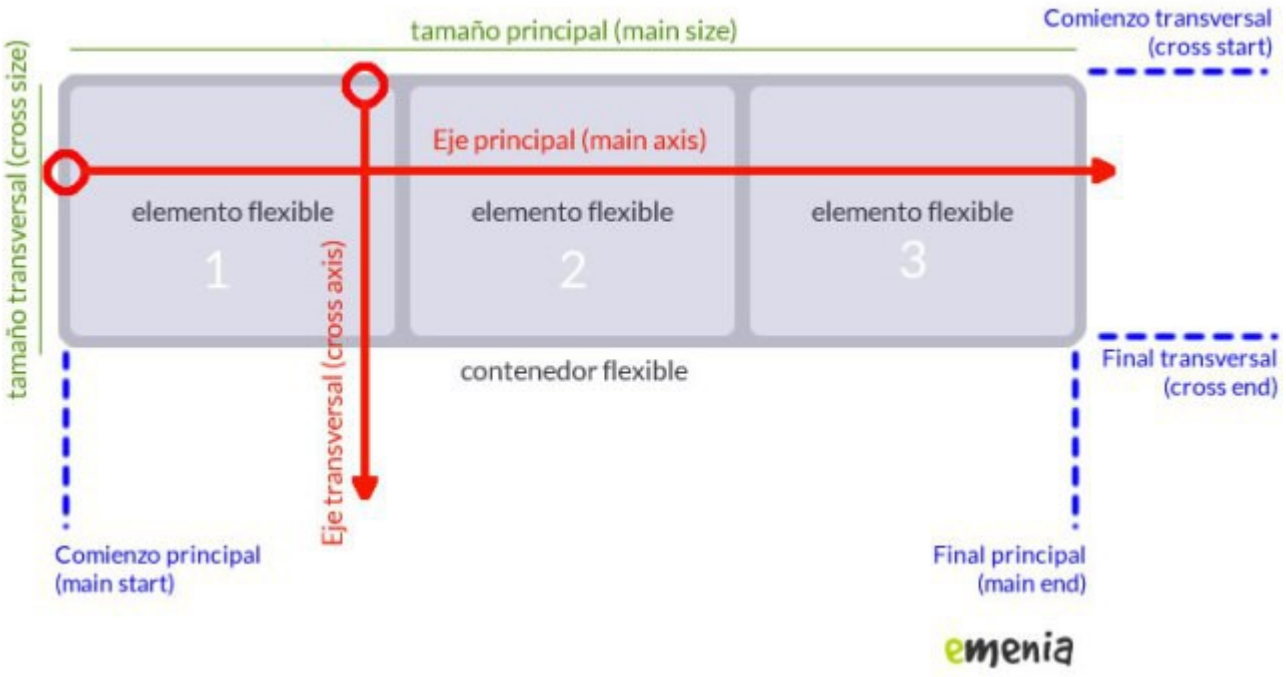
En qué consiste FlexBox

El objetivo de FlexBox es crear un modelo de caja o contenedor optimizado para los distintos dispositivos que usan los usuarios de una web. En CSS 2.1 se establecieron cuatro formas de crear una caja: block, inline, table y position. Ahora se crea un modelo de caja nuevo, **la caja flexible o FlexBox** con display: flex.

Funciona ajustando los tamaños y la disposición de los elementos que se encuentran dentro de un contenedor o caja, de tal manera que **se adapten siempre al espacio disponible**. Permite posicionar dichos elementos internos con gran facilidad, de manera independiente al orden en el que aparezcan en el código.

display: flex

La declaración display: flex; define un "contenedor flexible" y convierte de forma automática a sus "hijos" directos en "elementos flexibles". Un contenedor flexible tiene un Eje principal (main axis), que es la dirección en la cual se posicionan los elementos flexibles. Y tiene un Eje transversal, perpendicular al Eje principal. Ambos ejes tienen una serie de propiedades que controlan cómo se posiciona cada elemento flexible en relación a los demás. Puedes además poner contenedores flexibles uno dentro de otro, ya que la pripiedad display no se hereda de forma automática. Vamos a ver un gráfico que nos lo muestra mejor:



Empezamos con un ejemplo

Vamos a centrarnos en un ejemplo sencillo, perfecto para usar FlexBox (recordad que necesitáis un navegador moderno para ver bien la demo). Os podéis bajar aquí el código del ejemplo. Como podéis ver es un catálogo de viajes donde cada viaje tiene un título, una foto, una descripción y un botón de "Más información". Lo que queremos es que cada entrada tenga el mismo tamaño (anchura y altura), que la foto esté sobre el texto (aunque en el HTML no sea así) y que el botón de "Más información" esté siempre alineado abajo. Además, cuando cambiemos el ancho de pantalla se ajustará todo perfectamente sin necesidad de haber calculado complejos porcentajes. Con FlexBox conseguir esto es facilísimo!

