CSS – MÓDULO 2

1.0 – Cores

1.1 – Psicologia das cores

Cada cor desperta uma determinada emoção em alguém. Devemos saber harmonizar as cores, escolher bem uma cor para determinado tipo de conteúdo. Olhar a folha “O poder das cores”. A cor mais aceita e com a menor taxa de rejeição é o azul. Monocromático é diversificar tons da mesma cor.

1.2 – Representando cores com CSS3 (ex001)

As cores podem ser representadas de 4 formas: Por nomenclatura, hexadecimal, rgb e hsl.

1.2.1 - Nomenclatura

Basta escrever as cores conhecidas quando for colocar uma cor de fundo por exemplo: blue, White, purple, green, black...

```
<h2 style="background-color: blue;color: white;">Exemplo de cores</h2>
```

1.2.2 – Hexadecimal

Precisamos antes de tudo entender a diferença entre decimal e hexadecimal:

Decimal = 0 1 2 3 4 5 6 7 8 9

Hexadecimal = 0 1 2 3 4 5 6 7 8 9 A B C D E F

No hexadecimal, o valor menos significativo é o número 0 e o valor mais significativo é a letra F. Todas as cores são formadas por uma combinação entre Vermelho, Verde e Azul. Então para representarmos, por exemplo, a cor azul em hexadecimal, precisamos zerar a quantidade de vermelho, de verde e deixar cheio a quantidade de azul, chegando então no código: #0000ff. Branco é a presença total de cores: #ffffff. Preto é a ausência total de cores: #000000, e assim por diante.

```
<h2 style="background-color: #00ff; color: #ffffff;">Exemplo de cores</h2>
```

1.2.3 – RGB (Red, Green and Blue)

Sabemos então que todas as cores são formadas por um conjunto de combinações entre as cores Vermelho, Verde e Azul. No hexadecimal, vamos do número 0 até a letra F. No rgb, nos delimitados a quantidade em números de composição da cada uma dessas três cores, indo de 0 até 255. Exemplo então do azul: rgb (0, 0, 255). Branco: rgb (255, 255, 255)...

<h2 style="background-color: rgb(00, 00, 225); color: rgb(225, 225, 255);">Exemplo de cores</h2>

1.2.4 – HSL (Hue, Saturation and Luminosity)

Nesta forma de representação, fazemos uma mistura entre matriz, saturação e luminosidade. Azul: hsl (240, 100%, 50%). Branco: hsl (0, 0%, 100%).

<h2 style="background-color: hsl(240, 100%, 50%); color: hsl(0, 0%, 100%);">Exemplo de cores</h2>

1.2.4 – Ajuda do Visual Studio Code

Basta colocar na tag color o mouse sobre a escrita da cor, selecionar a cor desejada e clicar onde está escrito rgb para alternar entre os modos.

1.3 – Harmonia das cores

Usaremos como base o círculo cromático. Pegar a folha impressa para analisar melhor.

1.3.1 – Cores primárias, secundárias e terciarias.

Vamos estudar com ele: cores primarias: amarelo, vermelho e azul. Cores secundárias: laranja, violeta e o verde. Cores terciárias: Resto das cores. Para dar a nomenclatura, colocamos primeiro o nome da cor primária e depois da secundária. Então, são elas: amarelo-esverdeado, amarelo-alaranjado, vermelho-alaranjado, vermelho-arroxeado, azul-arroxeado e azul-esverdeado.

1.3.2 – Temperatura de cores.

Entre o amarelo esverdeado e o vermelho arroxeado, vamos fazer uma divisória (seguir na folha impressa). Esquerda, cores frias, e a direita, cores quentes.

1.3.3 – Classificações harmônicas

Uma paleta de cores necessita de uma harmonia, que pode ser feita de varias formas, contendo de três a cinco cores sempre tendo uma cor principal:

1.3.3.1 – Cores complementares

Cores que tem mais contrastes entre si, exemplo o violeta, quero saber qual é a cor que mais contrasta, traça uma linha reta: amarelo.

1.3.3.2 – Cores análogas

São cores que não tem tanto contraste entre si, são mudanças sutis, porém, perceptíveis. Basta selecionar a cor principal e pegas as cores que estão a direita e a esquerda. Exemplo: Violeta, esquerda, azul-arroxeado, e direita, vermelho-arroxeado.

1.3.3.3. – Cores análogas mais complementar

Mistura das duas acima. Paleta que tem cores análogas e uma de contraste. Violeta: Violeta, Azul-arroxeado, vermelho-arroxeado e amarelo.

1.3.3.4 – Cores análogas relacionadas

Criar um degrade com um pequeno contraste. Pega duas cores vizinhas, pula uma, e pega outra. Exemplo amarelo: Amarelo, amarelo-alaranjado e vermelho-alaranjado.

1.3.3.5 – Cores intercaladas

São parecidas, porém com mais contraste. Pega uma, pula uma, pega e pula novamente e pega mais uma. Exemplo Amarelo: Amarelo, laranja e vermelho.

1.3.3.6 – Cores triádicas

Bem contrastadas. Pega uma e pula três. Exemplo amarelo: Amarelo, vermelho e azul.

1.3.3.7 – Cores em quadrado

Ao invés de três, pulamos duas, formando quatro cores.

1.3.3.8 – Cores tetrádicas

Pegar uma cor, achar sua complementar. Pega outra cor e também acha o complementar.

1.3.3.9 – Monocromia

Pegar uma cor e modificar a saturação e o brilho. Um degradê.

1.4 – Paleta de cores

Ferramentas para formar paletas de cores: Pasta Dev.

1.5 – Degrade em CSS (ex002)

Para fazermos um degrade, no fundo usado como exemplo, usando CSS, não iremos usar o *background-color*, mas sim, *background-image* + *linear-gradient* . Primeiro nos colocamos a direção do gradiente nos parênteses: to right, to left, ou em graus, 90deg. Depois, as transições de cores, exemplo, de branco para vermelho, da esquerda para a direita:

Estilo local

<style>

Body{

Background-image: linear-gradient(to right, White, blue);

}

</style>

Podemos colocar para qualquer direção: To right, to left, 90deg, 45deg, to top (cima), to bottom (baixo), porém esses últimos, precisamos fazer uma configuração global na css, ficando assim:

Estilo local

<style>

****{ /* Configuração global das CSS */***

Height: 100%;

}

Body{

Background-image: linear-gradient(to right, White, blue);

}

</style>

Porém, quanto mais cores adicionarmos, fica melhor o degrade. Podemos usar qualquer coisa. Se colocarmos uma porcentagem ao lado da cor, determina a quantidade dela na tela (... *(to right, blue 80%, white)* mais azul do que branco). Fizemos com a logo do Instagram por exemplo, no ex002.

Usamos a tag *background-attachment: fixed;* para ele não bugar. Podemos também usar um *radial-gradiente (circle)* ex, para mudarmos o formato do degrade, para um círculo como exemplo.

1.6 – Criando um exemplo (ex003)

Bom, vamos criar um exemplo bonito de site agora. Tag *main* para dizer que ali está o conteúdo principal dentro de body. Também para facilitar para colocar em uma caixa.

TAGS USADAS:

font-family = Fonte das letras usadas

background-image + *linear gradient* = degrade de fundo da página

background-color = cor de fundo do quadrado (main é o conteúdo, quando colocamos o fundo de branco, fazemos um quadrado)

border-radius = arredondar cantos do quadrado

box-shadow: lado, baixo, expansão = sombra do quadrado

widht = largura do quadrado

padding = aumentar a sobra da borda do quadrado

margin: auto = deixar automaticamente no meio

color = Cor do objeto

text-align: center = Texto alinhado ao centro

text-shadow = Sombra do texto

text-align: justify = Texto justificado

1.0 – Tipografia

Tipografia é o estudo de como eu vou escrever coisas no papel. Estudo de como desenhar as letras.

1.1 – Anatomia do tipo

Olhar no pdf impresso. Mais especificamente em: pg 03-11.

1.2 - Família de fontes com CSS

As fontes possuem como se fossem famílias: das serifadas (Serif), não serifadas (sans-serif), handbrakes (feitas a mão), monoespaçadas.... Quando selecionamos *font-family* em CSS, se referimos a abertura e apresentação daquela família no nosso site, porém, nem todos os navegadores possuem aquela fonte, então é feito o que chamamos de *Web Safe Fonts Conbinations*, que basicamente são fontes que são mais seguras de abrirem em todos os navegadores.

1.3 - Tamanho de fonte e suas medidas

Temos duas formas de representar as medidas de fontes: Absolutas e Relativas. As absolutas (cm, mm, in(polegada), px, pt (point), pc (paica). As vezes o navegador não chega exatamente onde queremos no tamanho. As Relativas: em (relativa ao tamanho atual da fonte), ex (relativa ao tamanho x da fonte), rem (relativa a fonte que está configurada no body), vw (viewidth, relativa à sua tela). Recomendado o uso de px e em. O tamanho normal da fonte é 16px. Modificamos o tamanho da fonte com a tag:

Font-size: 16px;

Esses *16px geralmente = 1em (é o tamamanho da letra normalmente), logo 2em = 32px.*

1.4 - Peso, estilo e Shorthand font (ex004)

Peso significa deixar a fonte ***mais magrinha ou mais gorda.*** Para exemplificar, vamos utilizar uma fonte que contém todos os estilos para poder analisar as modificações (lembrando que nem todas as fontes possuem todos esses estilos que serão falados: ***WORK SANS.*** O peso da fonte é determinado pela seguinte tag:

Font-weight: (indicar o peso);

Os pesos das fontes podem ser: lighter (linhas mais finas, fonte mais leve), normal (fonte normal), bold (negrito um pouco mais forte) e bolder (super-

negrito). Você também pode determinar o peso da fonte através de números, que vão de 100 até 900, sendo 100 muito fina e 900 muito negrito. Exemplo: 400 seria o equivalente ao *normal*. Clicando as teclas *ctrl*+*space* pode-se verificar todas as opções de selecionar os pesos. Uma tag para a fonte normal, como exemplo, seria:

Font-weight: normal;

Agora, mudando de assunto, iremos falar sobre alterar estilos de fontes em css. Esta mudança é feita através da tag:

Font-style: (indicar o estilo);

A título de exemplo então, para deixar um determinado conjunto de fontes do seu site em itálico, por exemplo, se utiliza a tag:

Font-style: italic;

Então aprendemos a colocar o negrito (peso) na fonte, o itálico (estilo), formando então os conjuntos que mais usaremos em CSS:

```
h1 {
    font-family: 'work sans', sans-serif;
    font-weight: bolder;
    font-size: 3em;
    font-style: italic;
}
```

Nós podemos simplificar essas quatro linhas de código, pelo o que chamamos de shorthand, nesse caso, o shorthand ***font***. Veja a seguir:

```
h1 {
    font: italic bolder 3em 'work sans, sans-serif;
}
```

Para entender como funciona o shorthand, analise isso:

font: font-style + font-weight + font-size + font-family;

1.5 - Usando google fonts (ex005)

Nós podemos importar milhares de estilos de fontes através desta ferramenta, sem se limitar às oferecidas pelo vs studio code. Basta escolher o tipo de fonte desejada após realizar a filtragem, clicar em ***get font***, ***get embed code***, e por último ***@import.*** Copie este código e cole logo abaixo da tag style, do seu código de css. Para funcionar, após colar o import, na font Family, copie o código que está abaixo do import escolhido.

1.6- Usando fontes externas baixadas (ex006)

Muitas vezes, o seu cliente não vai precisar de mil fontes disponíveis gratuitamente, mas sim, de apenas uma, que ele pode te mandar em pdf, ou às vezes ele não está no google fonts.

Para baixar a fonte, vamos usar o dafonte.com. Pesquisaremos sobre o tipo de fonte desejada, ou, a escolhida pelo cliente. ***CUIDAR SE A LETRA CONTÉM TODOS OS GRIFOS (ACENTUÇÕES), SE NÃO TIVER, NÃO SERA POSSÍVEL ESCREVER NADA ACENTUADO!!!***

Vamos baixar ele em padrão zip, descompactar o arquivo, copiar e colar dentro da pasta do arquivo do nosso site **(se caso criar uma pasta separada, o normal a se fazer, para colocar a fonte, coloca o nome da pasta e uma barra antes da tag dentro dos parênteses da url, antes do nome do arquivo da fonte).** Após isso, vamos adicionar ela ao nosso site. Dentro de style, vamos abrir a seguinte tag:

```
@font-face {
```

Font-family: (nome que você quer que o programa reconheça essa fonte);

Src: url (nome EXATAMENTE IGUAL do arquivo com uma das fontes) format (formato do arquivo), /se caso tiver mais um/ url(nome EXATAMENTE IGUAL do segundo arquivo com outro formato da mesma fonte) format(formato do arquivo)

}

Começamos essa tag abrindo o *@font-face,* assim, automaticamente, ele irá dar todo este código. Nós podemos baixar essas fontes em vários formatos:

-opentype (otf)

-truetype(ttf)

-endedded-opentype

-truetype-aat (Apple Advanced Tipography)

-svg

Dentro então dos parênteses da *url*, vamos colocar o nome do arquivo que copiamos e colamos na pasta (exatamente igual, com ponto, espaço) entre as aspas solitarias. Depois da *url*, vamos colocar o tipo do arquivo com a tag *format*. Nem todos os navegadores suportam todos os tipos de formatos, então podendo variar os formatos seria melhor. Basta colocar uma virgula e mais uma *url tag* seguindo os mesmos passos do primeiro. Finaliza com ponto e vírgula.

1.7 - Capturando as fontes de um site

Para nós podermos pegar uma fonte de algum determinado site, usaremos uma ferramenta que está dentro do google: fonte ninja (salva nos favoritos em dev)

1.8 - Detectando fontes dentro de imagens

Agora, vamos identificar fontes dentro de imagens. Pode ser usado três tipos: what font is, font squirrel e o myfonts.

1.9 - Alinhamento de texto com CSS

Usamos geralmente a tag:

text-align: (alinhamento)

text-indent: (tab/início do parágrafo em px)

1.0 – Seletores Personalizados

Seletores Personalizados é basicamente formatar individualmente dois elementos da mesma classe.

1.1 – Usando ID com CSS (ex007)

Id (identificador) usamos para ***IDENTIFICAR*** um elemento. Como por exemplo: Supomos que em nosso site, temos dois h1, porém, um h1 é meu título principal, os outros dois são somente para nomear conteúdo do site. Dentro deste principal, abrimos uma tag *id* e nomeamos esse h1 específico:

```
<h1 id="principal">Título Principal de Exemplo</h1>
<h1>Título de Conteúdo</h1>
```

Agora vamos supor que eu quero que somente o h1 principal fique centralizado, como vou fazer isso dentro do CSS? Agora que esse h1 principal já está identificado, abrimos uma # dentro do CSS ao lado do seletor de h1, assim, modificando apenas os h1's que tem o identificador "principal", veja:

```
h1#principal {
    text-align: center;
}
```

Tudo que em HTML é id, em CSS é #. E tudo em HTML que é class, em CSS é ".". NÃO PODEMOS REPETIR O MESMO ID DENTRO DO MESMO HTML. SE VOCÊ TIVER DOIS H1'S E QUISER QUE ELES TENHAM A MESMA FORMATAÇÃO, DAI VOCÊ TERÁ QUE COLOCAR ELES DOIS EM CLASS. SÓ É PERMITIDO UM ID POR PÁGINA.

1.2 – As diferenças entre ID e Class (ex007)

Agora que já aprendemos a identificar um elemento, iremos classificar ele. Dentro de um HTML, podemos ter somente um id. Quando quisermos formatar partes específicas de textos dentro do site, que se repetem e seguem a mesma formatação, porém só queremos que elas duas recebam essa formatação específica, igual, teremos que colocar esses dois elementos dentro da tag *class*. **SEMPRE NOMEIE PELA FUNCIONALIDADE, COMO DENTRO DO EXERCÍCIO 007.** Siga a mesma coisa do id, porém mudando a tag de *id* para *class,* e também, alterando a # para um ponto sem colocar o seletor antes. Exemplo: Dentro do exercício, temos vários h2's divididos em básico, intermediário e avançado. Quero que todos os básicos sejam verdes, todos os

intermediários sejam amarelos e todos os avançado sejam vermelhos. Neste caso, ao invés de criar id's específicos para cada um, vamos criar classes. Vejá no código:

HTML:

```
<h2 class='básico'>html básico</h2>
<h2 class='intermediário'>html intermediário</h2>
<h2 class='avançado'>html avançado</h2>
<h2 class='básico'>css básico</h2>
<h2 class= 'intermediário'>css intermediário</h2>
<h2 class='avançado'>css avançado</h2>
```

CSS:

```
.básico {
        Color: green;
}
.intermediário {
        Color: yellow;
}
.avançado {
        Color: red;
}
```

É permitido que um elemento tenha mais de uma classe, e que também, um elemento com id participe de uma classe. Veja:

HTML:

```
<h1 id='principal' class='destaque'>TÍTULO PRINCIPAL</h1>
<h2 class='básico destaque'>html básico</h2>
<h2 class='intermediário'>html intermediário</h2>
<h2 class='avançado'>html avançado</h2>
<h2 class='básico destaque'>css básico</h2>
<h2 class= 'intermediário'>css intermediário</h2>
<h2 class='avançado'>css avançado</h2>
```

CSS:

```
.básico {
        Color: green;
}
.intermediário {
        Color: yellow;
}
.avançado {
        Color: red;
}
.destaque {
        Background-color: purple
}
```

Primeiro edite o geral, depois faça o *id* e depois o class. O *id* sempre irá sobrepor o *class.*

1.3 – Pseudo-Classes em CSS 9 (ex008)

Uma *div* é uma coisa que irá ocupar uma linha do nosso site. Vamos usar ela para entendermos Pseudo-classes. Depois vamos abrir um seletor para uma cor de fundo para elas. Depois vamos alinhas elas no centro. Vamos agora formatar elas, para que tenhamos três quadrados usando as tags *height (altura)* e *width (largura)*, colocando 200x200px. Depois vamos abrir uma *shorthand* de borda (*border*), caracteriza por largura (1px), tipo (solid) e cor (black). Agora vamos colocar uma do lado da outra com a tag *display: inline-block.* Agora vamos dar uma *id* para cada uma delas. Sabemos que # é id, que o "." é classe e agora aprenderemos que : é Pseudo-classes.

As Pseudo-classes têm de estar relacionadas com uma classe ou um elemento. Exemplo do exercício: *div*. Pseudo-classes estão relacionadas ao estado do elemento ou classe, ou seja, ativa, marcado, vazio, habilitado. Agora, dentro do exercício, a Pseudo-classe *hover,* indicando que quando o mouse estiver sobre a nossa classe, neste caso *div*, algo irá acontecer. Nesse caso, mudar de cor. Identificado pela tag: *div:hover + seletores.* Ou seja, todos os elementos do HTML *div,* irão mudar de cor quando o mouse estiver por cima.

EXEMPLO 2:

Para entendermos melhor as *div,* criamos a segunda página que mostra um texto escondido. Para isso, criamos o documento e colocamos o título, um h1 qualquer e uma div aberta pedindo para que seja passado o mouse sobre. Depois, abrimos um parágrafo dentro da *div*. Escrevemos texto escondido. Abrimos um parágrafo fora da *div* e escrevemos fim do exemplo. Agora vamos personalizar.

Abrimos uma tag body e modificamos a fonte. Depois, abrimos uma Pseudo-classes determinando que os parágrafos (filhos) dentro da *div*, deverão estão ocultos, pela tag *div > p {display: none;}.* Logo após, abrimos uma Pseudo-classes para quando o mouse estiver sobre a div com o parágrafo p, para que mostre o texto, com cor de fundo e cor de texto: *div>hover > p {display: block; color; background-color; width300px; }.* Para terminarmos, não tem como o cliente encontrar esse texto escondido, então precisamos abrir uma Pseudo-classes sobre alguma palavra aparente, para que quando passe sobre ela, mostre a palavra escondida. Tag: div:hover { color; }. Fim do exercício.

1.4 – Pseudo-Elementos em CSS

Vamos aprender mais duas Pseudo-classes em CSS. Abriremos o mesmo ex (008) e personalizaremos uma nova página (pg03). Dentro do exercício, vamos abrir uma lista sem ordem e colocar uma palavra que quando clicada irá levar a algum link. Depois vamos colocar a tag link na palavra. Vamos mudar a fonte dela. Vamos personalizar a tag link agora: escolher uma cor para o link, tirar o *text-decoration* (sublinhado do link) e colocar em negrito com a tag *font-weight.* Logo após, vamos abrir a nossa nova Pseudo-classe: *:visited.* Se caso o link já tiver sido visitado, aparecerá com outra cor. E a nossa outra Pseudo-classe: *:active,* dizendo que quando aquela coisa for clicada, mudará de cor (lembrando que as Pseudo-classes definem estados dos elementos selecionados). Todas essas novas Pseudo-classes são para o nosso link, a tag *a*.

Agora vamos falar de Pseudo-Elementos. Eles podem agir diretamente no conteúdo. Se eu quiser que todo o link tenha uma indicação que ele é um link (uma setinha antes e uma depois), usaremos esses pseudo-elementos, que atuam no conteúdo. Representamos ela por dois pontos repetidos: "*::*". Com os pseudo-elementos, consigo fazer com que coisas antes e depois apareçam no conteúdo. Então vamos trabalhar com o link: Queremos indicar que isso é um link, então vamos colocar uma seta antes e uma depois do link, apontadas para dentro. Vamos usar então os pseudo-elementos, já que não queremos alterar uma classe inteira, e sim somente um elemento. Fica assim:

a::after {

contente: (o que eu quero que apareça depois?) – no ex, copiei o emoji da seta)

text-decoration: none; (não quero que essa escrita tenha nenhuma maracação)

font-weight: normal;

color: darkgray;

}

Se eu quiser que apareça depois uma seta agora apontando para direita, ela terá que vir antes do link, então substituímos o *after* por *before..* No *content,* determino o que quero que apareça ANTES do link., abrindo assim um novo seletor.

Agora vamos supor que eu tenha um link mais importante que os outros, querendo assim uma formatação específica para ele. Eu pego ele e abro uma *class*, e determino o pseudo-elemento dessa *class,* veja:

.especial::after/ (ou) before {

Faço a formatação que eu quiser aqui. Se eu quiser que somente esse tenha as setas, por exemplo, formato assim somente com ele no meu código.

}

Aprendemos durante este módulo então estes seletores personalizados:

- # = id

- “.” = class

- “:” = pseudo-class

- “::” = pseudo-elements

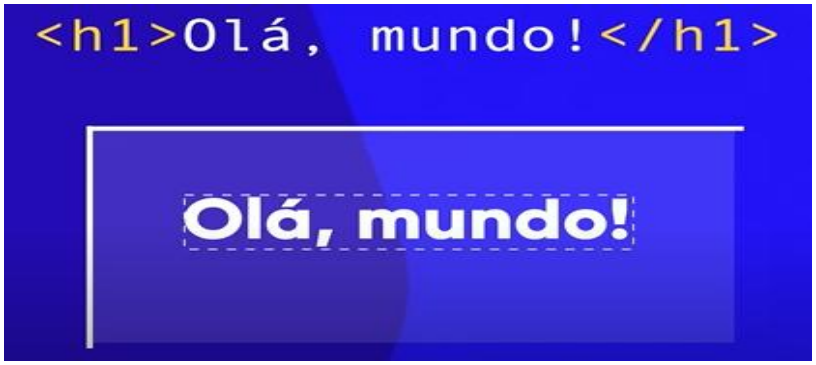
- > = children (filho de alguma coisa)

1.0 – Modelo de Caixas

1.1 – Modelos de Caixa: primeiros passos

É basicamente uma maneira de organizar o seu site. Tudo que é exibido em HTML é uma caixa. Nós também podemos colocar uma caixa dentro da outra, chamado aninhamento.

Vamos usar de exemplo a tag *<h1>Olá, mundo!</h1>.* Todo o elemento visível dentro do site é uma caixa. Isso é a caixa desse h1:

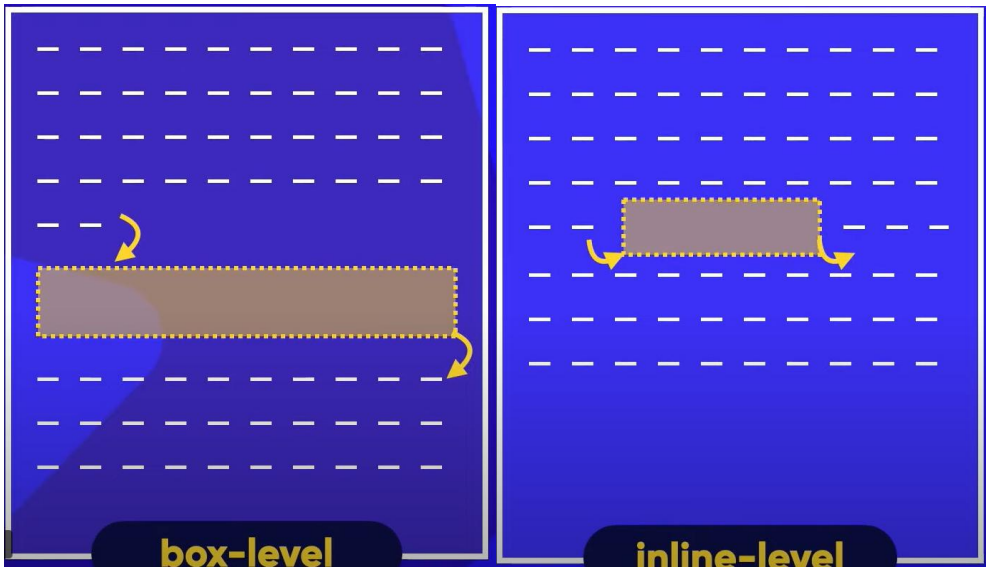


Uma caixa te um tamanho, determinada por duas medidas: altura, que chamamos de hight, e largura, width:

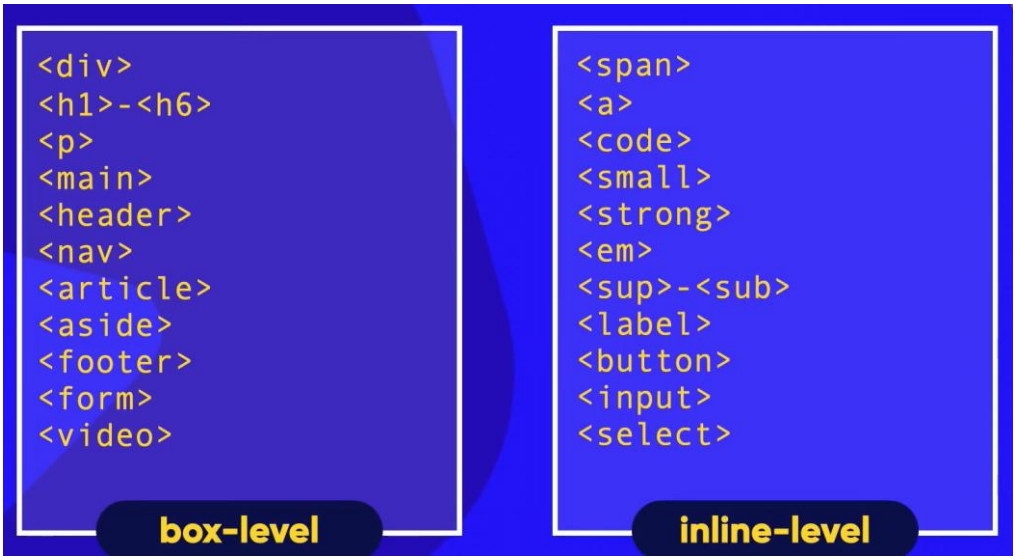
Nós podemos também podemos traçar uma linha envolta desse tracejado, que é o conteúdo da caixa, o qual chamamos de borda (border). Por padrão, a linha fica muito grudada no conteúdo, então podemos dar uma desafogada usando o padding:

Nós podemos também criar um espaço externo: a margem (margin). Outra coisa que podemos dentro da margem, o tracejado amarelo, que fica fora da borda.

Existe dois tipos de caixa: box-level e inline-level. Uma caixa do box-level, ela ocupa a largura total da tela sempre em uma linha nova. Agora o inline-level, ele pula para o lado, desenha a caixa, não ocupa a largura inteira, e não começa o conteúdo somente no próximo parágrafo. O tracejado seria vários parágrafos de conteúdo, e nós queremos colocar uma caixa dentro dele, e assim, mostrando a diferença.



Como usar? Olhe a imagem



1.2 – Modelos de Caixa na prática (p1 ex009)

Dentro do exercício pronto, nós vamos criar um h1 e um parágrafo para podermos exemplificarmos as box-level (ocupam uma linha inteira e sempre irão começar na linha seguinte). Porém, dentro do parágrafo, nós vamos abrir uma tag de link, e usaremos esse link para exemplificarmos a box inline-level.

Para podermos entendermos então sobre as caixas, vamos inspecionar o nosso próprio site. Olhar e mexer nas “caixinhas”, e ele vai mostrar todas as configurações desta caixa dentro do nosso site. Ele também irá mostrar a configuração padrão de CSS feita pelo USER AGENT (configurado automaticamente pelo nosso navegador, o user agent).

Vamos fazer depois, umas leves mudanças de estilo dentro do site. Primeiro na altura e na largura: 300x300 (height e width) assim, fazendo a caixa. Vamos colocar o background-color também. Dentro do próprio inspecionar, clicando no css no código desejado, podemos ver como ficaria com outras medidas. Vamos colocar uma borda: border-width (largura da borda) em 10px. Depois, um border-style (estilo da borda) em solid, ou seja, uma cor sólida. E por último, um border-color (cor da borda).

Agora vamos criar uma configuração de estilo para a nossa ancora, o link. Border-width em 10px, border-style em solid, e border color em uma cor destaque. Nós podemos configurar a borda em dashed (pontilhado), dotted (pontilhada), o double (linha dupla) e o groove (3d).

