2. CSS

# 2.1. Intro

Hasta ahora utilizamos estilos “inline” para definir dónde y cómo se muestran nuestros elementos en el pantalla.

Podría ahorrarnos mucho código definir un estilo único para el texto y luego aplicarlo para todos mis textos, o un estilo único para mis títulos.

Para esto usamos archivos CSS.

El CSS es también a libre interpretación del browser y/o sistema operativo. Hay situaciones en las que hay que especificar CSS para un determinado browser o SO.

Ejemplo:

Archivo HTML:

<html>

<head>

**<link rel="stylesheet" type="text/css" href=”/css/main.css" />**

</head>

<body>

<div class=”titulo”>¿Why should I trust Welcome To Uruguay?</div>

<div class=”parrafo”>In Welcome to Uruguay we know our country from side to side. We know its best locations, its best restaurants, its best night life.</div>

<div class=”titulo”> ¿How does it work?</div>

<div class=”parrafo”>Contact us, tell us what you’re looking for. ¿How many people are there in the group? ¿What activities do you like? ¿Fishing? ¿Nightlife? ¿Beach? ¿Pool? ¿Elegant dining? ¿Would you like a car? We’ll come back to you with an offer you can jump right into.<div>

<button id=”moreButton”>MORE INFO</button>

</body>

</html>

Archivo CSS:

.titulo{

position:relative;

width:100%;

font-size: 22px;

font-weight:bold;

text-align:center;

}

.parrafo{

position:relative;

width:100%;

font-size: 18px;

text-align:justify;

}

#moreButton{

position:relative;

width:100%;

font-size: 18px;

text-align: center;

background-color:#466457;

}

# 2.2. Position

* El concepto de position es de los más importantes para construir el front.
* Generalmente utilizamos 3 tipos de position: absolute, relative y fixed.

## 2.2.1. Absolute

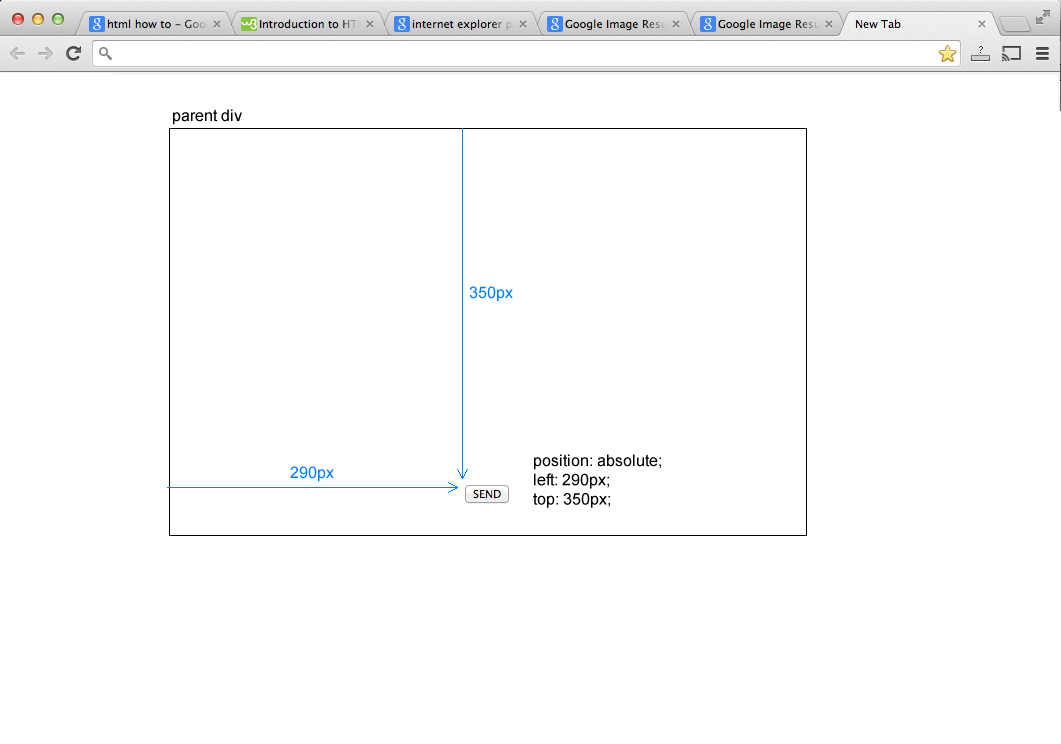
* Posiciona de manera fija un elemento dentro de su padre.
* Se utiliza junto con left, top, right y bottom.
* Confiable Cross-Browser.
* Fácil de usar.
* Necesario si queremos poner cosas por encima de otras.
* Útil en situaciones en las que sabemos que un elemento siempre va a ocupar esa posición.

