LEMONCODE

HTML

Diseño Web Responsivo (RWDD)

El diseño web responsive (RWDD) es un enfoque de diseño y desarrollo web que tiene como objetivo crear sitios web que se adapten y funcionen correctamente en diferentes dispositivos y tamaños de pantalla.

La idea principal es crear un diseño flexible que permita a los usuarios acceder al contenido cómoda y legible en cualquier dispositivo, ya sea móvil, tablet o ordenador.

Este enfoque se basa en el uso de media queries, reglas fluidas e imágenes flexibles.

El responsive web design es especialmente importante hoy en día debido al crecimiento exponencial del uso de dispositivos móviles para acceder a la web.

Un diseño adaptable garantiza que el contenido sea accesible y atractivo para una audiencia más amplia, independientemente del dispositivo que utilicen.

Además, tener un sitio web adaptable mejora la experiencia de usuario y puede mejorar el posicionamiento en los motores de búsqueda.

El concepto de Mobile First se refiere a la práctica de diseñar y desarrollar sitios web teniendo en cuenta primero los dispositivos móviles y luego adaptarlos para pantallas más grandes.

Esto significa que, por defecto, el navegador prioriza la visualización y el rendimiento en dispositivos móviles, utilizando el enfoque Mobile First.

A medida que aumenta el tamaño de la pantalla, se aplican estilos y ajustes adicionales para optimizar la experiencia de usuario en esos otros dispositivos.

Si trabajas en web, adoptar un enfoque Mobile First garantiza que los sitios web sean accesibles y funcionen correctamente en una amplia gama de dispositivos y tamaños de pantalla.

Uno de los componentes clave son las rejillas fluidas, que permiten estructurar diseños flexibles que se ajusten automáticamente a diferentes tamaños de pantalla.

En lugar de utilizar anchos fijos en píxeles, las rejillas fluidas utilizan unidades relativas como porcentajes para definir las anchuras de los elementos en función del tamaño de su contenedor.

También podemos utilizar unidades de Viewport para que se ajusten a la pantalla en la que se están renderizando.

Esto garantiza que los elementos se ajusten proporcionalmente al espacio disponible, lo que resulta en un diseño adaptable y escalable.

A continuación, veremos un ejemplo en HTML y CSS con un contenedor y elementos internos que utilizan porcentajes para ajustar su ancho.

```html
<!DOCTYPE html>
<html lang="es">
    <head>
        <meta charset="UTF-8" />
        <meta http-equiv="X-UA-Compatible" content="IE=edge" />
        <meta name="viewport" content="width=device-width, initial-scale=1.0" />
        <link rel="stylesheet" href="style.css" />
        <title>Responsive Web Design</title>
    </head>
    <body>
        <div class="container">
            <div class="element">Elemento 1</div>
            <div class="element">Elemento 2</div>
            <div class="element">Elemento 3</div>
        </div>
    </body>
</html>
```

Ya tenemos tres elementos, ahora vamos a ir al CSS y le vamos a indicar que el container tenga un width de 100%.

```css
/* style.css */

* {
    box-sizing: border-box;
    margin: 0;
    padding: 0;
}

.container {
    width: 100%;
    display: flex;
}

.element {
    flex: 1;
    padding: 20px;
    height: 100px;
    background-color: #f1f1f1;
}
```

En este ejemplo, como podemos ver, el contenedor tiene un ancho del 100% del espacio disponible, lo que significa que ocupará todo el ancho de la ventana del navegador.

Los elementos internos que son Element tienen la propiedad `flex: 1`, lo que indica que deben distribuir el espacio restante por igual entre ellos.

Esto nos garantiza que los elementos se ajusten automáticamente al ancho del contenedor adaptándose a diferentes tamaños de pantalla.

Tenemos otra herramienta para poder hacer estos layouts flexibles que son las media queries.

Las media queries son una herramienta esencial en el diseño web responsive, ya que permiten aplicar estilos específicos en función de las características del dispositivo, como el ancho de pantalla, la resolución y otros factores.

Esto significa que puedes adaptar el diseño y la apariencia de tu sitio web a diferentes dispositivos y tamaños de pantalla, mejorando así tanto la experiencia de usuario como la accesibilidad.

A continuación, veremos un ejemplo en HTML y CSS donde utilizaremos media queries para cambiar la disposición en función del ancho de pantalla de los estilos.

En este caso seguimos teniendo el mismo ejemplo anterior que contiene 3 `<div>` con la clase `element`.

En el CSS, para que se pueda ver más claro, le vamos a indicar a la clase `element` que queremos un `background-color: indigo` y además le vamos a añadir la media query, que se incluye tan fácil como decir media y aquí dentro le ponemos mean width de 768, que sería algo así como tablet.

Si le decimos que nuestro elemento tenga un width del 33% y además este background color que tenemos aquí de índigo lo vamos a incluir para que lo podamos ver aún más claro.

Lo vamos a poner de color pink.

```
/* style.css */

* {
    box-sizing: border-box;
    margin: 0;
    padding: 0;
}

.container {
    width: 100%;
    display: flex;
}

.element {
    width: 100%;
    padding: 20px;
    height: 100px;
    background-color: indigo;
}

@media (min-width: 768px) {
    .element {
        width: 33%;
        background-color: pink;
    }
}
```

En este ejemplo, los elementos `element` tenían un ancho del 100% en dispositivos con pantallas pequeñas, pero en un ancho de pantalla a partir de 768 se activa esta consulta a la media query y se aplica un ancho del 50%, tanto al elemento como cambia el color de background a pink.

Si quisiéramos tener otra media query con una pantalla más grande, podríamos hacer otra consulta de media query y se ajustaría el ancho de los elementos a otro porcentaje, creando una disposición de una, dos o tres columnas.

Estas consultas a las media queries nos permiten adaptar el diseño en función de las características tanto del dispositivo como de la pantalla, y nos proporcionan una experiencia de usuario muy optimizada en diferentes tamaños de pantalla.

Vamos a hacer un ejemplo responsive sin necesidad de utilizar las media queries.

Recordad que hemos utilizado flexbox y grid layout para poder hacer layouts más complejos. En este caso, vamos a utilizar un contenedor con un display grid y las cajas interiores, los elementos, van a estar en display flex para poder centrarlos.

```
<!DOCTYPE html>
<html lang="es">
    <head>
        <meta charset="UTF-8" />
        <meta http-equiv="X-UA-Compatible" content="IE=edge" />
        <meta name="viewport" content="width=device-width, initial-scale=1.0" />
        <link rel="stylesheet" href="style.css" />
        <title>Responsive Web Design</title>
    </head>
    <body>
        <div class="container">
            <div class="box">1</div>
            <div class="box">2</div>
            <div class="box">3</div>
            <div class="box">4</div>
            <div class="box">5</div>
            <div class="box">6</div>
        </div>
    </body>
</html>
```

Ya tenemos nuestro HTML. Vamos a los estilos.

```
/* style.css */

* {
    box-sizing: border-box;
    margin: 0;
    padding: 0;
}

.container {
```

```
    display: grid;
    gap: 1rem;
    grid-auto-rows: 20rem;
    grid-template-columns: repeat(auto-fill, minmax(15rem, 1fr));
}

.box {
    display: flex;
    justify-content: center;
    align-items: center;
    background-color: lightblue;
}
```

Recordad que `fr` era una fracción de unidad. Así que lo que tenemos aquí en nuestro layout es responsive y se comporta de manera fluida y dependiendo del tamaño que tenga nuestra pantalla van a entrar más o menos elementos. Esto lo conseguimos uniendo grid y flex.