E por último, vamos colocar uma medida de padding, aquela sobre entre o conteúdo e a borda. Sempre começamos em cima, e fazemos um sentido horário, ou seja, cima, direita, baixo e esquerda, usando: padding- top, right, bottom ou left.

1.3 – Modelos de Caixa na prática (p2) (ex009)

Vamos começar configurando a margem em 10px, seguindo aquele sentido horário. Com a margem, conseguimos separar itens um dos outros, de acordo com o local. Para você centralizar a caixa, usamos a tag margin: auto.

Nós também podemos criar um outline, que fica entre a borda e a margem, mas dentro da margem. Vamos colocar ele com o código: outline-width (largura do outline), outline-style: dashed (tracejado), em 5px e outline: color.

SHORTHANDS:

borda, ao invez daqueles três (border-width+style+color), colocamos somente border: width style color.

Padding: seguindo a ordem. Se caso repetir, no caso do exercício, basta somente colocar uma vez, senão, sentido horário (cima, direita, baixo e

esquerda). Você também pode colocar dois valores, o primeiro para o topo e baixo da caixa, e o outro para os lados.

Margem: margin: cima direita baixo esquerda. Pode simplificar para um valor também se repetir. Se caso eu quiser centralizar, nos valores da direita e da esquerda eu coloco *auto*.

Outline: largura estilo e cor.

Para você transformar algo em box-level ou inline, basta colocar display: block ou inline.

1.4 – Grouping tags em HTML5 (ex010)

Para que serve esse conceito das caixas? Todo o site usa isso. Quando aplicamos isso no site. Vamos aprender a agrupar as caixas de maneira inteligente e organizada.

Sempre se teve duas tags para elas: a div (box-level) e spam (inline-level). Era normal o pessoal pegar e criar varias div's uma dentro da outra e nomear elas com id's separados para cabeçalho, menu, rodapé.... Mas o html5 evoluiu.

CABEÇALHO:

Header (cabeçalho)

Nav (menus do cabeçalho, links)

CONTEÚDO

Main (onde está inserido o conteúdo)

Section (seções, ex: assuntos, notícias. Nomeados por id's)

Section > id notícias > article (notícia) aqui vai um h1, um p...> aside (conteúdo relativo ao conteúdo de antes, tipo o escritor)

RODAPÉ

footer

1.5 – Sombra nas caixas

Utilizando o modelo anterior (site de notícias) para a base deste desenvolvimento de sombras, vamos colocar uma pequena sombra dentro do nosso menu de navegação, que é determinado pela tag *nav*. Para isso, vamos encontrar a seleção de estilo para o menu *nav* e iremos adicionar um *box-shadow* (sombra para caixa). Depois iremos determinar três medidas em pixel para as direções da sombra nesta ordem: deslocamento horizontal, deslocamento vertical, espalhamento e cor da sombra. Isso é uma shorthand. Ficaremos com esse código dentro do menu de estilização de nav:

Box-shadow: 1px 1px 1px black;

As sombras não podem ser uma cor sólida, e também não podem ser coloridas. Para isso, diminuímos a transparência do preto dentro do visual studio code.

1.6 – Caixa com vértices arredondados

Iremos utilizar o mesmo exercício para estudar esta função. Para fazermos o arredondamento dos cantos de um quadrado, usaremos a seguinte tag:

Border-radius: (3 formas de expressarmos a shorthand)

1 – Porcentagem (50% = bola)

2 – Dois valores equivalentes à canto superior esquerdo e canto inferior direito na primeira pixelagem, e na segunda pixelagem referente ao canto superior direito e canto inferior esquerdo. (10px 0px)

3 – Valor único para todas as bordas. (10px)

BOLA: Para fazermos uma bola, vamos abrir uma tag *div* dentro do body. Em css, igualaremos a altura e a largura, definiremos uma cor de fundo, depois adicionamos o border-radius em 50%. Pronto, temos uma bola. Este é o código de exemplo:

html

```
<div id=””bola></div>
```

css

```
div#bola {
    height: 100px;
    width: 100px;
    background-color: white;
    border-radius: 50%
}
```

1.7 – Bordas decoradas

Serve basicamente para colocar imagem nas bordas. Não é necessário se preocupar. Se caso precisar, está na aula 7 do capítulo 16 do módulo 2 do curso.

1.8 – DESAFIO DO MÓDULO 2.

O projeto desenvolvido está dentro da pasta de desafios.