**Curso de HTML5 e CSS3 – Guanabara – Módulo 2**

**Capítulo 13**

Cuidado com contraste fundo preto e fonte branca, principalmente se tem muito texto.

Usando hexadecimal nas cores

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

Em hexadecimal: 00 (*red*) 00 (*green*) FF (*blue*)

Resultado: cor máxima de azul.

Usando HSL nas cores

*Hue, saturation, lightness/luminosity*

RGBA e HSLA, alpha, transparência.

Cores primárias: amarelo, vermelho e azul.

Cores secundárias: laranja, violeta e verde.

Cores terciárias, misturas das cores primárias e secundárias.

Uma paleta de cor, vai ter de 3 a 5 cores. Média de 4.

Cores análogas, são vizinhas.

Cores análogas relacionais, escolha duas próximas e pulo uma no círculo cromático. Fica interessante.

Cores intercaladas, pulo uma no círculo cromático a cada nova cor.

Cores tetrádicas, escolhe duas cores e suas complementares, respectivamente.

Monocromia, modifica a saturação e a luminosidade. Efeito de degradê.

Para fazer efeito degradê no CSS, usar a propriedade background-image e a função interna linear-gradient().

Dentro da função colocar direção e cores separados por vírgulas. Nas direções posso colocar o ângulo como 90deg, 120deg e -30deg. Exemplo:

body {

background-image: linear-gradient(to right, white, blue);

}

Posso usar radial-gradient também. Exemplo:

body {

background-image: radial-gradient(circle, white, blue);

}

Dá para colocar a porcentagem ao lado de cada cor também:

body {

background-image: radial-gradient(circle, White 30%, blue);

}

Configuração global em CSS:

\* {

}

Propriedade CSS:

background-attachment: fixed; - deixa o fundo fixo.

Tag HTML5: <main> = onde vai o conteúdo principal.

**Capítulo 14**

Quando se cria uma fonte, começa pela letra “x” para denominar a altura “x”, que vão ser a altura das minúsculas.

Temos a altura das maiúsculas.

O “vazamento” para cima das minúsculas recebe o nome de ascendente.

Quando o “vazamento” é para baixo (cauda da fonte), descendente.

Já a altura toda, é chamada de corpo (soma de **todas as alturas**).

A serifa cria uma linha imaginária que o nosso cérebro percebe e acaba lendo “automaticamente”. Facilita a leitura das palavras, pois o cérebro não lê as coisas letra por letra. Muito usada em papel, para tela sem serifa acaba sendo melhor.

Anatomias geométricas dos tipos:

* Hastes;
* Filete (união);
* Arco;
* Esporão (base);
* Vértice;
* Terminal.

Anatomia humanizadas dos tipos:

* Braço (cima);
* Perna (baixo);
* Pé (é onde tem a serifa, quando há);
* Espinha;
* Barriga;
* Olho;
* Orelha;
* Cauda.

Glifo = símbolo (de uma família tipográfica).

Categorias de Fontes:

* Serifada
* Sem serifa
* Mondoespaçada
* Handwriting (script)
* Display (comemorativas)

Recomenda-se usar fonte sem serifa para leitura de textos longos em telas.

Duas palavras dentro de font-family devem entrar entre aspas. Exemplo:

font-family: ‘Courier New’;

Sempre termine a combinação segura de famílias de fontes em CSS com uma genérica.

Tamanhos de fontes em CSS

Temos as Medias Absolutas:

* cm;
* mm;
* in;
* px;
* pt;
* pc;

E Medidas Relativas, algumas das mais usadas são:

* em (relativo tamanho da altura “M” maiúscula);
* ex (relativo ao “x” de uma fonte);
* rem (igual ao em porém relativo ao “root”, fonte que está configurada no body);
* vw (porcentagem da largura do viewport);
* vh (porcentagem da altura do viewport);
* %.

Trabalhar com **em**. W3C recomenda trabalhar com px e em.

Geralmente o tamanho da fonte em navegadores está em 16px.

Já 16px normalmente equivale a 1em.

Guanabara recomenda usar o em.

Variações de peso em fontes, nome (lighter, normal...), numérica (de 100 a 900). Se eu for com CMD + ENTER no Mac pelo VS Code ele abre as opções disponíveis.

No CSS, text-decoration permite sublinhar a fonte.

No CSS, shorthand simplifica uma declaração, condensa ela. Para fonte, usar font, a ordem a ser especificada é:

* font-style;
* font-weight;
* font-size;
* font-family.

Separa por espaço (quando tem mais de uma família de fonte, as fontes podem ser separadas por vírgula no último “item”, font-family).

No Google Fonts, usar as fontes no CSS pelo @import.

É possível baixar fontes e usar no CSS, para isso colocar na folha de estilo @font-face. O VS já disponibilizar uma opção que abre font-family e src juntos.

Formatos oft e ttf. Como depende do navegador, alguns funcionam melhor com otf e outros com ttf, é recomendado colocar os dois no cógido.

Quando usamos o @font-face, também é preciso especificar o formato das fontes, sendo:

* Opentype;
* Truetype;
* Embedded-opentype;
* Truetype-aat (Apple Adavanced Typography);
* Svg.

Existem fontes opentype que são formato ttf.

Exemplo usando o @font-face:

font-family: ‘Love’;

src: url(‘love story.otf’) format(‘opentype’), url(‘love story.ttf’) format(‘truetype’);

font-weight: normal;

font-style: normal;

Depois de preencher src e os demais atributos, **nomear a sua fonte**.

