CSS (Folhas de Estilo em Cascata) *é uma Linguagem de Marcação, usada para estilizar seu projeto.*

OS TIPOS DE APLICAÇÃO

Existem três formas de aplicar o CSS no HTML.

CSS Inline *– Quando se coloca a formatação do CSS em uma única linha de código.*

CSS Interno *– Quando a formação e feita dentro do HTML em forma de Bloco.*

CSS Externo *– Quando a formação e feita fora do HTML e se utiliza um Link para fazer uma ligação entre HTML e CSS.*

Cores

Em um projeto as cores são importantes pois elas podem levar uma mensagem, emoções e uma identidade visual para seu projeto.

Cor	Associada a	Usar em	Evitar
vermelho	amor, emoção, energia, raiva, perigo	comida, moda, entretenimento, serviços de emergência e saúde	luxo, natureza, serviços em geral
amarelo	felicidade, alegria, otimismo, covardia	dar luz, dar calma e felicidade, chamar atenção	pode indicar que algo é barato ou spam
laranja	divertimento, ambição, calor, cautela	comércio eletrônico, entretenimento, call-to-action	pode se tornar cansativo se muito explorado
verde	saúde, natureza, dinheiro, sorte, inveja	relaxamento, turismo, financeiros, meio ambiente	luxo, tecnologia, meninas adolescentes
azul	competência, sabedoria, calma, frio	tecnologia, medicina, ciências, governo	comida (reduz apetite)
roxo	criatividade, poder, sabedoria, mistério	produtos de beleza, astrologia, ioga, espiritualidade, adolescente	não prende muito a atenção, indiferente
marrom	terra, robustez, estabilidade, amizade	alimentação, imobiliária, animais, finanças	cor considerada conservadora
preto	elegância, autoridade, mistério, morte	luxo, moda, marketing, cosméticos	desconforto e medo
branco	pureza, limpeza, felicidade, segurança	medicina, saúde, tecnologia, luxo (com preto, ouro, cinza)	não chama atenção, deve ser combinado
cinza	formalidade, sofisticação, frieza, indiferença	bens de luxo, efeito calmante	dá a sensação de frieza
rosa	amor, romance, sinceridade, cuidados	produtos femininos e cosméticos	pode tornar muito sentimental e doce

Colocando Cores

background-color *tag usada para colocar cor de fundo.*

Color *tag usada para color cor em algum elemento.*

Existe 4 formas de colocar cores no seu projeto

Nome *– Basta colocar o nome da cor de você quer colocar. Exemplo: Blue, Red. e Green.*

Código Hexadecimal *– você pode combinar números de*

0 a 9 e letras de A até F, com um asterisco na frente para criar uma cor.

Exemplo: #FF7F50 Coral em Hexadecimal.

RGB *– RGB significa Vermelho, Verde e Azul, combinado a intensidade de cada cor, você pode criar uma diversidade de cores.*

Exemplo: R 120, G 120 e B 120, formam Cinza.

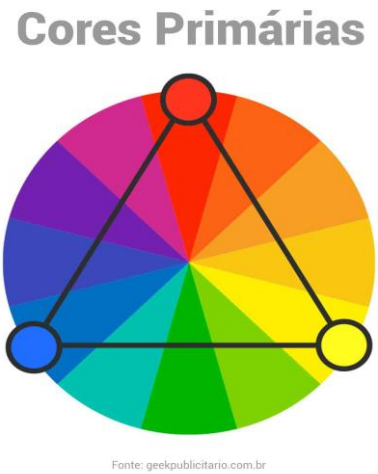
HSL *– Esse indica o tom da cor, a Saturação dela e a Claridade se vai ser escuro ou claro.*

Observação: *Em alguns casos e possível colocar o elemento* ALFA *que indica a Opacidade (Transparência) da cor.*

Cores Harmônicas

Em um projeto e fundamental a harmonia de cores, pois cria um ambiente confortável e agradável para seus usuários.

Para isso escolha uma paleta de cores harmônica e que faça sentido para seu projeto.



Cores Primarias

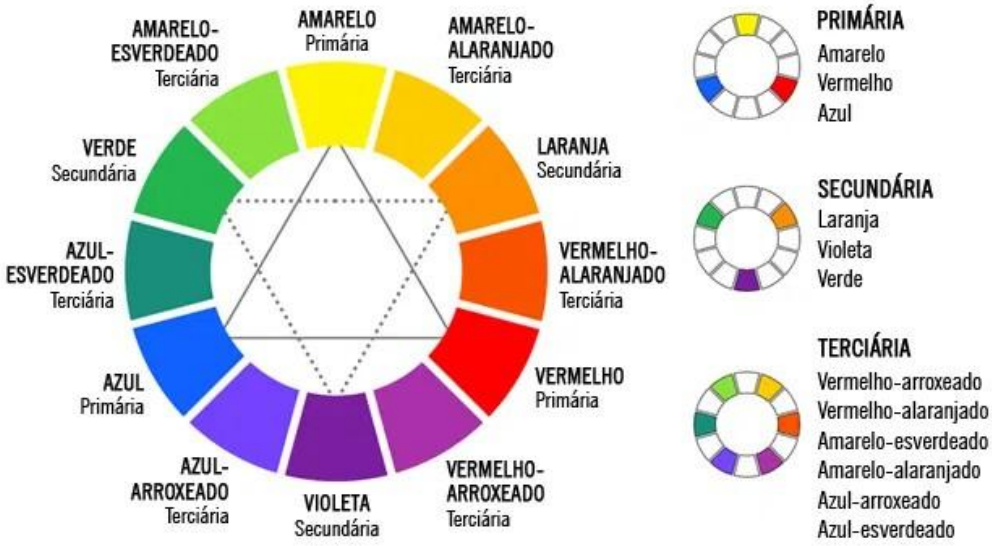
As cores primarias são Amarelo, Azul e Vermelho, são chamadas de cores primarias pois através delas que se dá origem a outras cores.

Cores Secundarias

As cores secundarias são resultado da mistura das cores primarias.

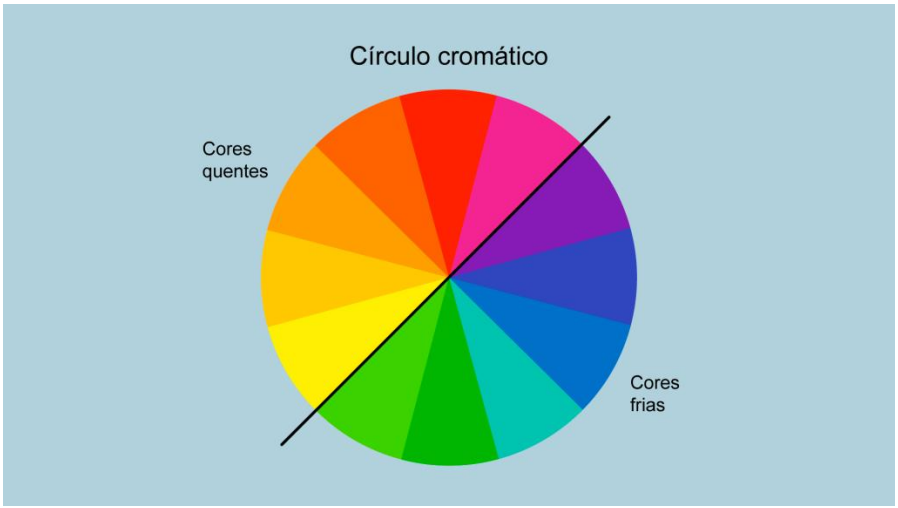
Cores Terciarias

As cores terciarias são a misturas das segundarias.



Cores Frias: *As cores frias são compostas por tons de azul, verde e violeta, e são conhecidas por seus efeitos calmantes e refrescantes. Cores frias também são frequentemente associadas à tranquilidade, serenidade e paz.*

Cores Quentes: *As cores quentes são o amarelo, o vermelho e o laranja, e são associadas ao fogo, ao sol e ao sangue. No círculo cromático, as cores quentes estão localizadas em uma das extremidades, enquanto as cores frias estão na outra.*



Degrade

Para colocar degrade no seu projeto, coloque a tag background-image: linear-gradient (), você pode escolher o sentido das cores, com a propriedades to right, to left,

to up e to bottom. E colocar as cores logo em seguida dentro dos parênteses. Obs. No máximo 5 cores.

Você também pode colocar a porcentagem que as cores vão ter de espaço no degrade.

Exemplo: blue 20% e red 15%

Existe também o background-image: radial-gradient (circle, white, blue). Que cria um degrade em forma de círculo.

Tipografia

É a arte e a técnica de organizar letras e textos de forma visualmente atraente e legível. Ela envolve a escolha de fontes, tamanhos, espaçamento e alinhamento.

E um recurso extremamente importante pois é têm a mesma importância das cores.

Escolha de fonte

Na hora de escolher uma fonte para seu projeto e necessário escolher o correto, pois ela ajudar a passar a mensagem que deseja.

Também vale ressaltar que em um projeto e recomendável não exagerar nas fontes, escolhes as que fazem sentido no assunto, não exagerar na quantidade, tenha entre duas ou três tipos de fontes diferentes no máximo.

E é claro que as fontes escolhidas, tendem a ser legíveis, para assim o usuário possa ler.

Colocando Fontes

Fontes Locais

Para colocar fonte local no CSS, coloque a tag font – family e escolha a fonte que você quer aplicar. Obs. Ao colocar diversos font – family, o CSS vai priorizar o primeiro.

Fontes Baixadas

Para fontes baixadas coloque a tag @font – face, na propriedade font – family coloque o nome que você quer para sua fonte, e no src: url coloque o endereço de onde a imagem está.

Fontes Da Internet

Para fontes da internet use @import, e na url o endereço de origem da fonte.

Tamanho da Fonte

Para mudar o tamanho da fonte, use a tag font – size e coloque tamanho em px ou em (recomendação da W3C).

A medida em equivale ao tamanho original da fonte (16px), ao colocar a medida em 2em, dobra o tamanho (32px).

Estilo de Fonte

Algumas fontes têm opções de estilização como Negrito e Itálico.

Para Negrito utiliza-se font – weight *e escolha os valores de Negrito forte, fraco ou normal.*

E para Itálico, coloque font – style *e escolha o valor italic*.

Observações: *Nem todas as fontes têm opções de Negrito e Itálico.*

Short Hands: *E um atalho para economizar tempo, basta colocar* font: *e os valores na ordem*, font – style, font – weight, font – size e font – family.

Textos Formatados

Existe algumas tags que permitem formatar o texto, como:

Text – transform transforma a fonte em Maiúscula ou Minúscula.

Font – variant: small-caps transforma as fontes em Maiúscula em tamanho reduzido.

Alinhamento de Texto

Para alinhar um texto, coloque a tag Text – aling *e coloque os valores de alinhamento.*

Left: *Alinhamento à esquerda*.

Right: *Alinhamento à direita.*

Center: *Centraliza o texto.*

Justify: *Distribui o texto de forma uniforme.*

Text – Indent: *E uma tag que permite criar um recuo no início do texto, basta colocar os valores em px.*

Ferramentas Para Conseguir Fontes.