Veamos en HTML que hemos usado. Nada especial:

```xhtml
<section id='viajes'>
    <div class='viajes-item uno'>
       <h2>Malta, la isla mediterránea por conocer</h2>
       <p>Dignissim placerat vel aenean porta, magna ac scelerisque facilisis, rhoncus est! Urna
       <img src='images/malta.jpg' alt='' />
       <button>MÁS INFO</button>
    </div>
    <div class='viajes-item dos'>
        <h2>Polonia, un país por descubrir</h2>
        <p>Dignissim placerat vel aenean porta, magna ac scelerisque facilisis.</p>
        <img src='images/oporto.jpg' alt='' />
        <button>MÁS INFO</button>
    </div>
    <div class='viajes-item tres'>
        <h2>Un paseo por el viejo Oporto</h2>
        <p>Cras quis mattis. Elementum, lectus magna, amet dis dis pulvinar scelerisque proin mon
        <img src='images/polonia.jpg' alt='' />
```

Los contenedores flexibles: flex e inline-flex

```css
#viajes {
    display: flex;
    display: -webkit-flex; /* Para navegadores webkit, como Safari */
}
```

Como veréis no usamos ni `position` ni `float`, ni ningún valor `block` o `inline` para `display`. Usamos `display: flex`. A partir de ese momento el contenedor (#viajes) será un contenedor flexible y los elementos dentro del contenedor serán elementos flexibles que se adaptarán a los espacios disponibles que les deje el contenedor. Podríamos haber puesto también:

```css
#viajes {
    display: inline-flex;
    display: -webkit-inline-flex;
}
```

en cuyo caso se comportaría en relación a otros elementos de la página de manera semejante a *display: inline*. En caso contrario se comporta igual que *display: block*.

Orientación: flex-flow, flex-direction y flex-wrap

Seguimos. Vamos a añadir al contenedor una propiedad que nos va a permitir decirle al navegador cómo se van a alinear los elementos que están dentro del contenedor (#viajes).

```css
#viajes {
    display: flex;
    display: -webkit-flex;
    flex-flow: row wrap;
    -webkit-flex-flow: row wrap;
}
```

Vamos a ver el CSS, pero ahora vamos a ir poco a poco:

Hemos añadido ***flex-flow: row wrap***. De esta manera establecemos que los elementos dentro del contenedor (los .viajes-item) se van a alinear en fila (row). *flex-flow* lo podríamos haber dividido en dos:

```css
#viajes {
    display: flex;
    display: -webkit-flex;
    flex-direction: row;
    -webkit-flex-direction: row;
    flex-wrap: wrap;
    -webkit-flex-wrap: wrap;
}
```

Hemos visto antes que el *eje principal* es la *dirección principal*, pero hay que tener en cuenta que esta no es siempre horizontal. La dirección del contenedor se puede cambiar con la propiedad ***flex-direction***, que especifica cómo se sitúan los elementos flexibles dentro del contenedor. Sus posibles valores son:

- *row*: se alinean en filas.
- *row-reverse*: en filas, pero con el orden inverso.
- *column*: se alinean en columnas.
- *column-reverse*: en columnas, pero con el orden inverso.

Es decir, que en móviles lo más lógico para nuestro ejemplo es tener una lista de una columna en lugar de varias columnas en horizontal. Esto se consigue en el ejemplo con flex-direction: column;.

En tamaños grandes se usaría flex-direction: row; para lograr que se distribuya horizontalmente. El valor por defecto de *flex-direction* es *row*, por lo que si no usáramos flex-direction los elementos aparecerían distribuidos horizontalmente.

Lo dicho: para probar *flex-direction:column* vamos a cambiar la orientación cuando el navegador tenga un ancho de 500px o menor. Así lo veremos mejor en los móviles:

```css
@media all and (max-width: 500px) {
    #viajes {
        flex-direction: column;
        -webkit-flex-direction: column; }
}
```

Probad a reducir el ancho del navegador y veréis el cambio al llegar a los 500px de ancho.

