**Datos Piolas CSS**

* DIV is for starting a new line. “**Block** level element”
* SPAN, HRefs are “**Inline** element” 🡪 To style something within a text already, so we don’t want to create a new line. (Usa unicamente el espacio necesario. Padding Margin are NOT respected. For that, I need to transform them into BLOCK)
* **Rules**: Everything will be inherited unless we change the property’s value specifically. (font-site: 1.5rem)

🡪 BORDER por ejemplo NO inherit.

* Using **REM** will help if the user changes the default browser settings (RESPONSIVE)
* **VW, VH**: Relativos a la pantalla. (RESPONSIVE)
  + Min-height: 100vh 🡪 **Full screen.**
* DevTools are perfect for **Debugging**. For example Margins and Paddings (Ctrl + Shift + I)
* **Calc()** Function es lo que usaremos para completar paginas al 100% de su capacidad. DEJAR ESPACIOS ENTRE ELEMENTOS Y LA OPERACIÓN A REALIZAR. **Calc (vh – px)**
* **CSS Box Model**: El elemento es siempre el texto y su área.
* Opacity: Entre 0 y 1 🡪 No se ve en 0, pero existe
* Visibility: hidden 🡪 No se ve, pero existe.
* Display: None 🡪 No se ve, lo remueve incluso de las devtools
* **BG: Overlay negra** 🡪 {bg: linear-g (rgba(0, 0, 0, 0.3)**,**

Rgba (0, 0, 0, 0.3))**,**

Url (“./Path.jpg”)}

* **BG URL ()** 🡪 url(“”) center/cover fixed/scroll no-repeat;
  + Podemos dar los valors pegados al URL y ahorrar espacio
* **Float:** Sacamos del “flow” normal y posiciona. **Clear:** Escapa el flow dado **(float: right 🡪 Clear: right)**
  + **Clear: Both 🡪** Si tenemos 2 floats a ambos lados.
* **Position: Relative 🡪** Mueve a los elementos desde su posición by default (inicial). Usaremos TOP, BOT, LEFT, RIGHT. The original space won’t be filled by anything. **{Position:Relative; Top: 200px}**
* **Pos: Absolute 🡪** Will be Relative to a parent (Body principamente, hasta que otro parent tenga POS:RELATIVE)

The initial space will be filled by the contenct following the element. Ej: Centrar Icons

* **Pos:Fixed** 🡪 Relative to PortView or Screen. Ej: NavBar

Ej: .nav {position: fixed; top:0; left: 0; width:100%}

* **Pos:Sticky 🡪** Relative until we touch the element, then it becomes fix. We can control the moment it becomes fixed.

Ej: {Position: sticky; top: 20px;} When it is 20px from the element It will stick.

* **@media screen and (min-width: px)** {

Body {properties: values;}

.nav {properties: values;}

}

🡪 This is RESPONSIVE: Mobile first, and use query to adapt to Tablet and Desktop.

* Z-Index 🡪 Doesn’t work with default position, and its **default is 0.** Positive or Negative. **Make elements closer or further to us**. {z-index: 10000}
* **DIV + IMG** 🡪 **Cubrir area** de DIV **con IMG{width:100%; display: block}**
* **::before ::after 🡪 Content:” “**; Pos: Absolute + top + left+ index, podemos crear sombras y cuadros y posicionarlos encimados.
* **Transform**: \_\_X () // \_\_Y () // \_\_ ( \_\_, \_\_ ) 🡪 Translate, Scale, Rotate, Skew
* **{Transition: property duration delay, property duration delay, etc.}**
* **@keyframes Name {} 🡪** En el medio ponemos 0%{} 100%{}. Podemos dar cualquier cantidad de % necesarios, usaremos 50% y 75%. **LOS % SE APLICAN A LA DURATION TAMBIEN.**
* .animation {animation: move 5s infinite}

🡪 **{Animation: Name Duration Count}**

* **{Animation-Fill-Mode: Forwards}** 🡪 Luego de terminar el COUNT la animación termina y queda como el 100% declarado en @keyframes.
* **Global Variables 🡪 :Root {- - varName: Value;}**
* **Element Variables 🡪 Div {- - varName: Value;}**
* **Call a Var 🡪 {Property: var(- - varName);}**

**:root** {--mainSpacing: 5px; --primaryColor: #f15025}

H1 {color: **var**(--primaryColor)}

* **Object-Fit:** Cover – Contain – Fill – None – Scale-down.

Width for Columns, General Heigth, object-fit to display imgs.

**For most Projects**

/\* Variables \*/

:root {

  --primaryColor: #f15025;

  --mainBlack: #222;

  --mainWhite: #fff;

  --offWhite: #f7f7f7;

  --darkGrey: #afafaf;

  --mainTransition: all 0.3s linear;

  --mainSpacing: 0.2rem;

}

/\* Global \*/

\* {

  margin: 0;

  padding: 0;

  box-sizing: border-box;

}

body {

  font-family: "Roboto", sans-serif;

  color: var(--mainBlack);

  background: var(--mainWhite);

  line-height: 1.5;

}

a {

  text-decoration: none;

}

img {

  width: 100%;

  display: block;

}

h1,

h2,

h3 {

  text-transform: capitalize;

  letter-spacing: var(--mainSpacing);

  margin-bottom: 1.25rem;

}

h1 {

  font-size: 3rem;

}

h2 {

  margin-bottom: 0;

}

p {

  font-weight: 300;

  font-size: 0.7rem;

  line-height: 2;

}

.btn {

  display: inline-block;

  padding: 0.5rem 1rem;

  text-transform: capitalize;

  border: 3px solid var(--primaryColor);

  color: var(--primaryColor);

  margin: 3rem;

  transition: var(--mainTransition);

}

.btn:hover {

  background: var(--primaryColor);

  color: var(--mainWhite);

}

* We can change any value, but this is a good start.