Anotações modulo 2 curso de html e css

Comentários para os vídeos: Conheci o Guanabara em 2018, comecei fazer o curso e desitir, hoje estou fazendo em 2023, me arrependo muito por não ter continuado, más agradeço por ter outra oportunidade. Por favor compartilhe e deixem o like, esse curso e o Professor merece. amo todos vocês e também os alunos que me ajuda muito nos comentários... pessoal por favor curta todos os videos e comente. obrigado equipe e guanabara. meu sonho realizado.janeiro de 2023

Como colocar cores no sites:

Coloque sempre o link abaixo de <title> cores no sites</title> para colocar em todas as páginas: no topo: <link rel="stylesheet" href="css01.css">

Para colocar nos títulos e tesxtos especifico:

Representação por nome: <h2 style="background-color: #000100; color: aqua;">Exemplos de cores 1</h2>

 <!--Representação por códigos hexadecimal -->

                <!-- decima: 0123456789-->

                <!--hexa decimal : 0123456789 a b c d e f-->

    <h2 style="background-color: #4b4bf9; color: ;">Exemplos de cores 2</h2>

<h2 style="background-color: rgba(247, 175, 21, 0.692); color: #33af33c2;">Exemplos de cores 3</h2>

<h2 style="background-color: #7cf94b5d; color: #184de0c0;"> coores editaveis no html</h2>

Harmonização de cores :

**Cores primarias:** amarelo, vermelho, azul (cores princiapis

**Cores secundarias:** laranja, violeta, verde (cores principais

**Cores tercearia:** amarelo-esverdeado , vermelho-alaranjado, azul-arroxeado, azul-esverdeado (mistura de cor principal com qualquer outras cores.

Cores frias: azul,verde, roxo forte

Cores quentes: amarelo,laranja,vermelho, roxo fraco

Cores complementares:

Ferramentas e sites parea gerar cores: adobe color : [https://color.adobe.com/pt/create/color-wheel#](https://color.adobe.com/pt/create/color-wheel)

<https://paletton.com/#uid=1000u0kllllaFw0g0qFqFg0w0aF>

**Como colocar degrade código:**

<style>

        \* { \*/configuraçoes globais\*/

            min-height: 100%;

        }

        body {

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

        }

    </style>

**Como cores ,degrade ,fundo nas páginas código:**

@charset "UTF-8";

\* {

    font-family: 'Franklin Gothic Medium', 'Arial Narrow', Arial, sans-serif;

    height: 100;

**Anatomia das letras( fonts)**

}

body {

    background-image: linear-gradient(to right, blue, #2000, #fff399);

}

main {

    background-color: white;

    width: 600px;

    border-radius: 10px;

    box-shadow: 5px 5px 15 px rgb(34, 4, 4);

    padding: 10px;

    margin: auto;

}

h1{

    color: aqua;

    text-align: center;

    text-shadow: 1px 1px 2px orange;

}

p {

    text-align: justify;

}

**Entendendo os códigos de fontes:**

<!DOCTYPE html>

<html lang="pt-br">

<head>

    <meta charset="UTF-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>fonte em css</title>

<style>

        body

        /\*medidas absolutas no site html:

        (cm,mm,in,px,ptpc)

        medidas relativas:

        (em,ex,rem,viu,vh,%)

        16px é igual 1em

        obs: sempre use o tamanho em e px\*/

        {

            font-family: Arial, Helvetica, sans-serif;

            font-size:2em;

            font-style: italic; /\* fonte para deixar italico\*/

            font-weight: normal; /\*  pesos para fontes codigos: lighter,normal,bold, bolder

            -fonte weigh para fazer o peso da fonte

            na pagina

            -fonte widt para fazer largura do site

            -fonte height para fazer altura do site \*/

            /\*obs: o shortrend font é uma forma de icurtar o codigo da fonte

             , seguindo a ordem certa: fonte-style, fonte-weight.font-size,font-family\*/

             font: italic bolder 2em arial,;

          }

          h1{

            font-family: italic arial, normal 2em ;

          }

    </style>

</head>

<body>

    <h1>Trabalhando com fontes</h1>

    <p>Lorem52 ipsum dolor sit amet consectetur adipisicing

         elit. Fugiat, quis a laudantium

          reprehenderit doloremque atque dolores natus

           suscipit animi sequi, dolor numquam voluptatem

        perferendis quia odit voluptatum,

        porro distinctio voluptas.</p>

        <h2>subtitilu do exercicio 01</h2>

        <p>Lorem ipsum dolor sit amet consectetur adipisicing

            elit. Fugiat, quis a laudantium

             reprehenderit doloremque atque dolores natus

              suscipit animi sequi, dolor numquam voluptatem

           perferendis quia odit voluptatum,

           porro distinctio voluptas.</p>

           <h2> Subtitulo do exercício 02</h2>

           <p>Lorem ipsum dolor sit amet consectetur adipisicing

            elit. Fugiat, quis a laudantium

             reprehenderit doloremque atque dolores natus

              suscipit animi sequi, dolor numquam voluptatem

           perferendis quia odit voluptatum,

           porro distinctio voluptas.</p>

</body>

</html>

</head>

<body>

    <h1>Trabalhando com fontes</h1>

    <p>Lorem52 ipsum dolor sit amet consectetur adipisicing

         elit. Fugiat, quis a laudantium

          reprehenderit doloremque atque dolores natus

           suscipit animi sequi, dolor numquam voluptatem

        perferendis quia odit voluptatum,

        porro distinctio voluptas.</p>

Usando fontes do google:

<!DOCTYPE html>

<html lang="pt-br">

<head>

    <meta charset="UTF-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Usando fontes do google</title>

    <style>

       @import url('https://fonts.googleapis.com/css2?family=Reggae+One&display=swap');

        body {

            font-family: 'great', Helvetica, sans-serif;

            font-size: 1em;

            color: black;

        }

        h1 {

            font-family: 'Reggae One', 'Times New Roman', Times, serif;

}

h2{

    font-family: Arial, Helvetica, sans-serif;

    font-size: 2em;

}

    </style>

</head>

<body>

    <h1>Meu titulo</h1>

    <p>Lorem ipsum dolor sit amet consectetur adipisicing elit.

         Nostrum repellendus minima excepturi exercitationem

         laudantium sed sint similique,

        consequuntur tenetur reiciendis. ipsum dolor sit amet consectetur adipisicing elit.

         Repudiandae enim id alias! Ipsam, quaerat?</p>

         <h2> sub titulo</h2>

@font-face { font-family: 'Love'; src: url(love\ larry\ ttf.ttf); /\*use o ctrl + espaço para selecionar o arquivo\*/ font-weight: normal; font-style: normal;

Colocando Fontes de outros sites diferentes no html:

<style>

        @font-face {

            font-family: 'monster';

            src: url(fontes/idroid.otf);

            font-weight: normal;

            font-style: normal;

        }

        h1 {

            font-family: 'monster', 'Times New Roman', Times, serif;

}

h2{

    font-family: Arial, Helvetica, sans-serif;

    font-size: 2em;

}

    </style>

        /\* tipos de fontes pra escolher no @font-face:

        -opentype: otf

        -truetype ttf

        -embedded- opentype

        -truetype-att (aple advance typegraphy)/// \*/

        body{

            font-family: Arial, Helvetica, sans-serif;

        }

        h1{

            font-family: 'monster';

            font-size: 3m;

            font-weight: normal;

        }

        p {

            font-family: 'monster';

        }

    </style>

Alinhamento de textos:

Basicamente temos quatro tipos de Alinhamento de Texto: - Alinhamento à esquerda - Alinhamento à direita - Centralizado - Justify Os textos no HTML (p, h1, h2 e etc.) já vem por padrão alinhados à esquerda da tela, mas se quisermos mudar esse alinhamento, devemos usar a Propriedade de Estilo "text-align", os valores mais comuns dessa propriedade são: center, right, left e justify O valor "center" alinha o texto no centro da tela, o valor "right" alinha o texto na parte direita da tela, o valor "left" alinha o texto na parte esquerda da tela... Sendo que o texto de PARÁGRAFOS normalmente tem um alinhamento nos DOIS LADOS, à esquerda e à direita... que é a função fundamental do valor "justify". Existe uma Propriedade de Estilo chamada "text-indent", que é responsável por dar aquele recuo no início de um parágrafo: Geralmente colocamos o valor dessa propriedade como "30px" de recuo.