Valores para o atributo text-align em CSS:

* left;
* right;
* justify.

text-indent coloca um espaço no início no parágrafo, colocar a quantidade como valor. Exemplo:

text-indent: 50px;

**Capítulo 15**

A recomendação é usar o *id* uma única vez. Quando for necessário mais de uma vez, usar a *class*.

Dar preferência a nomear *id* e *class* com relação a função deles e não a forma.

Em HTML a tag <span>, é um container genérico em linha, pode ser usada para agrupar elementos para fins de estilo.

É possível aplicar mais de uma *class* a um elemento para isso separar as duas classes com um espaço. Exemplo:

<h1 class = “avançado destaque”>Título</h1>

É possível usar um *id* e *class* em um elemento:

<h1 id=”principal” class=”avancado”>Título</h1>

Vale lembrar que o *id* se sobrepõe, caso o *id* e *class* estejam usando as mesmas propriedades.

Uma <div> é um espaço qualquer, uma divisão.

A propriedade *border* em CSS cria uma borda.

A propriedade *display*, define como os elementos vão ser apresentados na tela.

As pseudo-classes iniciam com “:”. Elas precisam estar relacionadas a um elemento ou uma classe. São relacionadas a um estado de um determinado elemento, como ativo, vazio, hover etc.

Exemplo de seu uso:

div:hover {

background-color: yellow;

}

Se eu usar o símbolo > em CSS significa que faço referência a um elemento dentro do outro. Por exemplo:

div > p {

}

Nesse caso estou lidando com o parágrafo dentro da div.

Alguns usos de pseudo-classes:

:visited – quando um link foi clicado.

:active – quando um link é clicado (ato de clicar).

Já os pseudo-elementos atuam diretamente no conteúdo. Usa-se dois pontos para eles “::”.

Exemplos:

::after

::before

Seletores personalizados

# = id

. = class

: = pseuco-class

:: = pseudo-element

> = children

**Capítulo 16**

HTML e CSS trabalham com modelos de caixa.

Seguem alguns elementos de anatomia de uma caixa.

Diagram

Description automatically generated

Diagram

Description automatically generated

Diagram

Description automatically generated

A *outline* fica fora da caixa (elemento), diferente de *border*.

Eixstem dois tipos de caixa:

box-level;

inline-level.

O box-level, sempre se inicia em uma linha nova e ocupa a largura total de uma tela.

Já o inline-level, continua na mesma linha, ocupa só o tamanho que precisa para o preenchimento e após o término também não pula linha.

Graphical user interface

Description automatically generated

Text

Description automatically generated

No *dev tools* do Chrome *user agent* é o navegador.

Elementos que tem display: block;, são box-level.

A ordem do padding é *top, right, bottom, left*.

margin: auto; centraliza as “caixas”. Para texto usar text-align: center;. Posso usar auto também onde entra esquerda e direita para centralizar.

*Shorthand* para border: (tamanho, tipo, cor).

Na *shorthand* quando eu uso duas medidas ele usa as duas primeiras para *top* e *bottom* e as outras para *right* e *left*.

margin: 10px 20px;

10px para altura e base e 20px para direita e esquerda.

Div e span são tags de agrupamento. Hoje com HTML5 é melhor usar as tags semânticas como tags de agrupamento. Os mecanismos de busca valorizam mais o uso dessas tags.

Na propriedade box-shadow, colocamos primeiro o valor para deslocamento horizontal, vertical, distorção, espalhamento e cor. Dá para colocar o *inset* na frente também para fazer a sombra interna. A distorção é o *blur* e o espalhamento o *spread*.

box-shadow: inset 2px 2px 4px 0.4px black;

Ordem do border-radius: topo esquerda, topo direita, base direita e base esquerda. Se eu colocar apenas duas medidas a primeira é para topo esquerda e base direita e a segunda é para topo direita e base esquerda.

Dá para usar porcentagem no border-radius também.

Para colocar borda personalizada em CSS usar a propriedade border-image-source. Outras propriedades também são necessárias para melhor visualização. Segue exemplo:

border-image-source: url('borda.png');

border-image-slice: 20;

border-image-repeat: repeat;

Shorthand para borda personalizada é border-image. A ordem acima deve ser seguida.

É preciso manter também a propriedade border para usar border-image.

**Capítulo 17**

É importante colocar o @charset no início do documento de estilo (CSS).

@charset “UTF-8”;

Posso colocar predefinições para o HTML usando :root

Posso definir cores, fontes etc

São declarações de variáveis em CSS.

Usar max-width e min-width para definir o tamanho da largura do site. É uma maneira mais básica, mas dependendo do projeto serve bem.

list-style-position – define onde os bullet points em uma lista iniciam.

columns – define o número de colunas em uma lista.

list-style-type – define o estilo para o bullet point. Isso não é compatível com todo navegador.

Posso colocar margin em nagtivo para fazer ajustes finos.

Como usamos os vídeos do Youtube e do Vimeo através de um <iframe>, é preciso fazer alguns ajustes para colocar o vídeo numa posição legal na tela.

A tag <iframe> foi envelopada como uma <div>, criado uma *class* e aplicado os passos abaixo no CSS:

div.video {

background-color: var(--cor5);

margin: 0 -20px 30px -20px;

padding: 20px;

padding-bottom: 50%;

position: relative;

}

div.video > iframe {

position: absolute;

top: 5%;

left: 5%;

width: 90%;

height: 90%;

}