Fonts Ninja: *Uma extensão do Chrome web store, que ativa pode te mostrar as fontes que estão sendo utilizada na página web.*

Google Fonts: *Uma biblioteca de fontes, escolha e incorpore ao seu projeto.*

DaFont: *Um site que pode ser usado para baixar fontes.*

Ferramentas para Fontes em Imagem

Existe algumas ferramentas que podem ser usadas para descobrir quais fontes está em uma Imagem.

what font is.com, Font Squirrel e Myfonts.

Seletores Personalizados

Os seletores Personalizados servem para identificar e agrupar elementos.

Existe dois seletores utilizados no CSS o id que pode ser identificado com um asterisco (#) e o class com um ponto (.)

Diferença entre id e class

O id seve para identificar um elemento de forma exclusiva, isso e só aquele elemento vai ter aquela identificação.

Já o class serve para identificar vários elementos, sem ser de forma exclusiva.

Pseudo – Classes

Pseudo – Classes e uma condição que você coloca depois de um seletor personalizado, com dois pontos, que serve para dar uma função para o elemento.

: hover *cria uma condição quando passa o mouse por cima.*

: link *para estilizar links para páginas não visitadas*

: *visited para estilizar links para páginas visitadas*

: active *para estilizar o link ativo*

: *enabled para classificar elementos como ativados*

: *disabled para classificar elementos como desativados*

: *checked para classificar elementos como marcados*

Pseudo – Elemento

Pseudo – elemento é uma palavra-chave adicionada, depois de 4 pontos :: a um seletor que permite que você estilize uma parte específica do elemento selecionado

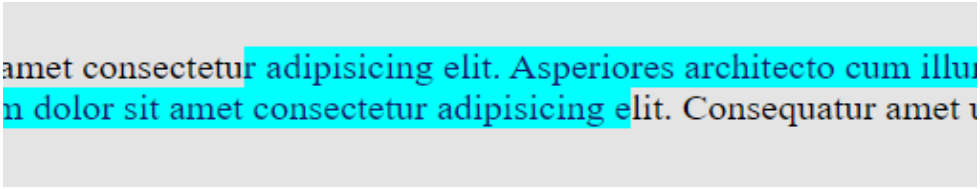
:: first-line *é usado para estilizar a primeira linha de um elemento.*

:: first-letter *é usado para estilizar a primeira letra de um elemento.*

:: before *é usado para colocar um conteúdo antes de um elemento.*

:: after *é usado para colocar um conteúdo depois de um elemento.*

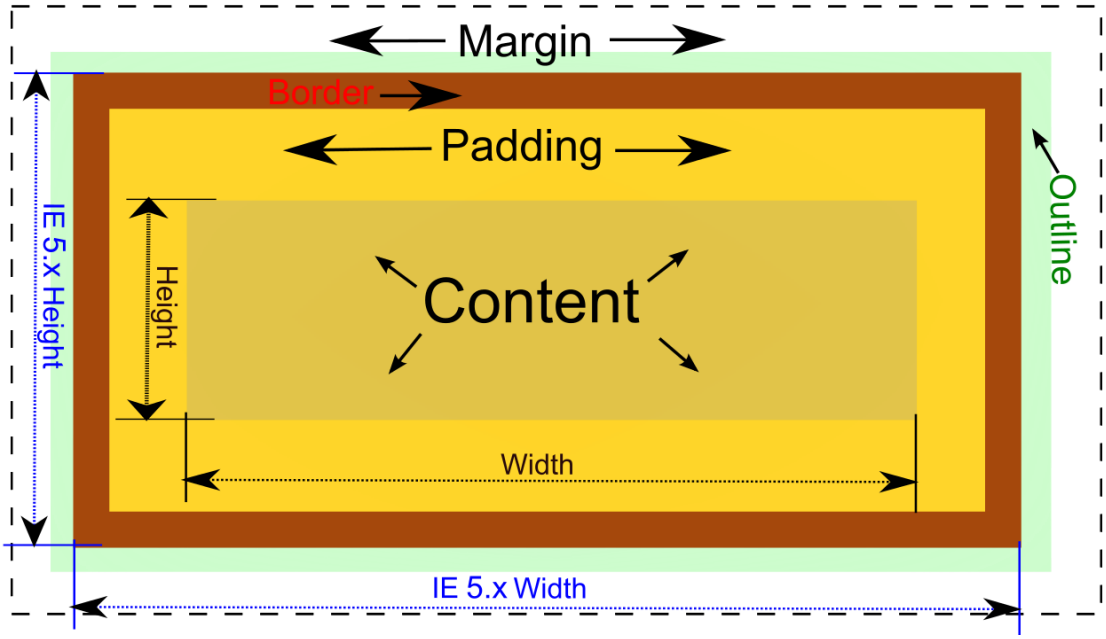
:: selection *é usado para alterar as características de um elemento selecionado, como cor ou cor de fundo.*



Modelo de Caixa / Box Model

E um conceito do CSS e HTML onde divide os elementos em caixas, facilitando a estilização.

Propriedades

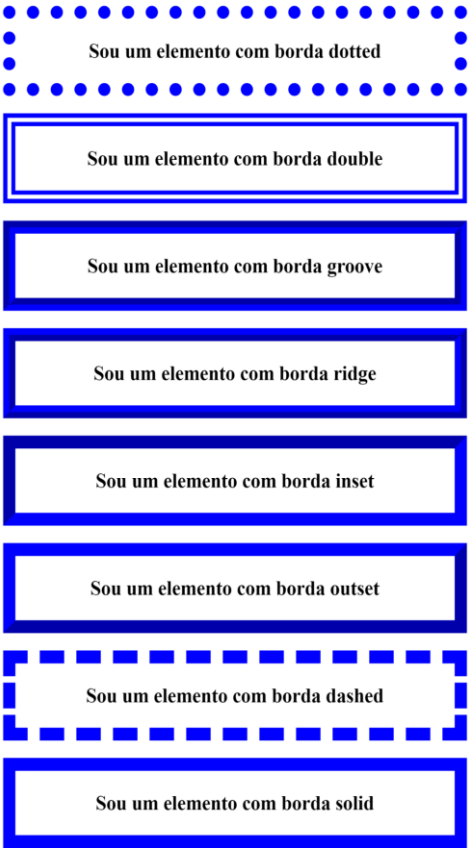


Content: *Conteúdo dentro da caixa*.

Padding: *Preenchimento de dentro da Caixa.*

Border: *Borda que fica ao redor do conteúdo da caixa.*

Tipos de Borda



Outline: *E o contorno que fica em volta da borda.*

Margin: *E o espaço que fica de fora da caixa.*

Colocando Propriedades

Height: Serve para configurar a Altura.

Width: Serve para configurar a Largura.

Ao configurar as propriedades do Modelo de Caixa, vale ressaltar que deve ser feito de maneira horaria, começando de cima.

Para configurar as propriedades coloque seus respectivos nomes e os valores que você quer, com Pixel (px), Vh (Viewport Height) ou Porcentagem (%).

Tipos de Caixa

Box – level / Block - level: esse elemento se trata de um tipo de caixa que se inicia em uma linha nova e ocupa a largura total da página ou elemento que está contido, logo após pula para a próxima linha e continua normalmente.

Exemplos de tags do tipo "box-level / block-level": <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 exemplo) sem pular para a linha de baixo.

Exemplos de tags do tipo "inline-level": <a> <code> .

Aninhamento de caixas

Aninhamento de caixa, e a forma de organizar uma caixa dentro da outra, trazendo um Visual mais complexo e dando características mais individuais.

Também pode ser chamado de Grouping Tags ou Semantic Tags.

Algumas tags são

Header *Cria áreas relativas a cabeçalhos. Pode ser o cabeçalho principal de um site ou até mesmo o cabeçalho*

de uma seção ou artigo. Normalmente inclui títulos ***<h1> - <h6>*** *e subtítulos. Podem também conter menus de navegação.*

Nav *Define uma área que possui os links de navegação pela estrutura de páginas que vão compor o website. Um <nav> pode estar dentro de um <header>.*

Main *É um agrupador usado para delimitar o conteúdo principal do nosso site. Normalmente concentra as seções, artigos e conteúdos periféricos.*

Section *Cria seções para sua página. Ela pode conter o conteúdo diretamente no seu corpo ou dividir os conteúdos em artigos com conteúdo específicos. Segundo a documentação oficial da W3C, "uma seção é um agrupamento temático de conteúdos, tipicamente com um cabeçalho".*

Article *Um artigo é um elemento que vai conter um conteúdo que pode ser lido de forma independente e dizem respeito a um mesmo assunto. Podemos usar um para delimitar um post de blog ou fórum, uma notícia etc.*

Aside *Delimita um conteúdo periférico e complementar ao conteúdo principal de um artigo ou seção. Normalmente um conteúdo está posicionado ao lado de um determinado texto ou até mesmo no meio dele.*

Footer *Cria um rodapé para o site inteiro, seção ou artigo. É um conteúdo que não faz parte diretamente do conteúdo nem é um conteúdo periférico (o que caracterizaria um), mas possui informações sobre autoria do conteúdo, links adicionais, mapa do site, documentos relacionados.*

Sombra nas Caixas

Ao colocar sombra na box – model e possível criar umas impressões de profundidade e dar destaque para essa caixa.

Para isso coloque a tag box – shadow e defina os valores de deslocamento horizontal, vertical, embaçamento e cor.

Sombra em Texto

Isso também vale para texto, coloque a tag Text – shadow e defina os mesmos valores.

Vértices Arredondas

Para arredondar as pontas das bordas, coloque a tag Border – radius e os valores que deseja, se for colocado apenas um valor, servira para todas as pontas.

Imagem como Borda

Para colocar uma imagem como bordar coloque, a tag Border – image : url() 30 repeat; coloque o endereço da imagem dentro da url, e o 30 repeat serve para repetir a imagem e formar uma borda com imagem.

Planejando um Projeto

Ao criar um Projeto e fundamental ter um planejamento, e para isso e necessário ter ferramentas que te ajude nisso.

O site MockFlow pode ajudar no planejamento de seu projeto.

Variável no CSS

Variável e uma forma de guardar valores ou informações, e que vale lembrar que toda linguagem de Programação tem variável, mas tem toda linguagem que tem variável e linguagem de programação.

Para criar uma variável no CSS, coloque a propriedade :root {} *e dentro dos colchetes coloque os valores das variáveis depois de dois traços*.

Exemplo:

```
:root {
    --cor00: #c5ebd6;
    --cor01: #83e1ad;
    --cor02: #3ddc84;
}
```

Seletor Global

*Ao colocar o asterisco (*), você selecionou um seletor global, isso é, um seletor que irá colocar um padrão em todos os elementos HTML.*

Você pode utilizar disso para padronizar os elementos HTML, e logo em seguida, colocar estilo de forma individual em cada elemento que deseja.

Espaçamento Entrelinhas

O espaçamento entrelinhas podem ser usados para ajudar na leitura, para configurar o espaçamento, coloque a tag Line – Height *e coloque os valores de tamanho em* em

Personalizando Lista

```
ul {
    list-style-type: '\2714\0020\0020';
    columns: 2;
    list-style-position: inside;
}
```

A tag list – style – type serve para colocar símbolos ou marcas, os números \0020 serve para dar um espaço em branco.

