrizione generata automaticamente



Immagine che contiene Software multimediale, testo, schermata

Descrizione generata automaticamente



Immagine che contiene Software multimediale, schermata

Descrizione generata automaticamente



Negli ultimi due esempi è stato modificato anche il centro della circonferenza che, come detto in precedenza, di default corrisponde al centro dell’elemento coinvolto. Il primo valore (300 pixel) rappresenta la coordinata x, mentre il secondo (100 pixel) la coordinata y.

Il gradiente di tipo ellittico funziona alla stessa maniera ma utilizza la parola chiave ‘ellipse’ al posto di ‘circle’. Per il resto valgono le stesse considerazioni fatte in precedenza in merito al centro, ai lati o agli angoli che l’ellisse deve lambire. Ecco un esempio di codice che utilizza un gradiente ellittico con due colori:

Immagine che contiene testo, Software multimediale, software, schermata

Descrizione generata automaticamente

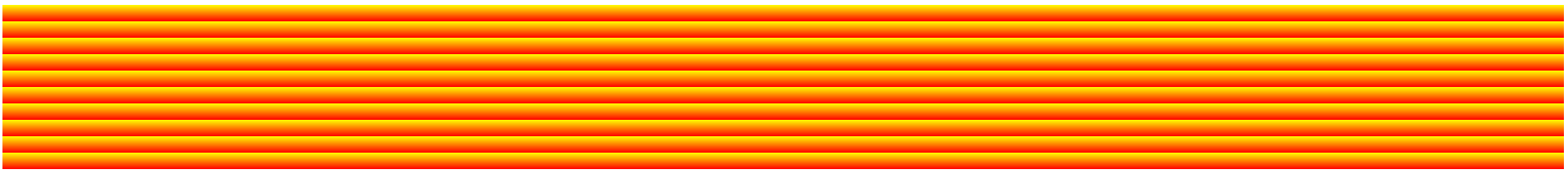


Per concludere, è utile sapere che si possono realizzare gradienti lineari e radiali che si ripetono:

Immagine che contiene testo, Carattere, Software multimediale, schermata

Descrizione generata automaticamente

Il codice sopra riportato definisce un gradiente lineare che utilizza due colori e si ripete ogni 20 pixel, ottenendo il seguente effetto grafico:

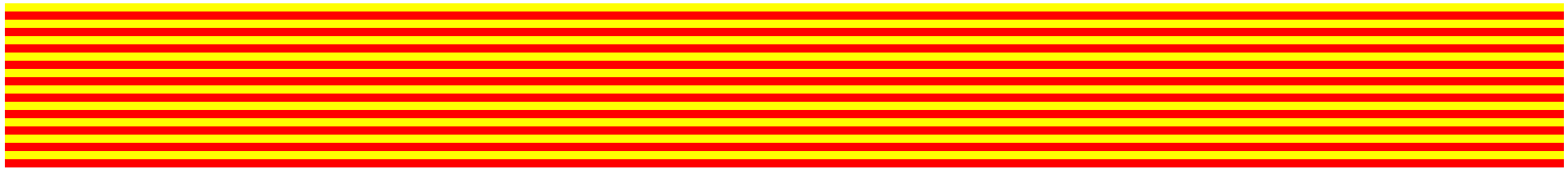


Qualora volessimo togliere le sfumature, potremmo utilizzare del codice simile al seguente:

Immagine che contiene Software multimediale, software, schermata

Descrizione generata automaticamente

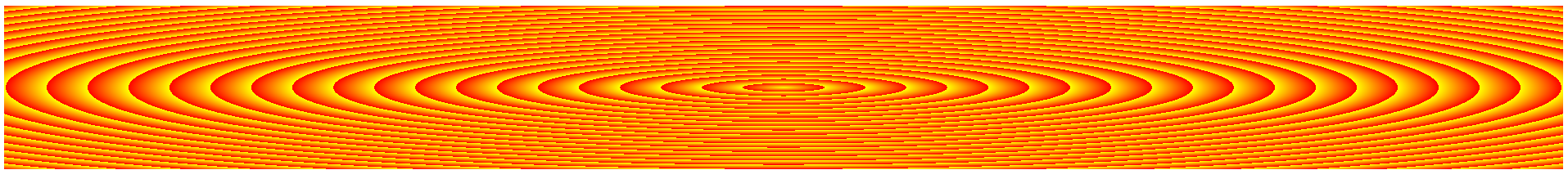
Per i primi 10 pixel abbiamo il colore giallo, per i secondi il colore rosso e così via, ottenendo quanto segue:



Analogo discorso vale per il gradiente radiale:

Immagine che contiene testo, Carattere, schermata

Descrizione generata automaticamente



Nel caso specifico, abbiamo ottenuto una serie di circonferenze concentriche.

**Unità di misura**

Fino ad ora abbiamo dimensionato gli elementi utilizzando i pixel come unità di misura oppure le percentuali.

Le unità di misura si dividono in due grandi categorie: assolute e relative.

Il pixel è un’unità di misura assoluta, mentre le percentuali sono relative in quanto fanno riferimento alla dimensione del genitore. Quest’ultime, però, hanno effetto solo se il genitore viene dimensionato in pixel.

Generalmente le unità di misura relative sono preferibili perché si adattano meglio al responsive.

Tra queste molto importanti sono quelle che si riferiscono al viewport, ossia alla dimensione corrente della pagina web. Tali unità di misura sono rispettivamente vw (viewport width) e vh (viewport height): la prima si riferisce alla larghezza della pagina, mentre la seconda alla sua altezza.

Se dimensioniamo un elemento con un’altezza pari a 1vh, stiamo dicendo che quell’elemento avrà un’altezza pari all’1% dell’altezza della pagina. Stesso discorso vale per la dimensione in larghezza.

Un’altra unità di misura relativa è ‘em’, utilizzata molto spesso per dimensionare il testo della pagina. Questo perché tale unità di misura fa riferimento alla ‘font-size’ dell’elemento genitore.

Una pratica molto usata consiste nel dimensionare in pixel la ‘font-size’ dell’elemento <body>, per poi specificare la ‘font-size’ degli altri componenti della pagina utilizzando ‘em’ come unità di misura.

Sostanzialmente, se impostiamo una ‘font-size’ a 1.2em stiamo dimensionando la grandezza dei caratteri di quell’elemento con un valore pari a 1.2 moltiplicato per la ‘font-size’ dell’elemento genitore.

Bisogna fare attenzione a questi fattori moltiplicativi, soprattutto negli elementi annidati, poiché si potrebbero avere degli effetti indesiderati.

Chiaramente l’unità di misura ‘em’ può essere utilizzata anche per le altre proprietà css, ma non sempre questa è una strategia vincente.

Per evitare il meccanismo dei fattori moltiplicativi negli elementi annidati, al posto di ‘em’ si può utilizzare l’unità di misura ‘rem’, che fa riferimento sempre alla ‘font-size’ del tag <html>. Questo potrebbe semplificare il dimensionamento del testo in molti casi.

**Affiancamento degli elementi con la proprietà float**

Ora torniamo ad affrontare il tema dell’affiancamento degli elementi. Al momento abbiamo visto solo la tecnica che fa uso della proprietà ‘display’ con valore ‘inline-block’. Analizziamone un’altra che utilizza la proprietà ‘float’:

Immagine che contiene testo, schermata, software

Descrizione generata automaticamente

Immagine che contiene testo, schermata, software, Carattere

Descrizione generata automaticamente

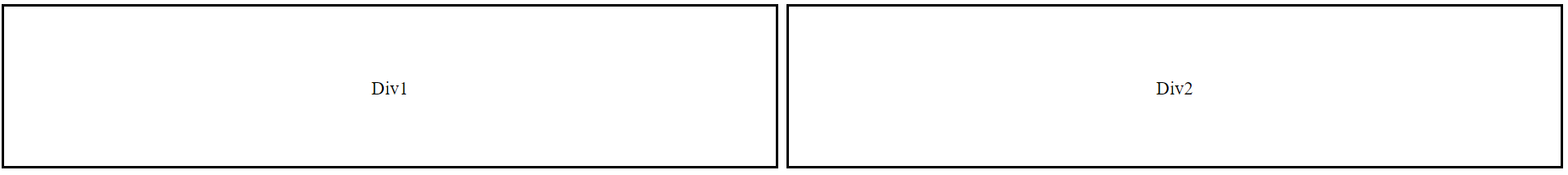
Immagine che contiene linea, diagramma, design