Ejemplo:

<div class=”parentDiv”>

<button class=”buttonElm”>SEND</button>

</div>



## 2.2.2. Relative

* Posiciona un elemento a continuación del anterior.
* Para mover el elemento utilizamos margin-left, margin-top, etc.
* Útil en situaciones en las que tenemos elementos dinámicos que pueden condicionar la altura del contenedor.
* Si usamos otros elementos adentro, hay que considerar que debemos definir “height:auto” y “overflow:hidden” para que conozca su height y empuje a los próximos elementos.

<div class=”parrafo”>

loreem ipsum….

</div>

<div class=”titulo”>

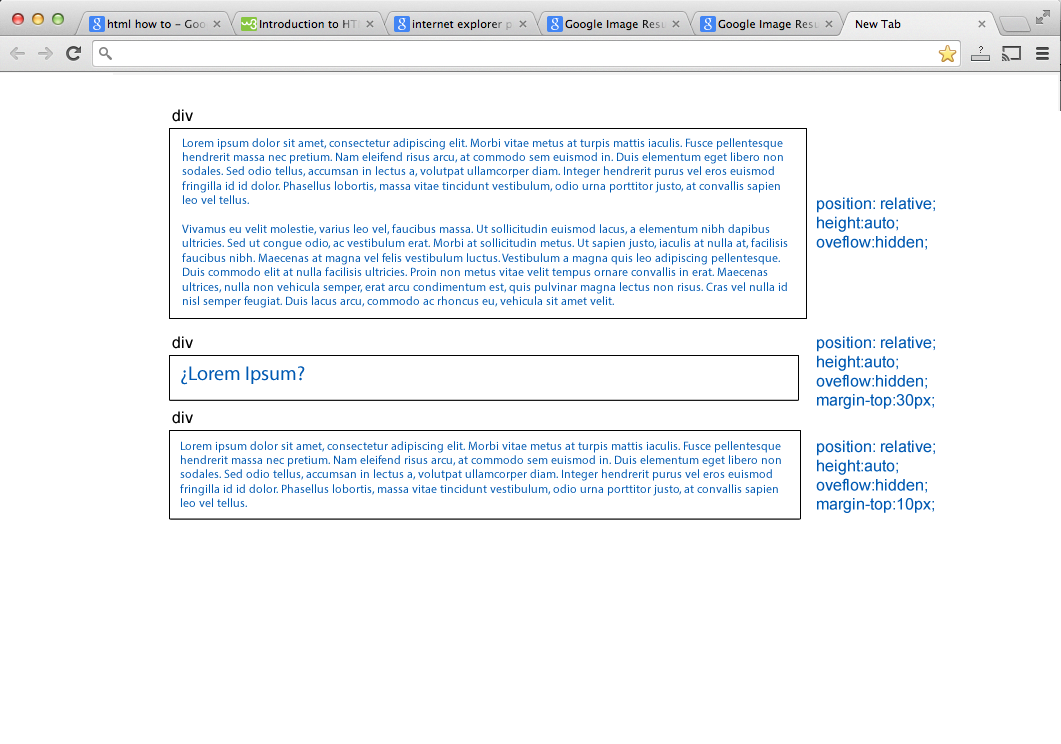
loreem ipsum….

</div>

<div class=”parrafo”>

loreem ipsum….

</div>



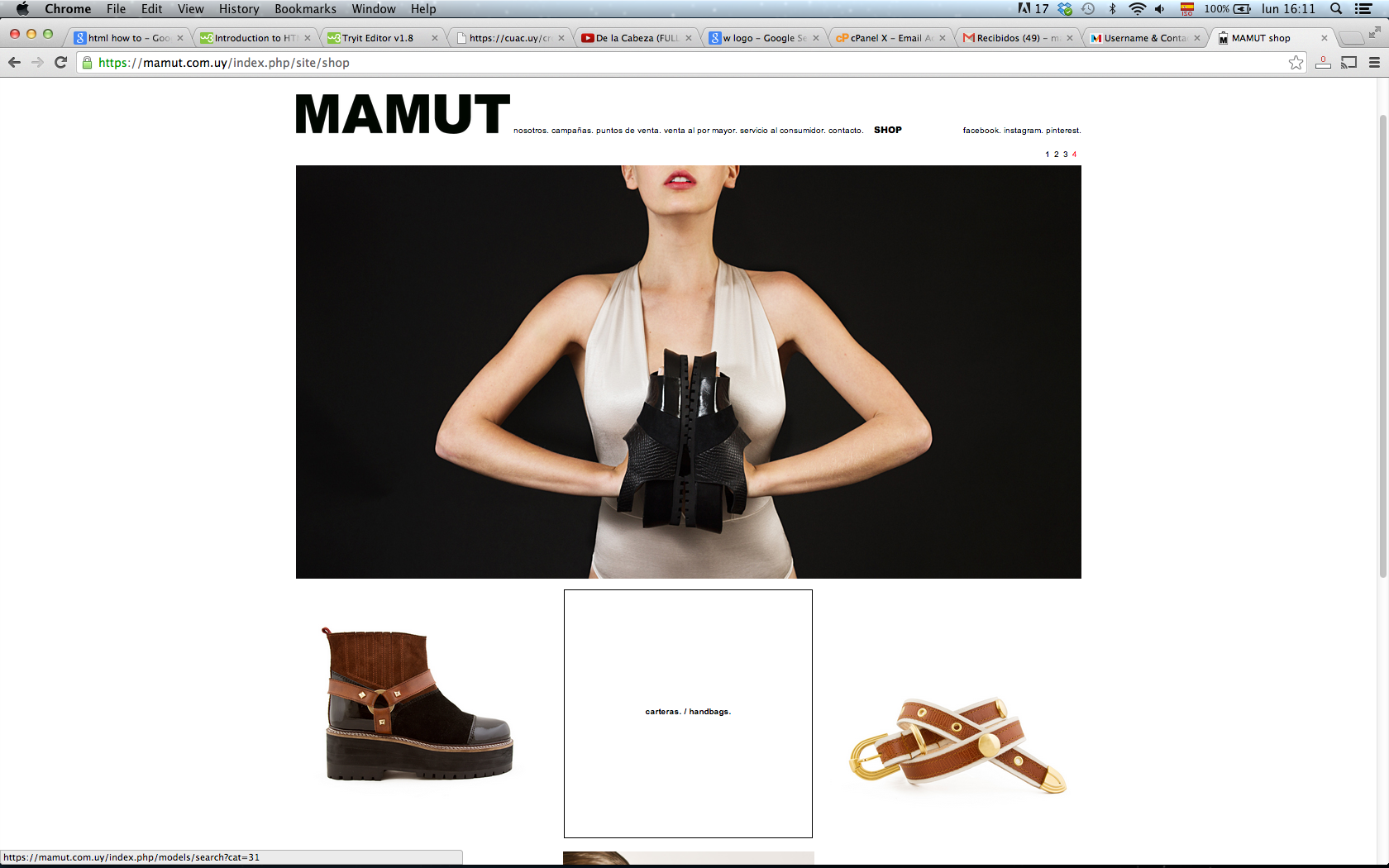
## 2.2.3. Fixed

* Igual que absolute, pero posiciona relativo a la pantalla, es decir que si hago scroll “me persigue”.
* Ideal para mensajes que deben ser atendidos antes de continuar.



# 2.2. Float

* Float nos permite poner elementos uno a continuación del otro horizontalmente.
* Sirve para casos en los que necesitamos específicamente lo anterior.
* También, si utilizamos este concepto por ejemplo en un menú, será más fácil de cambiar en un futuro, ya que sus elementos se ordenan solos.



# 2.3. Ejercicio

Hacer el código del siguiente diseño web.

Cuidado, hay elementos absolute, relative y float.

Investigar:

¿Cómo alineo horizontalmente sin saber el ancho de la pantalla del cliente?

¿Ponerle fondo a un div?

¿Opacidad?



# 2.4. Ejercicio

1. Buscar qué significa Google Fonts.
2. ¿Para qué sirve?
3. ¿Por qué es necesario?
4. Aplicar fonts diferentes de Google Fonts al menú, al título y al párrafo.

# 2.5. Ejercicio

1. Buscar qué significa hover, visited, etc.
2. Agregar un comportamiento a lo hecho anteriormente en el que se pone el cursor en pointer cuando le paso por arriba a la imagen.
3. Agregar un comportamiento hover en el que cambia el color de los items del menú cuando le paso por arriba
4. Investigar cómo usar **transitions** para que el cambio del punto (3) no sea instantáneo, sino fluído (animación).

# 2.6. Concepto Moon

* Usamos ***class*** cuando claramente pertenece a un grupo de elementos con el mismo estilo.

.button

* Usamos ***ids*** para darle formato a un element cuando este es único.

#loginButton

* La mayoría de las veces usamos ambos, la clase le da el formato inicial. Para cada elemento diferente agregamos estilos si son necesarios
* Este tipo de conceptos nos permite generar la misma solución más prolija, con menos líneas de código, más reutilizable y también nos permite entender el código de los demás participando del proyecto.

Por ejemplo, el botón el login es igual que todos los botones de mi sitio, rectangular, con fondo gris, texto negro de 16px,

PERO

El login button está abajo a la derecha relativo y el botón de registro está en el medio con posición absoluta.

