**CSS**

• CSS Seletores

Dividido em três tipos:

•CSS de elementos: <p>, <h1>, <body> usando esses elementos você muda todos os valores dessas tags no html.

p{

color: red;

}

•CSS de classe: diferente do de elementos você dá uma classe um elemento <p class = ”aclass” ></p> e só aquele elemento com a classe muda.

.aclass {

color: green;

}

•CSS de ID: a mesma coisa do de classe o que difere ele é o prefixo usado <p id = “sometext”></p>.

#sometext{

color: orange;

}

•CSS inline style: neste modo você estiliza tudo dentro da tag de html.

<p style=” color: red;”></p>

• Hierarquia das classes de CSS

**CSS inline Style > CSS de ID > CSS de classe > CSS de elemento**

Porém se você adicionar !important na frente de uma propriedade de classe ele se torna o primeiro.

h1{

color: green !important;

}

•Pseudoseletores

São colocados em frente a um elemento no CSS exemplo:

h2: hover{

color: red;

}

•:hover

Esse pseudoseletor faz com que quando você passe o mouse em cima do elemento ele mude de cor, ou tamanho, fonte, etc...

h2: hover{

color: red;

}

•:first-child

Esse pseudoseletor muda a cor do primeiro elemento de uma lista ou tabela.

li: first-child{

color: green;

}

•:last-child

Esse pseudoseletor muda a cor ou fonte do ultimo elemento de uma lista ou tabela.

li: last-child{

color: blue;

}

•:nth-child(coloque um número)

Esse pseudoseletor muda a cor ou fonte dos elementos que estão entre uma lista ou tabela. Exemplo uma lista com 5 linhas. Todos as linhas 3 terão as cores modificadas.

li: nth-child(3) {

color: yellow;

}

•:only-child

Esse pseudoseletor é usado quando uma lista ou tabela tem apenas um elemento/valor/filho.

<ul>

<li>Oi! </li>

</ul>

li: only-child {

color: orange;

}

• :link

Esse pseudoseletor é usado em <a> para estilizar a cor.

a:link {

color: gray;

}

•:visited

Esse pseudoseletor é usado em <a> para estilizar a cor quando um link já foi visitado.

a:visited {

color: pink;

}

•Seletores Avançados

•Seletor adjacente

Nesse seletor você estiliza tudo o que vem depois de um elemento, classe ou id. Independentemente do local ordem.

h2 + a {

color: white;

}

Todo <a> depois de h2 será modificado.

•Seletor irmãos

Nesse seletor você estiliza tudo o que vem antes de um elemento, classe ou id. O elemento, id, classe tem que vir exatamente a seguir.

textarea ~ button{

color: purple;

}

O conteúdo de <button> será modificado.

•Seletores filhos

Nesse seletor você estiliza todos os elementos, classe, id filhos do principal.

ul > li {

color: red;

}

Todos os elementos <li> da lista serão modificados.

•Seletor descendente

Nesse seletor você estiliza todos os elementos, classe, id filhos da classe principal

ul li {

color: black;

}

Todas as linhas dentro de <ul> serão modificadas.

•Seletores de Atributos

•CSS [attribute]

O [attribute] seletor é usado para selecionar elementos com um atributo especificado.

a[target] {  
  background-color: yellow;  
}

Seleciona todos os elementos <a> com um atributo de destino.

•CSS [attribute = "value"]

O [attribute="value”] seletor é usado para selecionar elementos com um atributo e valor especificados.

a[target="\_blank"] {

background-color: pink;

}

Seleciona todos os elementos <a> com um atributo target = "\_ blank"

•CSS [attribute ~ = "value"]

O [attribute~="value”] seletor é usado para selecionar elementos com um valor de atributo que contém uma palavra especificada.

[title~="flower"] {

border: 5px solid yellow;

}

Seleciona todos os elementos com um atributo title que contém uma lista de palavras separadas por espaço, uma das quais é "flower".