Mostrar menos

16

Responder

<style>

        body {

            font: normal 16px arial, verdana, serif;

        }

        h1 {

            text-align: center;

        }

        h2 {

            text-align: center;

        }

        p {text-align: justify ;

          text-indent: 10px;

        }

    </style>

</head>

<body>

    <h1>Tipos de alinhamento</h1>

    <h2>Subtitulo</h2>

    <p>Lorem ipsum dolor  sit amet consectetur,

         adipisicing elit. Fuga earum quisquam

         minus laboriosam, sed nobis dolorem

         distinctio quidem voluptatibus totam cum,

          rem dignissimos!

Seletor no css e html:

Código no css:

@charset "utf-8";

/\* -quando for fazer uma

ponte entre html e css, usa 'id' no html e # no css

 - quanddo  usa 'class' no html, no css usa '.'\*/

Obs: a tag id, não pode ser usado mais de uma vez no html, somente uma vez para identificação.

A tag class, pode ser usado varias vezes, para identificação

Exemplos:

@charset "UTF-8";

\* {

    font-family: 'Franklin Gothic Medium', 'Arial Narrow', Arial, sans-serif;

    height: 100;

}

body {

    background-image: linear-gradient(to right, blue, #2000, #fff399);

}

main {

    background-color: white;

    width: 600px;

    border-radius: 10px;

    box-shadow: 5px 5px 15 px rgb(216, 213, 17);

    padding: 10px;

    margin: auto;

}

h1{

    color: aqua;

    text-align: center;

    text-shadow: 1px 1px 2px orange;

}

h2{ color: blue;

    text-align:justify ;

    text-shadow: 1px 1px 1px;

}

p {

    text-align: justify;

body {

    background-color: aqua;

    font-family: Arial, Helvetica, sans-serif;

}

h1 {

    color: blue;

}

h1#principal {

    text-align: center;

    color: black;

    background-color: blueviolet;

}

h2 {

    color: rgb(42, 116, 27);

}

p {

    color: crimson;

}

Código no html:

<!DOCTYPE html>

<html lang="pt-br">

<head>

    <meta charset="UTF-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>seletores personaçizados</title>

    <link rel="stylesheet" href="estilos.css">

</head>

<body>

    <h1 id="principal">Criando sites com html e css</h1>

    <h1>aprendendo html</h1>

    <h2>html intermediário</h2>

    <p>Lorem ipsum dolor sit amet consectetur adipisicing

         elit. Voluptate a modi saepe ut, non sed, cumque

          veniam numquam quia nobis officiis mollitia

          alias, quasi

        laboriosam! Ea at exercitationem perspiciatis

        delectus?</p>

    <h2>html basico</h2>

**Pseudo classes com ‘div’ e ‘hover’**

No HTML, temos a tag "<div></div>", ela não tem uma função específica, serve apenas para demarcar um espaço qualquer dentro da nossa página. Podemos escrever textos dentro das tag div's... Como padrão, se tiver mais de uma, as div's ficam dispostas uma em cada linha, uma embaixo da outra, sendo que elas ocupam uma área que pega a largura inteira do navegador, de ponta à ponta. Exemplo: <body> <div>01</div> <div>02</div> <div>03</div> </body> Podemos mudar a altura e largura das div's com as Propriedades de Estilo "Height" e "Width", também conseguimos mudar a posição dos textos nas div's com a Propriedade text-align, sendo que podemos adicionar bordas às div's com a Propriedade "border" com o valor "solid" Nota: Ao utilizar a Propriedade de Estilo "border", devemos primeiro inserir o valor da espessura da borda em px, depois o tipo de borda (ex: solid) e posteriormente a sua cor. Exemplo de shorthand para bordas em CSS: " border: 1px solid black; " Como foi dito anteriormente, por padrão as div's ficam alinhadas uma em cima da outra, mas se quisermos que fiquem uma do lado da outra, devemos usar a Propriedade de Estilo "display" com o valor "inline-block", que seria um alinhamento de bloco em linha, assim fica tudo na mesma linha. Agora, referente à Pseudo-classes , elas precisam e são relativas a um elemento ou uma classe, elas são relacionadas ao ESTADO de um determinado ELEMENTO . No CSS, as Pseudo-classes são representadas por "dois pontos" ( : ) Para usar as funcionalidades de Pseudo-classes, no código CSS, devemos abrir um seletor personalizado para o elemento que queremos determinar um estado, digitando "dois pontos" ( : ) e especificando o estado desejado, abrindo chaves posteriormente. Alguns exemplos de ESTADOS de elementos (Pseudo-classes) são: Ativo, Marcado, Vazio, Habilitado e etc. Existe um estado (Pseudo-classe) chamado "hover" (hover é uma Pseudo-classe que é compatível com vários elementos), que significa: "quando eu passar o mouse por cima", quando estamos com alguma coisa por cima dele. Exemplo de elemento com Pseudo-classe abaixo: div:hover { background-color: brown; } Uma coisa legal que pode ser feita é inserir parágrafos (<p>) dentro de div's. Isso é enquadrado como um elemento dentro de outro elemento, e nesses casos, o elemento que está dentro é considerado " filho " ( children ) daquele que está fora (pai). Em seletores CSS, se quisermos personalizar os elementos que estão dentro de outros elementos ( children ), devemos indicar o elemento "pai" e usar o símbolo de maior " > " para indicar o elemento filho ( children ) imediato a ele. Exemplo: div > p { display: none; } A propriedade de estilo " display " com o valor " none " do exemplo acima faz com que os parágrafos que estejam dentro de div's ( ">" children ) fiquem com sua exibição oculta. Algo interessante que pode ser feito após isso, utilizando a Pseudo-Classe " hover ", é fazer com que o texto escondido apareça ao passarmos o mouse em cima da div. Para isso, abrimos um seletor para a div com a Pseudo-Classe hover (div:hover) e indicamos o seu filho (p) com o símbolo " > ", exemplo: " div:hover > p { " e inserimos dentro a Propriedade de Estilo " display " com o valor " block ". Dessa forma, ao passarmos o mouse sobre a div, o texto escondido contido na tag "p" filho irá aparecer. Exemplo: div:hover > p { display: block; color: white; background-color: red; width: 300px; }

Exemplo: <!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>aprendendo hover</title>

    <style>

        body {

            font-family: Arial, Helvetica, sans-serif;

        }

        div >p {

            display: none;

        }

        div:hover > p {

            display: block;

            background-color: blueviolet;

            color: blue;

            width: 300px;

        }

        div:hover {

            color: rgb(30, 153, 210);

        }

    </style>

</head>

<body>

    <h1> exemplo de hover</h1>

    <p>passe o mouse sobre o texto abaixo</p>

    <div>

        passe o mouse aqui

        <p>texto escondido0</p>

    </div>

    <p>fim do texto</p>

</body>

</html>

**Pseudo classe continuação com pseudo elemento:**

Outros exemplos de ESTADOS de elementos (Pseudo-classes) são: Visited e Active A pseudo-class "visited" é bastante usada em âncoras/links, ela serve para ativar uma configuração de estivo quando uma unidade de elemento for visitada. Já a pseudo-class "active" é um estado de elemento que serve para ativar uma configuração de estilo quando uma unidade de elemento for ATIVADA, ou seja... Quando clicarmos nela por exemplo. Agora, referente aos Pseudo-elementos , eles podem agir nas classes, nos elementos, nos id's e etc... Eles mexem diretamente no CONTEÚDO PERIFÉRICO DO ELEMENTO . No CSS, os pseudo-elementos são representados por " dois pontos " DUPLOS ( :: ) Suponha que queremos que todo link no final tenha uma indicação de que é um link, podemos abrir um seletor personalizado para links e usar o pseudo-elemento ( :: ) " after " para fazer isso. Exemplo de uso de pseudo-elemento: a::after { content: ' ⇦'; color: darkgray; font-weight: normal; } O código acima faz com que manipulemos o conteúdo periférico posterior (pseudo-elemento "after") de links, a Propriedade de Estilo " content " detém o valor que foi determinado para aparecer logo em seguida de todo link que inserirmos no site (seta para esquerda). Além disso, as outras propriedades de estilo do exemplo configuram o estilo de cor e peso desse conteúdo periférico posterior. Também existe o pseudo-elemento " before ", que funciona como o "after" mas de forma oposta, fazendo com que manipulemos o conteúdo periférico que antecede os elementos. Nota: Também conseguimos configurar pseudo-elementos e mexer em conteúdos periféricos quando usamos classes (class) . Exemplo: .especial::before { content: '⇨ '; color: darkgray; font-weight: normal; } .especial::after { content: ' ⇦'; color: darkgrey; font-weight: normal; } O código no exemplo acima utiliza pseudo-elementos para manipular conteúdo periférico de links que pertencem à classe (class) "especial". Observação: O sinal de maior ( > ) representa no CSS o " children ", que seria o filho (quando temos elementos dentro de elementos).