A Columns serve para colocar colunas nas listas.

E a list – style – position serve para colocar os símbolos e marcas dentro da caixa da lista.

Responsividade no CSS

A responsividade em um projeto e importante pois, ele pode ser visto em diversas telas.

Para isso coloque largura mínima e máxima min – widht e max – widht.

E Para a imagem ser responsiva coloque widht 100%.

Responsividade de Vídeo

Para fazer um vídeo importado do youtube se torna responsivo, vai precisar colocar o vídeo dentro de uma div e colocar algumas configurações.

```css
div.video {
    position: relative;
    background-color: var(--cor4);
    height: 0px;
    margin-left: -20px;
    margin-right: -20px;
    margin-bottom: 15px;
    padding-bottom: 59%;
}
```

```css
div.video > iframe {
    position: absolute;
    top: 5%;
    left: 5%;
    width: 90%;
    height: 90%;
}
```

A div. video serve para configurar as propriedades da div onde está o vídeo.

E o div. video > iframe configura as propriedades do vídeo.

Imagem de Fundo

Para colocar uma imagem de fundo no seu projeto, coloque a tag background – image: url (); e dentro da url coloque o endereço da imagem.

Configurações de Imagem de Fundo

Existe algumas configurações que dá para se fazer na imagem de fundo.

Background – size *serve para configurar o tamanho da Imagem de Fundo.*

Background – repeat *serve para escolher se a imagem vai se repetir ou não.*

Repeat – x e Repeat – y *serve para informar se a imagem vai se repetir de forma horizontal (x) ou vertical (y).*

Background – position *configura a posição da imagem ao fundo.*

Configurando Tamanho da Imagem de Fundo

Background - size: 100% 100%; *preenche a tela, porém acaba achatando a imagem e distorcendo.*

countain; *a imagem fica centralizada, porém cria laterais vazias, se o Repeat estiver ativo vai criar repetições se não vai ficar um fundo preto.*

cover; *cobre a tela totalmente, mesmo que necessite cortar alguns pedaços.*

Fixando Imagem de Fundo

E possível torna a imagem de fundo fixa, para isso coloque a tag background-attachment *e o valor* fixed *para conforme o conteúdo desce a imagem fica fixa.*

Short Hands de Background

O sentido da Short Hands.

background – color
background – image
background – position
background – repeat
background – attachment

Contêineres

O conceito de contêineres e de um elemento que pode ter outros elementos dentro de si, como no caso a tag DIV, que pode ter outras tags e elementos agrupados dentro.

Centralização Vertical de Caixas

Para centralizar uma caixa de forma vertical, crie uma caixa e dentro crie uma segunda caixa, a primeira vai ter configurações padrões como cor de fundo, tamanho e posição padrão, que e relative.

A segunda caixa, faça as configurações padrões (tamanho, e cor) e coloque position: Absolute. Com isso coloque as propriedades top e left em 50% e coloque a última tag transform: translate(-50%,-50%).

Efeito Parallax

E uma técnica que cria a impressão de que objetos ou pessoas estão em movimento.

Para criar esse efeito, crie uma caixa e coloque um id, dentro do id coloque uma imagem de fundo, e coloque as configurações de.

background-position: right center; *(para pegar o ponto da imagem que você quer)*

background-repeat: no-repeat; *(para a imagem não ficar se repetindo)*

background-size: cover; *(para a imagem ficar em um tamanho bom e responsivo)*

background-attachment: fixed; *(para criar um efeito parallax)*

Tirando a Barra de rolagem

Para tirar a barra de rolagem, e permitir que a página continue com a propriedade de rolar, coloque a propriedade: :: -webkit-scrollbar *com os valores* width: 0; e height: 0;.

Propriedade Float

define a posição de um elemento, colocando-o no lado esquerdo ou direito do seu recipiente.

Calc no css

Serve para fazer cálculo, e o valor adquirido e o resultado desse cálculo.

Exemplo:

Width: (100% - 50%) o Tamanho vai ser o resultado desse cálculo.

FlexBox

E um método que permite que os elementos dentro de um container se adaptem conforme o necessário.

Quem é o Pai? – e onde vai abrigar os elementos filhos.

Elementos Filhos – são os elementos que vão ficar dentro do elemento Pai.

Configurando direção (flex-direction)

A configuração e colocada no container Pai, e vai dar a orientação que os elementos têm que seguir como, row que é em linha começando da direita para a esquerda, row-reverse, que também e em linha, porém e da direita para esquerda, a orientação column, que vai ser uma coluna com orientação dos elementos de cima para baixo e o column-reverse que vai ser de baixo para cima.

Para colocar essas orientações use flex-direction:

Eixos no FlexBox

O eixo principal (main-axis) ele tem dois pontos o inicial (main-start) e o final (main-end), se por acaso estiver na direção row (linha deitada, que começa da esquerda para a direita), o ponto inicial (main-start) também vai começar da esquerda e o final (main-end) vai terminar na direita.

No row-reverse (linha reversa) o eixo principal muda e começa da direita para a esquerda e automaticamente, os pontos inicial e final também mudam para acompanhar o eixo principal. Sendo assim o (main-star) começa da direita e o final (main-end) termina na esquerda.

Na configuração Column (coluna que começa de cima para baixo), o sentido passa de horizontal (deitado) para vertical (em pé), o eixo principal (main-axis) começa de cima para baixo, e seu pontos inicial (main-start) e final (main-end) também acompanham o sentido.

E no Column-reverse (coluna que começa de baixo para cima), o eixo principal(main-axis) também começa de baixo para cima, e seu pontos inicial e final o acompanham automaticamente.

O eixo Transversal (cross-axis) ele também tem dois pontos o inicial (Cross-start) e final (Cross-end). Caso for na configuração row (linha deitada), ele começa de cima para baixo, sendo que o ponto inicial(cross-start) também começa de cima e o final(cross-end) acaba em baixo.

No row-reverse (linha reversa) o eixo transversal (cross-axis) e seus pontos inicial e final, não mudam o sentido, continua o mesmo.

Já no Column (coluna que começa de cima para baixo), o sentido do eixo transversal (cross-axis) muda e passa a ser de forma horizontal (deitado) no sentido da esquerda para a direita, e seus pontos Inicial (cross-start) e final (cross-end) também seguem esse sentido.

E no Column-reverse (coluna que começa de baixo para cima) os sentidos não mudam, continuam o mesmo da esquerda para a direita.

Flex-Wrap

Essa propriedade serve para configurar o comportamento da capsula Pai (onde vai os elementos filhos)

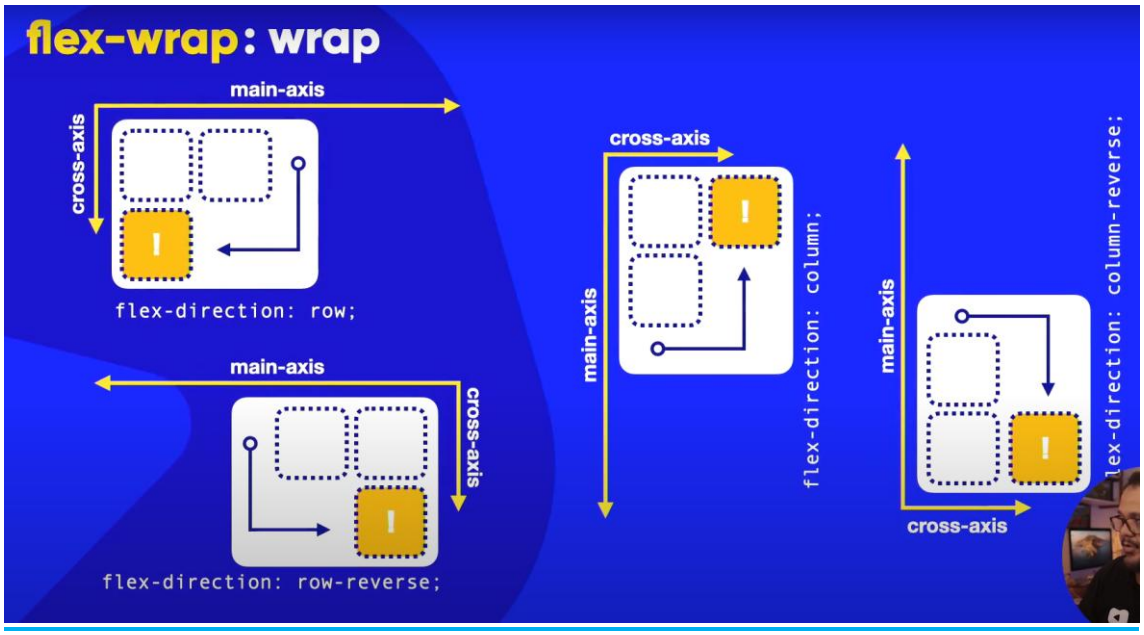
Flex-Wrap: nowrap; e a configuração padrão, informa que não deve quebrar a linha. Caso o bloco Pai, diminua de tamanho os elementos dentro vão encolher.

Ao só colocar display:flex; por padrão o Flex-Wrap se torna nowrap.

Atenção: o tamanho do bloco vai depender do conteúdo que vai dentro dele.

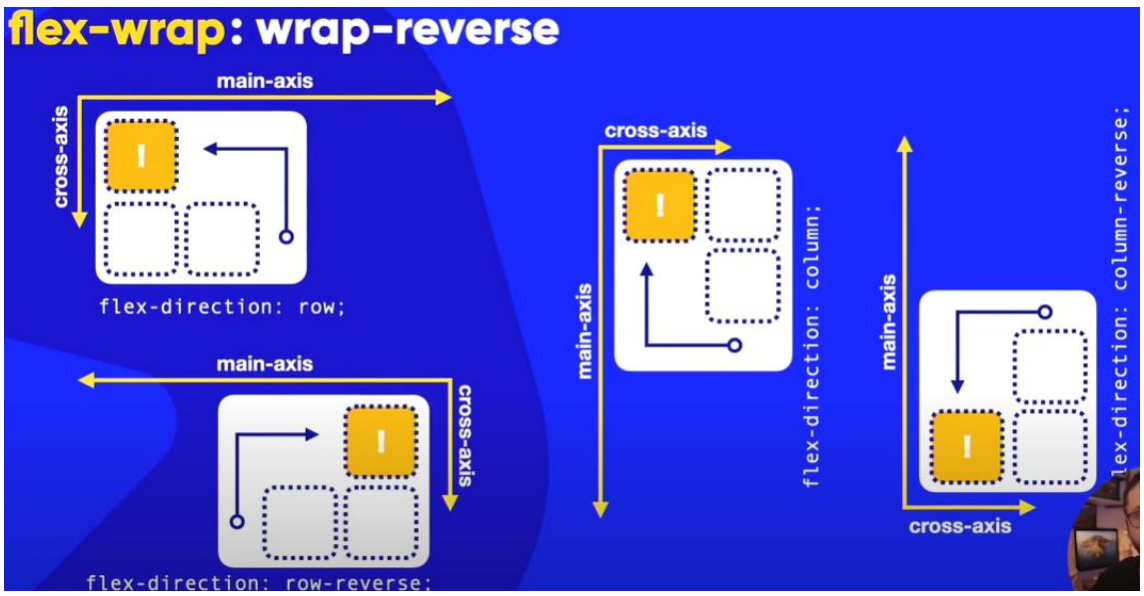
Flex-Wrap: Wrap; indica que o último elemento quebre para a direção do eixo transversal (cross-axis).