La propiedad ***flex-wrap*** controla si el contenedor flexible (en este caso #viajes) tiene una sola línea o múltiples líneas, así como la dirección en la que se colocan las nuevas líneas en el eje transversal. Sus posibles valores son:

- *nowrap:* El contenedor consta de una sola línea.
- *wrap:* El contenedor tiene múltiples líneas.
- *wrap-reverse:* El contenedor tiene múltiples líneas que se colocan en orden inverso.

En este caso, al ser múltiples líneas, hemos puesto *wrap*.

Bueno, ya hemos definido el comportamiento del contenedor (#viajes). **Ahora vamos a ver las propiedades que podemos asignar a los elementos que están dentro del contenedor** (los .viajes-item). Para empezar vamos a establecer que cada .viajes-item sea a su vez un contenedor flexible, donde sus componentes (el h2, el párrafo, la imagen y el botón) se van a alinear en columna:

```css
.viajes-item {
    display: flex;
    display: -webkit-flex;
    flex-direction: column;
   -webkit-flex-direction: column;
}
```

Orden: La propiedad order

Podemos establecer el orden en el que aparecen los componentes de una caja flexible. Por defecto aparecerán tal y como aparecen en el código HTML (equivale a *order: 0*), pero eso se puede cambiar de forma sencilla. Volvamos al ejemplo. Como hemos puesto una clase a cada viaje, podemos jugar con ellos:

```css
.uno {
    background-color: #bacee3;
    order: 3;
    -webkit-order: 3;
}
.dos{
    background-color: #b6ebb3;
    order: 2;
    -webkit-order: 2;
}
.tres {
    background-color: #7ebdbb;
    order: 1;
    -wbkit-order: 1;
}
```

Con la propiedad order podemos poner el orden que queramos. Hemos cambiado el orden del primero y del tercero. Es muy sencillo, como podéis ver (he puesto cada viaje de un color diferente para poder ver bien cada uno). ¿Os imagináis qué habríais tenido que hacer para conseguir esto con las herramientas que tenemos ahora mismo?

A propósito, al poner cada *.viaje-item* de un color diferente podéis ver **cómo todos tiene la misma altura independientemente de la de sus componentes**.

Vamos a aprovechar el media-query que iniciamos antes para volver a poner todo en su orden original (*order: 0* para los *.viajes-item*) para anchos inferiores a 500px:

```css
@media all and (max-width: 500px) {
    #viajes {
        flex-direction: column;
        -webkit-flex-direction: column; }
    }
    .viajes-item {
        order: 0;
        -webkit-order: 0;
        width: auto;
    }
```

Dentro de cada viaje queremos poner la imagen encima de todo. ¿Cómo lo conseguimos?

```css
.viajes-item > img {
    order: -1;
    -webkit-order: -1;
}
```

Como por defecto todos tienen *order: 0* si ponemos a la imagen *order: -1* se pondrá la primera.

Flexibilidad: La propiedad flex

Con la propiedad flex podemos establecer cómo crece o decrece un elemento flexible dentro del contenedor en relación a los demás. En el ejemplo que estamos siguiendo las tres columnas tienen el mismo ancho, por ello veréis en el código que todos tienen *flex:1*:

```css
    .uno {
        background-color: #bacee3;
    flex: 1;
    -webkit-flex: 1;
    -webkit-order: 3;
    }
    .dos{
        background-color: #b6ebb3;
    flex: 1;
    -webkit-flex:1;
    -webkit-order: 2;
    }
    .tres {
        background-color: #7ebdbb;
        flex: 1;
    -webkit-flex:1;
    -wbkit-order: 1;
    }
```

Si hubiéramos querido que *.uno* ocupara el doble que los otros dos habríamos hecho:

```css
    .uno {
        background-color: #bacee3;
        flex: 2;
        -webkit-flex: 2;
        -webkit-order: 3;
    }
```

Esta propiedad se puede volver más compleja, porque puede tener tres parámetros: ***flex-grow***, ***flex-shrink*** y ***flex-basis***.

- *flex-grow:* Especifica el factor de crecimiento, es decir, cuanto crecerá el elemento en relación a los demás cuando hay espacio disponible del contenedor a ocupar. Por defecto es '0', que es el valor que dimos en el ejemplo anterior a los tres elementos.
- *flex-shrink:* Determina el factor de reducción, es decir, cuanto decrecerá el elemento en relación a los demás cuando hay espacio negativo en el contenedor (el contenedor es más pequeño de los anchos combinados de los elementos que hay en su interior). Por defecto es '1'.
- flex-basis: Toma el mismo valor que la propiedad 'width' y establece el tamaño inicial del elemento antes de distribuir el espacio libre de acuerdo con los ratios de flex-grow o flex-shrink. Cuando se omite, su valor es 'main-size' (anteriormente, 'auto').

Para comprender este lío lo mejor es poner un ejemplo. Supongamos que tenemos un contenedor al que llamaremos "A" que tiene 300px de ancho. Hacemos que este contenedor sea flexible:

```css
A { display: flex;}
```

Supongamos que este contenedor tiene en su interior dos elementos, B y C, que no tienen un ancho especificado. Vamos a establecer los siguientes valores para ambos de la propiedad *flex: flex-grow, flex-shrink, flex-basis*:

```css
B { flex: 3 1 100px;}
C { flex: 1 2 100px;}
```

Como hemos establecido un *flex-basis* para cada elemento de 100px, nos quedarán aún 100px libres sin ocupar (300px del contenedor menos 100px del elemento B y menos 100px del elemento C). ¿Como se reparte ese espacio disponible entre los elementos B y C? En función del *flex-grow*: 3 partes para el elemento B (75px) y una parte para el elemento C (25px). Es decir, de inicio el elemento B ocupará 100px+75px = 175px de los 300px disponibles que mide el elemento A, y el elemento C ocupará 100px+25px=125px de los 300px disponibles del elemento A.

Ahora bien, supongamos que el elemento contenedor A mide 170px y no 300px. Eso quiere decir que habrá espacio negativo, porque los elementos B y C tienen un flex-basis de 100px cada uno, es decir, 200px, que es 30px mayor que los 170px del contenedor. En este caso el ratio que se usa es el flex-shrink, que recordemos que era 1 para el elemento B y 2 para el elemento C. Esos 30px se restarán del ancho de los elementos B y C en función de dicho ratio: al elemento B se le quitarán 10px y al elemento C se le quitarán 20px.

Parece un lío, pero seguro que dentro de poco nos habituaremos a hacer estos cálculos.

Alineación de los elementos flexibles

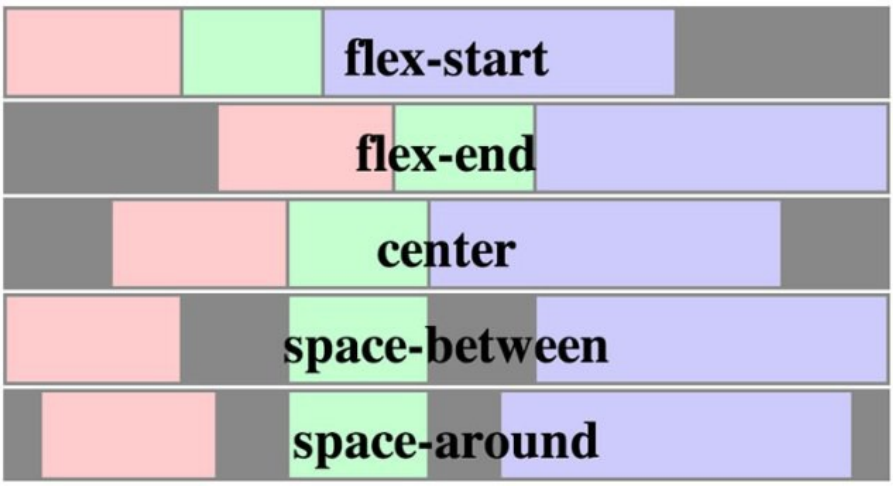
Podemos alinear los elementos flexibles en el Eje principal con justify-content y en el Eje transversal con align-items y align-self.

Si hay espacio extra dentro de un contenedor flexible la propiedad justify-content puede definir cómo se usa ese espacio que sobra. Las opciones son:

- **flex-start:** se distribuyen todos pegados al inicio.
- **flex-end:** se distribuyen todos pegados al final.
- **center:** se distribuyen todos alineados al centro.

- **space-between:** se distribuyen ocupando todo el espacio disponible, con separaciones iguales entre ellos, pero sin dejar espacio al inicio y al final.
- **space-around:** se distribuyen ocupando todo el espacio disponible, con separaciones iguales entre ellos, dejando espacio al inicio y al final.

