Disseny web adaptable

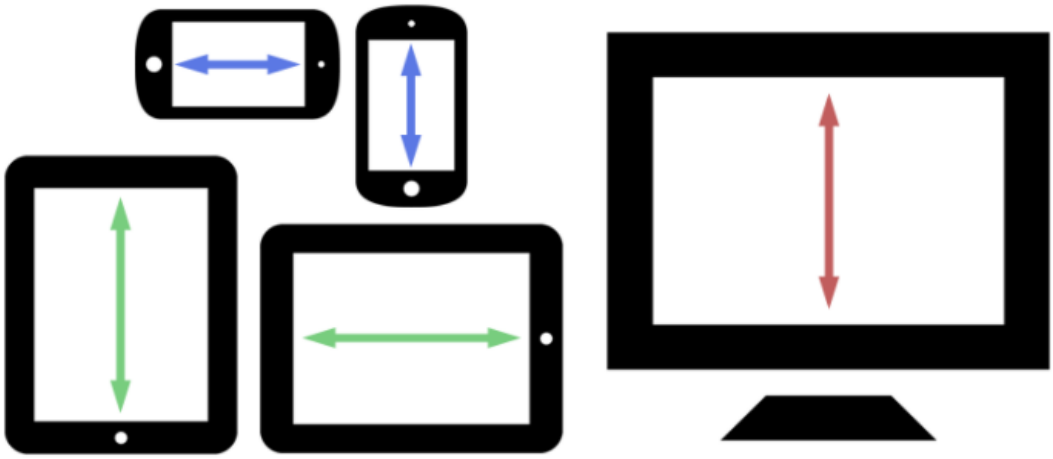
lloc: CIFP Francesc de Borja Moll
Curs: Llenguatges de marques i sistemes de gestió d'informació
Llibre: Disseny web adaptable
Imprès per: Joan Llompart Socias
Data: dilluns, 15 novembre 2021, 10:27

Taula de continguts

1. Disseny web adaptable

Un disseny web adaptable o *responsive* (RWD per les seves sigles en anglès), **és un mètode de creació de llocs web amb el qual una pàgina es mostra correctament sigui com sigui el dispositiu utilitzat per a visualitzar-la.**

La raó de què el disseny adaptable sigui cada vegada més important té a veure amb el gran creixement de l'ús de dispositius mòbils, tant si són telèfons mòbils com a tauletes o portàtils (dispositius que a vegades se'ls coneix com a pantalles horitzontals, per oposició als dispositius de sobretaula, als quals se'n diu pantalles verticals).



El propòsit d'un RWD és oferir la millor experiència de navegació possible, especialment per una fàcil usabilitat i la disminució del 'scrolling'

Principals característiques que defineixen un disseny web adaptable

- **Compatibilitat entre dispositius**: La seva capacitat per a adaptar la visualització d'un portal web a qualsevol dispositiu.
- **Una única versió**: El desenvolupament i manteniment del codi se simplifica, ja que comptem únicament amb una versió d'aquest.
- **Millora l'experiència d'usuari**: Per exemple, en un cas tan senzill com girar el mòbil, passant de posició vertical a horitzontal, amb el disseny web adaptable, l'usuari continuarà visualitzant els continguts de manera correcta.
- **Reducció de la càrrega del servidor**: Principalment perquè desapareix la necessitat

de realitzar redireccions de URL's en funció del tipus de dispositiu.

Tècniques de disseny web adaptable

Per poder crear pàgines web adaptables hi ha diverses tècniques:

- **Models de reixeta** ("modelos de rejilla" en castellà), que ens permeten ubicar els elements de la pàgina en funció de la pantalla.
- ***media queries***, que ens permeten especificar codi segons el tipus del dispositiu.

- **Models de reixeta**

  Molts dissenys actuals es basen en el model de reixeta, o GRID.

  - La reixeta és la base sobre la qual construirem el *layout (*el layout fa referència a la manera en què estan distribuïts els elements i les formes dins d'un disseny d'una pàgina, és a dir, la disposició dels elements).
  - És el component essencial per a assegurar que la nostra web s'adaptarà convenientment a diferents resolucions i pantalles.Bàsicament consisteix a dividir la pàgina en un nombre de columnes (normalment 12) i disposar les capes de manera que ocupin un nombre de columnes determinat.
  - Es basa en dues idees:
    - La primera és la de visualitzar les caixes en poques files amb moltes columnes quan la pantalla és el suficient ample.
    - La segona, és la de mostrar les caixes en moltes files amb poques columnes quan hi ha pocs píxels d'ample de la pantalla.