Exemplo:



Flex-Wrap: wrap-reverse; indica que o último elemento quebre no sentido oposto do eixo transversal (cross-axis).

Exemplo:



Flex-Flow e um short Hand que configura o Flex-direction e o Flex-Wrap em uma única configuração.

Exemplo:

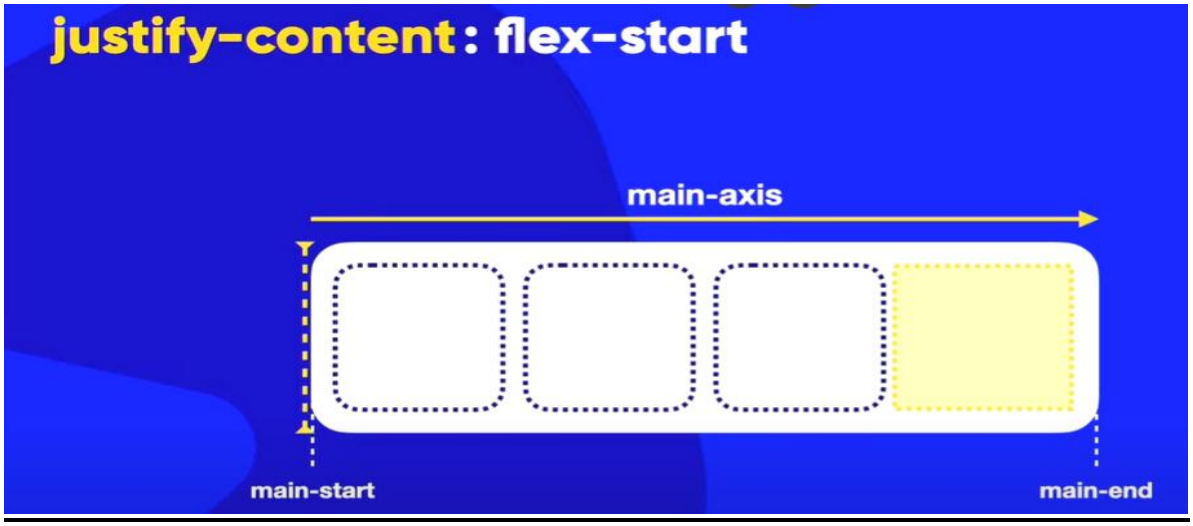
Flex-direction: row; e Flex-Wrap: nowrap; e o mesmo que Flex-Flow: row nowrap;

Justify-content

E uma forma de alinhar os itens dentro da capsula Pai, através dos pontos inicial (main-start) e final (main-end) do eixo principal (main-axis).

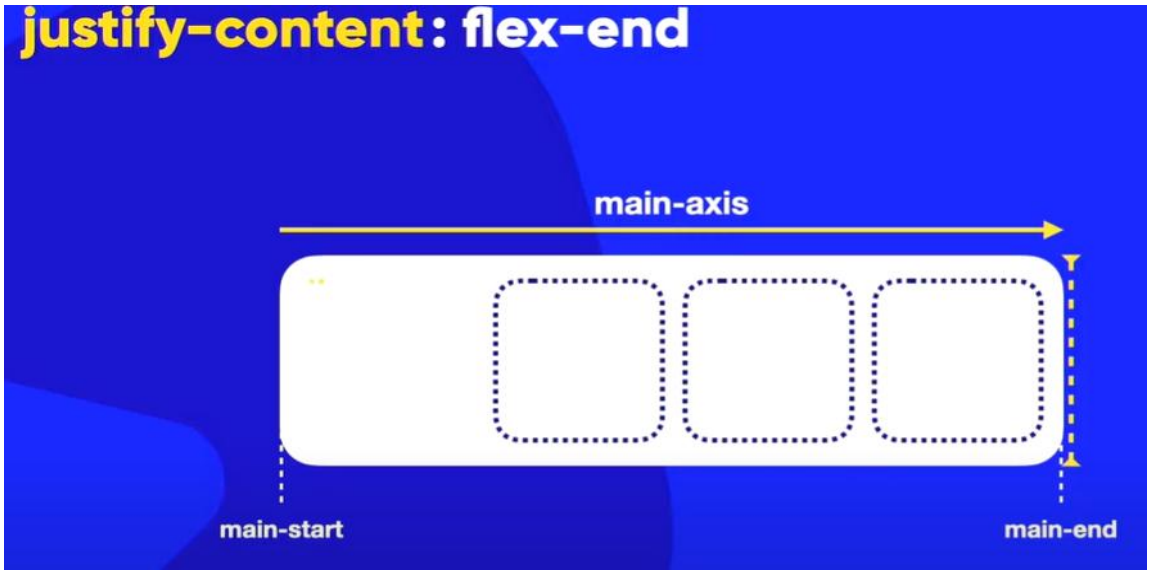
Justify-content: flex-start faz com que o primeiro item fique alinhado no ponto inicial (main-start), os outros elementos vão ficar grudados nele, o espaço vazio vai ser jogado para o final do eixo principal (main-end).

Exemplo:



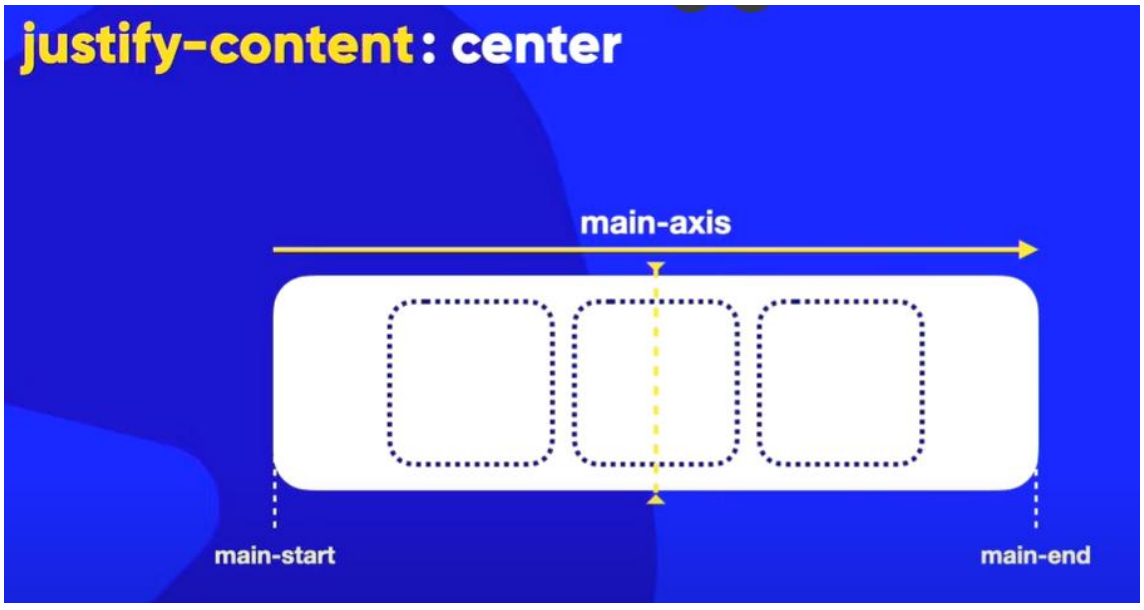
Justify-content: flex-end faz com que os elementos se alinhem no final do eixo principal (main-end) e deixa o espaço vazio no início do eixo (main-start).

Exemplo:



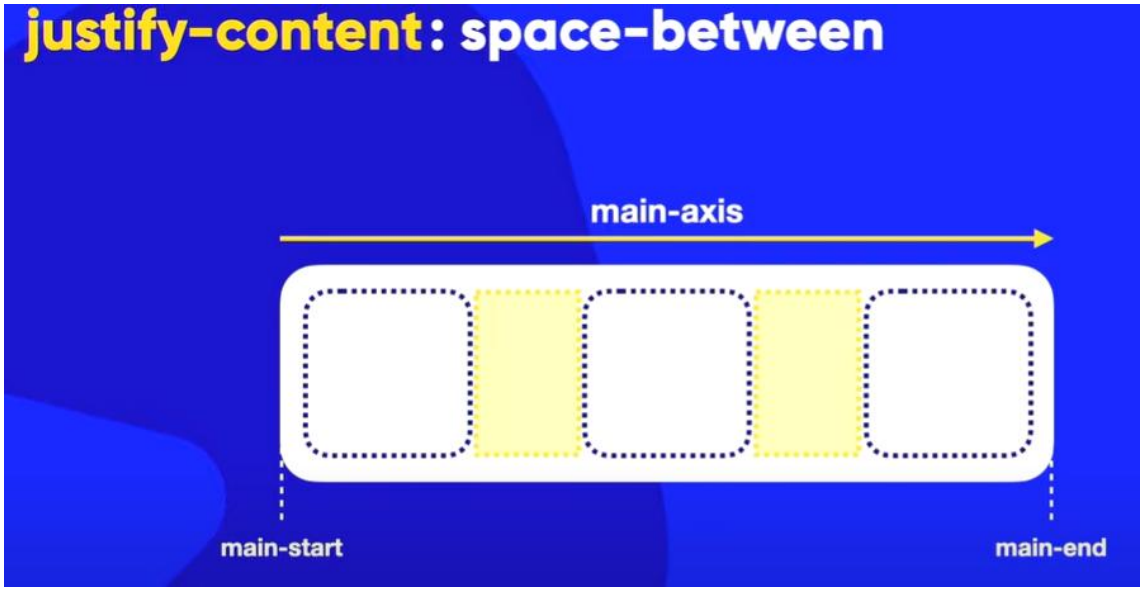
Justify-content: center vai centralizar os elementos e vai distribuir os espaços vazios de forma igual no (main-start) e no (main-end).