Vamos a ver un gráfico de W3C que nos lo explica mejor:



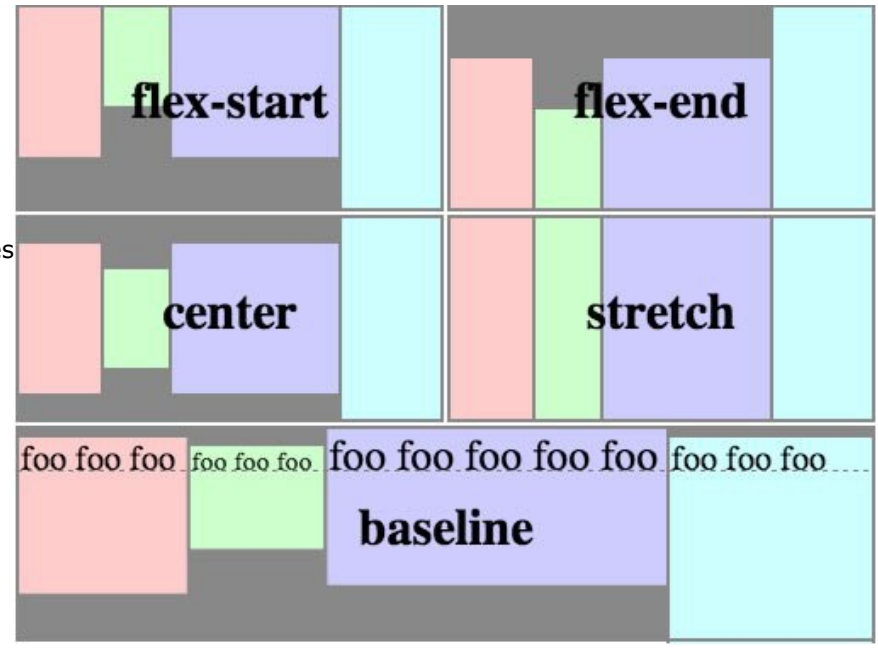
Podemos alinear los elementos flexibles en el eje transversal con align-items y con align-self. La primera establece el valor por defecto para todos, la segunda sirve para ser aplicada a elementos individuales sobre-escribiendo align-items para ese elemento.
Los valores posibles son:

- **auto:** Sólo se puede aplicar en align-self y equivale al valor de align-items del elemento padre, o a stretch si el elemento no tiene padre.
- flex-start
- flex-end
- center
- baseline
- stretch

Como una imagen vale más que mil palabras, aquí podéis ver cómo se distribuirían los elementos con cada una:

En el ejemplo inicial queríamos que el botón de "Más info" estuviera alineado en la parte inferior de cada caja flexible. Como todas tienen la misma altura, los tres botones estarán igualmente a la misma altura.

Eso lo conseguimos aplicando a la imagen *margin-top: auto*



```css
.viajes-item > button {
    margin-top: auto;
    width: 30%;
}
```

Además, queremos que tanto la imagen como el botón estén centrados dentro de su contenedor. Para ello utilizamos *align-self*:

```css
.viajes-item > img {
    order: -1;
    -webkit-order: -1;
    align-self: center;
    -webkit-align-self: center;
}
.viajes-item > button {
    margin-top: auto;
    width: 30%;
    align-self: center;
    -webkit-align-self: center;
}
```

Ojo: Hay propiedades de CSS que no funcionarán en un elemento que se encuentre dentro de un contenedor flexible, como *float*, *clear*, *column-* o *vertical-align*.

Bueno, hemos visto con un sencillo ejemplo las posibilidades de FlexBox. Hay mucho más que investigar dentro de esta nueva especificación.

Diferentes escenarios dependiendo del navegador

Si nuestros usuarios sólo utilizan los navegadores más modernos

Podemos usar FlexBox, sólo teniendo cuidado con:

- No usar flex-wrap ya que no lo soporta Firefox hasta la versión 28.
- Incluir el prefijo -webkit además del normal sin prefijo, para Safari 7.
- No usarlo para la estructura global de la página, como vimos al principio del artículo.

Si necesitas Explorer 10 y versiones anteriores de Firefox, Safari y Opera

Tendrás que combinar versiones anteriores de Flexbox con el actual, lo que es un poco tedioso (lo puedes solucionar con Autoprefixer. Tampoco deberás usar auto-márgenes.

Si necesitas IE8 e IE9 (seguro que lamentablemente, si)

En ese caso, y hasta que alguien no desarrolle un polyfill en condiciones, no hay nada que hacer salvo crear un diseño aparte para estos navegadores. Quizás con Modernizr detectar si soporta o no Flexbox y crear un diseño aparte….