**Media Query:**

Los “media queries” prácticamente son el corazón del responsive design. Se trata de leer ciertos estados en el navegador, puntualmente, el tamaño de pantalla. Existe 2 formas de acceder a esta propiedad, vamos a aprender cuál resulta mejor.

Existen 2 formas de agregar un media querie a un proyecto

1 ccs externo

<link rel=“stylesheet” href=“css/invie.css”/>

<link rel=“stylesheet” media="(max-width: 800px)" href=“css/media-querie.css”>

<meta name="viewport" content="width=device-width, initial-scale=1, user-scalable=no">Importante para que se muestren correctamente los media queries

Usar el meta viewport nos ayuda a mostrar correctamente nuestro contenido según el área visible del usuario. Si olvidamos esto, tendríamos cómo resultado nuestro sitio desktop con un zoom y todo muy pequeño y por lo tanto, la usabilidad es dañada. Dentro de todas las configuraciones disponibles, podemos tener un zoom por defecto o hasta bloquear el pinch to zoom.

donde los cambios se agregan a media-quierie.css

2 dentro del ccs

@media screen and (max-width: 800px) {

body {

background: peru;

border: 15px solid lightblue;

}

}

screen: para la pantalla del ordenador.

tv: para televisores

print: para la vista de impresión.

CSS  
**Selectores de atributo** Ejem. Todos los input de tipo texto

*input [type="text"],*

*input [type="email"]{*

*font-size: 16px;*

*padding:5px 0;*

*}*

**Zoom**

**z-index: 2;** Posiciona el elemento enfrente de los demás elementos.

[**https://icomoon.io/**](https://icomoon.io/)

para iconos

/\*Elemento estirado en toda la pantalla\*/

position:absolute;

left: 0;

top:0;

bottom:0;

right:0;

# Optimizando la carga de imágenes con lazy loading

https://cdnjs.com/ blazy

<script type="text/javascript" src="https://cdnjs.cloudflare.com/ajax/libs/blazy/1.8.2/blazy.min.js"></script>

var blazy = new Blazy({

selector: 'img'

});

# Videos responsive

<div class="video-responsive-contenedor">

<iframe class ="video-responsive-src" width="560" height="315" src="https://www.youtube.com/embed/WQ8E-AVgcUw" frameborder="0" allowfullscreen></iframe>

</div>

<video src="images/nature.mp4" ></video>

# Soportando multiples resoluciones

Install Python, make sure to add it to our environment variables

Open a command promt and type:

For python2.x: ***python –m SimpleHTTPServer 8000***

For python 3.7: ***python -m http.server 80***

So now we can open the browser and type or Ip Address in our new static local server

<img src="images/invie.png" srcset="images/invie.png 1x, images/invie-white.png 2x" width="186" height="60" alt="Invie logotipo"/>

With lazy loading:

<img class="izquierda" data-src="images/invie-classic.png|images/invie-classic2x.png" alt="Guitarra Invie Classic" width="350"/>

* **How to emulate retina screen on firefox**

You actually can using Firefox:

* Go to about:config
* Find layout.css.devPixelsPerPx
* Change it to your desired ratio (1 for normal, 2 for retina, etc.)

# Tabla responsive

We create an inner scroll

*.tabla .contenedor {*

*overflow: hidden;*

*padding: 0 10px;*

*}*

# Testing our site with remote debugging

# Gestures Hammer-JS

<https://hammerjs.github.io/>

Libraries <https://cdnjs.com/libraries/hammer.js>

# Optimizing our site load

Pagespeed

<https://developers.google.com/speed/>

-optimize images

1. go to <https://tinypng.com/>

2. download the images

- Eliminate render-blocking JavaScript and CSS in above-the-fold content

1. Only css elements within style tags should be located on the top part of the page all css and JavaScript links should be located on the bottom

Or…

2. Compress css styles with

<https://jonassebastianohlsson.com/criticalpathcssgenerator/>

and

<https://csscompressor.com/>

3. Add all the google font to our collection and use them as JavaScript

<script src="https://ajax.googleapis.com/ajax/libs/webfont/1.6.16/webfont.js"></script>

<script>

WebFont.load({

google: {

families: ['Montserrat', 'Allerta']

}

});

</script>