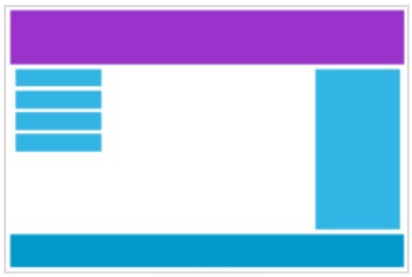
# Consejos para un diseño responsive

Se llama **diseño web responsivo** (responsive design) cuando se utiliza CSS y HTML para **redimensionar, ocultar, encoger, ampliar o mover el contenido** para que se vea bien en cualquier pantalla.


Desktop


Tablet


Phone

Pero, **¿qué herramientas o propiedades se recomiendan utilizar para hacer esto posible?** A continuación, se proporciona una lista.

**1)** Incluir el elemento *<meta>* ***viewport*** en el *<head>* de nuestro código HTML.

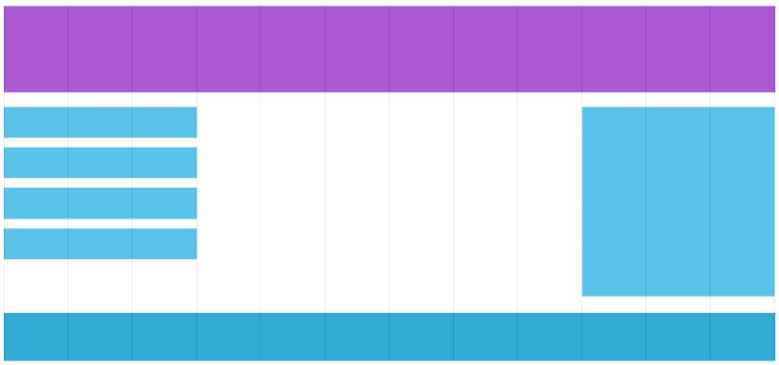
```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```


Without the viewport meta tag

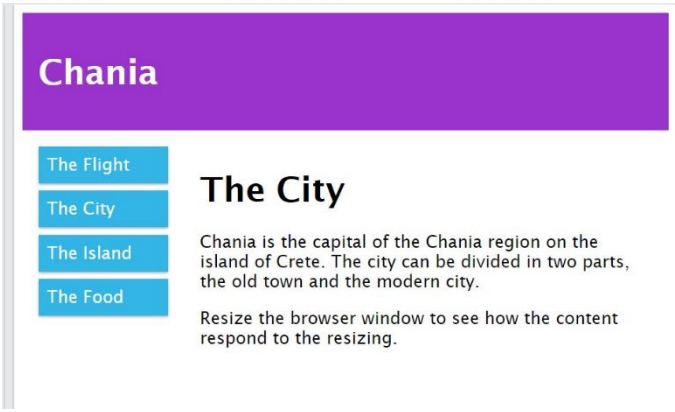

With the viewport meta tag

**2) Utilizar una tecnología de diseño responsive: Multicol, Flexbox o CSS Grid.**

a) **Multicol (Grid View)**: se basa en dividir el diseño de la página web en **12 columnas**, y una anchura total del 100%. Para ello tenemos que usar la propiedad de *box-sizing: border-box;* Es la tecnología más antigua.



```
<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<style>
* {
  box-sizing: border-box;
}

.row::after {
  content: "";
  clear: both;
  display: table;
}

[class*="col-"] {
  float: left;
  padding: 15px;
}

.col-1 {width: 8.33%;}
.col-2 {width: 16.66%;}
.col-3 {width: 25%;}
.col-4 {width: 33.33%;}
.col-5 {width: 41.66%;}
.col-6 {width: 50%;}
.col-7 {width: 58.33%;}
.col-8 {width: 66.66%;}
.col-9 {width: 75%;}
.col-10 {width: 83.33%;}
.col-11 {width: 91.66%;}
.col-12 {width: 100%;}

html {
  font-family: "Lucida Sans", sans-serif;
}

.header {
  background-color: #9933cc;
  color: #ffffff;
  padding: 15px;
}

.menu ul {
  list-style-type: none;
  margin: 0;
  padding: 0;
}

.menu li {
  padding: 8px;
  margin-bottom: 7px;
  background-color: #33b5e5;
  color: #ffffff;
  box-shadow: 0 1px 3px rgba(0,0,0,0.12), 0 1px 2px rgba(0,0,0,0.24);
}