Exemplo:



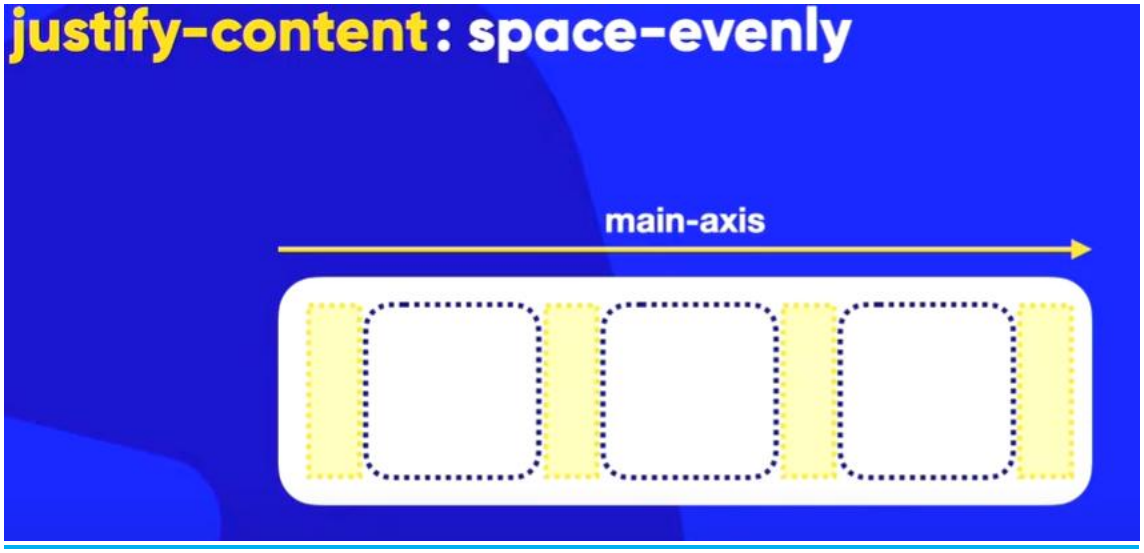
Justify-content: space-between vai alinhar o primeiro elemento no início do eixo principal (main-start), vai alinhar o último elemento no final do eixo (main-end) e centralizar os outros elementos, com espaços iguais.

Exemplo:



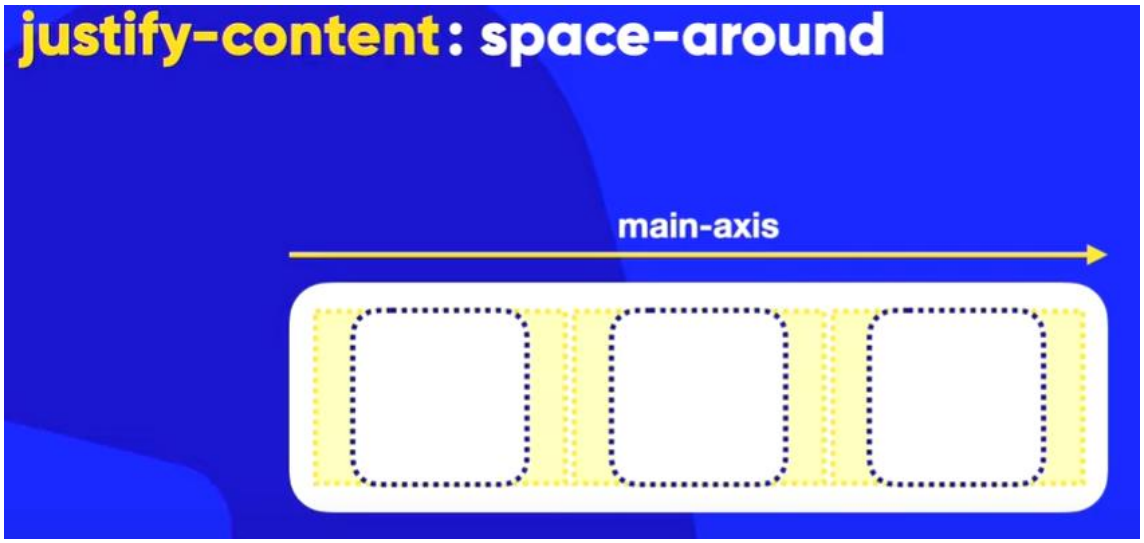
Justify-content: space-evenly alinha os elementos com os espaçamentos iguais, incluindo antes do ponto inicial (main-start) e antes do ponto final (main-end).

Exemplo:

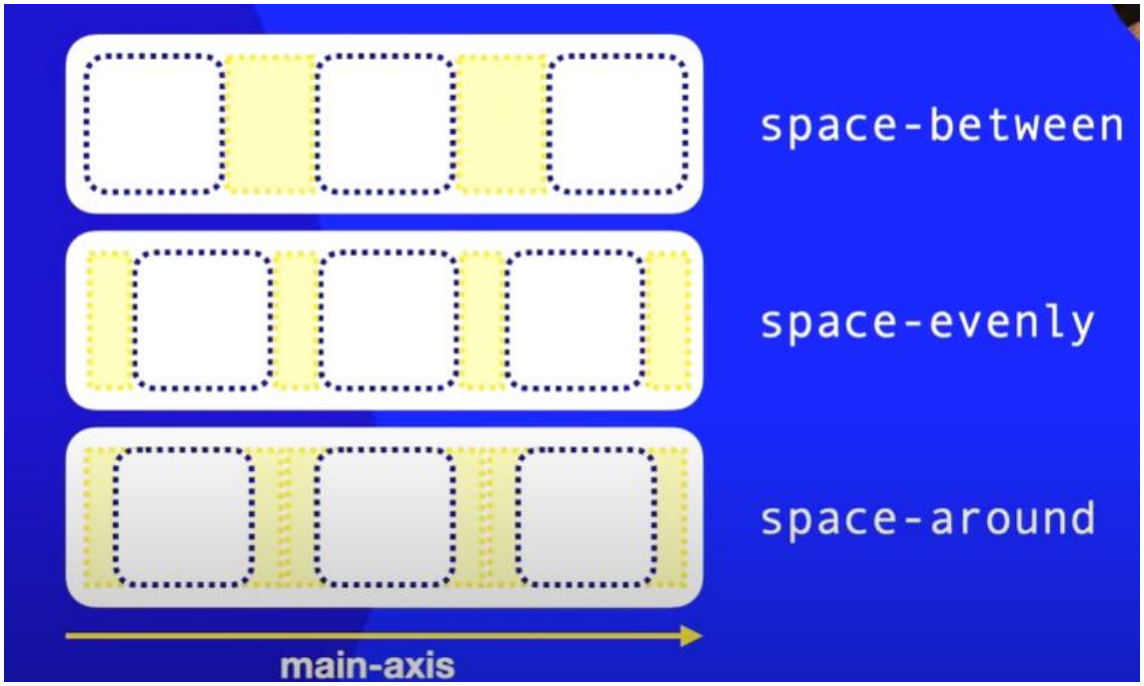


Justify-content: space-around divide em espaços da mesma quantidade dos elementos, e centraliza os elementos dentro desses espaços.

Exemplo:

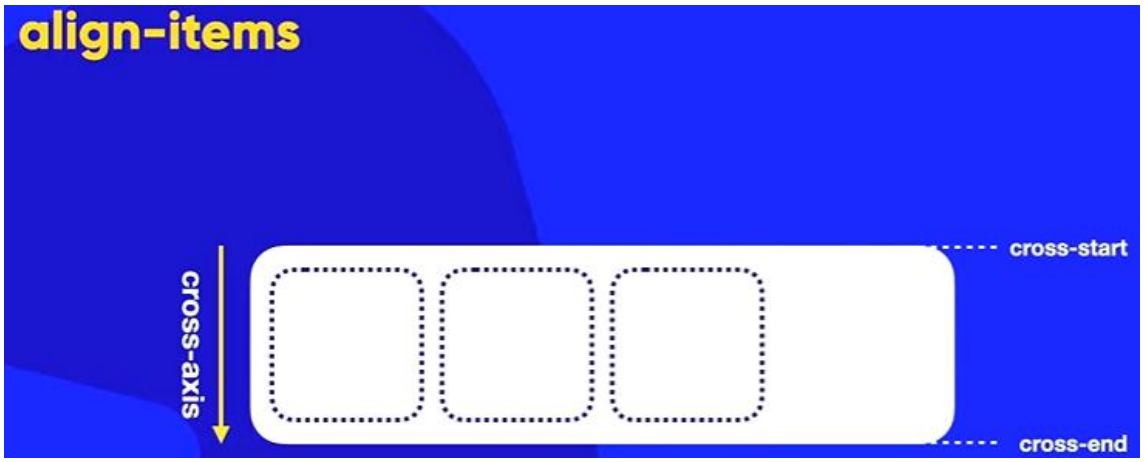


Diferenças entre os Space-between, Space-Evelyn e Space-Around.



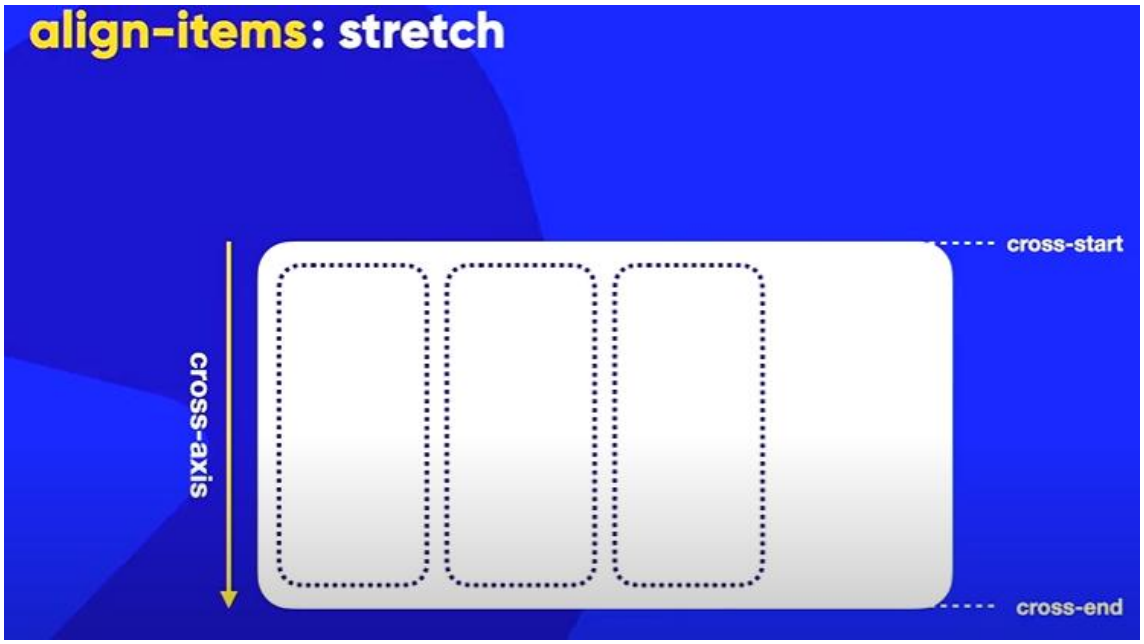
Alingn-items

E uma forma de alinhar os itens através do eixo transversal (cross-axis).



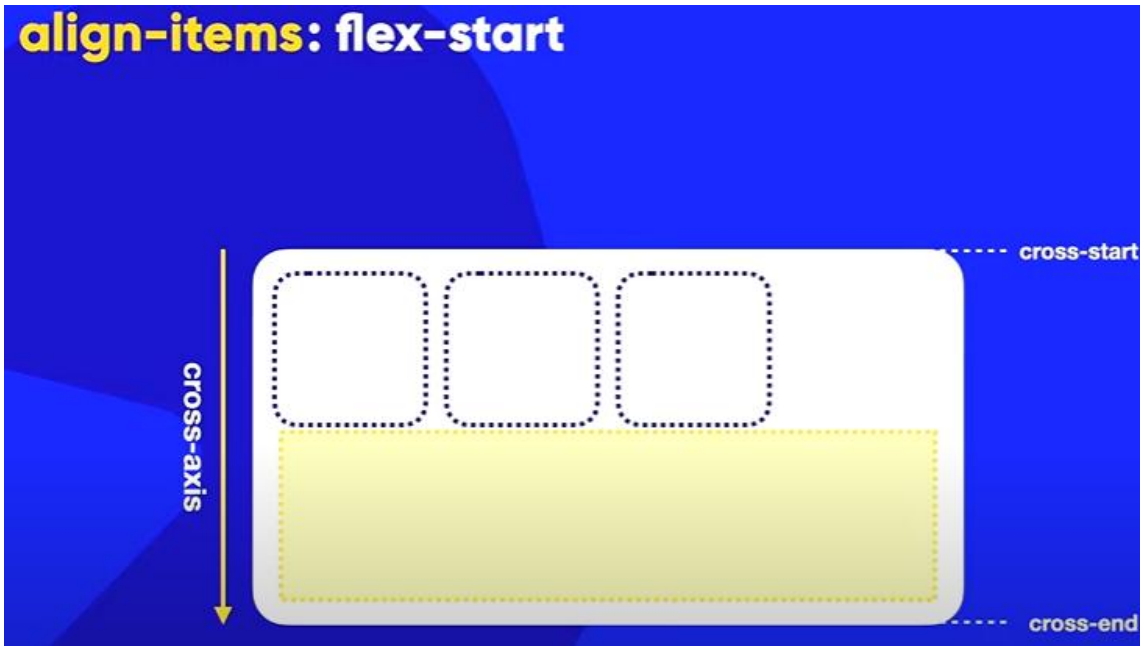
Aling-items: stretch configuração padrão, onde ao esticar o container Pai, irá esticar os elementos de dentro também.

Exemplo:



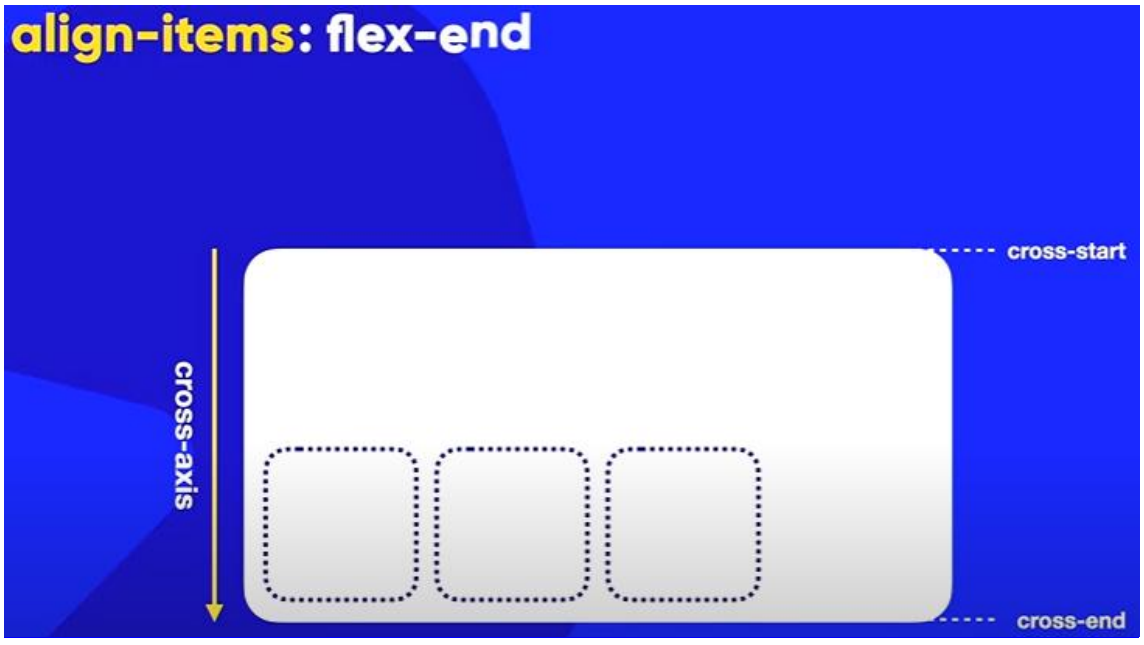
Aling-items: flex-start vai alinhar os elementos no início do eixo (cross-start).

Exemplo:



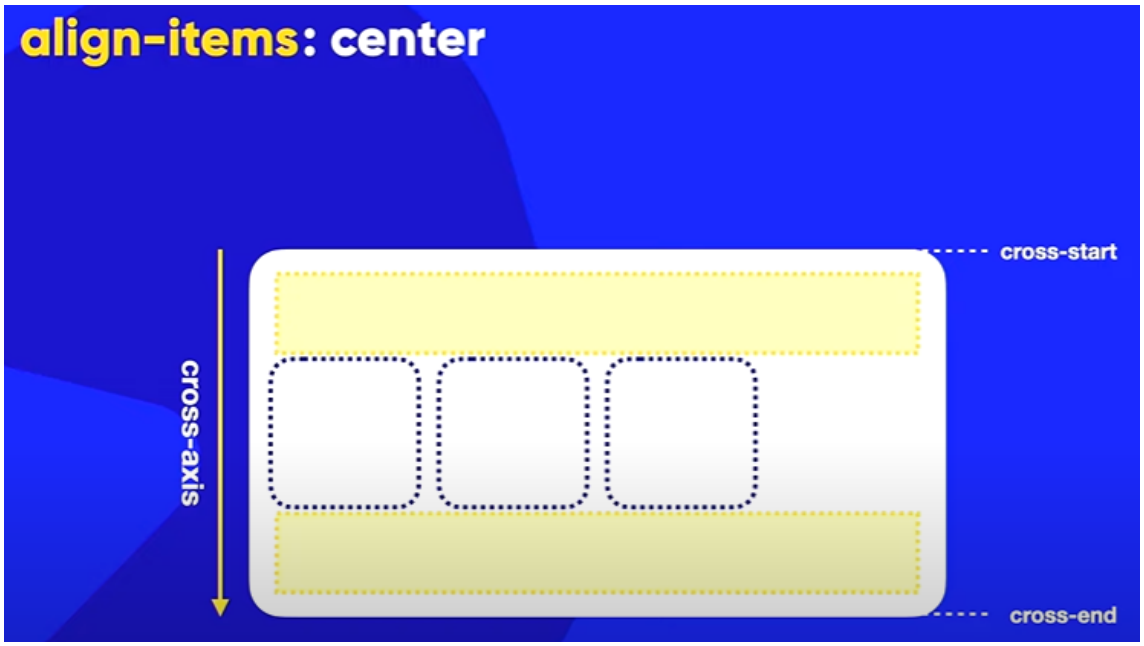
Aling-items: flex-end alinha os elementos no final do eixo (cross-end).

Exemplo:



Aling-items: center alinha os elementos ao centro e coloca espaços vazios de forma igual no início do eixo (croos-start) e no final (cross-end).

Exemplo:



Centralizando de Forma absoluta

Para centralização absoluta, coloque algumas configurações no container Pai como display: flex, justify-content: center e aling-items:center.

Exemplo:

```
#Pai {
    display: flex;
    justify-content: center;
    align-items: center
}
```

Conteúdo Empacotado

E o conjunto de elementos que são organizados dentro de containers.

ALIGN-CONTENT: serve para alinhar os elementos do eixo transversal quando eles estão empacotados.

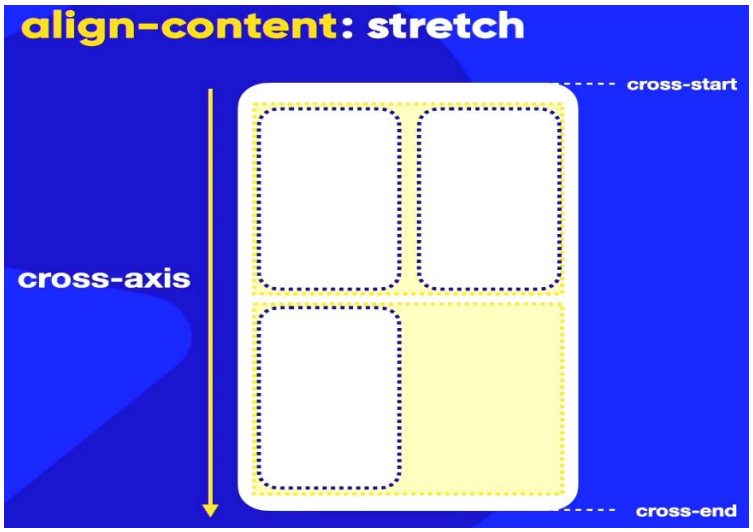
Diferenças do Justify-Content, Aling-Items e Aling-Content.

Justify-Content: Alinha os elementos através do eixo principal (main-axis).

Aling-Items: Alinha os elementos através do eixo transversal (cross-axis).

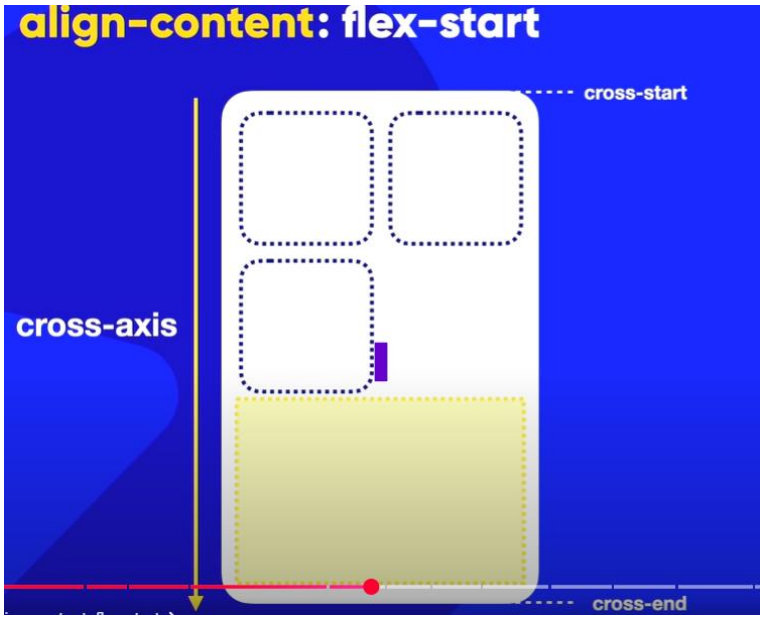
Aling-Content: Alinha os elementos do eixo transversal (cross-axis), quando eles estão empacotados.