Descrizione generata automaticamente

La proprietà ‘float’ con valore ‘left’ affianca i due elementi <div> alla sinistra della pagina. Come possiamo vedere dal layout, i due <div> vengono dimensionati in larghezza in base al loro contenuto. Se vogliamo che occupino tutto lo spazio disponibile in larghezza, dobbiamo opportunamente specificarlo:

Immagine che contiene testo, schermata, software, Carattere

Descrizione generata automaticamente

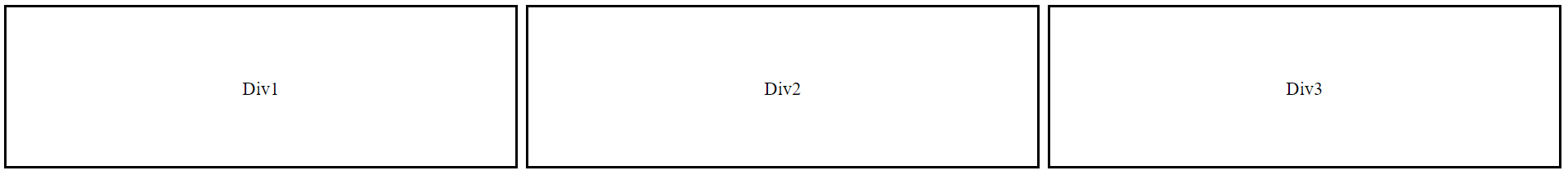


Come visto in precedenza, con la regola a riga 3 includiamo il bordo del <div> nella larghezza dell’elemento che poi dimensioniamo a riga 13, tenendo conto del margine impostato a riga 17. Nel caso specifico, abbiamo impostato un bordo destro di 10 pixel solo per il primo elemento. Questi 10 pixel li andiamo a sottrarre alla larghezza di entrambi gli elementi (diventano 5 pixel ciascuno).

Inserendo un terzo <div>, procediamo in modo analogo adattando la larghezza dei singoli elementi:

Immagine che contiene testo, schermata, software, Software multimediale

Descrizione generata automaticamente



Se andiamo ad analizzare con l’ispector la situazione del contenitore dei tre <div>, possiamo notare come quest’ultimo non abbia alcuna dimensione. Tale comportamento è una particolarità del float che rende invisibili gli elementi che la utilizzano rispetto a tutti gli altri. Pertanto, il <div> contenitore si comporta come se non avesse alcun contenuto.

Per risolvere questo problema ci sono diversi approcci.

Il primo consiste nell’utilizzare la proprietà ‘overflow’ nel contenitore dei tre <div>. Possiamo valorizzare tale proprietà con ‘auto’, ma questo valore genera una barra di scorrimento verticale nella pagina che non è un effetto gradito. L’alternativa è utilizzare il valore ‘hidden’ che, nel caso specifico, risolve il nostro problema, ma non è una soluzione sempre valida. Di seguito il codice nelle due situazioni:

Immagine che contiene testo, schermata, software, Software multimediale

Descrizione generata automaticamente

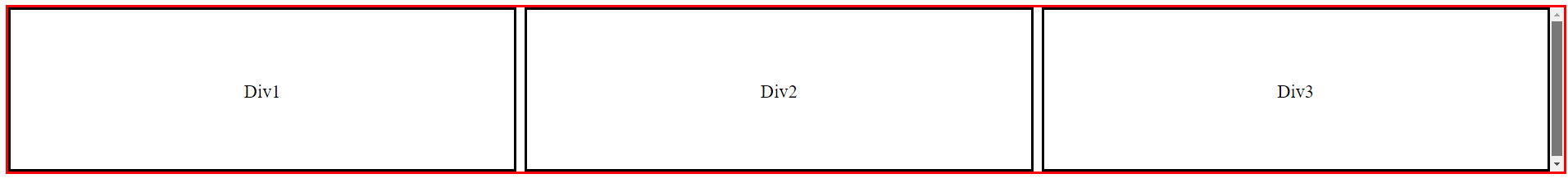
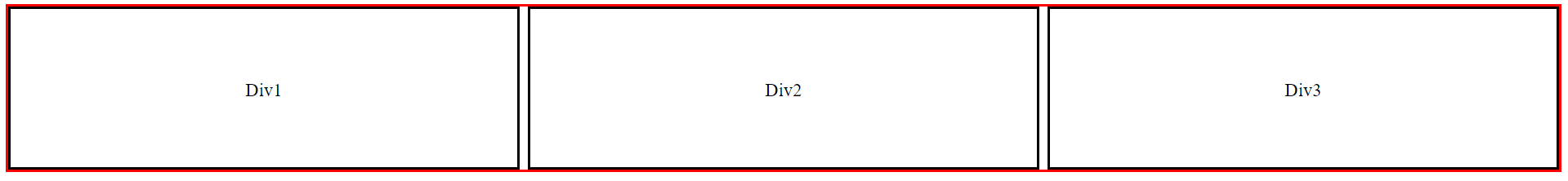


Immagine che contiene testo, schermata, software, Software multimediale

Descrizione generata automaticamente



È stato aggiunto un bordo rosso al contenitore per evidenziare il fatto che ore tale elemento racchiude effettivamente i tre <div>.

Poiché la proprietà ‘overflow: hidden’ in alcuni casi può creare altri problemi, un’altra soluzione potrebbe essere quella di aggiungere nel contenitore un ulteriore <div> come ultimo fratello, con contenuto vuoto, e assegnare a quest’ultimo la proprietà ‘clear’ con valore ‘both’:

Immagine che contiene testo, schermata, software, Software multimediale

Descrizione generata automaticamente

Immagine che contiene testo, schermata, software, schermo

Descrizione generata automaticamente

Come approfondimento è utile sapere che la best practice prevede di dare al contenitore delle card (degli elementi float in generale) una classe con le seguenti regole css:

.nome\_classe::after{

content: "";

display: block;

clear: both;

}

Per visualizzare i tre <div> in ordine inverso rispetto a come sono riportati nel codice html, basta valorizzare la proprietà ‘float’ a ‘right’.

In generale, per modificare l’ordine degli elementi con il css, senza toccare il codice html, si può agire sui valori ‘left’ e ‘right’ della proprietà ‘float’ ma, in alcuni casi, è necessario combinare tale proprietà con ‘display: inline-block’. Infatti, non tutte le combinazioni sono realizzabili solo attraverso ‘float’:

Immagine che contiene testo, schermata, software, schermo

Descrizione generata automaticamente

Immagine che contiene testo, schermata, software, Software multimediale

Descrizione generata automaticamente

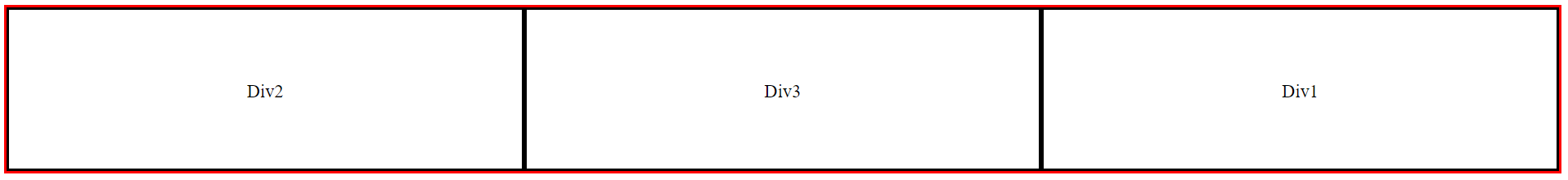


Immagine che contiene testo, schermata, software, Software multimediale

Descrizione generata automaticamente

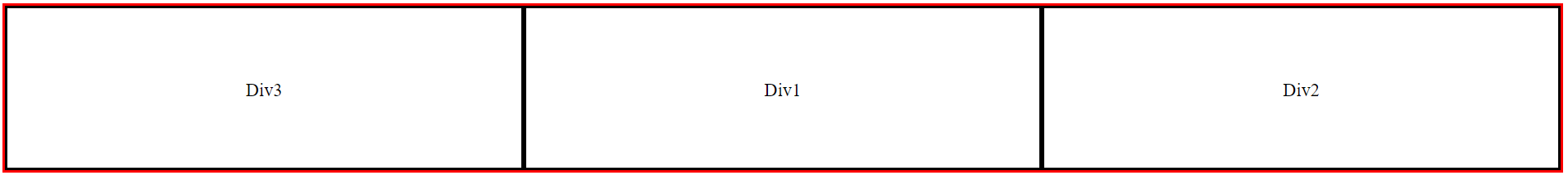
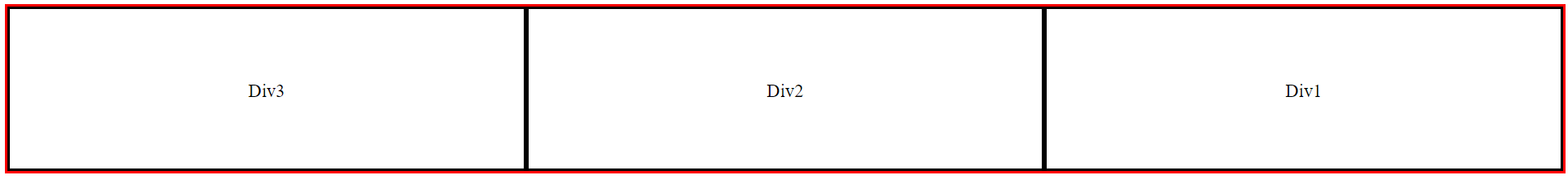


Immagine che contiene testo, schermata, software, Software multimediale

Descrizione generata automaticamente



**I tag semantici**

Fino alla versione 4 di HTML, la realizzazione dei siti web avveniva utilizzando principalmente gli elementi <div> per implementare le diverse sezioni, come l’intestazione, la barra di navigazione, il contenuto, il footer ecc. Ognuna di queste sezioni era costituita da un <div> con specifico id ad indicarne l’effettivo ruolo.

Dalla versione 5 di HTML, questo modus operandi è stato abbandonato grazie all’introduzione dei tag semantici. Di seguito vengono illustrati i principali:

<header> 🡪 all’interno di questo tag viene racchiusa l’intestazione della pagina web, che normalmente contiene il titolo e/o il logo del sito. Questo tag può essere utilizzato anche più volte all’interno di un sito, per rappresentare, ad esempio, l’intestazione di altre sezioni;

<nav> 🡪 questo tag è utilizzato per racchiudere i menu di navigazione e, anche in questo caso, può essere utilizzato più volte all’interno della stessa pagina web;

<main> 🡪 questo tag è destinato a inglobare il contenuto principale del sito;

<section> 🡪 viene utilizzato per creare delle sezioni distinte;

<aside> 🡪 tag destinato a contenere informazioni che non hanno a che fare con lo scopo del sito, come ad esempio bunner pubblicitari;

<article> 🡪 tag destinato a contenere contenuti indipendenti;

<footer> 🡪 viene utilizzato per realizzare il cosiddetto piè di pagina, ma anche per contenere la parte finale di altre parti del sito web;