Tags usadas: # é igual a id

. é class

: é igual a pseudo class

:: é igual a pseudo elemento

> é igual cllildem (filho)

<!DOCTYPE html>

<html lang="pt-br">

<head>

    <meta charset="UTF-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>links</title>

    <style>

        body {

            font-family: Arial, Helvetica, sans-serif;

        }

        a {

            color: brown;

        }

        a:visited {

            color: rgb(255, 0, 21);

        }

        a:active {

            color: burlywood;

        }

        a::after {

            content: '[link]';

            text-decoration: none;

            font-weight: normal;

            color: blue;

        }

    </style>

</head>

<body>

    <h1>personalizando links</h1>

    <li><a href="https://github.com/" target="\_blank">repositorio do githamb</a>></li>

    <li><a href="https://www.youtube.com/">canal do youtube</a></li>

    <li><a href="https://twitter.com/">site do google</a></li>

</body>

</html>

**Os conceitos do Modelo de Caixas ( Box**

**Height: altura**

**Width: largura**

**Padding: espaço nas bordas**

**Margin: margem**

Os conceitos do Modelo de Caixas ( Box Model ) são aplicáveis ao HTML e ao CSS. Se trata de conjuntos que irão facilitar muito na hora de organizarmos o design e as partes do nosso site. Quando temos um documento HTML, tudo aquilo que é exibido na tela, é exibido em forma de caixa... De tal modo que a grande maioria das caixas na HTML podem estar uma dentro da outra. Dessa forma, podemos formar hierarquias de caixas. Quando colocamos um elemento dentro do outro, o termo que descreve isso é o " ANINHAMENTO ", algo em forma de ninho, coisas que estão uma dentro da outra e assim por diante. Na prática, uma caixa seria todo elemento de um site visível na tela. Considerando que todos os elementos de um site estão dentro de uma caixa, quando formatamos um elemento com o CSS através de seletores, na verdade estamos configurando a caixa que vai estar exibindo esse elemento. ANATOMIA DAS CAIXAS: *PARTE DE CONTEÚDO* : Toda caixa possui um tamanho, que é chamado de " *PARTE DE CONTEÚDO* ", esse tamanho é especificado em duas medidas muito importantes, que são: Altura ( " height " ) e Largura ( " width " ) *BORDA* : Podemos traçar uma linha em volta da *Parte de Conteúdo* (que define o tamanho da caixa) e chamamos essa linha de " *BORDA* "... Sendo assim uma borda é uma linha que circunda o seu conteúdo. *PADDING* : Por padrão, a linha da Borda irá ficar muito próxima da Parte de Conteúdo , e se quisermos dar um distanciamento, devemos adicionar um espaço de " *PADDING* " (acolchoamento), que seria uma pequena lacuna entre a *Parte de Conteúdo* e a *Borda* , para dar aquela engordada no elemento, mantendo a nossa borda distante do conteúdo. *MARGIN* : Ao desenvolver um site, eventualmente iremos ter caixas uma embaixo da outra de forma seguida e, se uma dessas caixas estiver configurada com um espaçamento "padding", pode ser que elas fiquem muito próximas uma da outra e acabem "grudando" na borda do elemento de cima. Pensando nisso, podemos criar, além de um espaço interno (padding), um espaçamento EXTERNO logo após a borda , que seria um " *MARGIN* ". *OUTLINE/TRACEJADO* : Outra coisa que podemos fazer dentro do margin (espaçamento externo), logo do lado de fora da borda, é criar um tracejado, que não seria uma segunda borda e sim uma área chamada de " *OUTLINE* " (contorno/traçado). Sendo assim, ainda referente à anatomia das caixas, todo elemento e consequentemente toda caixa, tem a " Parte de Conteúdo ", tem uma " Borda ", dessa borda para dentro, temos o " Padding ", da borda para fora temos o " Margin " e, ainda dentro do margin e fora da borda, temos a área do " Outline ". Nota: Todas essas características anatômicas das caixas (tirando a *Parte de Conteúdo* ) geralmente só irão se manifestar se forem previamente configuradas no seu código. TIPOS DE CAIXA: Basicamente, existem dois tipos de caixas, as do tipo " *box-level / block-level* " e " *inline-level* ". box-level / block-level: Em uma caixa do tipo "box-level / block-level", o dito elemento sempre irá se iniciar em uma nova linha e como padrão sempre ocupará a largura total da tela ou do elemento onde ele está contido (100% do viewport), sendo que irá pular para a próxima linha antes de continuar o conteúdo posterior e aí sim inserir outros boxes. As caixas desse tipo são isoladas e independentes. Exemplos de tags do tipo "box-level / block-level": <div> <p> <main> <video> <h1> inline-level: Já uma caixa do tipo "inline-level" não vai começar em uma nova linha, e sim no ponto exato onde foram definidos, no meio de um parágrafo por exemplo. E a largura dele vai ocupar apenas o tamanho relativo ao seu conteúdo, sendo que não irá quebrar a linha ao terminar, continuando o conteúdo de forma direta (em um parágrafo por ex) sem pular para a linha de baixo. Exemplos de tags do tipo "inline-level": <a> <span> <code> <strong> <em>

Modelos das caixas nas práticas:

*PROPRIEDADES DE ESTILO DE CADA PARTE ANATÔMICA DE UMA CAIXA:* PROPRIEDADES - BORDA: As principais Propriedades de Estilo para bordas são: " border-width " (espessura da borda), " border-style " (estilo da borda) e " border-color " (cor da borda). Os valores mais utilizados para a Propriedade de Estilo que define o estilo de uma borda (border-style), são: " solid " (linha sólida), " dashed " (borda tracejada), " dotted " (borda pontilhada), " double " (linha dupla) e o " groove " (borda "3d") Sendo que podemos utilizar uma shorthand para inserir e simplificar todas essas declarações de bordas em uma única linha, exemplo: " border: 10px solid black; " (Siga a ordem de inserção de valores do exemplo na hora de usar uma shorthand, caso contrário não funcionará corretamente) PROPRIEDADES - PADDING: As principais Propriedades de Estilo para padding são: " padding-top " (acolchoamento da parte de cima de um elemento), " padding-right " (parte direita), " padding-bottom " (parte de baixo) e " padding-left " (parte esquerda) - O valor dessas propriedades geralmente é inserido em px. Existe um SENTIDO/ORDEM correto PARA INSERIR as propriedades de padding descritas acima. A "direção" correta é: 1) - *Top* 2) - *Right* 3) - *Bottom* 4) - *left* Exemplo: padding-top: 10px; padding-right: 10px; padding-bottom: 10x; padding-left: 10px; É importante declarar nessa ordem, pois é necessário ser condizente com a direção e sentido padrão do modelo de caixas do DevTools, que segue a direção de um ponteiro de relógio. Também podemos utilizar uma shorthand para inserir e simplificar todas essas declarações de padding em uma única linha, exemplo: " padding: 15px 10px 10px 15px; " OBSERVAÇÃO: Mesmo em *shorthands* como a do exemplo acima, os valores devem ser declarados na ordem orientada ( top , right , bottom e left ), para que consigamos nos nortear e especificar corretamente o espaçamento certo para o respectivo lado/sentido. Se quisermos configurar os quatro lados do padding de uma só vez e constituir um padding igualitário (mesmo valor nos quatro lados), podemos simplesmente inserir uma única vez a Propriedade "padding" com o valor em px desejado, esse valor será inserido nos quatro lados do elemento automaticamente, exemplo: " padding: 10px; " Também podemos colocar só dois valores em uma propriedade padding, exemplo: " padding: 10px 20px; " - Quando colocamos somente dois valores, significa que os sentidos de *CIMA E DE BAIXO* são representados pelo primeiro valor e os sentidos de *ESQUERDA E DIREITA* são representados pelo segundo valor . Atenção: A área do "body" na tela do site, tanto no HTML quanto no CSS, não é composta pelo corpo inteiro, por toda a tela do navegador (parte branca) e sim por uma box, que está contendo todo o nosso conteúdo.