  Bootstrap: És un *framework* CSS desenvolupat per Twitter en 2010, per a estandarditzar les eines de la companyia. Inicialment, es va dir *Twitter Blueprint* i, una mica més tard, en 2011, es va transformar en codi obert i el seu nom va canviar per a *Bootstrap*. Des de llavors va ser actualitzat diverses vegades i ja es troba en la versió 4.4. El *framework* combina CSS i JavaScript per a estilitzar els elements d'una pàgina HTML. Conté plantilles de disseny amb tipografia, formularis, botons, quadres, menús de navegació i altres elements de disseny basat en HTML i CSS, així com extensions de JavaScript addicionals. A diferència de molts frameworks web, només s'ocupa del desenvolupament front-end. A més de totes les característiques que ofereix el *framework*, el seu principal objectiu és permetre la construcció de llocs web responsive per a dispositius mòbils.
  https://es.wikipedia.org/wiki/Bootstrap_(framework)
  https://getbootstrap.com/

- ***media queries***

  Els m*edia queries* ja es varen plantejar en la primera versió de CSS (1994), però el 2012 és quan es va incorporar l'estàndard amb l'arribada de CSS3 i la recomanació del consorci W3C. Des d'un principi es va plantejar la necessitat de trobar una manera de representar el contingut HTML a pantalles amb diferents resolucions, sempre des del punt de vista de client.

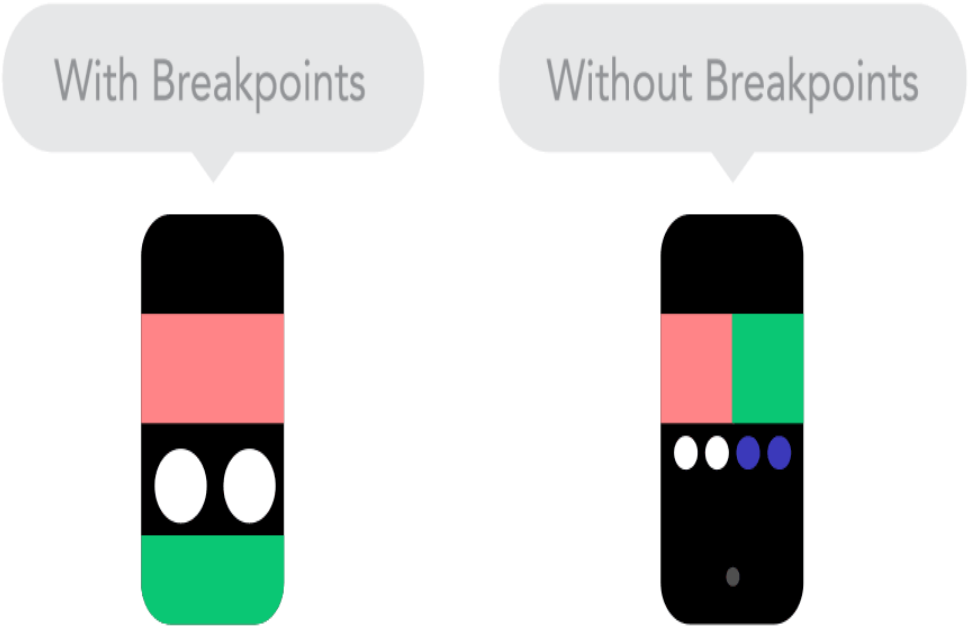
  - **Què són?** Són un tipus de regles de CSS que permeten crear un bloc de codi que només es processarà en els dispositius que compleixin els criteris específics a partir d'una condició.

```
Sintaxi:

Els media queries poden contenir una o més expressions, expressades com a funcions multimèdia, que es resolen en "true" o "false".
El resultat del query o consulta, retorna "true" si el tipus de mitjà especificat al media query, coincideix amb el tipus de dispositiu en què el document està sent mostrat. Quan un media query retorna "true", el corresponent full d'estil és aplicat, següent les regles habituals de CSS.

@media mitjà operador-lògic (condicions) {
 /* regles CSS */
 /* regles CSS */
     ...
}

On:
mitjà: serà el mitjà per on es visualitzarà la web.
operador-lògic: L'operador que ens permetrà definir l'aplicació del media querie.
condicions: Les condicions que es definiran perquè s'apliquin les regles CSS del punt d'interrupció.
-------------------------------------

Tipus de mitjans
- screen: Monitors o pantalles d'ordinador.
- print: Documents de mitjans impresos o pantalles de previsualització d'impressió.
- speech: Lectors de text per a invidents.
- All: Tots els dispositius o mitjans.

Operadors lògics: Es poden crear media queries complexos utilitzant certs operadors lògics.
```

- **L'operador "and":** Es utilitzat per combinar múltiples característiques en un sol *media querie*, requerint que cada funció retorni "true" perquè el *media querie* també ho sigui. Per exemple: @media screen and ((min-width: 700px) and (orientation: landscape)) { ... }
- **L'operador "not":** Es fa servir per negar un *media querie* completa. Per exemple: @media screen (not (monochrome) { ... }
- **L'operador "only":** Es fa servir per aplicar un estil només si el *media querie* complet és correcte.
- A més, **es poden combinar múltiples *media queries* separats per comes** en una llista: Si algun dels *queries* retorna "true", tota la declaració del *media querie* retornarà "true". **Això és equivalent a un operador "or".**

Punts d'interrupció

- Són punts en els quals el contingut del lloc web respon d'acord amb l'ample del dispositiu, la qual cosa li permet mostrar el millor disseny possible a l'usuari.
- No existeixen estàndards establerts que es puguin seguir per determinar-los. Per establir els punts d'interrupció s'utilitzaran "max-width" o "min-width", de manera que les propietats s'aplicaran quan la mida de l'àrea de visualització no superi aquests límits.



