**HTML5 E CSS3 – CURSO EM VÍDEO: MÓDULO 2**

**Psicologia das cores:** Todas as cores tem importância. Estudo diz que a escolha de um produto está relacionado com a cor e o visual (branding) está relacionado à compra dele. Esse efeito é instintivo (irracional). As cores representam coisas.

**Azul:** (Facebook, Dell, twitter, intell, linkdin): competência, confiança, sabedoria, calma, integridade, segurança, profissionalismo, integridade. Cor com **maior taxa de aprovação (46%)** e **menor taxa de rejeição (2%)** do mundo.

**Vermelho:** Amor, emoção, energia, raiva, perigo. Usada geralmente em sites de gastronomia (comida), moda e entretenimento, mas também serviços de **emergência e saúde.** Evitável para serviços de natureza, luxo e serviços em geral. (Ex: Netflix). Obs: fundos pretos com letras brancas causam canso visual grande – bom para quando temos pouca informação textual. KitKat.

**Amarelo:** Felicidade, alegria e otimismo. MacDonalds = Vermelho + Amarelo causa um impacto visual de comida feliz. MC também usa marrom e cinza para produtos premiun. Paola Krauze

**Laranja:** Divertimento, ambição, calor, cautela.

**Verde:** saúde, natureza, dinheiro, sorte, mas também remete à inveja. Muito utilizado para turismo, relaxamento, meio ambiente, empresas financeiras. Estudo diz que verde é pouco aceito para meninas adolescentes.

**Rosa:** amor, romance, cuidados, sinceridade. Muito usado para produtos femininos e de cosmético, evitar o rosa quando o produto não tiver pegada sentimental ou doce.

**Marrom:** Robustez, amizade, estabilidade. Uma cor com bastante rejeição, mas as vezes usada para chocolates, churrascarias <https://www.le28-lille.com/>

**REFERÊNCIAS DE SITES FODAS:** [**https://calem.pt/**](https://calem.pt/)[**https://takearecess.com/**](https://takearecess.com/)[**https://www.cedricpereira.com/**](https://www.cedricpereira.com/)[**https://yourplanyourplanet.sustainability.google/**](https://yourplanyourplanet.sustainability.google/)

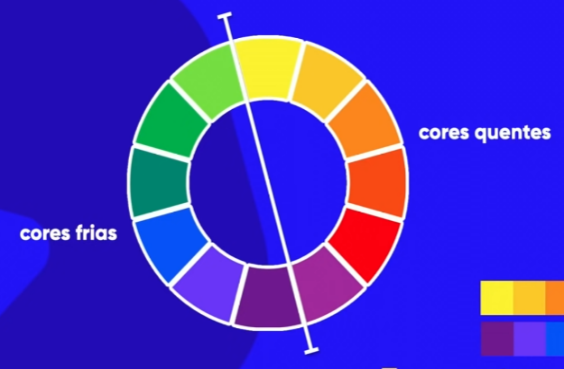
**Representando cores no CSS3:**

1. **Nomes:** É possível apenas colocando o nome das cores: style=” background-color: White;”}
2. **Hexadecimal:** Com 6 dígitos decimais: style=” bacground-color: #ffffff;”} dentre os dígitos [ 0 1 2 3 4 5 6 7 8 9 A B C D E F]. São códigos que são formados por **RGB – Red Green Blue Alpha,** por exemplo, azul seria #0000ff
3. **RGBa:** Por exemplo rgb(0, 0, 255) para azul e rgb(255, 255, 255) para o branco são as gradações de cada uma dessas cores. É como se fossem 3 leds de cada uma dessas cores e as quantidades são a intensidade que elas estão acesas.
4. **Hsla:** Através das características das cores. Hue (Matiz) Saturation (Saturação) Luminosity (Luz) Aplha (Transparência). Por exemplo hsl(240, 100%, 50%)

**Harmonização de cores:** Círculo cromático. Tem 65mi de cores. Por que as coisas são bonitas? R: simetria e cores.

**Primárias Secundárias**

**Terciárias:** Lê-se Primária-Secundária



**Temperatura das coras**

**\*PALETA DE CORES:** Geralmente teremos de 3 – 5 cores na paleta de cores de um site (sem considerar branco e preto). Uma média de 4 cores é ótimo.

**Cores COMPLEMENTARES:** São as cores que mais contrastam entre sí, ou seja, aquelas que se colocadas perto ou sobrepostas mais vão se destacar e diferenciar. Não necessariamente são as combinações mais bonitas, mas são as mais contrastantes. As complementares podem ser usadas junto com análogas.



**Cores ANÁLOGAS:** São cores que não contrastam entre sí, mas possuem uma harmonia entre elas. As análogas podem ser usadas junto com as complementares.



**Cores análogas relacionadas:** São cores usadas para criar degradês “agradáveis”, para isso escolhemos uma cor + outra imediatamente do lado (não importa o lado) depois pulamos uma casa e escolhemos a terceira pós salto.



**Cores intercaladas:** Também criam uma paleta, mas uma paleta um pouco mais “dura”, ou seja, com um pouco mais de contraste.

**Obs:** Para formar paletas também posso usar a técnica dos polígonos e formar com os traços: triângulo equilátero, um quadrado, um retângulo.

**Monocromático:** Pegamos uma cor e alteramos apenas a saturação e luminosidade. Um exemplo com a cor laranja seria:



**Degradês na CSS3:** com parâmetro { background-imge: linear-gradient (to right, White, blue);}. Background-image é o **parâmetro** para criar gradientes, linear-gradient é o **tipo** de gradiente, to right **é a direção** do gradiente (poderia ser, por exemplo, 90 deg) e em seguida vem as cores.

**Posicionamentos:** Para posicionar o degradê podemos usar: ângulos (45geg) (-45deg) (90deg) ou podemos usar a direção (to right) (to left) (to bottom) (to top).

**Tipos:** linear-gradient (linha reta + direção). Radial-gradient (circle, colors)

**Tamanha:** Após as cores podemos dar espaço e colocar uma % que retornará o tamanho do gradiente. (#b93590 10%).

**BOX-SHADOW:** Parâmetro para retornar sombra no objeto. Possui 4 valores 1) quanto ela anda para o lado 2) quanto ela anda para baixo 3) quando ela está espalhada 4) cor

**\*TIPOGRAFIA:** Surgiu no século 15 quando surgiu a imprensa. Quando não existiam máquinas, eram os monges copistas que faziam essa tarefa na Europa. Em 1400 e pouco Gutenberg, o pai da imprensa, criou a prensa mecânica. Com essa prensa e pecinhas de metal, rolhas e argilas o aproveitamento e pensamento de letras não ficava perfeito, as peças estragavam, e o espaço não era bem aproveitado surgindo a necessidade de criar mais tipos de letras. Foi aqui que nasceu a **tipografia. Tipo = imprimir no papal + grafia (escrita) 🡪 Estudo de como escrever coisas no papel**

**Forma:** A forma de escrever também transmite emoções

Diferentes tipos de representar a palavra amor: amor remete a algo delicado, duradouro, suave e escriptado (fonte de cima), já a de baixo é muito dura, remete à retrô.

Destacar a palavra importante requer um teor mais sério, direto com fácil leitura, imponente e não divertido e solto (como a fonte de cima).



A fonte de baixo com o texto “feito à mão” remete à ideia “literal” de que o produto vendido por esse anúncio também tenha sido feito à mão.

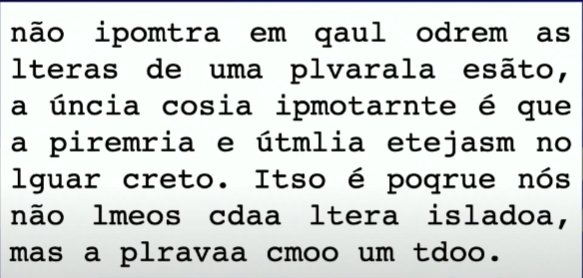
**\*ANATOMIA E FAMÍLIA DAS FONTES:** Tipos são cada uma das letras (glifos) que são usadas para constituir uma palavra. **Importante:** o ponto de partida da criação de qualquer fonte é o “x” com a altura “x”

** X:** O “x” minúsculo define a altura base de todas as outras letras minúsculas.

**Altura do corpo:** É a altura total de todos os tipos (o maior) de alturas = altura x + altura ascendente + altura descendente + altura das maiúsculas. Para fins de CSS3 a altura que usaremos é a altura do corpo (total de espaço ocupado).

****

**Serifas (serif):** São tracinhos que algumas fontes de uma família podem ter. Ela cria uma **linha imaginária** que guia nosso olhar para facilitar a leitura das palavras (no papel)

Por outro lado, existem estudos que comprovam que fontes sem serifa funcionam melhor para leitura (em telas).

**ANATOMIA DE FONTES:** Componentes geométricos:





**Glifos:** É cada unidade de letra usada em uma fonte. Fonte é o conjunto de todos os glifos (maiúsculos e minúsculos) de uma família.

**Família tipográfica:** São várias maneiras diferentes de representar o mesmo glifo (Open-Sans)

Negrito = bold. Nem toda família de fonte terá todas essas formas.

**Handwriting:** scripitadas (feitas à mão).

**Display:** Fontes comemorativas que foram feitas para remeter à ideia específica (geralmente filme). Jurassic Park, Toy Story, Os incríveis, Robôs, Kung fu panda, World of Warcraft. São fontes onde a **liberdade criativa,** são maiores. Precupam-se apenas com o aspecto visual e representação da ideia (não necessariamente com a leitura).

**Fontes no CSS3 – FONT-FAMILY E SAFE COMBINATION:** Quando declaramos o parâmetro font-family no css3 podemos colocar como valor, mais de uma fonte. Isso faz com que caso o dispositivo que está lendo a página não tenha como fonte o primeiro valor declarado, ele tenta o segundo, depois o terceiro e assim por diante, até achar alguma fonte que ele realmente tenha e consiga ler. Body { font-family: Arial, Helvetica, sans-serif;}

O ápice do safe combination é colocar como último valor alguma dessas: serif, sans-serif ou monospace; nesse caso o dispositivo tenta ler com algum tipo de fonte que ele tenha.

**FONT-SIZE:** Tamanhos de fontes.

**Medidas absolutas:** cm, mm, in, **px**, pt, pc

**Medidas relativas: em**, ex, rem, vw, vh, % (em = altura relativo ao glifo “m”; ex = altura relativa ao glifo “x”; view width; view height;).

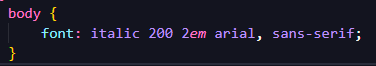
**As duas medidas que usaremos para web recomendadas pela W3C:** px (pixels) e em (altura relativa ao glifo “m”). Em pixels o “padrão” que os navegadores usam é 16px. Em em o “padrão” que os navegadores usam é 1em. **PADRÃO = 16px = 1em.**

**FONT-WEIGHT (PESO):** É o parâmetro que indica o peso da fonte (algumas podem ser simples e não ter variação de peso). Os valores existentes para font-weight são: { **lighter; normal; bold; bolder**}. Existem também fontes com mais variações (que são numéricas), por exemplo, famílias de fontes do Google que podem variar de: {**100; 200; 300; até ; 900**} em que 100 é a mais leve e 900 é a mais pesada. Geralmente o **normal** equivale a **300** ou **400**.

**FONT-STYLE:** É um parâmetro de estilo de fonte que veio para substituir as tags: <i> e <b> (não tem mais valor semântico pois agora é <em> e <Strong> respectivamente. Os valores para ele são: **{ italic; normal; oblique; inherit;}.**

**TEXT-DECORATION:** Parâmetro que estiliza com os seguintes valores: **{ dashed; dotted; double; none; line-through; underline; overline; solid}**

**\*SHORTHAND PARA FONTE 🡪 FONT:** FONT-STYLE > FONT-WEIGHT > FONT-SIZE > FONT-FAMILY

Posso colocar quantas e quaisquer propriedades dessas 4, desde que sejam colocadas em ordem.

**(MAIÚSCULO) TEXT-TRANSFORM / FONT-VARIANT:** Uppercase; ou small-caps;

**Google fonts:** Repositório externo e gratuito de fontes mantido pelo Google. Podemos fazer “incorporações” em “embed code” do Google fonts para <linkar> em nosso documento HTML5 ou @importar em nosso documento CSS3. O recomendado é @importar no css.

**Fontes externas baixadas:** Podemos baixar uma fonte em um site como Dafont e descompactar o arquivo na pasta fontes dentro da pasta **fonts** (irmã dos arquivos index.html). Por padrão ele virá com as extensões **.otf** ou **.ttf** . Cada navegador terá sua compatibilidade e para linkar essa fonte dentro da css3 existe um comando:

@font-face { font-family:Nome que quisermos ; src= url(‘love Larry.otf’) format (‘opentype’), url (‘love lary.ttf’) format (‘truetype’);}

Esse comando consiste parâmetro **url** + valor **‘nome do arquivo.extensão’** + **format** além disso ele também permite nos deixar escolher o nome da fonte em font-family

Os tipos de formato que existem são: opentype (p/ .otf); truetype (p/ .ttf); embedded-opentype; truetype-aat (p/ Apple Advanced Typography).

* É possível “roubar / descobrir” a fonte que um site está usando se ela estiver na forma de texto (não for uma imagem ou arte). Para isso basta baixar a extensão **Fonts Ninja** em Chrome Web Store.
* Também é possível “roubar / descobrir” a fonte que um site está usando se ela estiver na forma de imagem com 3 sites: <https://www.whatfontis.com/> <https://www.fontsquirrel.com/> <https://www.myfonts.com/>

**FONT-ALIGN (ALINHAMENTO DE FONTES):** Basicamente existem 3 tipos de alinhamento textuais: na esquerda (left) centralizado (center) e a direita (right), todos esses alinhamentos são passados com o parâmetro **text-align.** Além de alinhar a posição de um texto podemos fazer mais estilizações como:

**Text-align:** Justify; (ele justifica e coloca espaçamento padrão para os dois lados)

**Text-indent:** 1.9em; (ele cria identação, assim como num código. Seria o equivalente ao “Tab”, mas podemos escolher o tamanho dessa indentação.

**ID NA CSS3 (SELETORES PERSONALIZADOS):** A grande vantagem de se usar CSS3 é seu efeito em cascata, onde podemos estilizar várias elementos “iguais” de uma só vez. Por exemplo, quando escrevemos uma tag e passamos o parâmetro de background-color para ela, todas as tags iguais a ela também serão alteradas. Podemos resolver isso passando um parâmetro **id = identificator** para essa tag. Identificadores são representados pelo **#** (hashtags / jogo da velha) e **só existem um com cada valor.**

Segundo a W3C só deve existir **um único id** por documento.

**CLASS NA CSS3 (SELETORES PERSONALIZADOS):** Diferente dos **identificators (id)** as **class** podem ser múltiplas, portanto podem ser usadas em mais do que uma tag, inclusive com variações.

**\*DICA:** Não dê nome às classes e aos ids em virtude da forma, mas sim de acordo com a **FUNÇÃO –** Isso por que se coloco class=”verde” e o cliente pede alteração para azul teremos um class com nome verde e com a forma (cor real) azul. Então use pela funcionalidade.

“**.”** As classes são representadas na CSS3 por meio de um ponto “.”

**VÁRIOS VALORES:** Podemos inclusive passar vários valores no mesmo parâmetro class da seguinte maneira: <span class=”básico marca-texto”>. Para fazer isso basta separar os valores com um espaço.

**\*DICA:** Na hora de desenvolver com CSS3 comece estilizando com o que é mais genérico primeiro e depois parta para o que é mais específico, por exemplo, comece estilizando <p> depois <h2> depois .class depois #id. Isso por que as estilizações mais genéricas **herdarão** tudo que for igual a elas.

**PSEUDO-CLASSES:** Estão relacionadas a um **estado (**ativo, marcado, vazio, habilitado, hover)de determinado **elemento.** Pode ser escrito na css3 assim: div:hover {background-color: blue;} e deve ser lida 🡪 Nas divs “no estado” em que o mouse estiver em cima, mude o background para cor azul.

**EXEMPLO:** Podemos criar uma pseudo-class de estado de **sites visitados** para mudar a cor de um link quando o site já tiver sido visitado. **A:visited {color: grteen;}.**

**EXEMPLO:** Podemos criar uma pseudo-class de estado de **isso é um site** apenas usando a o estado hover quando o mouse estiver por cima de um link, assim: **a:hover {text-decoration: underline;}.**

**EXEMPLO:** Podemos criar uma pseudo-class de estado de **está clicado** apenas usando o estado **active** quando o botão do mouse está pressionando um link, assim: **a:active {color: black;}**

**\*DICA:** Na css3 e no html5 podemos representar elementos de várias formas. Veja a tabela abaixo.

|  |  |
| --- | --- |
| REPRESENTAR <TAGS> | DIV |
| REPRESENTAR ID | # |
| REPRESENTAR CLASS | . |
| REPRESENTAR PSEUDO CLASS | : |
| REPRESENTAR PSEUDO ELEMENTO | :: |
| REPRESENTAR <TAGS> FILHAS | DIV > P (parágrafo dentro da div) |

**PSEUDO ELEMENTOS:** Alteram o conteúdo periférico **(antes::before ou after::depois)** de um determinado objeto, para isso ele usará o **parâmetro content “ “** + valor que deverá aparecer antes ou depois de determinado objeto.

**EXEMPLO:** Para um pseudo elemento que aparece depois de um link posso criar: **a::after {contente “[ link clicável]”; color: purple;}.** Assim depois de todo link aparecerá o valor escrito entre as aspas.

**MODELO DE CAIXAS:** Box-size e display de caixas. A exibição do html e css basicamente acontece por meio de caixas (boxes), quando uma caixa está dentro de outra aconteceu o **aninhamento.** Todo elemento visível dentro de um site é exibido na forma de caixa.

**ANATOMIA DAS CAIXAS:**

**height e width:** são elementos relacionados ao conteúdo “de fato”. **Padding:** é o preenchimento entre a parte visível do conteúdo e a borda: **border:** é a parte mais externa do “próprio” elemento que pode envolve-lo. **Margin:** É a folga que existe para fora do elemento, mas que permite respiro e **outline** é a bora que não “pertence” ao elemento, e que está no campo da margin.

**Border: {**border: tamanha tipo cor} 10px solid black. Os tipos de bordas são: **solid** (linha contínua e sólida) **dashed** (tracejada) **dotted (**marcada com bolinhas) **double** (linha dupla) **groove (**borda 3d com chanfro)

**Padding:** É o preenchimento entre o conteúdo de fato e a borda (é um preenchimento vazio). Ele começa a contar no sentido horário. **{padding: top right bottom left}** podemos passar esses parâmetros separadamente ou em forma de um só vez, ou até mesmo, com shorthand.

**Shorthands {paddind: 10px} 🡪** coloca um padding de 10px para todos os lados de uma vez.

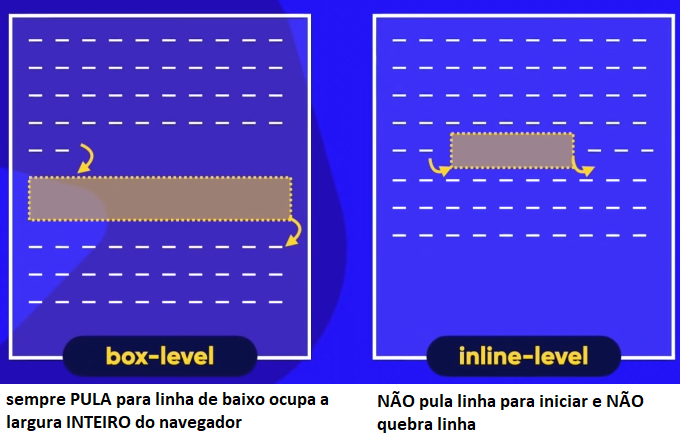
**Shorthands {padding: 10px 50px) 🡪** coloca padding 10px (top and bottom) e 50px (right and left)

**Margin:** É o que vem logo após a borda de um objeto e está do lado de fora do elemento.Começa a contar sentido horário **{margin: top right bottom left}** podemos passar esses parâmetros separadamente ou em forma de um só vez, ou até mesmo, com shorthand.

**SHORTHAND MARGIN-AUTO🡪** Centraliza (top right bottom and left) determinado objeto de acordo com o viewport

**SHORTHAND {margin: 20px auto 50px auto;} 🡪** Vai colocar 20px (top) 50px (bottom) e vai centralizar de acordo com o viewport (right and left).

{**Outline: 5px solid salmon}:** é um contorno que acontece pelo lado de fora da borda, mas dentro da margem (é o contorno da borda).

**box-level:** É isolado e independente. <div>

**Inline-level:** Não quebra linha e ocupa apenas o tamanha que precisa. <span>

**BOX-LEVEL (DISPLAY: BLOCK):** Todo elemento com {display: block;} vai ser um elemento com box-level

**GROUPING TAGS <HEADER> <MAIN> <FOOTER>:** São < > usadas **semanticamente** para **indicar / marcar** para os motores de busca as partes dos nossos sites .

**<HEADER>:** Cabeçalho, contém partes como logo, menu de navegação, área de login

**<MAIN>:** Conteúdo principal do site, geralmente composto por: <SECTION> 🡪 <H1> e <H2> 🡪 <ARTICLE> 🡪 <P> e <ASIDE>

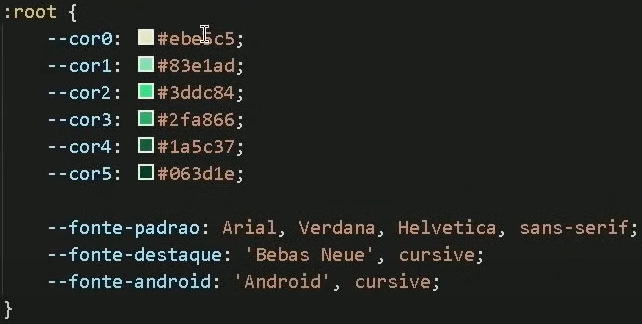
**<FOOTER>:** Rodapé

**BOX-SHADOW:** São sombras aplicadas a objetos. Ela já funciona com 4 parâmetros dentro do **shorthand**

**SHORTHAND BOX-SHADOW:** {box-shadow: x y blur spread color;}. X > 0 sombra anda para direita X < 0 sombra anda para esquerda. Y > 0 sombra anda para baixo, Y < 0 sombra anda para cima. Blur (desfoque / borrado) Spread (espalhamento / alcance). **{ box-shadow: -3px 4px 6px 1px black;}.** É possível colocar sombra interna se passar o primeiro parâmetro como insert. A sombra não pode ser uma cor sólia, ela tem que ter transparência para dar efeito de sombra (aquela região de penumbra onde há um pouco de luz e um pouco de sombra). **Dica:** para criar um sombra podemos usar a ferramenta DEV TOOLS do navegador.

**BORDER-RADIUS (vértices arredondados):** É o arredondamento dos vértices de uma box. A ordem de valores é: top-left > top-right > bottom-right > bottom-left. Existe o shorthand para facilitar {border-radius: 10px 10px 10px 10px}. Se forem passados apenas dois valores será {border-radius: 10px 40px;} 10px (top-left and bottom-right) e 40px será (top-right and bottom-left).

**Bordas decoradas:**

**\*APLICANDO VARIÁVEIS EM CSS3:** São “slots” (gavetas) onde se guardam valores. Elas facilitam muito a vida pois depois de “cadastrarmos” um valor dentro de uma variável, podemos “**chamar / requisitar”** esse valor declarando a variável. Podemos, por exemplo, cadastrar uma lista de cores em variáveis e depois apenas fazer a requisição. Em qualquer linguagem de programação temos uma área para declaração de **variáveis globais,** no css3 essa área é: **:root { }** e dentro de { } declarar variáveis com **“- -“. :root { - -padding1; - - padding2; - -cor1; - -cor2; - -shadow; - -font-defaut; - -font-highlighted}.** Depois de declaras essas variáveis podem ser chamadas dentro do parâmetro com o valor: **var(- -cor1);**

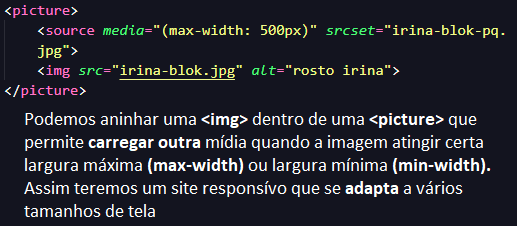
As variáveis facilitam nosso trabalho na criação de um site, mas a **GRANDE VANTAGEM** das variáveis está na manutenção do código. Se decidirmos mudar a paleta de cores ou outra coisa, basta mudar o **valor** dessas variáveis na declaração de variáveis dentro de :root { }

**\*CONFIGURAÇÃO GLOBAL EM CSS3:** Quando se cria um site, tem algumas configurações que sempre mudamos para o mesmo formato (margem). Por isso podemos usar o seletor **“ \* ”** e declarar tudo o que queremos padronizado para o documento. **\* { margin:0; padding:0; }**

**\*RESPONSIVIDADE:** É a função de **adaptar** o tamanho e conteúdo do site para os vários tamanhos de telas diferentes. O site vai “responder” se adaptando ao tamanho de tela onde está sendo exibido. Os dois grande **objetivos** da responsividade são: **primeiro** (evitar rolagem lateral para conteúdos grandes exibidos em telas pequenas) **segundo** (evitar que uma leitura ou visualização fique muito grande lateralmente, por exemplo, ler um parágrafo na tela de uma tv de 60 polegadas ou num telão). Existem várias maneiras de aplicar responsividade e a mais simples delas é: Para dar responsividades a imagens um recurso que pode ser usado é: main img { width: 100%;}. Para fazer um parágrafo quebrar linhas e se ajustar à largura podemos passar os seguintes parâmetros 🡪 header > p { max-width: 500px; padding-right and left: 10px; margin: auto;}

**PLUGGIN PARA GOOGLE CHROME (WINDOW RESIZER)**

1. **Determinando (limitando) tamanhos:** Determinar tamanhos mínimos e máximos de largura de um site com **min-width (:320px;)** and **max-width (:800;)**

****

Colocando a correntinha de link externo com pseudo elemento.