Aling-content: Stretch Configuração padrão, onde divide em áreas da mesma quantidade de elementos, colocando os elementos dentro dessas áreas, e esticando-os para preencher o espaço que sobrou. Exemplo:

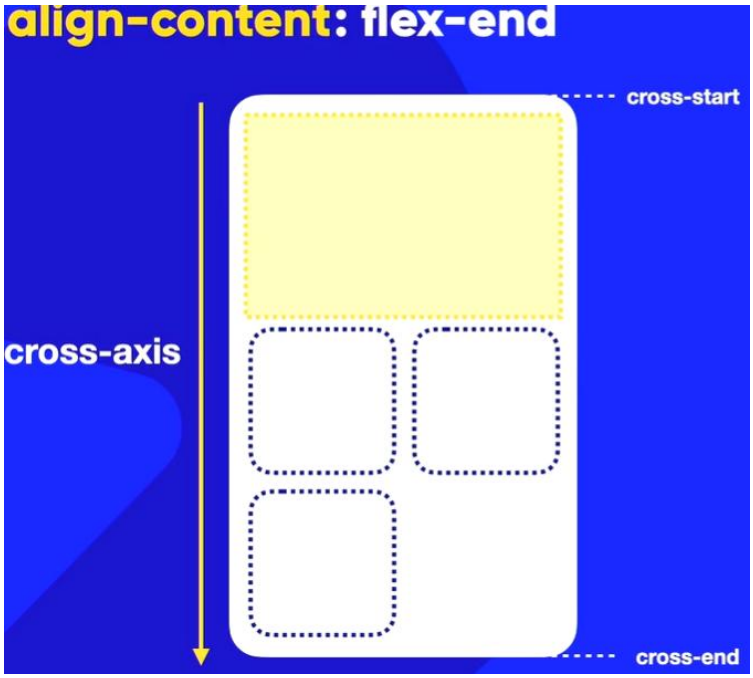


Aling-content: flex-start faz com os elementos se alinhe no início do eixo transversal (cross-axis) e empurra o espaço vazio para o fim do eixo.

Exemplo:

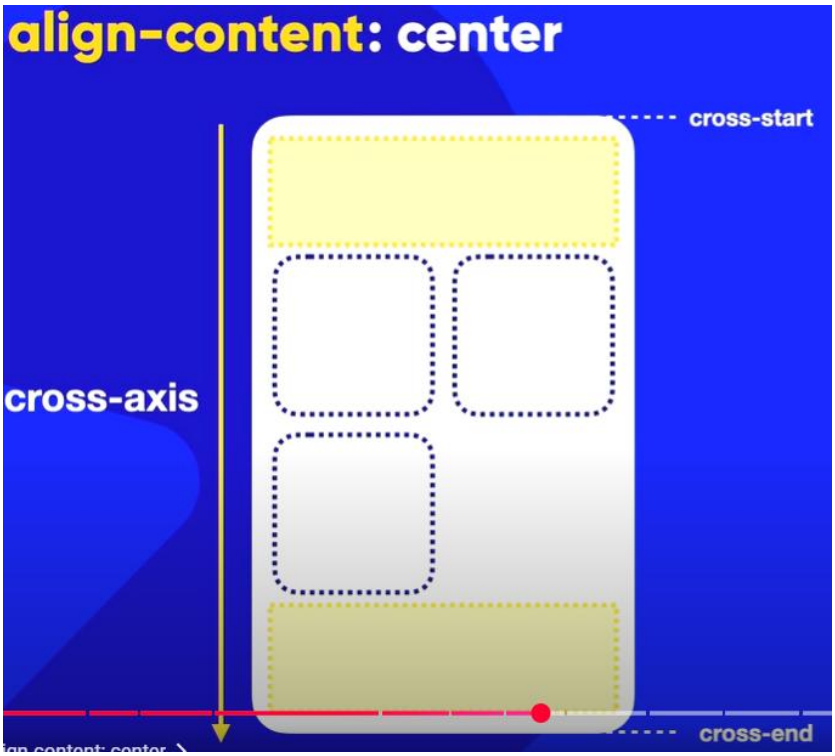


Aling-content: flex-end faz com que os elementos se alinhem no final do eixo transversal (cross-end) empurrando o espaço vazio, para o início do eixo (Cross-start). Exemplo:

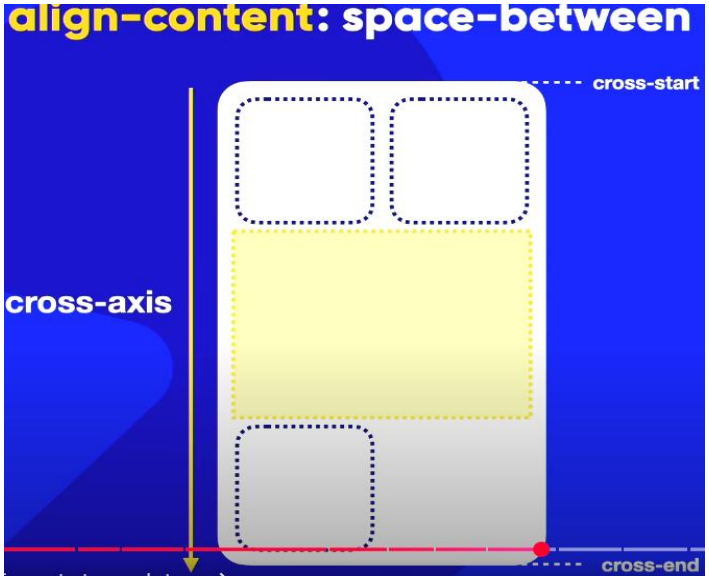


Aling-content:Center Centraliza os elementos no meio do eixo transversal, e colocar espaços iguais no início e no final do eixo (cross-start) e (cross-end).

Exemplo:

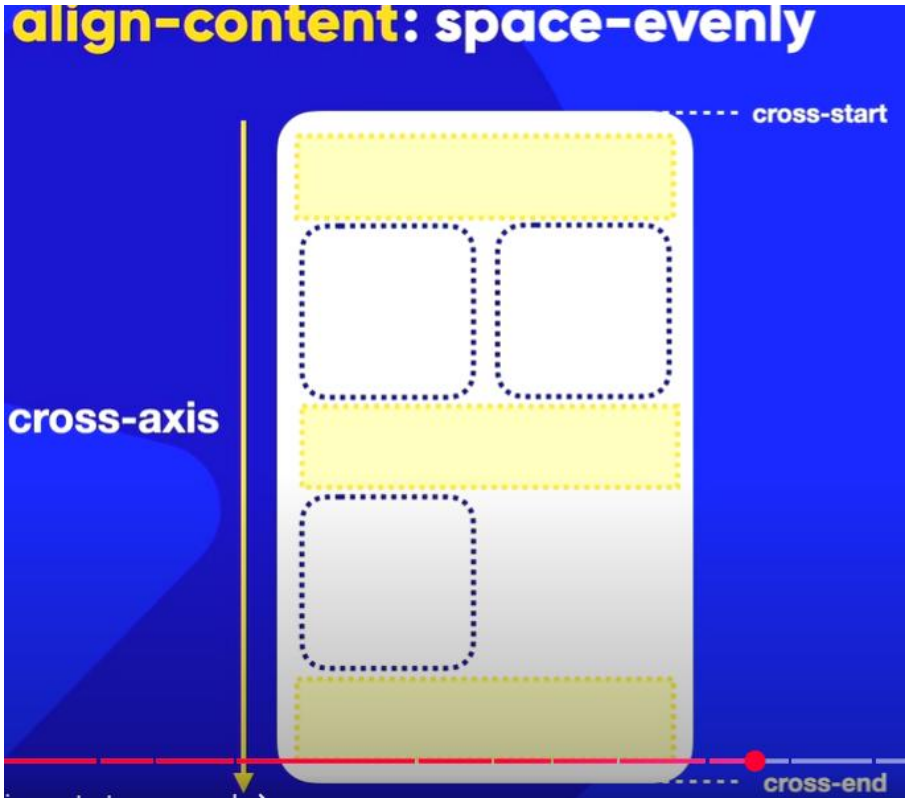


Aling-content: Space-between alinha os primeiros elementos no início do eixo (Cross-start) o último elemento no final do eixo (Cross-end) e centraliza ou o espaço em branco ou os elementos no meio. Exemplo:



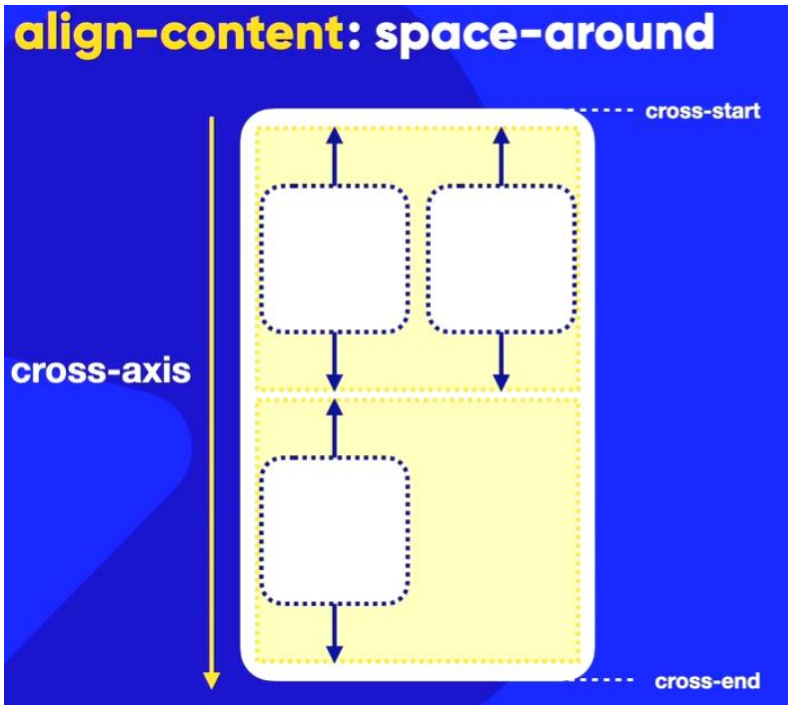
Aling-content: Space-evenly cria um espaçamento igual no início do eixo, no final e entre os elementos.

Exemplo:



Aling-content: Space-around vai dividir o container em áreas iguais e colocar os elementos dentro dessas áreas, centralizando-os. Sendo que o espaço do meio e o dobro do espaço do início e do final do eixo.

Exemplo:



Itens Flexíveis ou Flex-item

São itens que se localiza dentro do container pai. Elas podem ter configurações, que podem alterar suas características e ordem.

Propriedade Order propriedade que te dá a capacidade de configurar a ordem dos elementos que estão dentro do container Pai. Por padrão a propriedade Order vai esta configurada com o valor 0 (zero).

Outras configurações com outros valores vão fazer com que o CSS organize os elementos de forma crescente, do menor ao maior, sendo que os valores negativos vão vir antes.

Para colocar a propriedade, selecione o elemento que você quer e coloque a propriedade order com o valor desejado.

Aling-self configuração que vai usar o eixo transversal (cross-axis) como base.

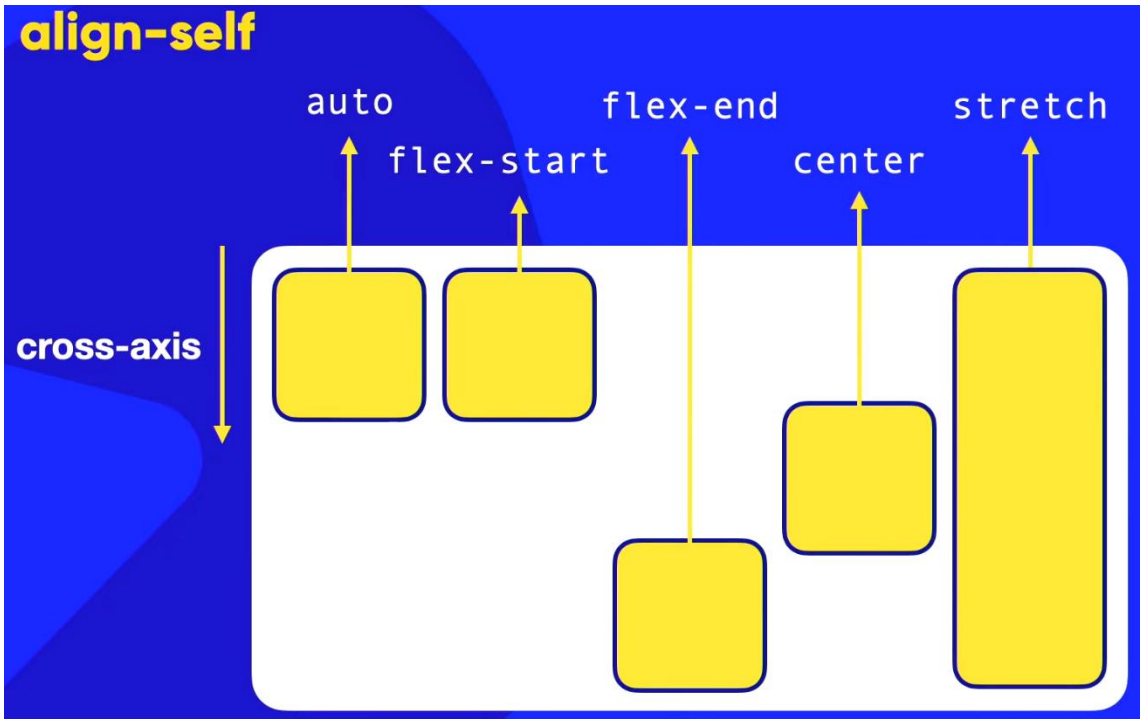
Aling-Self:auto; configura o item a seguir os padrões impostos pelo container Pai. Através do Aling-items.

Aling-Self:flex-start alinha o item no início do Eixo Transversal. (Cross-start).

Aling-Self:flex-end alinha o item no final do Eixo Transversal. (Cross-end).

Aling-Self:center alinha o item no centro do eixo Transversal.

Aling-Self:stretch estica o item dentro do container Pai.



Flex-basis configura o tamanho dos itens dentro do container Pai, se orientando através do eixo principal (main-axis).

Flex-basis: auto faz com que o tamanho do elemento seja dado, pelo tamanho do conteúdo que esteja dentro.

Exemplo:

Caso o contêiner esteja com nowrap (sem opção de levar o elemento para baixo) vai diminuir os outros elementos (a e c) que vão encolher para que caiba todos.

E possível colocar um valor no Flex-basis como por exemplo: 200px isso faz com que os elementos Filhos, tenha seu tamanho padronizado independentemente do tamanho do conteúdo. Porém também vai depender do tamanho do container Pai, pois se por acaso o contêiner for menor. O tamanho dos elementos vai ser reduzido de forma semelhante entre todos. E se por acaso o flex-wrap estiver com o valor nowrap ativado e o tamanho do container for reduzido, ele pode quebrar.

Aumentando e Diminuindo o Tamanho dos Itens Flexíveis

Para aumentar e diminuir Itens Flexíveis no Css coloque as propriedades Flex-shrink e Flex-grow.

O Flex-shrink e responsável por encolher os elementos e o Flex-grow e responsável por aumentar os elementos.

Por padrão o Flex-shrink vai ter o valor 1 e o Flex-grow vai ter o valor 0. O número 0 significa que o valor não vai ser aplicado e acima do valor 0 o valor vai ser aplicado.

Exemplo:

Flex-shrink: 0 não vai encolher. Se o valor for 1 vai encolher.

Flex-grow: 1 vai crescer. Se o valor for 0 não vai crescer.

Vale ressaltar que o valor colocado nessas propriedades, tem proporções, isso é, ao colocar um número em uma propriedade e colocar outro de número em outra propriedade de mesmo nome, vai ter aplicação de forma proporcional.

Exemplo:

flex-grow: 1;

flex-grow: 2; esse vai crescer o dobro do primeiro.

flex-shrink: 1;

flex-shrink: 2; esse vai encolher o dobro.

Porém.

flex-grow: 1;

flex-grow: 4; esse vai crescer 4 vezes mais que o primeiro.

flex-shrink: 1;

flex-shrink: 5; esse vai encolher 5 vezes mais que o primeiro.

As propriedades flex-basis, flex-grow e flex-shrink costumam trabalhar juntos.

Propriedade Flex

Essa propriedade serve para encurtar na hora da escrita.

A propriedade Flex e igual flex-grow + flex-shrink + flex-basis. Quando for colocar os valores, coloque nessa ordem.

Simplificação

A configuração padrão do Flex é Flex: 0 1 auto ou Flex: initial que acaba sendo a mesma coisa.

A configuração Flex: 0 0 auto e o mesmo de Flex: none e serve para tornar os elementos mais rígidos, sem ser flexíveis, diminuído ou aumentando de tamanho.

A configuração Flex: 1 1 auto, pode ser resumido por Flex: auto que serve para tornar os elementos mais flexíveis, podendo diminuir ou almentar de tamanho de forma automática.

Se por acaso for usado apenas um valor no Flex, esse valor será usado para o flex-grow. As outras propriedades vão estar no automático.

Grid Layout

E uma tecnologia do CSS que permite organizar os elementos da tela. Através de grades, que são separações de linhas e colunas.

Display: Grid Informa que a configuração a ser feita, vai utilizar o grid layout.

Container Pai: Vai abrigar os elementos dentro de si. E possível colocar algumas configurações, como grid-template-columns que te dá possibilidade de configurar a quantidade de colunas, que vai ser ter.

Exemplo: grid-template-columns: 200px 200px 200px 200px vai criar 4 colunas com a largura de 200 pixels.

Propriedade Auto

Propriedade auto que cria um tamanho de coluna que se adeque ao tamanho da tela de forma automática.

Lembrando que é possível mesclar o tamanho auto com os valores em pixels.

Grid-template-rows

Essa propriedade faz a mesma coisa do grid-template-columns, porém, ao invés de ser com colunas (vertical) ela faz com linhas (horizontal).

Os valores de auto e pixels também valem para o grid-template-rows.

Valores fr

Os valores fr também servem para dar tamanhos para seus elementos, os fr significa fração, isso e 1fr significa 1 fração da tela. Esse valor e responsivel, dando dinâmica para seus elementos, também é possível, mudar essas frações com os números que vem a frente. Exemplo: 2fr são duas frações da tela e 4fr e quatro frações da tela.

Repeat ()

Essa propriedade permite colocar um valor para que se repita por diversas vezes.

Exemplo: grid-template-columns: repeat (5, 200px) vai criar 5 colunas com 200 pixels cada. Também é possível usar os valores de tamanhos como auto e fr.

A Propriedade repeat pode também ser usada em grid-template-rows. E possível, colocar vários repeat na mesma propriedade. Exemplo: repeat (5, 200px) repeat (2, 1fr).

Minmax

Te dá a possibilidade de dar um tamanho mínimo para seus containers e um tamanho máximo que eles podem alcançar.

Exemplo: minmax (100px, 300px) o tamanho mínimo e de 100 pixels e o máximo e de 300. E possível colocar outras propriedades como repeat, auto e fr.

Gap

E uma propriedade que dá a possibilidade de dar um espaçamento entre as linhas e colunas.

Exemplo: gap: 20px o espaçamento entre as linhas e colunas e de 20 pixels.

Caso queira colocar valores diferente entre as colunas e linhas, as configurações do gap é gap: row (linhas) e columns (colunas).

Valores individuais.

Columns-gap: da o espaçamento entre as colunas de forma individual.

Row-gap: dá o espaçamento entre as linhas de forma individual.

Justify-items

Justify-items alinha os itens de dentro da caixa de forma horizontal. Valores start alinha no início da caixa, center alinha ao centro, end alinha no final e stretch (padrão) que estica para preencher a caixa.

Aling-items

Alinha os itens de forma vertical. E os valores são os mesmos, start, center, end e stretch (padão).

Justify-content

Serve para alinhas as caixas ou container de forma horizontal. E os valores são start, center, end, space-between, space-around e space- Evelyn.

Aling-content

Alinha na vertical e os valores são os mesmos do justify-content.

Atenção as vezes e necessário colocar altura nos seus containers.

Place-items

place-items é um atalho para definir as propriedades align-items e justify-items em uma única declaração

Exemplo: Place-items: center (aling-items - vertical) start (justify-items – horizontal).

Grid-Template-Áreas

Técnica onde você pode criar áreas com nomes e os locais que elas vão ocupar (como por exemplo: duas colunas) e logo em seguida com uma div colocar os nomes dessas áreas.

Exemplo:

grid-template-areas:

'M M' A letra M vai ocupar duas colunas.

'C B' A letra C vai ocupar uma coluna e o B vai ocupar outra.

'O O' A letra O vai ocupar duas colunas.

Em seguida:

Coloque o identificador ou nome da DIV, com a propriedades grid-area: com o nome do elemento que você colocou no grid-template-areas: como por exemplo a letra M.

Grid-auto-rows

Propriedade que coloca os tamanhos das linhas (horizontal).

Exemplo: grid-auto-rows 200px as linhas vão ter altura de 200 pixels.

Caso coloque dois valores, o primeiro valor vai servir para a primeira linha, pula uma linha e configurar o tamanho da linha seguinte, e o segundo valor vai servir para a segunda linha e configura uma linha depois de pular uma linha.

Exemplo: grid-auto-rows: 100px 300px; a primeira e terceira vai ter 100px e a segunda e quarta linha vai ter 300px.

Grid column

Serve para colocar a quantidade de colunas que um elemento filho, que está dentro do container, vai ter e onde vai ocupar.

Exemplo: grid column: 2 mostras que o elemento vai ter duas colunas e vai ocupar a segunda.

Propriedade Span da a possibilidade de fazer um elemento ocupar todas as áreas das colunas solicitadas.

Exemplo: grid column: span 3 vai ocupar três colunas de uma vez.

Grid row

Seve para indicar qual linha o elemento vai ocupar.

Exemplo: grid row: 4 esse elemento vai ocupar a quarta linha.

Justify-self

Alinha o elemento de forma horizontal. Os valores que podem ser colocados, são start, para alinhar no começo, center, para alinhar no meio, end, para alinhar no final e stretch para preencher o espaço todo.

Aling-self

Serve para alinhar os elementos de forma vertical. E os valores são os mesmos do Justify-self. Start, center, end e stretch.