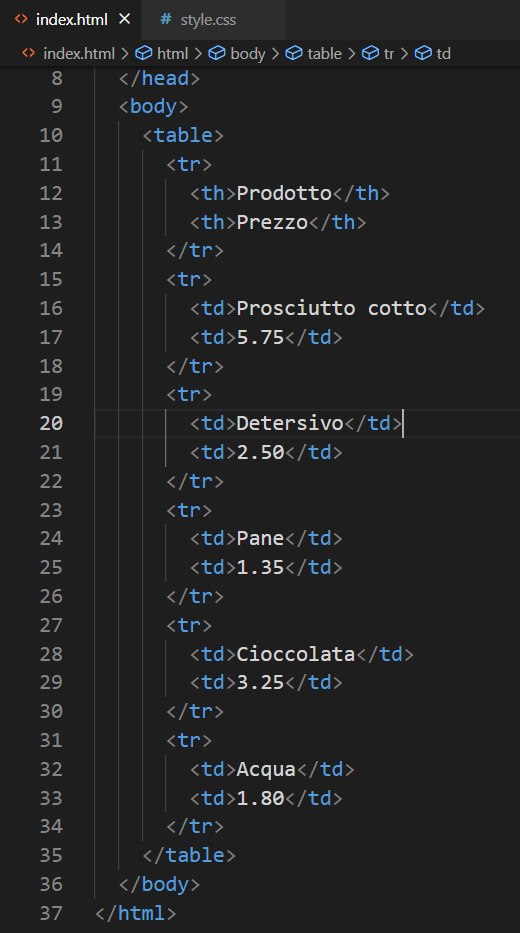
<time> 🡪 viene utilizzato per inserire una data, a volte arricchita da un orario. Quando si usa questo tag va specificato l’attributo ‘datetime’ il cui valore indica la data a cui si fa riferimento in uno specifico formato: YYYY-MM-DDTHH:MM:SS.MS+GMT (esempio 2024-01-11T16:30:25.56+01:00). L’informazione relativa all’orario è opzionale. Quando il tag <time> viene inserito all’interno di un elemento <article>, generalmente si utilizza anche l’attributo ‘pubdate’ ad indicare che si tratta della data di pubblicazione dell’articolo;

<details> serve per specificare dei dettagli riguardanti un determinato argomento. Annidato al suo interno troviamo il tag <summary>, ad indicarne il titolo, seguito dalla descrizione del dettaglio. Si tratta di un elemento non molto utilizzato.

**Le tabelle**

Fino alla versione 4 di HTML, le tabelle venivano utilizzate anche per realizzare il layout della pagina. Questa pratica ormai è da considerarsi obsoleta, pertanto le tabelle vanno utilizzate solo per impaginare dei dati.

Di seguito un esempio di tabella:



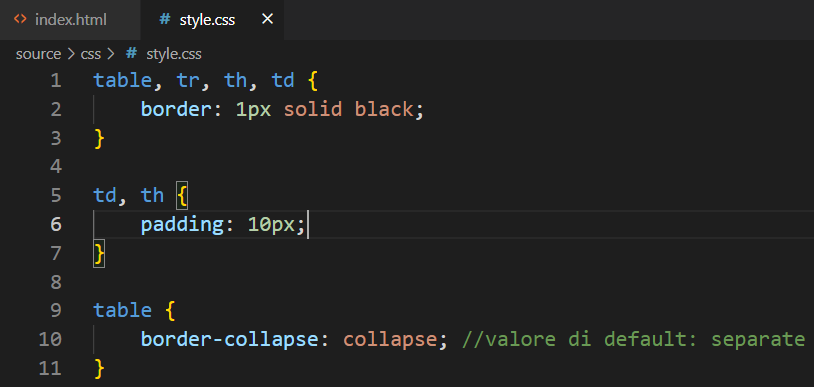


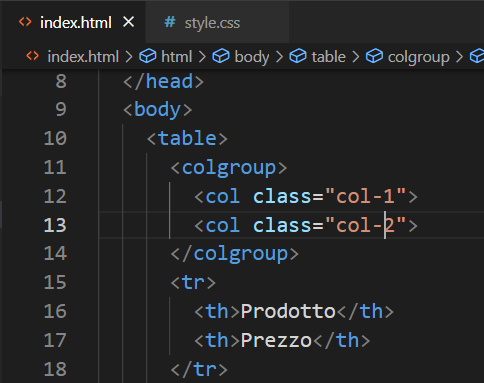
Immagine che contiene testo, schermata, Carattere, numero

Descrizione generata automaticamente

L’elemento <tr> viene utilizzato per creare una riga della tabella. All’interno di questo tag possiamo trovare:

* <th> per l’intestazione;
* <td> per le singole celle contenenti i dati da mostrare.

