Il modello Flexbox e il suo funzionamento

Il modello di layout **Flex** è stato introdotto per aggiungere una funzionalità adattabile alle nuove esigenze responsive delle applicazioni web. Fino alla sua introduzione esistevano quattro principali modelli di layout:

- **block** — il modello generale;
- **inline** — principalmente pensato per posizionare blocchi di testo;
- **table** — utilizzato per posizionare tabelle:
- **positioned** — vale a dire absolute, fixed, static e relative, che consentono di posizionare elementi in maniera indipendente.

Il modello Flex possiede delle caratteristiche in grado di riposizionare gli elementi a seconda delle dimensioni del contenitore e si rapporta a due assi: il **main axis** (orizzontale - x) e il cross axis (verticale - y).

L'immagine precedente è presa dalla sezione del W3C che tratta direttamente l'argomento Flexbox ed esemplifica in maniera chiara il funzionamento del modello.

Impostazioni iniziali

Per esemplificare il funzionamento delle varie proprietà del modello Flex, utilizziamo dei semplici parametri iniziali.
Il contenitore ha uno sfondo blu, mentre gli elementi, numerati, sono di colore arancio con bordo bianco. In realtà il bordo bianco lo otteniamo tramite **box-shadow inset** per mantenere gli esempi compatibili con **Internet Explorer**, il quale calcola il bordo come un'aggiunta alle dimensioni degli elementi.
Impostiamo inoltre una dimensione semi-fissa per il contenitore (w100% * h200px) e una dimensione relativa per gli elementi (25% * 25%), in modo da avere una visione più omogenea dell'esempio.

flex-direction

Regola la scelta dell'asse lungo il quale il contenitore allinea gli elementi e la loro direzione.

- **row** distribuisce gli elementi da sinistra a destra lungo il **main axis**;
- **row-reverse** agisce come row, ma da destra a sinistra;

- **column** distribuisce gli elementi dall'alto verso il basso lungo il **cross axis**;
- **column-reverse** agisce come column, ma dal basso verso l'alto.

flex-wrap

Definisce se gli elementi devono essere distribuiti lungo l'asse oppure no. Definisce inoltre la direzione di distribuzione.

- **nowrap** stabilisce che gli elementi **non devono** essere distribuiti su più righe o colonne;
- **wrap** stabilisce che gli elementi **devono** essere distribuiti su più righe o colonne in modo da mantenere le loro dimensioni prefissate;
- **wrap-reverse** agisce come wrap, ma gli elementi devono seguire il senso contrario dell'asse.

justify-content

Definisce la gestione dello spazio residuo tra gli elementi lungo il **main axis**.

Per le colonne

- **flex-start** posiziona tutti gli elementi all'inizio del contenitore, lasciando lo spazio residuo dopo *l'ultimo* elemento;
- **flex-end** posiziona tutti gli elementi alla fine del contenitore, lasciando lo spazio residuo prima del *primo* elemento;
- **center** posiziona tutti gli elementi al centro del contenitore, distribuendo lo spazio residuo in modo uniforme prima del primo elemento e dopo l'ultimo elemento;
- **space-between** distribuisce lo spazio residuo in modo uniforme tra gli elementi;
- **space-around** distribuisce lo spazio residuo in modo uniforme *attorno* gli elementi;

align-items

Definisce la distribuzione dello spazio residuo tra gli elementi lungo il **cross axis**.

- **flex-start** agisce come quello di **justify-content**;
- **flex-end** agisce come quello di **justify-content**;
- **center** agisce come quello di **justify-content**;
- **stretch** allunga gli elementi per coprire l'intera altezza del contenitore, eliminando lo spazio residuo;
- **baseline** allinea gli elementi seguendo la baseline del testo in essi contenuto. Come si può osservare nell'esempio, il testo dei tre elementi ha dimensione differente. L'elemento con il testo di dimensioni maggiori viene posizionato all'inizio dell'asse e gli altri elementi vi si allineano seguendo la baseline del testo.

align-content

Funziona esattamente come **justify-content**, ma si applica al **cross axis**.

Proprietà degli elementi

Gli elementi hanno delle proprietà che ci permettono di attribuire loro comportamenti collettivi o indipendenti.

Nell'esempio seguente ho indicato il valore corrispondente all'interno dei vari elementi.

- **order** definisce la posizione dell'elemento nell'ordine degli elementi. Se **order** non è impostato, gli elementi vengono ordinati secondo il loro ordine di creazione. Accetta solo un valore numerico positivo.
- **flex-grow** stabilisce l'ordine di ridimensionamento degli elementi: più alto è il valore, più l'elemento si espande per occupare lo spazio residuo. È necessario specificare un valore di base (ad esempio **flex-grow: 1;**) per gli elementi che non devono espandersi.
- **flex-shrink**, come per **flex-grow**, più alto è il valore, più l'elemento si restringe. Non è necessario specificare un valore di base.
- **flex-basis** stabilisce la dimensione base che gli elementi devono raggiungere prima di iniziare la gestione dello spazio residuo.
- **align-self** sovrascrive l'allineamento predefinito per l'elemento.

Proprietà non supportate

Proprio per le sue caratteristiche di allineamento e distribuzione automatiche degli elementi, il modello Flexbox ignora/altera il funzionamento di alcune proprietà.

- **float** e **clear** vengono ignorati dagli elementi di tipo Flex.
- **vertical-align** viene ignorato.
- Tutte le proprietà di **column-** non sono utilizzabili per gli elementi di tipo Flex.
- **::first-line** e **::first-letter** non sono utilizzabili.
- Un elemento con **position: absolute** non parteciperà al riordinamento degli altri elementi.

Supporto browser

Il supporto browser rimane al momento limitato alle ultime versioni dei principali browser. Di seguito una tabella riassuntiva. Per i dati mi sono basato sulla tabella di Caniuse.com, al momento aggiornata al 19 gennaio 2014.

Altre risorse

- W3C - CSS flexible box layout module level 1
- Caniuse.com - Flexible box layout module
- A complete guide to Flexbox by Chris Coyier
- Flexplorer by Bennett Feely
- Diving into Flexbox by Bocoup

Conclusioni

Come abbiamo potuto vedere, il modello Flexbox ci fornisce degli ottimi strumenti per costruire layout complessi e per controllarne facilmente il comportamento in base alle differenti viewport. Questa caratteristica risulta molto efficace nel **responsive webdesign**, ma non solo. È in grado di rendere più semplice anche la progettazione di applicazioni web dinamiche nelle quali, ad esempio, vogliamo ridimensionare e riordinare alcuni elementi in base a un comportamento dell'utente. Diventa sufficiente impostare alcune proprietà e il modello Flexbox esegue i calcoli al nostro posto.
L'unica nota dolente rimane il supporto browser che confina l'utilizzo di questa tecnica alle ultime versioni. Non credo che il supporto a Flexbox verrà esteso alle vecchie versioni, di conseguenza l'unica pratica possibile rimane quella di progettare delle solide applicazioni web utilizzando tecniche standard e arricchirle nella visualizzazione sui browser di ultima generazione con tecniche più moderne, come consigliato da Dave Methvin e Rey Bango nel loro articolo "Cross browser development standards & interoperability best practices | Modern.IE".

Come sempre, potete trovare il codice relativo alla lezione a questo indirizzo!