• CSS [attribute | = "value"]

O [attribute|="value”] seletor é usado para selecionar elementos com o atributo especificado começando com o valor especificado.

[class|="top"] {

background: yellow;

}

Seleciona todos os elementos com um valor de atributo de classe que começa com "top"

•CSS [attribute ^ = "value"]

O [attribute^="value”] seletor é usado para selecionar elementos cujo valor de atributo começa com um valor especificado.

[class^="top"] {

background: yellow;

}

Seleciona todos os elementos com um valor de atributo de classe que têm que começar com "top"

•CSS [attribute $ = "value"]

O [attribute$="value"] seletor é usado para selecionar elementos cujo valor de atributo termina com um valor especificado.

[class$="test"] {

background: yellow;

}

Seleciona todos os elementos com um valor de atributo de classe que termina com "test"

•CSS [attribute \* = "value"]

O [attribute\*="value"] seletor é usado para selecionar elementos cujo valor de atributo contém um valor especificado.

[class\*="te"] {

background: yellow;

}

Seleciona todos os elementos com um valor de atributo de classe que contém "te"

•CSS tipo de cores

-Hexcode = #35HJ

-rgb = (255,265,0)

-hsl (0, 100%, 50%)

•Propriedades de fundo

Quando colocamos uma imagem pelo CSS usando o background: url() o CSS coloca como default para imagem se repetir se ela for menor.

•Background-repeat

Ele vem como default depois de colocar uma imagem pelo url para retirarmos a repetição caso a imagem for pequena e não cubra o total do container. Usamos o atributo no-repeat.

Background-repeat: no repeat;

•Background-size

Nessa propriedade podemos mudar o tamanho da imagem dentro do CSS. Para cobrir todo o container usamos o atributo cover.

Background-size: cover;

•Opacidade e Transparência

No CSS para termos transparência e opacidade nas cores usamos o rgba(c1,c2,c3,1) que é a mesma coisa que o rgb(c1,c2,c3) porem adicionando o a podemos mexer na opacidade e transparência.

a de alpha em rgba só pode ter valores entre [0, ... ,1] onde zero fica tudo transparente e 1 fica do mesmo jeito.

color: rgba(125,145,333,0.2)

•Gradientes

Existem dois tipos de gradientes lineares e radial.

•Linear

Para modicar o grandiente de forma linear temos duas formas a primeira:

background: linear-gradient(to top, rgb(44, 158, 139),rgba(44, 158, 139,0));

Aqui modificamos uma cor de cima para baixo. Você vê que ela vai ficando mais transparente.

A segunda forma é usando ângulos:

background: linear-gradient(215deg,rgb(44, 158, 139),rgba(44, 158, 139,0));

•Radial

Para modificar o grandiente radial usamos:

background: radial-gradient (circle, blue 20%, yellow 40%, green 55%);

Aqui modificamos o fundo de uma forma gradiente, o formato default é elipsal, dentro do exemplo mudamos a forma para circular que também muda a proporção das cores, a porcentagem significa a quantidade das cores.

•Repeating

Podemos repetir os padrões de gradiente usando este comando, serve para linear e radial.

background: repeating-linear-gradient ( 90deg,

yellow 0px,

blue 40px,

green 40px,

red 80px

);

•Tipos de Unidades

Existem dois tipos de unidades as absolutas e a relativas.

•Absolutas

São medidas que não mudam conforme o design do site.

•mm (milímetros)

Define as medidas em milímetros.

p {word-spacing: 15mm;}

•cm (centímetros)

Define as medidas em centímetros.

div {margin-bottom: 2cm;}

•in (polegadas)

Define as medidas em polegadas. 1 polegada = 96 pixels = 2.54 centímetros.

p {word-spacing: .15in;}

•px (pixels)

Define as medidas em pixels. 1 pixel = 1/96 de 1 polegada

Os pixels (px) são relativos ao dispositivo de visualização. Para dispositivos de baixa resolução, 1 px é um pixel de dispositivo (ponto) da tela. Para impressoras e telas de alta resolução, 1px implica vários pixels do dispositivo.

p {padding: 25px;}

•pt (pontos)

Define as medidas em pontos. 1 ponto = 1/72 de 1 polegada

body {font-size: 18pt;}

•pc (paica)

Define as medidas em paicas. 1 paica = 12 pontos

p {font-size: 20pc;}

•Relativas

São medidas que mudam conforme outra medida do site.

•%

Define a medida em porcentagem relativa a outro valor.

p {font-size: 16pt; line-height: 125%;}

•em

Relativo ao tamanho da fonte de um elemento (2em significa 2 vezes o tamanho da respectiva fonte)

p {letter-spacing: 7em;}

•ex

Relativo ao x-altura da respectiva fonte (raramente usados)

p {font-size: 24pt; line-height: 3ex;}

•ch

Relativo à largura de “0”.

div {font-size: 3ch; }

•rem

Relativa ao tamanho do elemento fonte.

div { font-size: 3rem; }

•vw

Relativa a 1% da largura de um viewport.

h1 { font-size: 20vw;}

•vh

Relativa a 1% da altura de um viewport.

h1 { font-size: 20vh;}

•vmin

Relativa a 1% de um viewport de dimensão pequena.

h1 { font-size: 20vmin;}

•vmax

Relativa a 1% de um viewport de grande dimensão.

h1 { font-size: 20vmax;}

•Manipulação de Texto

•text-transform

O text-transform tem a função de mudar as letras para três tipos uppercase(maiúsculo), lowecase(minúsculo) e capitalize (A primeira letra de qualquer palavra será maiúscula).

•text-decoration

O text-decoration tem a função de decorar as palavras, a tag <a> do HTML vem com o text-decoration default: underline (uma linha abaixo do link) porém você pode mudar para none (sem nenhuma linha) ou overline (encima do link uma linha).

body{ text-transform: uppercase;}

•text-align

O text-align tem a função de mover o texto, ele tem 4 funções o center(todo texto fica localizado no centro da página), o right(todo texto fica a direita do site), o left(todo texto fica a esquerda do site) e o justify(que é usado para justificar o texto, é bastante usado em textos acadêmicos).

h1 {text-align: center;}

•text-ident

O text-ident é usado para dar espaço na primeira linha do texto.

p { text-indent: 50px;}

•text-shadow

O text-shadow é uma propriedade para adicionar sobras no texto.

h1 { text-shadow: 2px 2px;}

•Tamanho de Fonte

Para manipular o texto usamos algumas funções;

•font-size

Muda o tamanho das letras do texto.

p { font-size : 2 em;}

•font-weight

Usado para aumentar o negrito do texto.

p { font-weight:800; }

•font-style

O font-style serve para determinar uma parte do texto e ele tem 3 funções o italic(muda o texto para itálico), o normal que mantém o mesmo texto e o oblique(que é um itálico mais diferente).

•Fonts

Para adicionar fonts ao nosso site usamos alguns sites que disponibilizam fonts um deles é o google fonts. Para adicionar uma font entre no site selecione a font e adicione o link para colocar:

<link href="https://fonts.googleapis.com/css2?family=Oxygen:wght@300;400;700&family=Poppins:ital,wght@0,500;0,800;1,600&display=swap" rel="stylesheet">

Não se esqueça de colocar acima do stylesheet do CSS para carregar a font antes de entrar no site.

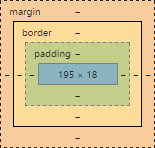
Agora no stylesheet colocamos as fontes que adicionamos no html. Os sites de fontes já dão o comando na hora de adicionar a fonte no html logo abaixo.

font-family: 'Oxygen', sans-serif;

font-family: 'Poppins', sans-serif;

•CSS Layout

Dividido em quatro partes:



[Esta Foto](https://stackoverflow.com/questions/42888544/css-input-height-in-chrome) de Autor Desconhecido está licenciado em [CC BY-SA](https://creativecommons.org/licenses/by-sa/3.0/)

•Padding

Representa o espaço entre a borda e o conteúdo.

•Border

A borda é o divisor entre a margem e o padding. Pode ser estilizado usando a propriedade border.

border: 1px solid #ff00ff

•Margin

A margin é o espaço entre a borda e todo o outro conteúdo.

•Mudando o tamanho do conteúdo

Usualmente numa página a melhor maneira de usar as unidades são absolutas para height (px, cm, pt) e relativas para weight (vw, %, em).

•CSS Borders

Para estilizar uma borda usamos:

border: size(tamanho) style (tipo) color(cor)

border: 2px solid rgba (255,236,210,0.6)

Nos quatro estilos de borda os mais comuns são solid (linha solida), dashed (linha tracejada), dotted (linha em pontinhos) e double (borda dupla).

•Margin & Padding

Para mexer no padding e na margin temos quatro maneiras:

1° Usar uma propriedade simples e dar apenas um valor:

padding: 20px;

margin: 20px;

Neste caso as medidas mudam em todos os lados.

2° Usar a propriedade porem indicar a direção:

padding-top: 5px;

padding-left: 5px;

margin-right: 5px;

margin-bottom: 5px;

Neste caso as medidas mudam apenas para a direção indicada.

3° Usar uma propriedade simples porem com quatro valores:

padding: top right bottom left;

Esta é a ordem dos valores seguidas. A mesma ordem também serve para margin.

padding: 10px 100px 40px 0;

4°Usar uma propriedade simples, porém com dois valores:

margin: top/bottom left/right;

Esta é a ordem dos valores seguidas. A mesma ordem também serve para padding.

margin: 80px 40px;

Aqui vemos que a primeira medida serve para top e bottom e a segunda serve para left e right.

•Float e Tipos de Display

•Display

É a propriedade mais importante do CSS para controlar o layout. Cada elemento tem um valor padrão para o display dependendo de seu tipo. O valor padrão na maioria dos elementos é normalmente block (O conteúdo fica separado), inline (o conteúdo fica alinhado na mesma linha), inline-block (o conteúdo fica separado e na mesma linha), none (esconde o conteúdo).

•Float

A propriedade float especifica como o elemento deve flutuar no site. Os valores mais usados da propriedade float são: right (elemento pra direita), left (elemento pra esquerda), none (o elemento não flutua).

•Flexbox

O Flexbox nos ajuda a organizar esses elementos sem a ajuda do float e também nos ajudam a sanar problemas de Box Model que normalmente acontecem quando acrescentamos, padding, margin e border além da largura do elemento.

O Flexbox é acessado quando um elemento pai do container tem um display: flex;

No html

<div class=”container”>

<div class=” item”> 1 </div>

<div class=” item”> 2 </div>

</div>

Sem um usar a propriedade display:flex; na div class=”container” temos os itens dentro do container um abaixo do outro. Aplicando a propriedade os itens dentro do container estarão um ao lado do outro.

•Flex Direction e Wrap

•Flex Direction

As direções dos itens dentro do container podem ser mudadas ou as formas que vemos usando as propriedades flex-direction.

Dentro de flex-direction temos: flex-direction: row;(é o padrão inicial e deixa todos os itens um ao lado do outro) , flex-direction: reverse-row;(muda a direção dos itens da página e a ordem porem fica um do lado do outro), flex-direction: column; (deixa todos os itens do container em forma de coluna), flex-direction: reverse-column ; (muda a direção dos itens da página e a ordem proem fica em forma de coluna) .

•Wrap

Dentro de uma página tipo em um container tem vários elementos e eles excedem o espaço nos flexbox podemos usar a função flex-wrap os elementos dentro do container que excederão o espaço irão para uma nova linha assim mantendo um padrão. O flex-wrap tem dois tipos mais comuns usados o flex-wrap: wrap; e o flex-wrap: wrap-reverse; (esse muda as posições tipo o que está em cima vai pra baixo e o debaixo vai pra cima).

•Content Alignment

•Justify-Content

No flexbox é usado para dividir os espaços para as colunas os mais usados são:

-justify-content: flex-start;

O conteúdo do container é separado no início da box.

-justify-content: flex-end;

O conteúdo do container é separado no final da box.

-justify-content: center;

O conteúdo do container é separado no centro da box.

-justify-content: space-between;

Os conteúdos do container têm espaços iguais dentro da box.

-justify-content: space-around;

Os conteúdos do container são separados da margem porem com o mesmo tamanho dentro da box.

-justify-content: stretch;

Os conteúdos dos containers se esticam para completar a box.

•Align-items

A propriedade align-items especifica o alinhamento padrão para itens dentro do container flexível.

Os valores mais usados nessa propriedade são align-items: flex-start;(Os itens dentro do container são colocados no início do container), align-items: flex-end;(Os itens dentro do container são colocados no final do container), align-items: center;(Os itens são colocados no centro do container) , align-items: baseline;(Todos os itens do container não importando o tamanho são alinhados a uma linha de base), align-items: stretch;(Os itens dentro do container são esticados para caber no container).

•Shrink, Grow & Basis

-flex-grow;

A propriedade flex-grow especifica quanto o item crescerá em relação ao restante dos itens flexíveis dentro do mesmo contêiner.

-flex-shrink;

A propriedade flex-shrink especifica como o item será reduzido em relação ao restante dos itens flexíveis dentro do mesmo contêiner.

-flex-basis;

A propriedade flex-basis especifica o comprimento inicial de um item.

•Property “flex”

A propriedade flex serve para setar as mesmas propriedades flex-grow, flex-shrink, flex -basis.

A propriedade flex e composta de:

flex: grow shrink basis

Exemplo:

flex: 1 1 50px

• Item Alignment

A propriedade align-self especifica o alinhamento do item selecionado no container porem de forma vertical.

•Grid

O CSS grid é uma forma de montar os layouts de um site por containers.

Propriedades de um Grid Container (elemento pai):

display: grid;

Gera uma grade em nível de bloco.

display: inline-grid;

Gera uma grade de nível em linha.

Para gerar as colunas e linhas usamos as propriedades:

-grid-templates-columns;

-grid-templates-rows;

Para alinhar um conteúdo dentro de um grid usamos a propriedade justify-content e a propriedade align-content.

Para dar espaço entre os conteúdos no grid usamos as propriedades grid-column-gap e grid-row-gap.

Porém temos uma maneira mais fácil de setar esses valores usando apenas uma propriedade o grid-gap.

-grid-gap: linha(row) coluna(column);

-grid-gap: 200px 100px;

Para dividir os espaços dos containers do grid usamos as seguintes propriedades grid-column e grid-row.

-grid-column: 1 / 3;

Começa na linha 1 e termina na linha 3.

-grid-row: 1 / 2 ;

Começa na linha 1 e termina na linha 2.

Para especificar a distância do espaço entre os containers usamos a palavra span.

-grid-column: 1 / span 2;

Grid-Area

Acima vimos que usamos o grid-row e o grid-column para definir os espaços porem temos uma maneira mais fácil que usamos uma só propriedade a grid-area.

-grid-area: row-ponto de início / column-ponto de início / row-ponto final / column-ponto final;

-grid-area: 1 / 1 / span2 / span3;

A linha começa na linha 1 e termina na linha 2 e a coluna começa na coluna 1 e termina na coluna 3.

•Transition

A propriedade transição é usada para dar animação ou movimenta aos itens selecionados.

transition: (primeiro o elemento a ser mudado) (o tempo de duração da animação) (estilo da transição) (delay da transição);

transition: background 2s linear 1s;

O tempo de duração pode ser usado em segundos(s) milissegundos(ms).