.menu li:hover {
  background-color: #0099cc;
}
</style>
</head>
<body>

<div class="header">
  <h1>Chania</h1>
</div>

<div class="row">
  <div class="col-3 menu">
    <ul>
      <li>The Flight</li>
      <li>The City</li>
      <li>The Island</li>
      <li>The Food</li>
    </ul>
  </div>
  <div class="col-9">
    <h1>The City</h1>
    <p>Chania is the capital of the Chania region on the island of
Crete. The city can be divided in two parts, the old town and the
modern city.</p>
    <p>Resize the browser window to see how the content respond to the
resizing.</p>
  </div>
</div>

</body>
</html>
```



b) **Flexbox**. En flexbox, los ítems flex se encogen o crecen, distribuyendo el espacio entre los elementos según el espacio de su contenedor. Es un sistema **unidimensional**.

c) **CSS Grid**. Es un sistema **bidimensional**, lo que significa que puede manejar tanto columnas como filas.

**3)** Utilizar **media queries**. Normalmente *@media screen* para adaptar el contenido de nuestra página web al tamaño de pantalla del dispositivo del usuario.
En los tamaños de pantalla en los que introducimos el cambio, se le denominan **breakpoints**. Para ello podemos usar la propiedad *min-width* o *max-width*.

```
@media screen and (min-width: 80rem) {
  .container {
    margin: 1em 2em;
  }
}
```

**4)** Utilizar **imágenes, vídeos y fondos responsive**.

**a. <u>Imágenes y vídeos</u>**:

- Si la propiedad de `width` se establece en un **porcentaje** y la propiedad de `height` se establece en "**auto**", la imagen (o vídeo) será responsive y se escalará hacia arriba y hacia abajo.

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

- Si en vez de utilizar `width`, utilizamos `max-width`, la imagen (o vídeo) nunca se ampliará para ser más grande que su tamaño original.

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

**b. <u>Fondos</u>**. Podemos hacer un fondo con las siguientes propiedades:

- `background-size: contain;` la imagen de fondo se escalará y tratará de ajustarse al área de contenido.

- `background-size: 100% 100%;` La imagen de fondo se extenderá para cubrir toda el área de contenido.

- `background-size: cover;` La imagen de fondo se escalará para cubrir toda el área de contenido. Hay que tener en cuenta que el valor de "cover" mantiene la relación de aspecto, y alguna parte de la imagen puede ser recortada.

**5)** Utilizar **tipografía responsive**, mediante la combinación de media queries y las unidades relativas como **rem** o **viewport units** (vh y vw).

- **1vw** es igual al **1% de la anchura de la ventana gráfica**, lo que significa que si estableces el tamaño de la fuente utilizando vw, siempre estará relacionado con el tamaño de la ventana gráfica.

```
h1 {
  font-size: 6vw;
}
```

- En cuanto a los **rem**, establece el **tamaño en proporción al tamaño de fuente base**. Por ejemplo, si utilizamos 4rem en nuestra tipografía, significa que esta será cuatro veces nuestro tamaño de fuente base.

```
@media (min-width: 1200px) {
  h1 {
    font-size: 4rem;
  }
}
```

**6)** Utilizar **plantillas con diseño responsive**.

**7)** Utilizar **Frameworks**. En cuanto a los Framework CSS que tenemos hoy en día en el mercado, podemos encontrar los siguientes: