DISEÑO DE INTERFACES WEB UD-2

RESPONSIVE DESIGN

ÍNDICE

[RESPONSIVE DESIGN 3](#_Toc148995926)

[A. RESPONSIVE 3](#_Toc148995927)

[B. DISPOSITIVOS MÓVILES 3](#_Toc148995928)

[C. HERRAMIENTAS PARA DESARROLLADORES (VISIÓN DE DISPOSITIVOS) 3](#_Toc148995929)

[D. VIEWPORT 4](#_Toc148995930)

[1. Layout viewport 5](#_Toc148995931)

[2. Visual viewport 5](#_Toc148995932)

[E. <meta name=”viewport”> 6](#_Toc148995933)

[F. UNIDADES VIEWPORT 9](#_Toc148995934)

[G. MEDIA QUERIES 10](#_Toc148995935)

# RESPONSIVE DESIGN

## 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 discapacidad (especialmente visual). Hay que dotarla de accesibilidad.

## DISPOSITIVOS MÓVILES

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 sí, 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.

## HERRAMIENTAS PARA DESARROLLADORES (VISIÓN DE DISPOSITIVOS)

Si abrimos cualquier página en un navegador, con la vista de “Herramientas para desarrolladores” podemos activar la visualización para dispositivos móviles pulsando en el icono:



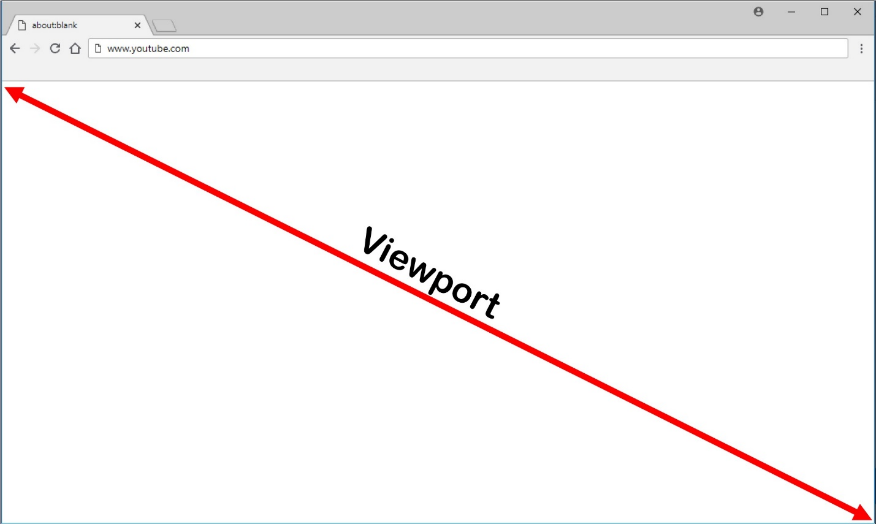
Gracias a estas herramientas podemos ver el aspecto que tiene la página web en diferentes tipos de dispositivos:

Interfaz de usuario gráfica, Sitio web

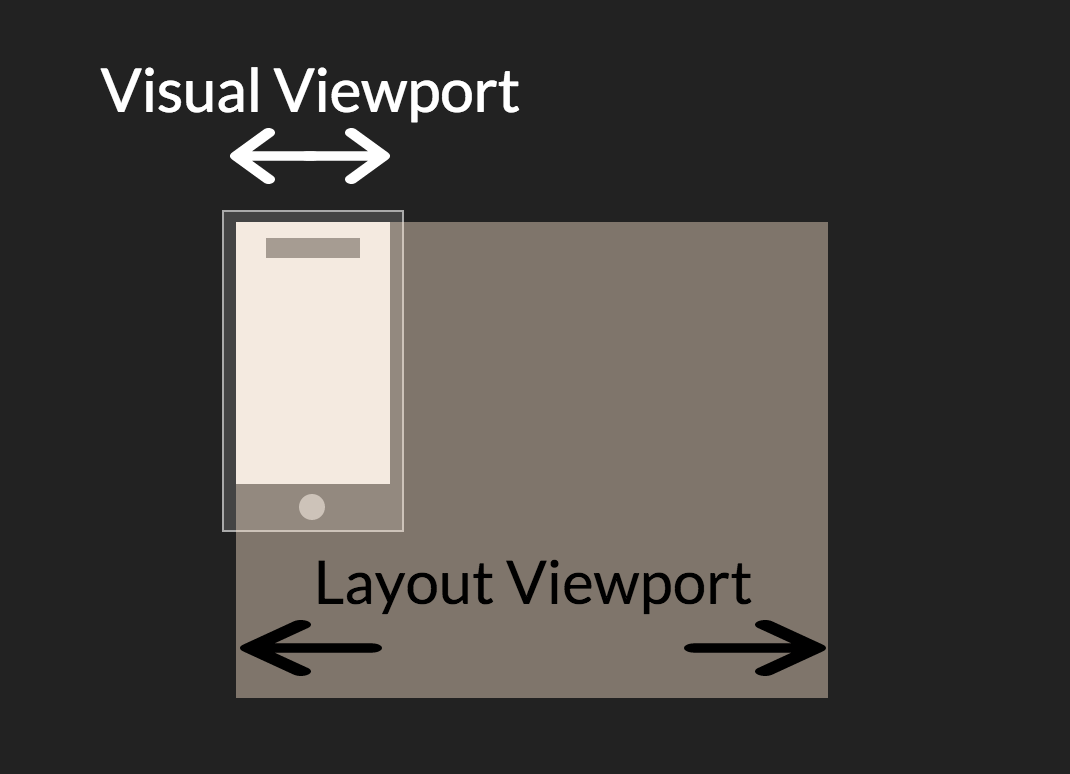
Descripción generada automáticamente

## VIEWPORT

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



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.



1. Layout viewport

Es el viewport a partir del cual los ficheros CSS tomarán 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.

Cada navegador móvil configura a su tamaño de layout viewport como considera óptimo y no cambia nunca.

1. Visual viewport

Es la parte de la página que actualmente se muestra en pantalla. El usuario puede hacer scroll sobre ella o zoom (en este caso cambiará el tamaño del visual viewport).

Muchos móviles muestran inicialmente las páginas 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 son las mismas. Si giramos el móvil, 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.

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 adptará al layout viewport (el grande). Para definir un ancho de trabajo y de paso unificar los diferentes anchos del layout viewport podemos usar una etiqueta meta con el nombre viewport.

## <meta name=”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. Funcionalidad 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. Poniendo a 1 conseguimos el mismo efecto que con device-width. Si lo ponemos a 2 hacemos que el visual sea el doble que el layout, con lo que el navegador necesitará ampliar, acercarse.
* mínimum-scale / máximum-scale: escala máxima y mínima que podemos aplicar. Si lo dejamos todo en 1 no permitiremos al usuario poder hacer zooms.

Vamos a partir de la siguiente web:

Texto

Descripción generada automáticamente con confianza media

Cuyo código HTML es el siguiente:

Texto

Descripción generada automáticamente

Si ahora añadimos la etiqueta meta con un valor de contenido de ancho de 400px lo que haremos es que la web se replantee sobre un ancho de esa cantidad.

Texto

Descripción generada automáticamente

Pantalla de computadora con letras

Descripción generada automáticamente con confianza media

En el siguiente ejemplo eliminaremos la parte de “historia”:

Texto

Descripción generada automáticamente

Si usamos ahora el initial-scale=1 conseguiremos lo siguiente:

Texto

Descripción generada automáticamente

Imagen que contiene Esquemático

Descripción generada automáticamente

Cada píxel de CSS es igual a un visual pixel. Si ponemos un 2:

Texto

Descripción generada automáticamente

Escala de tiempo

Descripción generada automáticamente

El layout ahora ocupa la mitad que antes y el efecto que se ha conseguido es que el zoom se acerca y las letras se ven más grandes.

## 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 que 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, y 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 adaptadas al escritorio y las adaptadas a dispositivos móviles.

Ambas 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 cómo cargar un CSS u otro en función del dispositivo, problema que se soluciona con la propiedad media.

Desde CSS2 las páginas pueden ser sensibles al dispositivo en el que van a ser rentabilizadas, gracias a la propiedad media del tag link.

Entre los dispositivos que podemos elegir tenemos:

* screen (pantallas de ordenador).
* print (papel).
* braille (máquinas braille).
* tty (consolas de texto).
* handled (dispositivos de mano).
* aural (sintetizadores de voz).
* tv (televisiones).
* projection (proyectores).



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 sino también en las especificaciones de este. Podemos utilizar una consulta (una query) al dispositivo para decidir qué CSS cargamos. Para ello podemos hacer uso de and y or (representando con ‘,’) y de la consulta de varias variables, entre las que podemos consultar:

* width (el ancho del viewport).
* height (el alto del viewport).
* 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 que se puede indicar que el máximo o el mínimo sea el valor asignado. Por ejemplo:



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

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza media

O incluso dentro de un import:



Para ver su funcionamiento real vamos a partir de este HTML:

Texto

Descripción generada automáticamente

Y este CSS:

Texto

Descripción generada automáticamente

Que producen este resultado:

Imagen que contiene Teams

Descripción generada automáticamente

Ahora vamos a crear una regla que haga pasar de una disposición en columna a una disposición en rejilla.

Texto

Descripción generada automáticamente

Con esto conseguimos lo siguiente:

Imagen que contiene reina, cuarto

Descripción generada automáticamente

Además, ahora en función de la anchura del navegador tendremos más o menos elementos por fila:

Imagen que contiene cuarto

Descripción generada automáticamente

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Ahora queremos que cuando quede una única columna consigamos que no quede el resultado de la siguiente manera sino completamente centrado en pantalla:

Interfaz de usuario gráfica, Aplicación, Teams

Descripción generada automáticamente

Para ello haremos uso de las media-queries:

Texto

Descripción generada automáticamente

Esta regla se aplicará sobre todos los dispositivos que tengan un ancho máximo de 750px y colocará los elementos de clase bloque (los figures) como elementos en bloque con un ancho auto.

El resultado será el siguiente:

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Como regla las media-queries se deben situar al final del documento CSS para evitar que haya sobreescrituras de las propiedades CSS.

Otra manera de conseguir la rejilla es haciendo que el contenedor de las figuras sea flexible con un flujo de elementos en fila (row) y subdiviendose en n-filas necesarias (wrap).

Texto

Descripción generada automáticamente

Para la media-query tendremos que afectar a ese flujo pasándolo a columna y sin wrap:

Texto

Descripción generada automáticamente

Otro ejemplo de uso sería un menú de cabecera:

Pantalla de computadora con letras

Descripción generada automáticamente con confianza media

A partir de este HTML vamos a intentar crear un menú desplegable, para ello vamos a comenzar con el siguiente CSS:

Texto

Descripción generada automáticamente

Importante tener en cuenta la última regla que hará que el primer elemento de todos no sea visible al lanzar el documento:

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente

Ahora vamos a crear una media-query para que en un tamaño móvil ese elemento “menú” sea visible y se comporte como un menú desplegable:

Texto

Descripción generada automáticamente

Esto hace que los elementos ul pasen a tener un display: none, es decir, pasan a ser invisibles y ese “menú” sí que es visible ahora:

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Ahora para otorgarle funcionalidad de menú desplegable debemos irnos a JS:

Texto

Descripción generada automáticamente

Lo primero que hacemos es añadir un evento de escucha al objeto window para que en el momento de carga de la página lance la función arrancaMenu() que lo que hace es capturar el elemento a de clase .menu-responsive y añadirle un evento de escucha de tipo click que lanza la función despliegaMenu().

Esta función despliegaMenu() lo que va a hacer es coger el listado ul que se encuentra dentro del nav y añadirle una clase si no la tiene o quitársela si ya la tiene (toggle).

Ahora tenemos que crear en el CSS la clase .desplegar:

Texto

Descripción generada automáticamente

Ahora si hacemos click sobre el menú, se nos desplegarán las opciones:

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Para mejorar el visionado de las opciones podemos hacer que los li también se comporten como elementos en bloque:

Texto

Descripción generada automáticamente

De esta manera conseguiremos el efecto siguiente:

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente