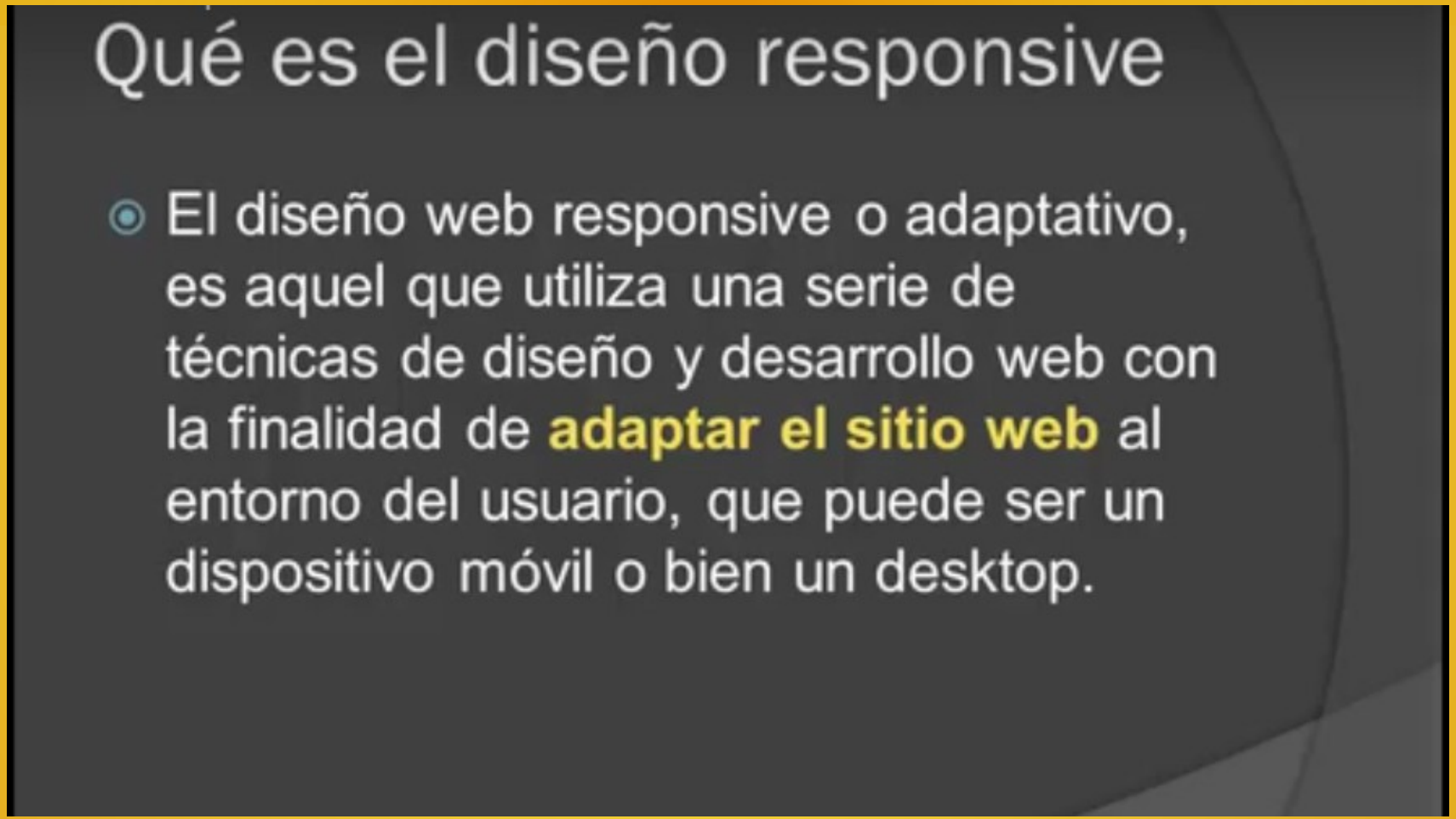
DISEÑO WEB RESPONSIVE



DISEÑO WEB RESPONSIVE

En qué se basa (teóricamente) el responsive design

- **Media queries de CSS3**. Básicamente lo que hacen los media queries es aplicar un estilo CSS u otro en función del ancho del navegador. Así, podemos decir que si el ancho de pantalla es menor a 480px el tamaño de fuente sea uno y si es mayor que sea otro.

- **Diseño web líquido o fluido**. Si hasta ahora habíamos utilizado tamaños fijos para determinar los elementos, como por ejemplo 960px, ahora lo haremos con valores porcentuales, como 80% o 90%. Esto nos va a servir para hacer que si por ejemplo navegamos con un smartphone la barra lateral en se vaya debajo del contenido central dándole un ancho del 100%.

- **Imágenes líquidas o fluidas**. Las imágenes fluidas son aquellas que escalan según el ancho del navegador a fin de adaptarlas al dispositivo en cuestión. Esto también lo veremos con otros archivos multimedia, como son los vídeos.

DISEÑO WEB RESPONSIVE

Y por tanto los tres pilares básicos para una web responsive son:

- Los media queries
- El viewport
- El layout líquido o fluido

DISEÑO WEB RESPONSIVE

Resolución de pantalla vs. ancho de ventana del navegador

- Hemos de **distinguir** entre la resolución de pantalla y el ancho de ventana del navegador.
- Ejemplo navegador PC.

El ancho de ventana de navegador como máximo podrá ser el tamaño de la resolución de pantalla del dispositivo.

DISEÑO WEB RESPONSIVE

Compara las siguientes web con diseño responsive

- http://www.pcbox.com/
- http://veign.com/?ckattempt=1

Y estas otra que no usan un diseño adaptativo completo

- http://www.appinformatica.com/index2.php
- https://www.amazon.es

Los media queries

- Con los media queries lo que hacemos básicamente es decidir cómo se va a mostrar una página web en función de los parámetros que indiquemos.
- En el caso tradicional, nos encontrábamos con dos versiones de una web, la que era para pantalla o screen y la que era para imprimir o print.

Los media queries

Ejemplo media query tradicional

```
@media screen {
    body { width: 70%; }
}
@media print {
    body { width: 100%; }
}
```

Los media queries

- Los media queries de CSS3 llevan la idea de los media queries tradicionales un paso más allá para **adaptar el diseño web a los diferentes dispositivos**.
- Tendremos sobre todo en cuenta el **ancho de la ventana del navegador**, así como la resolución del dispositvo.

Los media queries

Ejemplo Media Queries de CSS3

- Ejemplo de media queries de CSS3:

```
@media screen and (min-width:320px) {
     /* Código CSS */
}
```

- Las condiciones indicadas son las siguientes:
  - **screen** - indica que el dispositivo de salida es una pantalla (y no una impresora u otro dispositivo).
  - **min-width:320px** – indica que el ancho de ventana ha de ser como mínimo de 320px.

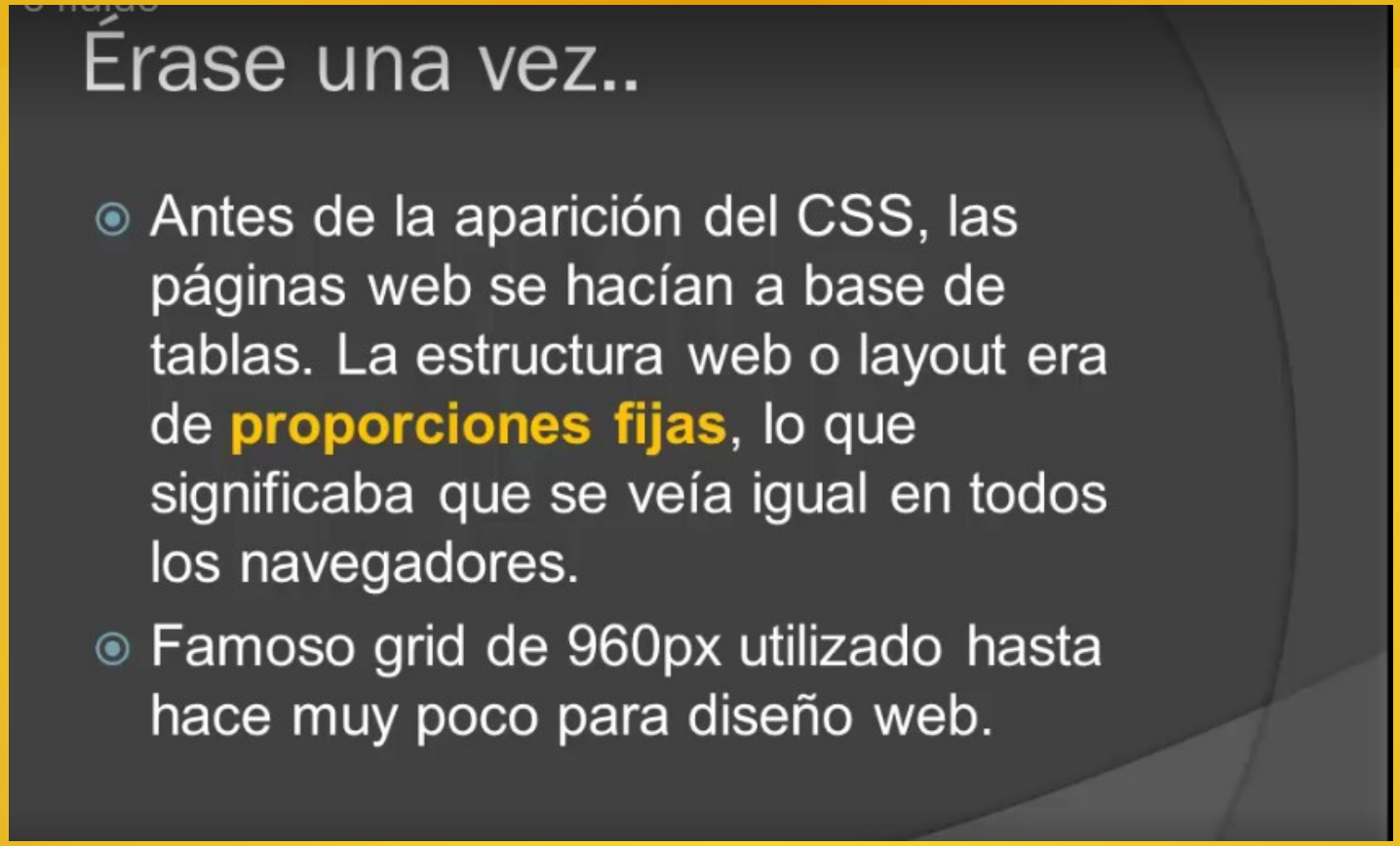
Los media queries

Uso de AND en las condiciones

- Para separar las diferentes condiciones utilizamos la palabra and y podemos especificar tantas como creamos necesarias. Ejemplo:

```
@media screen and (min-width: 480px)
and (max-width: 767px) {
    /* Aquí nuestro CSS */
}
```

Diseño web fluido



Diseño web fluido

Aparecen las mega pantallas..

- Con la aparición de pantallas de mayor resolución, el **grid de 960px se fue quedando pequeño**, con lo que empezaron a aparecer otros que permitían diseñar para una resolución mayor.
- El problema entonces era que había usuarios que **seguían utilizando pantallas de baja resolución**, con lo que tenían que utilizar la barra lateral horizontal para desplazarse por el contenido, lo cual no proporcionaba precisamente una gran experiencia de usuario.

Diseño web fluido

La solución

- Para evitar todos estos problemas, se llegó a la conclusión que era mejor dejar de **dejar de diseñar para tamaños fijos** en píxeles.
- Así, se llevo al modelo de layout líquido, que consiste en **utilizar porcentajes**.

Qué es el diseño web fluido

- El diseño web fluido o líquido **se adapta al ancho de ventana** que tenemos disponible del navegador, de modo que **cambia las proporciones** en función del ancho de ventana.
- En el caso del diseño web líquido lo que utilizaremos serán **porcentajes**.

Diseño web fluido .Ejemplo

Diseño web fluido

CSS básico del ejemplo

```
@media screen and (max-width: 479px)
{
     #main { width:100%; }
     #sidebar { width:100%;}
}
@media screen and (min-width: 480px)
{
     #main { width:70%; }
     #sidebar { width:30%; }
}
```

Diseño web fluido

Pros y contras del layout fluido

PROS

- **Se adapta a toda resolución de pantalla** de una manera muy amigable.
- Ofrece una **gran experiencia de usuario**

CONTRAS

- Hay que tener en cuenta un montón de cosas para que todo se vea bien.
- Por ello, es **bastante más difícil de diseñar y programar**.

El viewport

- El Viewport es una de las etiquetas más representativas de la web móvil, que nos permite configurar cómo debe interpretar una página el navegador web para móviles.
- Se trata de una etiqueta META a través de la cual se puede definir:
  - si el usuario puede o hacer zoom en la página
  - qué zoom inicial debe aplicarse al entrar
  - la anchura que debe simular la pantalla del dispositivo.

El viewport

El problema..

- Prácticamente todos los navegadores de smartphones al entrar a un sitio analizan el tamaño total y **lo escalan para que se muestre completo en la pantalla**.
- Así, si diseño mi web para 320px con los media queries de CSS3, el iPhone 4S la escalará a 960px, que es su resolución máxima.

El viewport

- El viewport es un atributo del tag <*meta*> que debe incluirse entre las etiquetas <*head*> de un documento HTML , siguiendo este patrón básico:

```
<meta name="viewport"
content="width=device-width"/>
```

- Con sólo agregar esto, no se escalará nuestro contenido.

El viewport

Propiedades del viewport

Atributo	Valores	Descripcion
width	Valor integral (en pixels) o constante device-width	Define el ancho del viewport
height	Valor integral (en pixels) o constante device-height	Define el alto del viewport
initial-scale	Cualquier numero real de 0.1 en adelante. 1 representa no escala	Define la escala inicial del viewport
user-scale	"yes"/ "no"	Define los permisos para que el usuario pueda escalar el viewport
minimum-scale	Cualquier numero real de 0.1 en adelante. 1 representa no escala	Define la escala minima del viewport
maximum-scale	Cualquier numero real de 0.1 en adelante. 1 representa no escala	Define la escala máxima del viewport

El viewport

Propiedades del viewport

Atributo	Valores	Descripcion
width	Valor integral (en pixels) o constante device-width	Define el ancho del viewport
height	Valor integral (en pixels) o constante device-height	Define el alto del viewport
initial-scale	Cualquier numero real de 0.1 en adelante. 1 representa no escala	Define la escala inicial del viewport
user-scale	"yes"/ "no"	Define los permisos para que el usuario pueda escalar el viewport
minimum-scale	Cualquier numero real de 0.1 en adelante. 1 representa no escala	Define la escala minima del viewport
maximum-scale	Cualquier numero real de 0.1 en adelante. 1 representa no escala	Define la escala máxima del viewport

El viewport

Qué es el device-width

- Ejemplo:

```
<meta name="viewport"
content="width=device-width"/>
```

- El **device-width** es equivalente al 100% del ancho de la pantalla del dispositivo, independiente de su tamaño, posición o resolución.

El viewport

El height en el viewport

- El alto de la pantalla también es configurable con las mismas propiedades a través del atributo height , aunque – salvo condiciones muy especificas- **no es necesario definirlo**.
- Esta propiedad se asignará de manera automática a través del scroll.

El viewport

Escala del viewport

- Podemos controlar la escala de la vista con el atributo **initial-scale**. Por ejemplo con este código el sitio se mostrará al doble de su tamaño original:

```
<meta name="viewport"
content="width=device-width; initial-
scale=2"/>
```

El viewport

Opción de zoom

- Con el viewport se puede también bloquear la opción de zoom mediante el atributo user-scalable. Veamos un ejemplo:

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

- Esto en la mayoría de casos no es nada recomendable, pues al usuario no le gusta que le impidamos hacer zoom.

El viewport

Formato recomendado

- El formato recomendado para la mayoría de casos es el siguiente:

```
<meta name="viewport"
content="width=device-width, initial-
scale=1.0">
```

LAS FUENTES FLEXIBLES. Medidas em y rem

En la definición de tamaños de las fuentes también es muy útil trabajar con unidades relativas, dado que así todos los tamaños del texto se adaptarán al elemento raíz.

em corresponde con la medida actual de la fuente en un contenedor.

El tamaño de fuente predeterminado en el navegador es usualmente de 16px, por lo que, si no lo has alterado en algún lugar de tu CSS, **1em equivale a 16px**

Por ejemplo, si has modificado el tamaño de la fuente en un elemento y le has asignado font-size: 20px, entonces dentro de ese contenedor (y sus hijos si no redefinimos ese tamaño), 1em será igual a 20px.