Mostrar menos

Continuação:

*PROPRIEDADES DE ESTILO DE CADA PARTE ANATÔMICA DE UMA CAIXA:* PROPRIEDADES - MARGIN: As principais Propriedades de Estilo para margin são: " margin-top " (margin da parte de cima de um elemento), " margin-right " (parte direita), " margin-bottom " (parte de baixo) e " margin-left " (parte esquerda) - O valor dessas propriedades geralmente é inserido em px. Nota: Para centralizar um elemento (CAIXA) na tela, de forma *automática* , devemos usar a Propriedade de Estilo " margin " com o valor " auto ". Isso fará com que elementos (caixas) sejam centralizados horizontalmente de forma automática, mesmo que o tamanho da tela mude. Como no "padding", também existe um SENTIDO/ORDEM correta PARA INSERIR as propriedades de margin descritas logo acima. A "direção" correta é: 1) - *Top* 2) - *Right* 3) - *Bottom* 4) - *left* Exemplo: margin-top: 10px; margin-right: 10px; margin-bottom: 10px; margin-left: 10px; É importante declarar nessa ordem, pois é necessário ser condizente com a direção e sentido padrão do *Modelo de Caixas do DevTools* , que segue a direção de um ponteiro de relógio. Também podemos utilizar uma Shorthand para inserir e simplificar todas essas declarações de margin em uma única linha, exemplo: " margin: 15px 10px 10px 15px; " OBSERVAÇÃO: Mesmo em shorthands como a do exemplo acima, os valores devem ser declarados na ordem orientada ( top, right, bottom e left ), para que consigamos nos nortear e especificar corretamente o espaçamento certo o para respectivo lado/sentido. Se quisermos configurar os quatro lados do margin de uma só vez e constituir um margin igualitário (mesmo valor nos quatro lados), podemos simplesmente inserir uma única vez a Propriedade "margin" com o valor em px desejado, esse valor será inserido nos quatro lados do elemento *automaticamente* , exemplo: " margin: 10px; " *IMPORTANTE* : Se quisermos centralizar um elemento (CAIXA) de forma automática, mas ainda assim , alterar o margin do sentido de cima (top) ou de baixo (bottom), devemos inserir a seguinte declaração shorthand: " margin: 10px auto 10px auto; " Também podemos colocar só *dois valores* em uma propriedade margin, exemplo: " margin: 10px 20px; " - Quando colocamos somente dois valores, significa que os sentidos de *CIMA E DE BAIXO* são representados pelo primeiro valor e os sentidos de *ESQUERDA E DIREITA* são representados pelo segundo valor . PROPRIEDADES - OUTLINE: As principais Propriedades de Estilo para outline são: " outline-width " (espessura da outline), " outline-style " (estilo da outline) e " outline-color " (cor da outline). Os valores mais utilizados para a Propriedade de Estilo que define o estilo de uma outline (outline-style), são: " solid " (linha sólida), " dashed " (outline tracejada), " dotted " (outline pontilhada), " double " (linha dupla) e o " groove " (outline "3d") Sendo que podemos utilizar uma shorthand para inserir e simplificar todas essas declarações de outline em uma única linha, exemplo: " outline: 10px solid black; " (Siga a ordem de inserção de valores do exemplo na hora de usar uma shorthand, caso contrário não funcionará corretamente)

Exemplos: 1

<!DOCTYPE html>

<html lang="pt-br">

<head>

    <meta charset="UTF-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>modelos de caixas</title>

    <style>

        h1 {

            background-color: aquamarine;

            height: 300px;

            width: 300px;

            border-width: 10px;

            border-style: solid;

            border-color: blanchedalmond;

      /\*forma simplificada dos comandos\*/      border:  10px solid darkblue;

            padding-top: 10px;

            padding-right: 10px;

            padding-bottom: 10px;

            padding-left: 10px;

            /\*forma simplificada dos comandos\*/    padding: 10px;

            margin: auto;

            margin-top: 10px;

            margin-right: 10px;

            margin-bottom: 10px;

            margin-left: 10px;

            /\*forma simplificada dos comandos\*/    margin:  10px;

            outline-width: 20px;

            outline-style: dashed;

            outline-color: black;

            /\*forma simplificada dos comandos\*/    outline: 20px dashed salmon;

        }

        a {

            border-width: 10px;

            border-style: groove;

            border-color: brown;

             /\*forma simplificada dos comandos\*/ border: 10px groove brown;

            padding-top: 10px;

            padding-right: 10px;

            padding-bottom: 10px;

            padding-left: 10px;

        }

    </style>

</head>

<body>

    <h1>modelo de caixas box-level</h1>

    <p>paragrafos tambem são exemplos de box-level, mas os <a href="">os links são exemplos de cixas inline-level</a>vamos ver como tudo isso funciona</p>

</body>

</html>

Exemplos: 2

[Tiago Sampa](https://www.youtube.com/channel/UCAAK8RU7-wL65LZzCJBWbdg)

[há 2 semanas](https://www.youtube.com/watch?v=JPMm-jyKOaM&lc=UgzCI3od263uuYTOWrR4AaABAg)

Tags de agrupamento (grouping tags): - Sem valor semântico: <div> -> block <span> -> inline - Com valor semântico: <header> -> cabeçalho <nav> -> para menus, links, navegação <main> -> principal <section> -> seção <article> -> artigo <aside> -> periférico ao artigo (ex: escritor / autor) <footer> -> rodapé

Ao utilizar sombras, devemos tomar cuidado para não exagerar. Não temos que mostrar de forma forçada para o usuário que em determinados pontos do site existem sombras... É como vimos nos capítulos anteriores de Cores e Fontes, o cérebro irá perceber a harmonia das propriedades de forma automática e inconsciente. As sombras funcionam já com uma shorthand (simplificação de várias declarações em uma linha) e os recursos do *DevTools* ajudam muito na hora de criarmos e configurarmos sombras. Para inserir *SOMBRAS DE CAIXA* , temos que utilizar a Propriedade de Estilo shorthand " box-shadow ", essa propriedade shorthand geralmente possui quatro parâmetros (cinco na verdade) que devem ser preenchidos obrigatoriamente na seguinte ordem : (1 - Deslocamento horizontal - O quanto a sombra irá andar para o lado direito ou esquerdo (Valor em negativo para ir pra esquerda, ex: " -10px ") - X OFFSET (2 - Deslocamento vertical - O quanto a sombra irá andar para baixo ou para cima (Valor em negativo para ir pra cima, ex: " -10px ") - Y OFFSEY (3 - O quanto essa sombra irá espalhar/embaçar - BLUR (4 - Nível de alcance/tamanho que essa sombra irá ter (ESSE VALOR GERALMENTE NÃO É USADO, É SÓ NÃO INSERI-LO NA SHORTHAND NESSE CASO) - SPREAD (5 - Qual cor essa sombra vai ter e o nível de transparência (alpha) que essa cor irá ter *Exemplo de Shorthand* (sem o valor "spread" ): " box-shadow: 1px 1px 1px [#0000007e](https://www.youtube.com/hashtag/0000007e); " (Siga a ordem de inserção de valores do exemplo na hora de usar uma shorthand, caso contrário não funcionará corretamente) Ao configurar a sombra de uma caixa, é interessante abrir o DevTools e formatar os *valores da shorthand* pelo Sub Menu de Regulagem "shadow editor" da interface DevTools de maneira manual e observando as mudanças em tempo real na tela do navegador, até encontrar a configuração ideal... Passando essa configuração ideal para o código no VSC posteriormente. Também é possível inserirmos sombras do LADO DE DENTRO de uma caixa . Apesar de não ser muito recomendável botar isso em prática na maioria dos casos... Para fazer isso basta inserir o parâmetro " inset " como o *primeiro valor* na Shorthand, exemplo: " box-shadow: inset 1px 1px 1px [#0000007e](https://www.youtube.com/hashtag/0000007e); " Frequentemente usamos somente a cor preta para representar sombras, isso porque sombras coloridas geralmente ficam estranhas e com um efeito "não natural". Normalmente, após configurar os valores para a sua sombra, ela ainda continuará com uma cor muito forte (sólida) ... Para amenizar isso e tornar a exibição algo mais "natural" e suave, adicionamos transparência à cor (black), de forma geral 50% de transparência é o recomendado.

<!DOCTYPE html>

<html lang="pt-br">

<head>

    <meta charset="UTF-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>grouping tag</title>

    <style>

        body {

            font-family: Arial, Helvetica, sans-serif;

            background-color: blue;

        }

        header {

            background-color: aqua;

            padding: 10px;

            margin: 10px;

        }

        nav {

            background-color: aqua;

            padding: 3px;

            margin: 10px;

        }

        nav > a {

            text-decoration: none;

            color: bisque;

            font-weight: bolder;

            margin-right: 10px;

        }

        nav > a:hover {

            text-decoration: underline;

        }

        article {

            background-color: aliceblue;

        }

        article > aside {

            background-color: rgb(11, 226, 118);

        }

        footer {

            background-color: darkgreen;

            color: coral;

            text-align: center;

            padding: 1px;

            margin: 0px;

        }

    </style>

</head>

<body>

    <header>

        <nav></nav>

        <h1> menu site</h1>

        <a href=""> link</a>

        <a href=""> link</a>

        <a href=""> link</a>

        <a href=""> link</a>

    </nav>

    </header>

    <main>

        <section id="assuntos">

        </section>

            <section id="noticias">

                <article>

                    <h2>noticias sobre o Brasil</h2>

                    <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Eum ad eius ea vel ipsam similique facilis. Adipisci molestiae tenetur doloremque earum soluta placeat quam laudantium accusamus reprehenderit maiores. Corrupti, quia.</p>

                    <aside>

                        <p>Artigo escrito por emerson22233232</p>

                    </aside>

                </article>

            </section>

        </section>

    </main>

    <footer>

        <p>Artigo escrito por emerson</p>

    </footer>

</body>

</html>

Vertes arredondados:

[Tiago Sampa](https://www.youtube.com/channel/UCAAK8RU7-wL65LZzCJBWbdg)

[há 2 semanas](https://www.youtube.com/watch?v=PGIrTzQqpqo&lc=Ugx_XHRWKESHVZiGhY94AaABAg)

LEMBRE-SE border-radius: ; Shorthand: Inicia no canto superior esquerdo (border-top-left-radius) e segue o sentido horário. ex: border-radius: 10px 20px 30px 40px; Com 2 valores: 1º valor: border-top-left-radius e border-bottom-right-radius 2º valor: border-top-right-radius e border-bottom-left-radius ex: border-radius: 10px 20px; Transformando uma div em bola: div { border-radius: 50%;

Podemos criar caixas com pontas arredondadas, caixas com vértices arredondados... Para fazer isso utilizamos a Propriedade de Estilo " border-radius ". PROPRIEDADES - BORDER-RADIUS: As principais Propriedades de Estilo para Vértices Arredondados são: border-top-left-radius (arredondamento da parte superior esquerda da box) border-top-right-radius (arredondamento da parte superior direita da box) border-bottom-right-radius (arredondamento da parte inferior direita da box) border-bottom-left-radius (arredondamento da parte inferior esquerda da box) Existe um SENTIDO/ORDEM correta PARA INSERIR as propriedades border-radius descritas acima. A "direção" correta é: 1) - Top Left - *Superior Esquerda* 2) - Top Right - *Superior Direita* 3) - Bottom Right - *Inferior Direita* 4) - Bottom Left - *Inferior Esquerda* Exemplo: border-top-left-radius: 10px; border-top-right-radius: 10px; border-bottom-right-radius: 10px; border-bottom-left-radius: 10px; Também podemos utilizar uma *SHORTHAND* para inserir e simplificar todas essas declarações de border-radius em uma única linha, exemplo: " border-radius: 15px 10px 15px 10px; " OBSERVAÇÃO: Mesmo em shorthands como a do exemplo acima, os valores devem ser declarados na *ordem orientada* ( Top Left, Top Right, Bottom Right e Bottom Left ), para que consigamos nos nortear e especificar corretamente o arredondamento certo para o respectivo lado/sentido. Se quisermos configurar os quatro lados do border-radius de uma só vez e constituir um border-radius igualitário (mesmo valor nos quatro lados), podemos simplesmente inserir uma única vez a Propriedade "border-radius" com o valor em px desejado, esse valor será inserido nos quatro lados do elemento automaticamente, exemplo: " border-radius: 10px; " Também podemos colocar só dois valores em uma propriedade border-radius, exemplo: " border-radius: 10px 20px; " - Quando colocamos somente dois valores, significa que os sentidos *SUPERIOR ESQUERDO E INFERIOR DIREITO* são representados pelo Primeiro Valor e os sentidos *SUPERIOR DIREITO E INFERIOR ESQUERDO* são representados pelo Segundo Valor . NOTA: O design prima pela discrição , então cuidado para não exagerar ao utilizar bordas arredondadas. Quando temos um elemento na forma QUADRADA , uma <div> com *altura e largura* de 100px por exemplo, podemos transformas esses elementos quadrados em um círculo/bola com a Propriedade de Estilo " border-radius " com o valor " 50% " ou " 100px ".

Borda com imagem:

Caso queira uma borda com um visual personalizado, podemos adicionar uma imagem como borda e ainda configurar alguns *aspectos desta imagem* . Podemos definir uma imagem para servir de borda. Para fazer isso, usamos a Propriedade de Estilo " border-image ". PROPRIEDADES - BORDER-IMAGE: As principais Propriedades de Estilo para "Imagem como Borda" (border-image) são: border-image-source (onde inserimos o endereço da imagem adequada dentro do campo "url") border-image-slice (propriedade que "fatia" a imagem que inserimos, é necessário colocar um valor numérico, sem unidade de medida) border-image-repeat (comportamento do fatiamento, os valores mais usados para essa propriedade são "stretch" e "repeat") Também podemos utilizar uma *SHORTHAND* para inserir e simplificar todas essas declarações de border-image em uma única linha, exemplo: " border-image: url (imagens/exemplo.jpeg) 30 repeat; " (Siga a ordem de inserção de valores do exemplo na hora de usar uma shorthand, caso contrário não funcionará corretamente) IMPORTANTE: Para utilizar as Propriedades de Estilo border-image descritas acima, devemos primeiramente configurar uma borda comum (ex: border: 10px solid transparent;) para o elemento que escolhemos... Caso contrário as configurações border-image acima não terão efeito. Ao configurar as propriedades border-image, é interessante abrir o DevTools e formatar os *valores das propriedades de borda e border-image* pelo Sub Menu de Regulagem da interface DevTools de maneira manual e observando as mudanças em tempo real na tela do navegador, até encontrar a configuração ideal... Passando essa configuração ideal para o código no VSC posteriormente. Dependendo do efeito que queremos, também é necessário uma imagem adequada para dar esse efeito ideal. Observação: Geralmente bordas com imagens (border-image) ficam feias e estranhas, então essa técnica não é muito recomendada... Use apenas em último caso.

**Montando o site do zero:**

Quando vamos iniciar um projeto, antes de começarmos a mexer no código, é essencial que façamos um planejamento de como vai ser a estrutura desse site. O planejamento de um site é algo muito importante, tão importante que inclusive existem algumas ferramentas de planejamento de sites. Na prática, para fazer o planejamento, temos que ter em mente três itens: 1) - Layout do Site 2) - Paleta de Cores do Site 3) - Fontes de Texto do Projeto *LAYOUT:* Devemos prototipar a estrutura do layout do nosso site através de um " Mock " ou " Wireframe ", ambos tem funções bem semelhantes: Mock: Objetos mock, objetos simulados ou simplesmente mock em desenvolvimento de software são objetos que simulam o comportamento de objetos reais de forma controlada, simula a estrutura básica e o formato de um site (um Mock NÃO se preocupa com cores ou imagens, mas sim somente com a forma da estrutura básica, com a modelagem). Wireframe: Um wireframe de site web ou website wireframe é um protótipo usado em design de interface para sugerir a estrutura de um sítio web e relacionamentos entre suas páginas . Um wireframe web é uma ilustração (que pode ser *colorida e conter imagens* ) semelhante do *layout* de elementos fundamentais na interface. A ferramenta utilizada no vídeo para elaborar a prototipação do layout do nosso projeto foi o " MockFlow ", atualmente (08/01/2023) essa ferramenta ainda está funcionando e possui todos os recursos mostrados no vídeo... Entretanto o site atualizou e agora tem uma interface inicial um pouco diferente. Como ferramenta alternativa ao "MockFlow", também existe o " Figma ". Ao logar e entrar na ferramenta MockFlow, devemos abrir o menu da interface e começar a montar o protótipo da estrutura do nosso site, inserindo e organizando as caixas que o nosso projeto terá. Você deve manipular os elementos disponíveis no menu, como browsers frames, retângulos, caixas de títulos, caixas de texto, caixas de imagens e etc... Você também deve manipular a altura (height) e largura (width) desses elementos e deixar tudo isso documentado, simule cada parte da interface do site (header, nav, main, sections, articles, footer e tec) e coloque elementos dentro de outros elementos... Faça isso até formar um protótipo que considera ideal. O modelo de layout apresentado no vídeo será usado no nosso projeto, mas lembre-se que não existe somente esse modelo e que você pode usar sua criatividade para criar seus próprios ou se inspirar em modelos de outros sites e tentar replicá-los. *PALETA DE CORES:* Devemos criar uma Paleta de Cores harmônica com a ferramenta do "Adobe Color" ou qualquer outra ferramenta já apresentada. No caso a Paleta utilizada no vídeo é monocromática, com diferentes tons de verde (cor da logo do Android). *FONTES DE TEXTO:* Devemos escolher um conjunto adequado de fontes de texto para o projeto, que podem ser padrões, externas web ou externas de forma local (baixadas). Use as ferramentas Google Fonts e uma biblioteca de fontes locais de sua preferência. Nota: A parte do planejamento pode ser um pouco demorada, mas montar o mock, essa representação visual estruturada e simplificada do conceito do site e etc, é muito útil... Pois teremos uma ideia de como o nosso site será, isso vai te levar a montar essa diagramação muito mais facilmente lá na frente na horar de colocar a mão no código.

Mostrar menos

Exemplos:

No HTML, criaremos dentro do "body" a estrutura do projeto, inspirada no mock que prototipamos anteriormente... Basicamente, a organização das Grouping Tags SEMÂNTICAS do projeto foi a seguinte: Estrutura básica do HTML do projeto: <header> <h1></h1> <p></p> </header> <nav> <a href="#"></a> <a href="#"></a> <a href="#"></a> <a href="#"></a> </nav> <main> <article> <aside> </aside> </article> </main> <footer> <p></p> </footer> Tome muito cuidado na hora de montar a *estrutura HTML* , cuidado para não colocar as tags do lado de fora ou no lugar errado... Se você errar no HTML e os seus blocos não estão com a hierarquia certa, não adianta tentar formatar os estilos com CSS já que o HTML estará errado... *IMPORTANTE:* É fundamental que um aspirante a desenvolvedor/programador mantenha o seu código sempre bem arrumado, a INDENTAÇÃO é muito importante! Após construir a estrutura do seu site com as Grouping Tags Semânticas acima, o nosso conteúdo estará aglomerado e sem formatação, mas já estará SEMÂNTICAMENTE ORGANIZADO em caixas com uma estrutura hierárquica... Por hora não teremos efeitos visuais evidentes, mas já teremos um efeito puramente semântico da organização das nossas caixas, que serão modeladas, formatadas e estilizadas com CSS . Já na parte do " *head* ", inserimos o favicon logo abaixo de "title"... Como temos uma pasta para imagens e fontes, e estamos elaborando um projeto, é importante mantermos os nossos arquivos organizados... Por isso CRIAMOS UMA PASTA ( "estilos" ) para armazenar o nosso *documento CSS* , que será externo ( Você pode criar a pasta "estilos" pelo próprio VSC ou no Explorador de Arquivos do projeto ) Já no nosso documento CSS a regra *@charset "UTF-8"* foi inserida no início do código (Regra de codificação de caracteres utilizado no documento) Para testar se o arquivo CSS que criamos está funcionando, modifique a cor de fundo do corpo da página com a propriedade "background-color", se o corpo mudar de cor o nosso CSS está linkado e tudo funcionando como deveria.

Uma dica, pessoal: quando for criar vários links ou uma lista, ao invés de colocar a tag várias vezes, é só escrever (tag\*número de vezes) e dar enter. Por exemplo: a\*5 = 5 links li\*10 = 10 elementos da lista.

**Link da aula :** [**https://www.youtube.com/watch?v=D7jnoo7UHKE&list=PLHz\_AreHm4dlUpEXkY1AyVLQGcpSgVF8s&index=34**](https://www.youtube.com/watch?v=D7jnoo7UHKE&list=PLHz_AreHm4dlUpEXkY1AyVLQGcpSgVF8s&index=34)

***SELETOR GLOBAL:*** Basicamente, quando carregamos um site, existem algumas configurações que são "globais" e automáticas, feitas pelo "use agent" (nosso navegador). Essas configurações globais normalmente são de valores de margin nos elementos... Acontece que essas configurações globais automáticas muitas vezes acabam atrapalhando e tornando moroso o nosso processo de desenvolvimento para chegar a um resultado pretendido. Pensando nisso, para não sofrermos com "erros" de margins e paddings excessivos na página, podemos fazer CONFIGURAÇÕES GLOBAIS , onde nenhum elemento/bloco *inicialmente* irá ter margins ou paddings... Para isso precisamos *selecionar todos os elementos HTML* , para botar isso em prática existe um seletor especial das CSS, que é o asterisco ( \* {...} ) SELETOR GLOBAL . Ele tem uma função muito especial, pois basicamente ele aplica uma configuração padrão para TODOS os elementos do código HTML ao qual o estilo está sendo aplicado. Exemplo: \* { margin: 0px; padding: 0px; } O Seletor Global acima ( \* {...} ) seleciona todos os elementos HTML e *toda declaração* que estiver dentro de suas chaves aplica isso para *todos esses elementos* ... Portanto, inicialmente zeramos os valores de margin e padding de todos os elementos... Claro, após fazer isso, todos os elementos da página irão "grudar" um no outro e teremos que configurar parte por parte, mas de qualquer forma isso ainda será mais proveitoso e facilitará o nosso processo de desenvolvimento, pois vai permitir personalizar as medidas que vão aparecer na tela para cada elemento individualmente.

*VARIÁVEIS EM CSS:*É possível utilizar variáveis dentro do CSS. Isso não torna a linguagem CSS uma linguagem de programação... Ela ainda continua sendo uma linguagem de marcação, assim como o HTML. Toda linguagem de programação tem variáveis, mas nem toda linguagem que tem variável é uma linguagem de programação. As Variáveis GUARDAM VALORES para serem *usados mais na frente* no nosso projeto/programa. Por que deveríamos usar variáveis em CSS? Porque isso facilita, e muito, o nosso *processo de desenvolvimento* e posteriormente a manutenção do projeto , caso desejemos alterar algo no futuro... Fora que tudo fica ainda mais organizado. Usando a *Paleta de Cores* de um projeto qualquer como exemplo, geralmente alguns desenvolvedores simplesmente fixam a Paleta de Cores (códigos das cores) de um site em um comentário no documento CSS, e copiam e colam esses códigos sempre que precisam alterar uma cor... Na realidade esse não é o modo mais adequado de se fazer isso e existe uma maneira muito melhor e prática de se obter um melhor resultado, com variáveis CSS. Para declarar variáveis em CSS, devemos criar uma Pseudo-classe especial dentro do documento CSS atual, que seria uma espécie de referência dentro desse documento. As *variáveis CSS* são definidas dentro da Pseudo-classe " root " ( :root {...} ). Essa pseudo-classe define as configurações para a " RAIZ DA ÁRVORE ", que vai servir para o *documento inteiro* . Tudo que colocarmos e definirmos dentro dessa pseudo-classe será válido para todo o documento, já que essa pseudoclasse (root) tem uma especificidade muito alta, ou seja, aplicado ao HTML, :root representa o elemento <html> e é idêntico ao seletor html, abrangendo todo o projeto... Dessa forma, basicamente podemos criar Variáveis "Globais" dentro de root. Observação: Por exemplo, em linguagens de programação, normalmente temos uma "área principal", que é onde podemos declarar as nossas variáveis globais... No CSS, essa "área principal" é na Pseudo-classe root. Todas as *Variáveis em CSS* tem que começar com " dois traços " ( -- ), após isso é preciso dar um nome (de sua preferência) para essa variável, posteriormente inserindo o valor da variável (código da cor nesse caso). Exemplo: :root { --cor0: [#ebe5c5](https://www.youtube.com/hashtag/ebe5c5); --cor1: [#83e1ab](https://www.youtube.com/hashtag/83e1ab); --cor2: [#3ddc84](https://www.youtube.com/hashtag/3ddc84); --cor3: [#2fa866](https://www.youtube.com/hashtag/2fa866); --cor4: [#1a5c37](https://www.youtube.com/hashtag/1a5c37); --cor5: [#063d1e](https://www.youtube.com/hashtag/063d1e); } Nota: Não cadastramos as cores branco e preto nas variáveis do exemplo, mas isso é opcional e se você quiser pode cadastrar... Já que branco e preto sempre vai estar em um site, independentemente da Paleta utilizada. IMPORTANTE: Referente ao nome que damos para as nossas variáveis, ele é de sua preferência e você pode nomear como no exemplo acima, mas você também pode nomear de acordo com a função da variável. Por exemplo, no caso das cores da nossa Paleta, podemos nomear cada cor (variável) com o seu respectivo encargo, exemplo: "cor-de-destaque", "cor-de-fundo" e etc. Recapitulando... Para declararmos variáveis CSS, devemos abrir um seletor para a pseudo-classe "root" (referência e raiz da árvore) no início do código CSS (logo após os códigos de regra) e posteriormente declarar as nossas variáveis ( -- ) dentro desse seletor. Depois de declarar as suas variáveis "globais" de cor dentro do root, na hora de *formatarmos um elemento* , qualquer elemento que seja, ao invés de digitar ou copiar e colar um código de cor, simplesmente usamos o atalho "Ctrl + Espaço", avançamos até o fim da lista de sugestões e lá estará a nossa *Paleta de Cores* declarada no root (valores guardados por variáveis)... Posteriormente basta escolher a cor desejada e adicioná-la. Ao selecionar variáveis na lista de sugestões como valor da respectiva Propriedade, o VSC coloca a função "var(--ex-nome);" como o valor (também é possível digitar a especificação da variável ao invés de usar a lista). O legal das variáveis é que podemos pode usá-las em quaisquer elementos e depois, se um cliente pedir por exemplo, facilmente trocar o tema do site inteiro só mudando o valor que está configurado na variável (dentro de root)... É tipo um CSS dentro do CSS, trocando o valor da variável global que está no :root, você troca no site inteiro ... Tudo que foi configurado com essa cor/variável irá ser automaticamente adaptado , sem precisar modificar propriedade por propriedade, item por item... Lembrando que nos exemplos anteriores, usamos variáveis contendo valores de cores, mas ainda dentro do mesmo seletor root que abrimos, podemos também declarar variáveis de fontes de texto, tanto fontes padrão, externas ou locais... E tudo irá funcionar exatamente como as variáveis de cores. Exemplo: :root { --fonte-padrao: Arial, Verdana, Helvetica, sans-serif; --fonte-destaque: 'Bebas Neue', sans-serif; --fonte-android: 'Android', sans-serif; } ---> 'Bebas Neue' e 'Android' são fontes externas e locais respectivamente, que foram configuradas como valor de variáveis. Observação: Ao iniciarmos a configuração CSS de um novo projeto, por uma questão de organização, é muito indicado que você vá abrindo os seletores de estilização CSS de acordo com a ordem das tags da estrutura HTML , assim os seletores CSS ficarão organizados ao invés de aglomerados e completamente desordenados, e na hora de fazer alguma alteração pontual você irá se nortear de forma muito mais fácil pelo código CSS.

Mostrar menos

REVISÃO: Extensão Chrome recomendada: Window Resizer Largura Responsiva: min-width: 320px; (tela de cel mais antigos) max-width: 800px; margin: auto; Para imagens: img { width: 100% } - Envelopar a <img> junto com ' source:media ' dentro da tag <picture>, para imagem dinâmica (que altera de acordo com a largura da tela). Ex: <picture> <source media="(max-width:600px)" srcset="imagens/irina-blok-pq.jpg"> <img src="imagens/irina-blok.jpg" alt="Irina Blok"> </picture>

Obs: Quando devemos usar o sinal de maior ( " > " ) para manipular elementos que estão dentro de outros elementos (filho/children)? Talvez alguns de vocês estejam na dúvida, porque tem vezes que o Guanabara usa o sinal, e outras não... E às vezes, quando inserimos o sinal nos nossos seletores, as declarações simplesmente não tem efeito... ESCLARECENDO: Nesse caso, você precisa lembrar dos conceitos de COMBINADORES DE SELETORES, que nada mais é do que criar uma relação útil entre os seletores. Você usa o " > " quando quer indicar que deseja estilizar um FILHO DIRETO de um elemento. Segue um exemplo: <header> <h1></h1> <p></p> </header> No caso do exemplo acima, o "p" é um filho direto da div especial <header>, e portanto é um elemento de Primeiro Nível. Da mesma forma, veja que o "h1" identicamente é um filho direto de header e portanto é um elemento de Primeiro Nível também. É como se <p> e <h1> fossem IRMÃOS. Mas quando você NÃO USA o " > ", você quer indicar os elementos que não são necessariamente filhos, mas são DESCENDENTES. Veja outro exemplo: <main> <article> <h1></h1> </article> </main> *SELETOR NÃO SIMPLIFICADO PARA O EXEMPLO ACIMA:* " main > article > h1 {...} ". H1 é filho direto de article, que é filho direto de main. Tem como simplificar? Sim. Nesse caso, perceba que h1 é descendente de main (como se fosse um NETO ), então você pode DEIXAR DE USAR o " > " sem problema nenhum, dessa forma estará indicando elementos que não são necessariamente filhos, e sim *descendentes.* Exemplo: *MESMO SELETOR ANTERIOR, SÓ QUE SIMPLIFICADO:* " main h1 {...} " Obs.: Se tivéssemos criado o seletor simplificado do exemplo acima, mas com o sinal " > " ( ex: main > h1 {...} ), não funcionaria porque esse h1 NÃO É FILHO DIRETO do main, e sim neto. Sendo que o sinal de maior ( ">" ) serve apenas para indicar filhos diretos. É por isso que o Guanabara não usa o sinal de maior (>) às vezes, ele está simplificando o seletor (quando possível). Isso facilita muito, porque você não precisará de uma pancada de classes e deixa o seu código mais limpo. Créditos: Alisson Radam

Link da aula pra adequar o vídeo do youtube ao tamanho do site quando minimiza : <https://www.youtube.com/watch?v=u9NE0jInb_c&list=PLHz_AreHm4dlUpEXkY1AyVLQGcpSgVF8s&index=42>

Primeiro site (projeto,do mudulo 1 e 2. Html, css)

Html:

<!DOCTYPE html>

<html lang="pt-br">

<head>

    <meta charset="UTF-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Como surgiu o mascote do android</title>

    <link rel="shortcut icon" href="imagens/favicon.ico" type="image/x-icon">

    <link rel="stylesheet" href="estilo.css">

</head>

<body>

    <header>

        <h1>Curiosidades de tecnologia</h1>

        <p>Tudo aquilo que voce sempre quis saber sobre o mundo em um unico lugar.</p>

    </header>

    <nav>

        <a href="">home</a>

        <a href="">noticias</a>

        <a href="">curiosidade</a>

        <a href="">fale com nosco</a>

    </nav>

    <main>

    <article>

<h1>História do Mascote do Android</h1>

<p>Provavelmente você sabe que o sistema operacional <strong>Android</strong>, mantido pelo <strong>Google</strong> é um dos mais utilizados para dispositivos móveis em todo o mundo. Mas tavez você não saiba que o seu simpático mascote tem um nome e uma história muito curiosa? Pois acompanhe esse artigo para aprender muita coisa sobre esse robozinho.</p>

<h2>A primeira versão</h2>

<p>A primeira tentativa de criar um mascote surgiu em 2007 e veio de um desenvolvedor chamado <a href="https://androidcommunity.com/dan-morrill-shows-us-the-android-mascot-that-almost-was-20130103/" target="\_blank" class="externo">Dan Morrill</a>. Ele conta que abriu o <a href="https://inkscape.org/pt-br/" target="\_blank">Inkscape</a> (software livre para vetorização de imagens) e criou sua própria versão de robô. O objetivo era apenas personificar o sistema apenas para a a sua equipe, não existia nenhuma solicitação da empresa para a criação de um mascote.</p>