Per questioni legate all’accessibilità del sito web, nelle tabelle sono molto usati i tag <colgroup> e <col>, che ci permettono di facilitare l’impostazione dello stile per le colonne. Questi tag vanno inseriti subito dopo l’elemento <table>:



Con questa tecnica possiamo modificare solo alcune proprietà css, come il colore di sfondo, la larghezza, il bordo e la visibilità. Tutte le altre proprietà non verranno prese in considerazione.

Altri tag che dovrebbero essere sempre usati sono:

* <thead> per contenere l’intestazione della tabella;
* <tbody> per contenere il corpo della tabella;
* <tfoot> per il footer della tabella, dove generalmente vengono inseriti eventuali consuntivi.

Immagine che contiene testo, schermata, software, Software multimediale

Descrizione generata automaticamente

Immagine che contiene testo, schermata, software, Software multimediale

Descrizione generata automaticamente

Per costruire tabelle con colonne tutte della stessa larghezza, senza doverlo calcolare a mano, è possibile utilizzare una strategia che prevede:

* l’utilizzo della proprietà ‘table-layout: fixed’ per l’elemento <table>;
* impostare la proprietà ‘width: 100%’ per le celle <th> e <td>.

Con questa tecnica il browser si andrà a calcolare automaticamente la larghezza corretta delle colonne.

Gli attributi ‘rowspan’ e ‘colspan’ applicati all’elemento <td> permettono di accorpare più celle rispettivamente su righe di verse o su colonne diverse, variando in questo modo la struttura della tabella.

Per specificare il titolo di una tabella si dovrebbe utilizzare il tag <caption> annidato all’interno di <table>., la cui proprietà css ‘caption-side’ viene utilizzata per specificarne la posizione (sopra, sotto, a fianco della tabella ecc.).

**I link**

Per inserire dei link all’interno di una pagina web si utilizza l’elemento <a>. Questo tag è di tipo inline ma, a differenza degli altri dello stesso tipo, può contenere al suo interno qualunque altro elemento, anche di tipo block.

Al tag <a> sono associati due attributi importanti:

* ‘href’ serve per specificare il percorso della risorsa che deve essere aperta al click sul link. È necessario sempre associare un valore a questo attributo, altrimenti il tag <a> perderebbe la sua funzionalità. Se inizialmente il valore da inserire non è conosciuto, al suo posto convenzionalmente si può utilizzare il carattere ‘#’;
* ‘target’ viene utilizzato per fare in modo che la nuova risorsa venga aperta in un’altra scheda del browser, in modo da non chiudere la pagina web attuale. Per convenzione, il valore associato a tale attributo è ‘\_blank’. Qualora non venisse utilizzato, al click del link la pagina web attuale verrebbe automaticamente sostituita dalla risorsa collegata.

Il tag <a> viene utilizzato anche per creare dei collegamenti all’interno della stessa pagina o con altre pagine dello stesso sito. Questo comportamento viene implementato con l’ausilio dell’attributo id, che viene definito nel tag html che deve fungere da ancora. Per fare riferimento a quest’ultimo, l’attributo ‘href’ del link deve essere valorizzato nel modo seguente: ‘#tag\_id’.

Se il link fa riferimento ad un’ancora che si trova in un’altra pagina html dello stesso sito, il valore associato all’attributo ‘href’ deve avere la seguente forma: ‘percorso\_pagina\_html#tag\_id’.

Un altro attributo importante del tag <a> è ‘download’ che viene utilizzato quando il link fa riferimento ad un’immagine o ad un file scaricabile.