Exemples:

```
@media screen and ((min-width: 400px) and (max-width: 700px)) { ...
}

@media screen and (min-width: 1280px)  {...}
```

- **Propietats dels *media queries***
  - **width**: És l'ample de l'àrea de visualització (display area). Accepta els prefixos min / max.
  - **height**: És l'alt de l'àrea de visualització (display area). Accepta els prefixos min / max.
  - **min-width**: Si el dispositiu té una amplada major que la indicada.
  - **max-width**: Si el dispositiu té una amplada menor que la indicada.

Exemple: Indica quina regla CSS s'aplicarà a amplades entre 400 i 700 píxels.

```
@media screen and (min-width: 400px) and (max-width: 700px) {
   body{
      background-color: blue;
      font-family: Times New Roman;
   }
   header h3{
      text-align : center;
   }

@media screen and (min-width: 700px) and (max-width: 1250px) {
   body{
      font-family: Courier;
   }

   h1{
     color: green;;
   }
 }
```

  - **device-width:** Descriu l'amplada del dispositiu de sortida (ja sigui la totalitat de la pantalla o pàgina) i no l'àrea del document a renderitzar. (accepta els prefixos max/min).

```
Exemple: Per aplicar un full d'estil a un document quan aquest es mostri a una pantalla de menys de 800px d'ample.

@media screen and (max-device-width: 799px) { /*Regles CSS*/ }
```

- **device-height**: És l'alt total del dispositiu (rendering sourface). És lo mateix que l'anterior però en altura.
- **orientation**:
  - Amb valors: landscape ò portrait:
  - Si el dispositiu està posat en mode vertical o apaïsat.
  - No accepta els prefixos min/max.
  - El valor de "orientation" és "portrait" (retrat) quan el valor de l'altura del mitjà de sortida és major o igual al valor de l'amplada d'aquest mitja, al contrari el valor és "landscape" (paisatge).

```
Exemple:

@media all and (orientation:portrait) { /*Regles CSS*/ }
@media all and (orientation:landscape) { /*Regles CSS*/ }

@media all and (orientation:portrait) { /*Regles CSS*/ }
@media all and (orientation:landscape) { /*Regles CSS*/ }
```

- https://developer.mozilla.org/es/docs/Web/CSS/Media_Queries/Using_media_queries

Exemples

Exemple 1: Aplicar color de fons a una classe en funció del mitjà (aquest cas és una pantalla o "screen")

```
@media screen and (max-width: 640px) {
  .menu {
    background: blue;
  }
}

@media screen and (min-width: 640px) and (max-width: 1280px) {
  .menu {
    background: red;
  }
}

@media screen and (min-width: 1280px) {
  .menu {
    background: green;
  }
}
```

On: Tenim una classe "menu" amb un color de fons depenent del tipus de mitjà amb què es visualitzi la pàgina.
- Blau per a resolucions menors a 640 píxels d'amplada (mòbils).
- Vermell per a resolucions entre 640 píxels i 1280 píxels d'amplada (tauletes).
- Verd per a resolucions majors a 1280 píxels (escriptori).

També es poden indicar els *media queries* des d'HTML, utilitzant l'etiqueta <link>. L'equivalent al codi anterior seria:
- <link rel="stylesheet" href="mobil.css" media="screen and (max-width: 640px)">
- <link rel="stylesheet" href="tauleta.css" media="screen and (min-width: 640px) and (max-width: 1280px)">
- <link rel="stylesheet" href="escriptori.css" media="screen and (min-width: 1280px)">

Exemple 2: Aplicar un tipus de font si és per pantalla o per impressora.

```
@media screen {
 * { font-family: times }
}

@media print {
 * { font-family: arial }
}
```

**No fa falta que treballis amb un smarphone, una tauleta i un ordinador al mateix temps per a provar el bon funcionament dels media queries. Per a simular l'amplària d'una pantalla, només has de reduir o augmentar la talla de la finestra del teu navegador; també pots emprar l'opció "inspeccionar" del navegador; o pots afegir alguna extensió al teu navegador amb aquest propòsit, per exemple: "Resolution test".*

1.1. Àrea de visualització o viewport

El viewport és la mida visible d'una pantalla d'un dispositiu i l**a resolució real de la pantalla** és la quantitat de píxels que té aquesta. Aquestes dues mesures gairebé quasi mai coincideixen.

```
viewport: És l'àrea útil on es mostrarà la pàgina web.
```

Imagen hecha por Nora Ferreirós

El *viewport* va ser creat per la companyia Apple pels seus productes iPhone, iPod Touch o iPad, amb la finalitat de solucionar el problema sobre la mida de la pantalla a l'hora de visualitzar i navegar en un lloc web. Així per exemple, les premeres tauletes iPad d'Apple, l'iPad 1 i el 2, tenien una resolució de pantalla de 1024x768 píxels, i la mida per defecte de la seva viewport era exactament el mateix. No obstant això, quan van llançar l'iPad 3 que ja tenia una pantalla "retina", la resolució física de la mateixa era 2048x1536, mentre que la mida del viewport seguia sent el mateix: 1024x768.

