ceedcv
CENTRE ESPECÍFIC
D´EDUCACIÓ A DISTÀNCIA DE
LA COMUNITAT VALENCIANA

UD03. Responsive
Conceptos básicos

Diseño de interfaces WEB
2º Curso

- Responsive
- viewport
- media queries

ceedcv
CENTRE ESPECÍFIC
D´EDUCACIÓ A DISTÀNCIA DE
LA COMUNITAT VALENCIANA

Responsive

Los contextos de trabajo han cambiado:

- De un ordenador se ha pasado a móviles, tabletas, televisores...
- Las páginas web son más potentes y se les exige más. Entre otras cosas generar más información, parte de ella en otros entornos físicos, como el papel.
- La web se ha convertido en una herramienta de trabajo a nivel mundial y hay que dar cabida en ella a todo tipo de usuarios incluidos aquellos que tienen algún tipo de minusvalía (especialmente visual). Hay que dotarla de accesibilidad

Dispositivos móviles

El problema

El problema con los dispositivos móviles es el tamaño de la pantalla.

- En general, un menor tamaño de pantalla viene asociado con un menor número de píxeles, por lo que si trabajamos con elementos porcentuales, no es lo mismo un 10% de 2000px (200px) que de 400px (40px). En el segundo caso es muy probable que no pueda leerse cómodamente.
    - Hay que recordar que un zoom no cambia el tamaño de los elementos, únicamente acerca o aleja la cámara. Así que si, por ejemplo, una columna resulta muy estrecha y sólo caben dos letras, por hacer un zoom no vamos a hacer que quepan más letras, sólo las veremos más grandes
- Incluso con el mismo número de píxeles, el tamaño de la pantalla sigue siendo pequeño (se utilizan densidades muy altas) por lo que los elementos se seguirán viendo pequeños, teniendo poca legibilidad

ceedcv
CENTRE ESPECÍFIC
D'EDUCACIÓ A DISTÀNCIA DE
LA COMUNITAT VALENCIANA

Viewport

Cuando hablamos del *viewport* estamos haciendo referencia al área de la ventana del navegador.

Viewport

Layout vs visual

Con este nuevo concepto podríamos simplificar el problema diciendo que lo que tenemos es un *viewport* muy pequeño, muy estrecho y que lo que necesitamos es que sea más grande. Para ello, en los dispositivos móviles el concepto *viewport* cambia, apareciendo dos *viewports*: el *layout viewport* y el *visual viewport*.

- El *layout viewport* es el *viewport* a partir del cual los ficheros CSS tomaran las referencias. El elemento *<html>* del documento se adaptará a ese *viewport*. En general, tiene un valor mayor que el de la pantalla del dispositivo, para asegurar que el comportamiento sea el diseñado teniendo en mente un navegador de escritorio (es el *viewport* grande que buscábamos)

  Cada navegador móvil configura su tamaño de *layout viewport* como considera óptimo y no cambia (¿casi?) nunca.

- El *visual viewport* es la parte de la página que actualmente se muestra en pantalla. El usuario puede hacer scroll sobre ella o zoom (en cuyo caso cambia el tamaño de *visual viewport*).