<picture>

    <source media="(max-width: 800px )" srcset="imagens/dan-droids-pq.png">

    <img src="imagens/dan-droids.png" alt="Primeiro mascote do android"></picture>

<p>Essa primeira versão bizarra até foi batizada em homenagem ao seu criador: seriam os <strong>Dandroids</strong>.</p>

<h2>Surge um novo mascote</h2>

<p>A ideia de ter um mascote foi amadurecendo e a missão foi passada para uma profissional da área. A ilustradora Russa <a href="https://www.irinablok.com/android" target="\_blank" class="externo"class="externo">Irina Blok</a>, também funcionária do Google, ficou com a missão de representar o pequeno robô de uma maneira mais agradável.</p>

<picture>

    <source media="(max-width: 800px )" srcset="imagens/irina-blok-pq.jpg">

    <img src="imagens/irina-blok.jpg" alt="irina block criadora do bigdoid "></picture>

<p>A ideia principal da Irina era representar tudo graficamente com poucos traços e de forma mais chapada. O desenho também deveria gerar identificação rápida com quem o olha. Surgiu então o <strong>Bugdroid</strong>, o novo mascote do Android.</p>

<img src="imagens/bugdroid.png" class="pequena" alt="bugdroid">

<p>A principal inspiração para os traços do novo <strong>Bugdroid</strong> veio daqueles bonequinhos que ilustram portas de banheiro para indicar o gênero de cada porta. Conta a lenda que a artista estava criando em sua mesa no escritório do <strong>Google</strong> e olhou para o lado dos banheiros e a identificação foi imediata: simples, limpo, objetivo.</p>

<div id class="video"><iframe width="560" height="315" src="https://www.youtube.com/embed/I\_vi2q6sC1k" title="YouTube video player" frameborder="0" allow="accelerometer; autoplay; clipboard-write; encrypted-media; gyroscope; picture-in-picture; web-share" allowfullscreen></iframe></div>

 <aside>

    <h3>Quer aprender mais?</h3>

    <p>

        Outro assunto curioso em relação ao Android é que cada versão sempre foi nomeada em homenagem a um doce, em ordem alfabética a partir da versão 1.5 até a 9.0.

    </p>

<ul>

<li>1.5 - <abbr title="doce bde mel">Cupcake</abbr></li>

<li>1.6 - <abbr title="rosquinha">Donu</abbr>t</li>

<li>3.0 - <abbr title="bomba">Eclair</abbr></li>

<li>2.2 - <abbr title="iorgurt">Froyo</abbr></li>

<li>2.3 - <abbr title="biscoito">Gingerbread</abbr></li>

<li>3.0 - <abbr title="five de mel">Honeycomb</abbr></li>

<li>4.0 - <abbr title="sanduiche de sorvete">Ice Cream Sandwich</abbr></li>

<li>4.1 - <abbr title="jujuba">Jelly Bean</abbr></li>

<li>4.4 - <abbr title="kitkat">KitKat</abbr></li>

<li>5.0 - <abbr title="pirulito">Lolipop</abbr></li>

<li> 6.0 - <abbr title="machimelo">Marshmallow</abbr></li>

<li> 7.0 - <abbr title="torrone">Nougat</abbr></li>

<li> 8.0 - <abbr title="oreon">Oreo</abbr></li>

<li>9.0 - <abbr title="torta">Pie</abbr></li>

</ul>

    <p>Infelizmente, o <strong>Android</strong> Q não existiu, pois o Google resolveu pôr fim a essa divertida prática e começou a usar numerações, o que deu origem ao Android 10.</p>

    <p>Acesse aqui o site <a href="https://www.android.com/" target="\_blank" class="externo">Android History </a>para conhecer a sequência das versões "adocicadas" e o que cada uma trouxe para o sistema Android.</p>

</aside>

<p> Então é isso! Espero que você tenha gostado do nosso artigo com essa curiosidade sobre o sistema Android e seu simpático mascote.</p>

    </article>

    </main>

    <footer>

       <p> Site criado por <a href="https://gustavoguanabara.github.io/">Gustavo Guanabara </a>para o <a href="https://gustavoguanabara.github.io/" target="\_blank">CursoemVideo</a>.

    </p>

    </footer>

</body>

</html>

Css:

@charset "utf8";

/\* --cor0: #ebe5c5; --cor1: #83e1ab; --cor2: #3ddc84; --cor3: #2fa866; --cor4: #1a5c37; --cor5: #063d1e; \*/

  @import url('https://fonts.googleapis.com/css2?family=Bebas+Neue&family=Reggae+One&display=swap');

  @font-face {

    font-family: 'Android';

    src: url(idroid.otf) format('opentype');

    font-weight: normal;

}

:root {

    --cor0: #ebe5c5;

    --cor1: #83e1ab;

    --cor2: #3ddc84;

    --cor3: #2fa866;

    --cor4: #1a5c37;

    --cor5: #063d1e;

    --fonte-padrao: arial, verdana, helvetica, sans-serif;

    --fonte-destaque: 'bebas neue', 'cursive';

    --fonte-android: 'Android', 'cursive';

}

\* {

    margin: 0px;

    padding: 0px;

}

body {

    background-color: var(--cor0);

    font-family: var(--fonte-padrao);

}

a.externo::after {

    content: '\1F517';

}

header {

    background-image: linear-gradient(to bottom, var(--cor3), var(--cor5));

    min-height: 150px;

    text-align: center;

    padding-top: 30px;

}

header > h1 {

    color: white;

    font-family: var(--fonte-destaque);

    font-size: 3em;

    font-weight: normal;

    margin-bottom: 20px;

    text-shadow: 2px 2px 0px black;

    }

header > p {

    font-family: var(--fonte-padrao);

    font-size: 1.2em;

    color: white;

    max-width: 500px;

    padding-right: 10px;

    padding-left: 10px;

    margin: auto;

    margin-bottom: 20px;

}

nav {

    background-color: var(--cor5);

    padding: 10px;

    border-radius: 5px;

    box-shadow: 0px 7px 9px black;

}

nav a {

    color: rgb(255, 255, 255);

    padding: 10px;

    text-decoration: none;

    font-weight: bold;

    transition-duration: 2s;

}

nav a:hover {

    background-color: var(--cor3);

    color: var(--cor5);

}

main {

    min-width: 300px;

    max-width: 1000px;

    margin: auto;

    margin-bottom: 30px;

    background-color: white;

    padding: 20px;

    box-shadow: 0px, 0px, 10px black;

    border-bottom-left-radius:10px;

    border-bottom-right-radius: 10px;

}

main h1 {

    font-family: var(--fonte-android);

    font-weight: normal;

    font-size: 1.8px;

}

main h2 {

    font-family: var(--fonte-android);

    color: var(--cor4);

    font-size: 1em;

    font-weight: normal;

    background-image: linear-gradient(to right, var(--cor1), transparent);

    text-indent: 8px;

}

main p {

    margin: 15px opx;

    text-align: justify;

    text-indent: 30px;

    font-size: 1em;

    line-height: 2em;

}

main strong {

    color: var(--cor4);

    font-weight: bold;

}

main a {

    text-decoration: none;

    font-weight: bold;

    color: var(--cor5);

    background-color: var(--cor1);

    padding:  2px 6px;

}

main a:hover {

    text-decoration: underline;

    color: var(--cor4);

}

main img.pequena {

   max-width: 350px;

    display: block;

    margin: auto;

}

div.video {

    background-color: var(--cor4);

    margin: 0px -20px 30px -20px;

   padding: 20px;

   padding-bottom: 56%;

   position: relative;

}

div.video > iframe {

    position: absolute;

    top: 5%;

    left: 5%;

    width: 90%;

    height: 90%;

}

aside {

    background-color: var(--cor1);

    padding: 10px;

    border-radius: 10px;

    box-shadow: 3px 8px 3px black;

}

aside > h3 {

    background-color: var(--cor4);

    color: white;

    padding: 10px;

    margin: -10px -10px 0px 10px;

    border-radius:  10px 10px 0px 0px;

}

aside > ul {

    list-style-type: '\2714\00a0\00a0';

    list-style-position: inside;

    columns: 2;

}

footer {

   background-color: var(--cor5);

   color:white;

   text-align: center;

   font-size: 0.8em;

   padding: 5px;

}

footer a {

    color: white;

    font-weight: bolder;

    text-decoration: overline;

}

Obs: os códigos acima de html e css, faz parte do mesmo site...