Os estilos de transição mais comum são:

linear: especifica o efeito de transição com a mesma velocidade do começo ao fim.

ease: especifica o efeito de transição com um começo devagar, depois rápido, e terminando devagar.

ease-in: especifica o efeito de transição com um começo devagar.

easy-in-out: especifica o efeito de transição com um começo e um final devagar.

ease-out: especifica o efeito de transição com um final devagar.

•Transform

A propriedade aplica uma transformação ao elemento para 2D ou 3D.A propriedade transform permite que você gire o elemento, mova, mecha nos eixos x|y se for 2D, e nos eixos x|y|z se for 3D.

Valores mais comuns da propriedade transform:

none: define que não terá uma transformação aqui.

matrix: define uma transformação 2D, usando seis valores.

transform: matrix((scaleX), (skewY usar valores decimais), (skewX usar valores decimais), (scaleY), (translateX), (translateY);

translate: move um objeto usando os eixos x e y.

transform: translate(20px,10px);

scale: aumenta o tamanho do objeto transformado pela escala.

transform: scale (2.5);

rotate: faz a rotação do objeto definido pelo ângulo.

transform: rotate (45deg);

skew: define a rotação do objeto pelo eixo x e y.

transform: skew (10deg, 20deg);

•Animation

Animações permitem que mudem aspectos do html sem javascript ou flash!

Para começar uma animação usamos a propriedade @keyframes.

@keyframes exemplo {}

Dentro do keyframes temos 2 opções uma que só meche de um valor para outro:

@keyframes exemplo {

from {background: red;}

to {background: green;}

}

Essa é a mais comum usada quando temos mais de 2 valores:

@keyframes exemplo {

0% {background: red;}

50% {background: green;}

100% {background: white;}

}

Para ativar uma animação de um botão precisamos usar as seguintes propriedades:

#div{

animation-name: exemplo;

animation-duration: 4s;

animation-delay: 2s;

animation-iteration-count: infinite;

animation-direction: reverse;

animation-timing-function: ease;

animation-fill-mode: initial;

}

A propriedade animation-duration indica a duração da animação.

A propriedade animation-delay indica o período de descanso da animação.

A propriedade anime-iteration-count indica a quantidade de vezes que essa animação irá fazer dentro dessa propriedade o que destaca e você poder fazer um loop infinito da animação usando o valor infinite.

A propriedade animation-direction indica a direção da animação existem quatro valores principais o alternate, normal, alternate-reverse e o reverse.

A propriedade animation-timing-function indica o estilo da animação que funciona tipo como um transform e seus valores.

Porem temos uma forma mais comum de usar as animações em uma linha só usando a propriedade:

animation: nome tempo de duração timing-function delay iteration-count direction fill-mode;

#div{

animation: exemplo 4s ease 2s infinite reverse initial;

}

•Bezier Curves

Nas animações CSS, as curvas de Bezier são usadas com a função cubic-bezier. A forma da curva representa o desempenho da animação. A curva vive em um sistema de coordenadas 1 por 1. O eixo X deste sistema de coordenadas é a duração da animação (pense nela como uma escala de tempo), e o eixo Y é a mudança na animação.

A função cubic-bezier consiste em quatro pontos principais que ficam nessa grade 1 por 1: p0, p1, p2 e p3. p0 e p3 são definidos para você - eles são os pontos inicial e final, sempre localizados respectivamente na origem (0, 0) e (1, 1). Você define os valores x e y para os outros dois pontos e, onde os coloca na grade, determina a forma da curva a ser seguida pela animação. Isso é feito no CSS declarando os valores x e y dos pontos "âncora" p1 e p2 no formato: (x1, y1, x2, y2). Reunindo tudo, aqui está um exemplo de uma curva de Bezier no código CSS:

animation-timing-function: cubic-bezier(0.25, 0.25, 0.75, 0.75);