ceedcv
CENTRE ESPECÍFIC
D'EDUCACIÓ A DISTÀNCIA DE
LA COMUNITAT VALENCIANA

Viewport

visual viewport

Layout viewport

Tamaño definido por cada navegador móvil

ceedcv
CENTRE ESPECÍFIC
D'EDUCACIÓ A DISTÀNCIA DE
LA COMUNITAT VALENCIANA

Viewport

visual viewport

Layout viewport

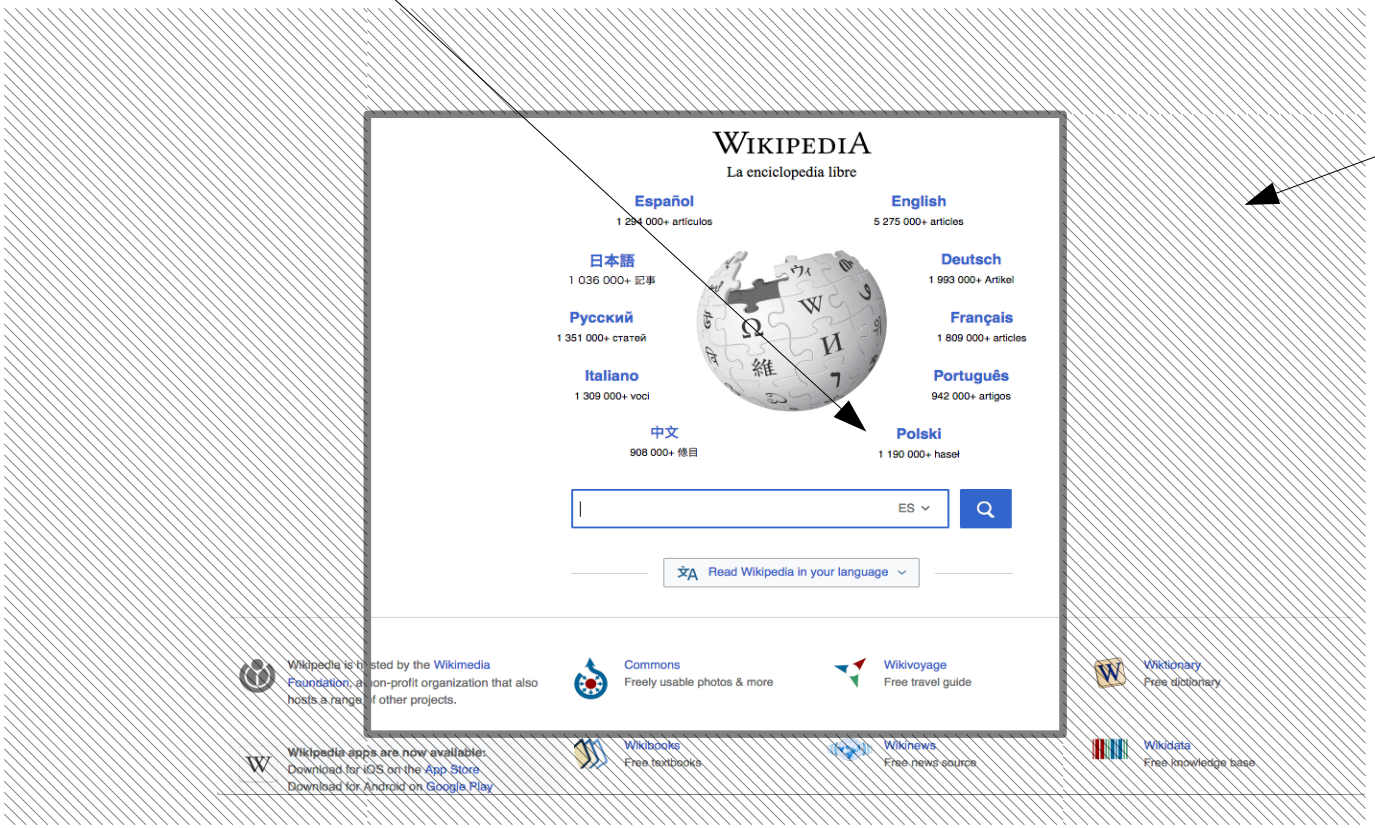
Se ha aplicado un scroll al visual viewport

ceedcv
CENTRE ESPECÍFIC
D'EDUCACIÓ A DISTÀNCIA DE
LA COMUNITAT VALENCIANA

Viewport

visual viewport

Layout viewport



Se ha aplicado un zoom (alejarse) al visual viewport (se ve más area del layout)

Viewport

Muchos móviles muestran inicialmente las página haciendo un *zoom out*, es decir, alejándose, de tal manera el tamaño del *visual viewport* es el tamaño del *layout viewport*.

- Las dimensiones (el ancho en realidad) del *layout* siempre es el mismo. Si giramos el móvil (a apaisado), lo que cambian son las dimensiones del *visual viewport*. Al mantenerse el *zoom out* máximo, al tener más ancho en el *visual viewport*, hará falta alejarse menos y por lo tanto la web se verá un poco más grande.

Viewport

Con estos dos *viewports* podemos hacer que la página en un móvil se comporte como la teníamos diseñada con un ordenador de escritorio, pero aún quedan más problemas. Aunque la página se comporte igual, muy probablemente se verá muy pequeña.

La solución consiste en diseñar no sólo para escritorio, sino también para móviles (pensando en el orden de magnitud en píxeles del *visual viewport*)

El problema es que ese diseño se adaptará al *layout viewport* (el grande). Para definir un ancho de trabajo y de paso unificar los diferentes ancho del *layout viewport* podemos usar una etiqueta meta con el nombre *viewport*.

viewport

A esta etiqueta podemos aplicarle varios contenidos:

- width: *valor-ancho*. Con un valor numérico le indicamos al navegador móvil cual ha de ser el tamaño del *layout viewport*. Con el valor *device-width* asignamos el valor del ancho del *visual viewport*
- *height: valor-alto. F*uncionalidad similar al *width* pero con respecto al alto
- *initial-scale*: *valor-numérico*. Se le da la escala inicial al *layout viewport*. La relación de escala es *visual : layout*. Poniéndolo a 1 conseguimos el mismo efecto que con *device-width*. Si lo ponemos a 2 (por ejemplo) hacemos que el *visual* sea el doble que el *layout*, con lo que el navegador necesitará ampliar, acercarse.
- *minimum-scale/maximum-scale*: escala máxima y mínima que podemos aplicar. Si lo dejamos todo en 1 no permitiremos al usuario poder hacer zooms.
    - El concepto escala es un concepto como mínimo contradictorio. El *layout* no se modifica salvo en la carga, luego no tiene sentido indicar como el usuario puede o no modificarlo. Posiblemente un mejor nombre para la propiedad sería *minimun/maximun-zoom*

Unidades viewport

La aparición del concepto *viewport* ha hecho que a su vez aparezcan otro tipo de unidades relativas relacionadas con él:

- *vw*: 1/100 del ancho del viewport.
- *vh*: 1/100 del alto del viewport.
- *vmin*: 1/100 del lado más pequeño
- *vmax*: 1/100 del lado más grande

  Su uso puede parecer innecesario ya que indicar que el ancho o alto del elemento <html> es el 100% es lo mismo trabajar con relación al *viewport*. Pero sin embargo, si queremos hacer que un elemento hijo de html trabaje con relación a esas dimensiones mediante porcentajes, debemos ir arrastrando ese 100% por todos ellos, ya que el porcentaje se toma con relación al elemento padre.

media queries

En este punto parece claro que es necesaria la creación de dos versiones de las páginas web: las adaptados al escritorio y las adaptadas a dispositivos móviles.

Ambas dos deben tener la misma estructura (el mismo documento HTML) y ha de ser el CSS el que varíe en función del dispositivo en el que se va a renderizar.

El problema se centra en como cargar un CSS u otro en función del dispositivo, problema que se soluciona con la propiedad *media*.

Desde CCS2 las páginas pueden ser sensibles al dispositivo en el que van a ser renderizados, gracias a la propiedad *media* del tag *link*.

Entre los dispositivos que podemos elegir tenemos: *screen* (pantallas de ordenador), *print* (papel), *braille* (maquinas braille), *tty* (consolas de texto), *handled* (dispositivos de mano ¿móviles?, pantallas pequeñas), *aural* (sintetizadores de voz), *tv* (televisiones) y *projection* (proyectores)

```
<link rel="stylesheet" href="print.css" media="print">
```

media queries

Esta definición es un buen punto de partida, pero le siguen faltando cosas: por ejemplo, no todos los móviles son iguales.

A partir de CSS3 las posibilidades para los valores media se amplían permitiendo que no únicamente nos centremos en el dispositivo, también en las especificaciones del mismo. Podemos realizar una consulta (una *query*) al dispositivo para decidir que CSS cargamos. Para ello podemos hacer uso de *and* y *or (*representado con *,)* y de la consulta de varias variables, entre las que podemos consultar:

width (el ancho del *viewport*), *height* (el alto), *device-width* (el ancho del *layout viewport*), *device-height* (el alto del *layout viewport*), *orientation*, *aspect-ratio* (relación entre el ancho y el alto), *color*, *resolution* (densidad en píxeles)...

La gran mayoría de estas variables pueden ir prefijadas con *max* y *min*, de manera se puede indicar que el máximo o el mínimo sea el valor asignado. Por ejemplo:

```
<link rel="stylesheet" media=”screen and (max-width: 480px) and (resolution: 163dpi)" href="prueba.css”>
```

media queries

Además de poder crear un CSS diferente y enlazarlo, es posible incluirlos con la etiqueta *@media* dentro del CSS

```
@media screen and (max-width: 480px) and (resolution: 163dpi) {
    body {
        background: red;
    }
}
```

O incluso dentro de un *import*

```
@import (“prueba.css”)screen and (max-width: 480px) and (resolution: 163dpi);
```

Diseño adaptable

Otra de las técnicas recomendables es no usar *float* y utilizar *display* con valores *inline-block*.

El *inline-block* define un tipo de elemento que es una combinación de *inline* y *block*, de manera que aunque internamente es un bloque, su relación con el resto es como si fuera inline, lo que le permite posicionarse uno al lado del otro.

> Por ejemplo,un ejemplo de aplicación podría ser el diseño que surge cuando queremos visualizar una página con miniaturas de fotos a modo de rejilla, que vaya cambiando su disposición en función del tamaño de la ventana del navegador

Otra de las opciones, posiblemente más sencilla y potente es usar el potente contenedor *flexbox*.

Diseño adaptable

De manera general para hacer que una página sea adaptable conviene:

- No usar px. Usar porcentajes o unidades relativas al *viewport*
- Limitar los anchos (o altos, aunque esto es más raro) con *max-width*
- Evitar el uso de *position absolute* y *fixed*
- Evitar que las imágenes de fondo que no deben repetirse, lo hagan