LAS FUENTES FLEXIBLES. Medidas em y rem

Básicamente, la mejor unidad para trabajar textos adaptables en CSS es el **rem. Esta unidad de medida quiere decir "root em", y es aquella que nos remite al tamaño de fuente que tenemos en la raíz.**

Ejemplo, si nosotros tomamos el BODY y tiene un fontsize de 100% (lo que corresponderá generalmente con unos 16 px)sabemos que siempre rem será equivalente a esos 16px, lo pongas donde lo pongas

La unidad em es relativa al padre, que si tiene a su vez otra unidad en em, se hace relativa al padre y así, en la jerarquía de elementos hasta llegar al elemento raíz, el BODY.

Esto , en la práctica puede resultar difícil de calcular ya que dicho tamaño depende de una innumerable cantidad de contenedores que pueden estar anidados, hasta llegar al contenedor donde estamos indicando las medidas.

Calculando un valor en em y rem

Imaginemos el tamaño base de la fuente es de 16 píxeles.

Si queremos que un H1 de 24 píxeles,

Tenemos que aplicar **la regla "target/context"** el target es la medida que le queremos dar al H1, 24px y el contexto es la medida del contenedor donde estamos 16px. De modo que la medida que le tenemos que poner al H1 sería de 24/16 = 1.5em.

Si dentro del H1 tenemos un link y queremos darle un tamaño de letra menor, por ejemplo 12px-->

Target sería 12px y context 24px, de modo que 12/24 = 0.5em es el tamaño en em que deberíamos poner al enlace.

Para calcular el valor en rem la fórmula "target/context" es igualmente válida y además el valor de target va a ser siempre el mismo, es decir el tamaño definido en la raiz.

Imágenes responsive con la etiqueta Picture

Para adaptar una imagen muchas veces es suficiente con asignarle con CSS una anchura relativa, de modo que si su contenedor aumenta de tamaño, la imagen también aumentará.

La problemática sobre todo la encontramos con imágenes pesadas, como fotografías, donde la elección de su tamaño y calidad dependerá de la pantalla donde se visualicen.

Lo ideal es poder distribuir imágenes acorde con el tamaño de la pantalla

El objetivo es:

- Poder indicar una imagen y varios archivos de alternativa para distintos escenarios: pantallas pequeñas, medianas o grandes, velocidades de conexión, etc.
- Que el navegador sólo descargue una de las imágenes posibles y no todas. O sea, si una imagen tiene tres alternativas de tamaños, se le entregue al navegador solamente una de ellas, ahorrando la descarga de los otros tamaños alternativos que no necesita.

  Para ello usaremos el elemento PICTURE

Elemento PICTURE

El elemento PICTURE en sí no representa contenido alguno, sirve sólo como un contenedor para escribir varias imágenes y que sea el propio navegador el que elija la más apropiada en cada caso.

Para indicar las imágenes dentro de PICTURE se indicarán varias etiquetas SOURCE.

ejemplo:

```
<picture>
<source media="(min-width: 900px)" srcset="grande.png">
<source media="(min-width: 550px)" srcset="media.png">
<img src="archivo-pequeno.jpg">
</picture>
```

Imágenes responsive

Para adaptar una imagen muchas veces es suficiente con asignarle con CSS una anchura relativa, de modo que si su contenedor aumenta de tamaño, la imagen también aumentará o si la imagen es mayor que dicho contenedor no desborde a éste.

Para evitar que el layaout se rompa se usará:

img{max-width:100%}

Esto no sólo sirve para las imágenes, sirve para cualquier objeto multimedia:

img,object,embeb,video{max-width:100%}

Imágenes responsive

Alto (height) en imágenes responsives.

Lo mejor para el alto es indicar un valor auto, es decir, delegar al navegador que calcule el alto del elemento.

En la mayoría de los casos, esto hará que el alto no pierda proporción con respecto al ancho. Si el ancho de la imagen se reduce para adaptarse a responsive, el alto también lo hace. Si el ancho se queda fijo, el alto también lo hará. Así que completemos nuestra regla para imágenes responsives:

```
img {
height: auto;
max-width: 100%;
}
```