Així veurem la pàgina web a dispositius mòbils sense declarar "viewport"

- # Etiqueta viewport
  - És la'etiqueta que millor representa la web adaptable, ja que ens permet indicar com es veurà un projecte web en els diferents dispositius que existeixen al mercat.
  - Definim quina àrea de pantalla està disponible quan es renderitza un document.
  - També defineix el nivell d'escalat i el nivell de zoom que s'ha de mostrar inicialment.

Sintaxi:
<meta name="viewport" content="width=device-width, initial-scale=1.0, maximum-scale=1" >

On "content" pot contenir els següents valors:

- width = Valor integral (en píxels). - device-width= Valor constant.
Si posam content="width=device-width ... estem indicant al navegador que l'ample de la finestra gràfica de disseny ha de ser el mateix que l'ample de la finestra del dispositi

- initial-scale = Qualsevol número real de 0.1 en endavant. 1 representa no escala: L'escala inicial del document.

- minimum-scale = Qualsevol número real de 0.1 en endavant. 1 representa no escala: L'escala mínima que es pot posar en el document.

- maximum-scale= Qualsevol número real de 0.1 en endavant. 1 representa no escala: L'escala màxima configurable en el document.

- user-scalable="no/yes": Si es permet o no a l'usuari fer zoom. (per defecte és "yes").

Exemples:

- Adaptam l'àrea visible de la web a l'ample de la pantalla del dispositiu. No es permet a l'usuari l'ampliació ni l'escalat de la web.

```
<meta name="viewport" content="width=device-width, user-scalable=no">
```

- Definint un ample fix de 320 píxels.

```
<meta name="viewport" content="width=320"/>
```

- Adaptam l'àrea visible a l'ample de la pantalla fixant una escala mínima i una escala màxima.

```
<meta name="viewport” content=”width=device-width, minimum-scale=2, maximum-scale=3“/>
```

Quan es construeix una web responsive, és recomanable definir un viewport adequat, en cas contrari és molt probable que la pàgina no llegeixi correctament els *media Queries* i que es vegi en format molt reduït, sent necessari fer zoom per veure el contingut.

Mesures de alguns telèfons mòbils
Iphone 6, 6s, 7, 8: 750px x1334px
IPhone 6+, 6s, 7+, 8+: 1242px x 2208 px
Mesures de tauletes
iPad tots els models, així com iPad Mini: 1024 x 768
Galaxy Tab 2 i 3 (10.1 polzades): 800 x 1280
Ordinadors d'escriptori
Pantalles petites (usades per exemple en netbooks): 1024 x 600
Pantalles mitjanes: 1280 x 720 / 1280 x 800
Pantalles grans: amplada superior a 1400 píxels, exemple 1400 x 900 o 1600 x 1200.
Rellotges
Apple Watch: 42mm d'amplada de pantalla, viewport de 256px (calculat en relació a la mida de pantalla de l'iPhone)
360 Moto Watch: 218 x 281

1.2. Flexibilitat

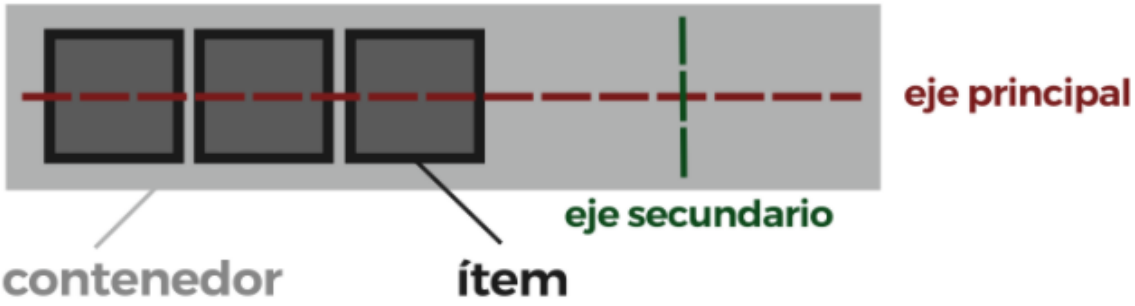
Tradicionalment, en CSS s'ha utilitzat el posicionament (static, relative, absolute...), els elements en línia o en bloc (i derivats) o els float. Això és basa en un sistema de creació de dissenys bastant tradicional que no encaixa amb els reptes que tenim avui dia: sistemes d'escriptori, dispositius mòbils, múltiples resolucions, etc.

En un disseny web adaptable hem de permetre que les pàgines siguin flexibles, de manera que els elements s'adaptin a l'espai disponible quan l'amplada de l'àrea de visualització es troba entre els punts d'interrupcions. **Això es pot aconseguir amb el model de caixes flexibles.**

Caixes flexibles (Flexbox)

- **CSS Flexible Box Layout** o **Flexbox**, és un mòdul de CSS que defineix un model de caixa CSS optimitzat. **Amb aquest model el disseny s'adapta de forma flexible a la pantalla on es mostra**.
- Aquest model no necessita utilitzar els mètodes tradicionals de posicionaments de caixes (static, relative, absolute, float…), si no que organitza els elements utilitzant contenidors flexibles.
- El propòsit d'introduir el model de disseny de caixes flexibles és proporcionar una manera més eficient d'organitzar un recipient (contenidor) amb els seus sub-elements, mitjançant l'alineació i distribució de l'espai. Per aconseguir-ho, Flexbox funciona amb dos eixos:
  - **eix principal**, que sol ser l'horitzontal.
  - **eix transversal** o vertical.

  Mitjançant aquests eixos, els elements s'organitzen dins de la caixa i es distribueixen en relació entre ells.
- **Hi ha un contenidor principal**, on els seus fills (*flex items*) es poden disposar en qualsevol direcció i poden "flexionar" les seves mides, ja sigui creixent fins a omplir l'espai no utilitzat o reduint-se per evitar el desbordament.



  - **Contenidor**: És l'element pare que tindrà al seu interior cadascun dels ítems

flexibles.
    - **Eix principal** (direcció principal): Els contenidors flexibles tindran una orientació principal específica. Per defecte, és en horitzontal (en fila).
    - **Eix secundari** (direcció secundària): Els contenidors flexibles tindran una orientació secundària, perpendicular a la principal. Si la principal és en horitzontal, la secundària serà en vertical, i viceversa. Una vegada esgotat l'espai a l'eix principal, els elements es poden anar ordenant en funció de l'altra eix.
  - **Item**: Cada un dels fills flexibles que tindrà el contenidor al seu interior.

```
Per exemple: Imaginem una pàgina web d'una botiga en línia, on tenim multitud de productes representats per una llista.

El contenidor podria ser la llista.
Els items podrien ser els productes.
Si els hi assignam propietats flexibles, els productes podrien adaptar-se a l'ample i/o alt de seu contenidor i distribuir-se per l'espai lliure de manera flexible, per exemple canviant el seu ample/alt.
```

Tipus de caixes flexibles

- "**flex**":
  - Es comporta com a bloc, és a dir, ocupa tot l'espai horitzontal disponible.
  - Amb "flex" establim les propietats a l'element pare (al contenidor). Es definirà posant: "display:flex".
- "**inline-flex**":
  - Es comporta com un element en-línia (ocupa només l'espai horitzontal necessari).

Exemple 1:

```
<!DOCTYPE html>
<html lang="es">
<head>
  <title>Exemple 1 Flex</title>
  <meta charset="UTF-8">
  <style>
    .caixa
     {
      display: flex;  /* Significa que els tres elements secundaris es converteixen en elements flexibles. */
      justify-content: space-between; /* distribueix l'espai entre i al voltant del items flex, al llarg de l'eix principal del seu contenidor */
     }
  </style>
</head>
<body>
  <div class="caixa">
     <div>Primer item del contenidor principal</div>
     <div>Segon item del contenidor principal</div>
     <div>Aquest és el tercer item <br> del contenidor principal <br> que ocupa més<br>d'una línia </div>
  </div>
</body>
</html>
```

Exemple 2:

```
<!DOCTYPE html>
<html lang="es">
  <head>
    <title>Exemple 2 Flex</title>
    <meta charset="UTF-8">
    <style>
      div {
          display: flex; /*Defineix el model flexible*/
          margin-bottom: 30px;
          margin-left: auto;
          margin-right: auto;
          flex-direction: row-reverse; /*ubica en ordre invers els
items a la seva declaració en el document.*/
          }
        p {
            width: 400px;
            border: black 2px solid;
            padding: 20px;
            background-color: beige;
          }
       </style>
   </head>
   <body>
      <div>
         <p> Lenguajes de marcas y sistemas de información</p>
         <p> Programación</p>
         <p> Base de datos</p>
         <p> Entornos de programación</p>
         <p> Sistemas informáticos</p>
     </div>
</body>
</html>
```

Una àrea del document que conté un "flexbox" s'anomena contenidor flex. Quan s'ha definit un contenidor flexible, els fills directes d'aquest contenidor es tornen "ítems flex".

Com amb totes les propietats de CSS, es defineixen alguns valors inicials, així que quan creiem un contenidor "flex" tots els "ítems flex" continguts es comportaran de la següent manera:

- Els ítems es despleguen sobre una fila (la propietat "flex-direction" per defecte és

"row").
- Els ítems comencen des del marge inicial sobre l'eix principal.
- Els ítems s'ajustaran per omplir la mida de l'eix secundari.

- **Propietats dels elements flexibles:**
  - **"flex-direction"**. Permet especificar l'eix prioritari del contenidor. Pot tenir els següents valors.
    - row: Disposa als elements fill de forma horitzontal o en fila. (Predeterminat).
    - row-reverse: Igual al anterior, però els ubica en ordre invers a la seva declaració en el document.
    - column: Disposició vertical o en columna.
    - column-reverse: Disposició vertical en sentit invers.
  - **"flex-wrap".** Estableix si els elements es posen en una sola línia o en varies.
    - no-wrap: Els elements se posen en una sola línia.
    - wrap: Els elements se posen en vàries línies si no caben en una sola línia.
    - wrap-reverse: Els elements se posen en vàries línies si no caben en una sola línia, però les línies se mostren de abaix cap a dalt.
  - **"flex-flow".** És una propietat composta i permet establir simultàniament les dues propietats "flex-direction" i "flex-wrap".
  - **"flex-basis".** La mida inicial dels elements flexibles ve determinada en principi pel seu contingut. Si hi ha espai suficient en la direcció principal, els elements s'eixamplen per mostrar tot el seu contingut en una sola línia. La propietat flex-basis estableix la mida inicial de l'element abans de que es reparteixi l'espai lliure. El valors que pot prendre són:
    - content: La mida inicial ve determinada pel contingut de l'element.
    - auto: La mida inicial ve determinada per les propietats "width i/o height".
  - **"flex-grow".** És el factor d'expansió dels elements flexibles. Aquesta propietat fa que els elements creixin fins ocupar tot l'espai disponible en la direcció principal.
    - Pren valores sencers que indiquen la proporció en que es reparteix l'espai.
    - Si tots els elements prenen el mateix valor, l'espai es reparteix a parts iguals.
  - **"justify-content"**. Defineix com el navegador distribueix l'espai entre i al voltant del items flex, al llarg de l'eix principal del seu contenidor. Admet els següents valors:
    - flex-start: Per ajustar els fills a l'inici del contenidor (predeterminat).
    - flex-end: Per ajustar els fills al final.
    - center: Centra els items.
    - space-between: Justifica els items.
    - space-around: Justifica els items deixant buit en els extrems.
  - **"align-items".** Permet alinear els elements a l'eix secundari. Els valors possibles són:

- flex-start: Col·locar els elements a la part superior de l'element pare.
- flex-end: Col·locar el elements a la part inferior de l'element pare.
- center: Les centra.
- baseline: Els alinea segons l'alineació base.
- stretch: Fa que ocupin tot l'alt del contenidor pare (en el cas de que la direcció sigui en forma de fila).

- "**Align-content**". Similar a l'anterior, però alineant els elements fill pel que fa a l'eix principal, és a dir, com es distribueixen les files d'elements en el contenidor pare, en conseqüència, només afecta si hi ha més d'una fila d'elements. Els seus valors poden ser: flex-start | flex-end | center | space-between | space-around | stretch.

Exemple 3: "display:flex" i "flex-direction" en columna.

```
<!DOCTYPE html>
<html lang="es">
  <head>
    <title>Columna</title>
    <meta charset="UTF-8">
    <style>
      div {
          margin-bottom: 20px;
          margin-left: 10%;
          margin-right: 10%;
          background-color: rgb(240, 136, 136);
          display: flex;
          flex-direction: column;
        }
      p {
          width: 400px;
          border: black 1px solid;
          margin: auto;
        }
       </style>
   </head>
   <body>
      <div>
         <p> Lenguajes de marcas</p>
         <p> Programación</p>
         <p> Base de datos</p>
         <p> Entornos de programación</p>
         <p> Sistemas informáticos</p>
     </div>
</body>
</html>
```

Exemple 4: "Flex-direction" en fila.

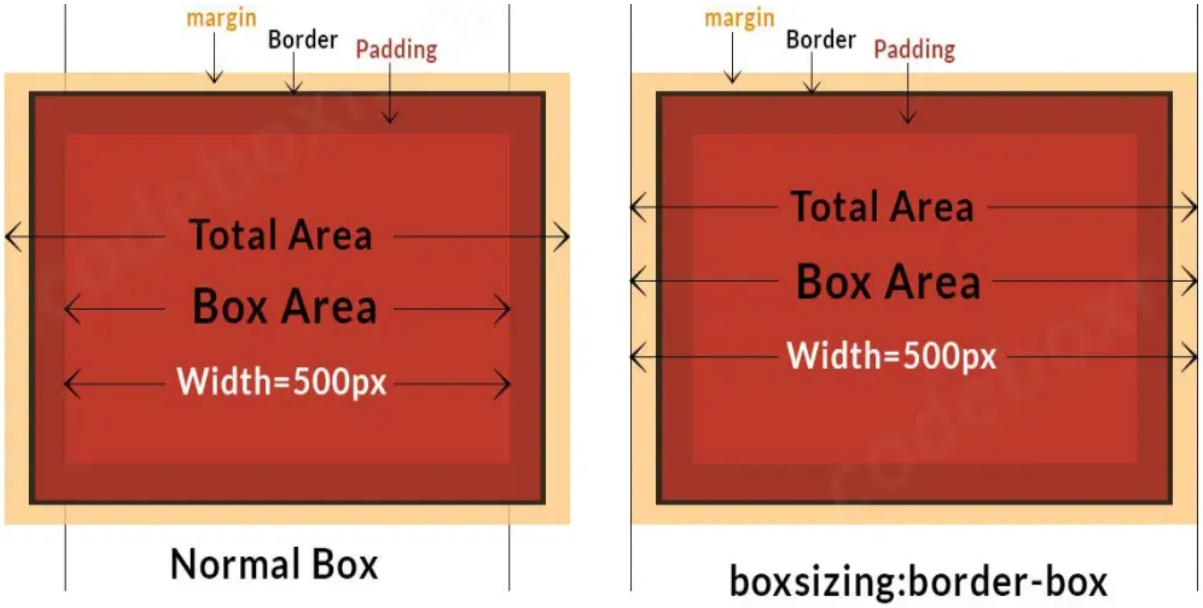
```
<!DOCTYPE html>
<html lang="es">
  <head>
    <title>Fila</title>
    <meta charset="UTF-8">
    <style>
      div {
          flex-direction: row;
          background-color: rgb(240, 136, 136);
          display: flex;
        }
      p {
          border: black 1px solid;
        }
       </style>
   </head>
   <body>
      <div >
         <p> Lenguajes de marcas</p>
         <p> Programación</p>
         <p> Base de datos</p>
         <p> Entornos de programación</p>
         <p>Sistemas informáticos</p>
     </div>
</body>
</html>
```

Propietat box-sizing

- Aquest propietat permet canviar el model de caixes que ve per defecte als navegadors.
- És la propietat que ens permet decidir com es calcularà la mida o grandària d'un element i forçar als navegadors a incloure el farcit i la vora a la mida declarada per la propietat "width".

```
box-sizing: border-box.

Amb això, indicam al navegador que prengui com la vora de la caixa l'àrea definida per qualsevol grandària que posem.
```

- Els valors disponibles són: "content-box" i "border-box":
  - "**content-box**": Per defecte, el valor de la propietat "box-sizing" es declara com a "content-box", que és el comportament és el que tenia el model de caixes tradicional.
  - "**border-box**": En canvi, si assignam el valor "border-box", el navegador inclourà el farcit i la vora com a part de l'amplada. El valor border-box s'utilitza per facilitar-nos la tasca de calcular i ajustar les dimensions dels elements perquè hi capin posat horitzontalment.



Exemple: Amb "box-sizing: border-box".

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scal
e=1.0">
    <title>Box-sizing</title>
    <style>
      * {
         margin: 0px;
         padding: 0px;
        }
      section {
        float: left;
        width: 65%;
        padding: 20px;
        margin-right: 5%
        background-color: #999999;
        box-sizing: border-box;
```

```
        }
      aside {
        float: left;
        width: 30%;
        padding: 20px;
        background-color: #CCCCCC;
        box-sizing: border-box;
       }
     p {
        text-align: center;
       }
   </style>
</head>
<body>
    <section>
       <p>Hola</p>
    </section>
    <aside>
       <p>Adiós</p>
    </aside>
</body>
</html>

Es declara la propietat "box-sizing" amb el valor "border-box" per
els dos elements. Amb això ja consideram el farcit quan calculam la
mida dels elements (65 + 5 + 30= 100).
```

Text adaptable

Quan declaram un tipus de lletra amb una grandària fixa, com per exemple 14 píxels, el text es mostrar sempre amb aquesta mida, independentment del dispositiu.

Per ajustar la grandària de la lletra al dispositiu ho podem fer utilitzant:

- **Proporcions escalables o adaptatives**:
  - "**em**":
    - És una unitat escalable que és igual a la mida de la font actual, per exemple, si la mida de la font del document es 16px, 1em és igual a 16px.
    - Són de naturalesa escalable, per lo que 2em seria igual a 32px.
    - La unitat "em" sempre depèn del seu element pare. Por exemple, si l'element body té una mida de font de 16px i un element fill té una font amb mida 1.3em, aquest text es mostrarà amb una mida un 30% més gran que la del body (20.8px).
    - Per tant "em" sempre ens proporciona valors relatius al "font-size" de

l'element.
- S'hauria d'aplicar a propietats que necessiten mantenir la proporcionalitat respecte al font-size d'un estil en concret.

Exemple:

```
<!DOCTYPE html>
<html lang="es">
<head>
    <title>Exemple 2 Flex</title>
    <meta charset="UTF-8">
    <style>
        .uno {
            font-size: 1em;
        }
        .dos {
            font-size: 2em;
        }
    </style></h
</head>
<body>
    <p class="uno">paràgraf a 1em</p>
    <p class="dos">paràgraf a 2em</p>
</body>
</html>
```

- "**rem**":
  - És molt similar a "em", amb la única diferència de que no és escalable, és a dir, que no depèn de l'element pare, si no de l'element arrel del documento (l'element HTML).
  - Això significa que si l'element HTML té una mida de font de 16px (com és per defecte), llavors 1rem, seria igual a 16px, i si volem aplicar una mida basada en rem a qualsevol element de la pàgina, no importarà quin sigui el mida de font que tingui associat aquest element, ja que 1rem sempre serà igual a 16 píxels tret que es modifiqui l'element arrel.

Exemple:

```
<!DOCTYPE html>
<html lang="es">
<head>
    <title>rem</title>
    <meta charset="UTF-8">
    <style>
        html {
            font-size: 16px;
        }
        div {
           font-size:20px;
        }
        .text {
           font-size: 2rem;
        }
    </style>
</head>
<body>
    <p>Amb font per defecte</p>
    <div>
       <p class="text">Això és una prova amb "rem", és dues veg
ades la font per defecte</p>
       <p>sense "rem", és la mida del "div"</p>
    </div>
</body>
</html>
```

- **Proporcions relatives a la mesura de la pantalla de visualització (*viewport*):**
  - "**vh**" i "**vw**": Permeten crear mides de font (amb "font-size") que s'ajustin a l'ample o alt de les pantalles de visualització:
    - 1vh = 1% de l'alçada del viewport.
    - 100vh = alçada del viewport.
    - 1vw = 1% de l'ample del viewport.
    - 100vw = ample del viewport

    - Són molt útils per texts molt grans a pantalles gegants i texts petits a pantalla petites, i per establir la mesura d'una imatge de fons, marc contenidor o una caixa de text.

    Aquests unitats de mesura, també es poden utilitzar per les vores, marges, etc.

Imatges adaptables

- Quan treballem amb imatges a les webs adaptables ens podem trobar amb 2 problemes:
  - D'una banda, hi pot haver imatges que a mida que es van fent més grans o més petites, no queden bé.
  - En altres casos, la imatge es veu bé, però carregar imatges de gran resolució en dispositius petits no és el més adequat. Podem accelerar el seu temps de càrrega reduint aquesta resolució per una altra de més adequada al dispositiu.

- **Podem indicar la mida de la imatge en percentatge**

Exemple 1: L'amplada dels elements "img" dins "article" del document, serà del 100% i per tant les imatges tendran un ample igual al contenidor.

```
article img {
   width: 100%
}
```

Al assignar un percentatge a l'ample de la imatge, es força al navegador a calcular la mida de la imatge d'acord a la mida del seu contenidor.
També podem declarar valors menors a 100%, però la mida de la imatge sempre serà proporcional a la mida del seu contenidor, lo que significa que si el contingut s'expandeix, la imatge es podria presentar amb una mida més grossa que l'original.

Per evitar la problemàtica anterior, podem utilitzar: "max-width" i min-width". Mitjançant aquestes regles indicarem que la imatge tengui un ample amb píxels, però com a màxim estigui al 100% de l'amplada del contenidor. Per exemple, si el contenidor té 300px d'amplada i la imatge es de 400px, la imatge ocuparà els 300px, si el contenidor té una amplada de 400px, la imatge tendrà 400px, és dir el 100% de l'ample del contenidor, però si el contenidor té una amplada de 1200px, la imatge serà de 400px. És a dir, la regla deixa que la imatge s'expandeixi fins la seva mida original. La imatge serà tan ample com el seu contenidor a no ser que el contenidor tengui una mida més gran que la mida original de la imatge.

Exemple 2: En aquest exemple empram la propietat "box-sizing", el % a l'imatge.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>Imatges adaptables</title>
    <style>
      * {
          margin: 0px;
          padding: 0px;
        }
      section {
         float: left;
         width: 100%;
         padding-right: 260px;
         margin-right: -240px;
         background-color: #f3d0d0;
         box-sizing: border-box;
       }
      aside {
         float: left;
         width: 240px;
         padding: 20px;
         background-color: #97b6f0;
         box-sizing: border-box;
        }
       p {
         text-align: center
       }
       section img {
           max-width : 100%;
       }
   </style>
</head>
<body>
    <section>
       <img src="simpson.jpg">
    </section>
    <aside>
       <p>Imatge dels Simpsons</p>
    </aside>
</body>
</html>
```

- **Canviar la resolució**
  - Si utilitzam la sintaxi ja coneguda per inserir una imatge (<img src="imatge.png">) no podrem canviar de resolució, ja que només inserim una imatge amb una resolució determinada (el navegador només apunta a un arxiu).
  - Hi ha una altra sintaxi, si afegim els atributs: "srcset" i "sizes", que ens permetran proporcionar imatges addicionals i suggerències per ajudar al navegador triar la imatge correcta.

Exemple:

```
<img srcset="imatge1.png 320w, imatge2.png 480w, imatge3.png 800w" sizes="(max-width: 320px) 280px, (max-width: 480px) 440px, 800px" src="imatge.png">


"srcset":
- Defineix el conjunt d'imatges que el navegador podrà triar i la mida de cada imatge.
- Cada conjunt d'informació d'una imatge està separat de l'anterior per una coma.
- El conjunt d'informació és: un nom d'arxiu d'imatge, una amplada de la imatge en píxels (atenció, la unitat és "w", no "px"; w ens indica la mida real de la imatge).

"sizes":
- Indica quina mida d'imatge seria millor triar quan es compleixen certes condicions dels mitjans (max-width: 600px, per exemple).
- Després d'un espai, s'indica l'ample que la imatge ocuparà quan la condició sigui certa.
```

- **La propietat "height" en imatges *responsives***
  - Normalment solem guiar-nos per l'amplada dels dispositius i contenidors per ubicar-hi el nostre contingut. Poques vegades en disseny web el valor height és un element a tenir en compte. El que es fa normalment és definir un valor auto a aquesta propietat.