**Meu caderno de anotações para a formação front-end da Alura**

**HTML:**

Toda estrutura HTML segue uma padronização para seu funcionamento, com o body dividido em três elementos principais:

Header - Cabeçalho

Main – Corpo e conteúdo

Footer – Rodapé

Section – Uma tag semântica utilizada para definir uma seção de conteúdo, muito utilizada para agrupar conteúdos e tornar o conteúdo do website semântico.

**CSS**

Reset-CSS:

O reset CSS é um recurso importante a ser utilizado para limpar as configurações padrão de CSS que já são importadas junto ao navegador, para utilizar o reset CSS pode-se usar a documentação padrão, ou um reset definido manualmente, utilizando o seletor \*

Altura: 100vh – Utilizamos a View High para demonstrar ao navegador que aquele elemento tem de ocupar toda a altura da minha página, não deixando nada fora da sua altura (o mesmo serve para o vw (width))

Como a altura dos dispositivos é diferente, e o conteúdo do site pode ser mostrado em diferentes dispositivos, a vh é importante de ser utilizada para padronizar a altura de leitura em n dispositivos.

Box-Sizing: Border-Box

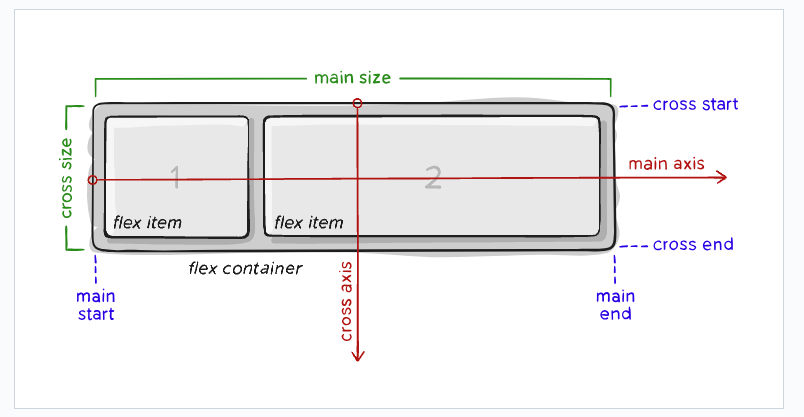
Propriedade padrão do front-end para definir que uma caixa/conteúdo não saia do seu pai, com essa propriedade o conteúdo sempre ficará dentro das suas respectivas caixas pai (evitando problemas de saída de elementos)

Todo elemento raiz tem 16px como padrão do navegador de tamanho de fonte

Flexbox:

É a tecnologia utilizada para alinhar as caixas e posicioná-las conforme o dev deseja, assim, utilizamos no elemento pai dos elementos que queremos ajustar na página utilizando o parâmetro display:flex para dizer a máquina que utilizaremos flexbox.

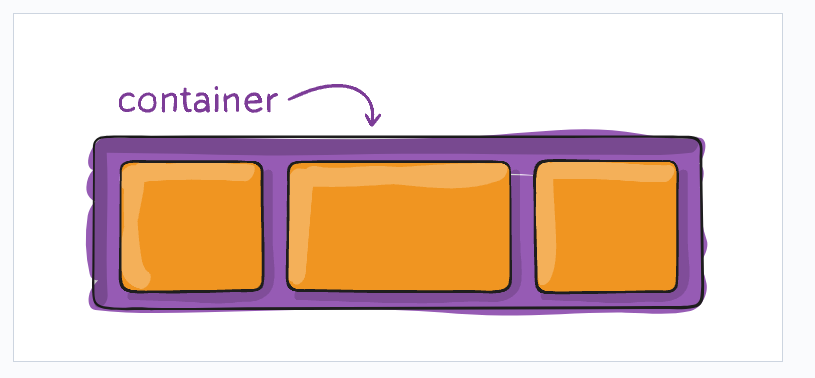
Se o leiaute "padrão" é baseado nas direções block e inline, o leiaute Flex é baseado em direções "flex flow". Veja abaixo um diagrama da especificação, explicando a ideia central por trás do leiaute Flex.



**Flexbox: Seletores e parâmetros para container**

Display: Flex

Esta propriedade define um flex container; inline ou block dependendo dos valores passados. Coloca todos os elementos-filhos diretos num contexto Flex. Sempre será utilizada no seletor do elemento pai, para definir o flexbox dos elementos filhos.



Flex-Direction:

Estabelece o eixo principal, definindo assim a direção em que os flex items são alinhados no flex container. O padrão é em linha, ou seja na horizontal da esquerda para a direita, mas é possível transformar em linha-reverso, coluna, ou coluna-reversa

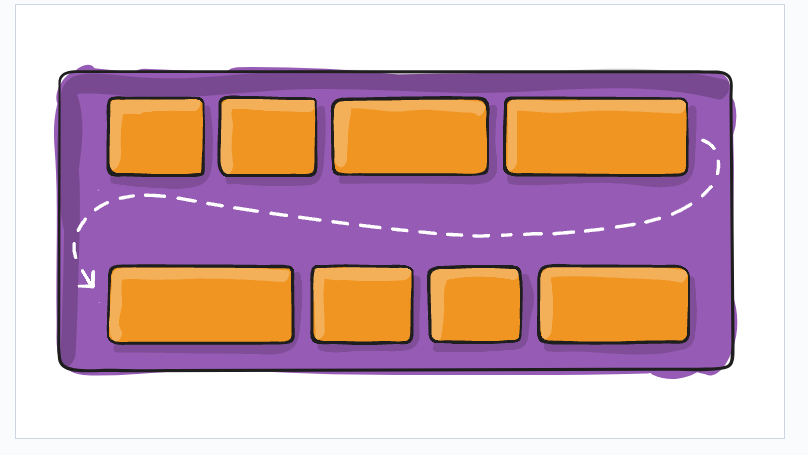
.flex-container {

flex-direction: row | row-reverse | column | column-reverse;

}

Flex-wrap:

Por padrão, os flex items vão todos tentar se encaixar em uma só linha. Com esta propriedade você pode modificar esse comportamento e permitir que os ítens quebrem para uma linha seguinte conforme for necessário.



.flex-container {

flex-wrap: nowrap | wrap | wrap-reverse;

}

É possível juntar as propriedades do flex-direction e do flex-wrap em uma só propriedade, o flex-flow

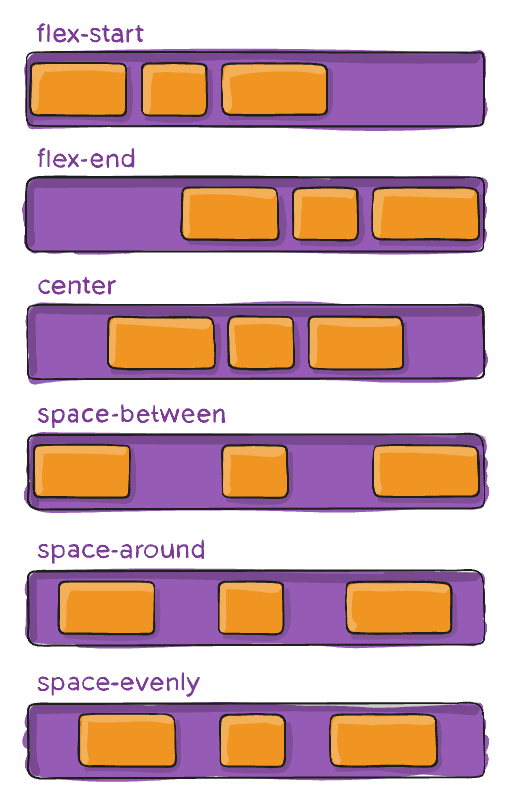
Assim:

.flex-container {

flex-flow: row nowrap | row wrap | column nowrap | column wrap;

}

Justify-content:



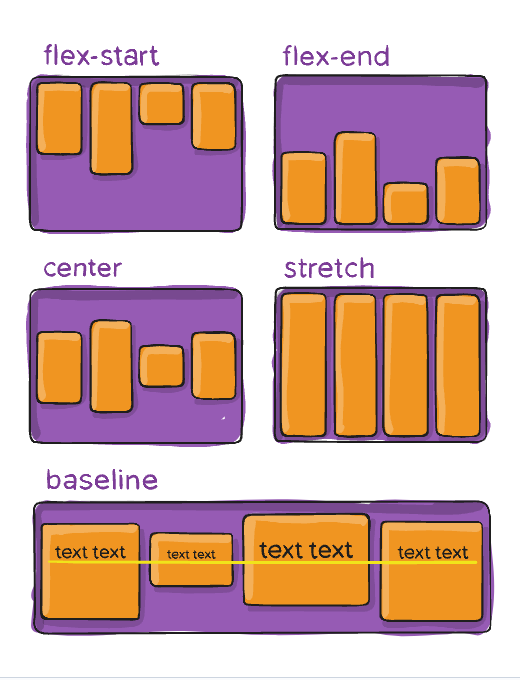
.flex-container {

justify-content: flex-start | flex-end | center | space-between | space-around | space-evenly;

}

* flex-start (padrão): os ítens são alinhados junto à borda de início (start) de acordo com qual for a flex-direction do container.
* flex-end: os ítens são alinhados junto à borda final (end) de acordo com qual for a flex-direction do container.
* start: os ítens são alinhados junto à borda de início da direção do writing-mode (modo de escrita).
* end: os ítens são alinhados junto à borda final da direção do writing-mode (modo de escrita).
* left: os ítens são alinhados junto à borda esquerda do container, a não ser que isso não faça sentido com o flex-direction que estiver sendo utilizado. Nesse caso, se comporta como start.
* right: os ítens são alinhados junto à borda direita do container, a não ser que isso não faça sentido com o flex-direction que estiver sendo utilizado. Nesse caso, se comporta como start.
* center: os ítens são centralizados na linha.
* space-between: os ítens são distribuídos de forma igual ao longo da linha; o primeiro ítem junto à borda inicial da linha, o último junto à borda final da linha.
* space-around: os ítens são distribuídos na linha com o mesmo espaçamento entre eles. Note que, visualmente, o espaço pode não ser igual, uma vez que todos os ítens tem a mesma quantidade de espaço dos dois lados: o primeiro item vai ter somente uma unidade de espaço junto à borda do container, mas duas unidades de espaço entre ele e o próximo ítem, pois o próximo ítem também tem seu próprio espaçamento que está sendo aplicado.
* space-evenly: os ítens são distribuídos de forma que o espaçamento entre quaisquer dois itens da linha (incluindo entre os ítens e as bordas) seja igual.

Align-items:



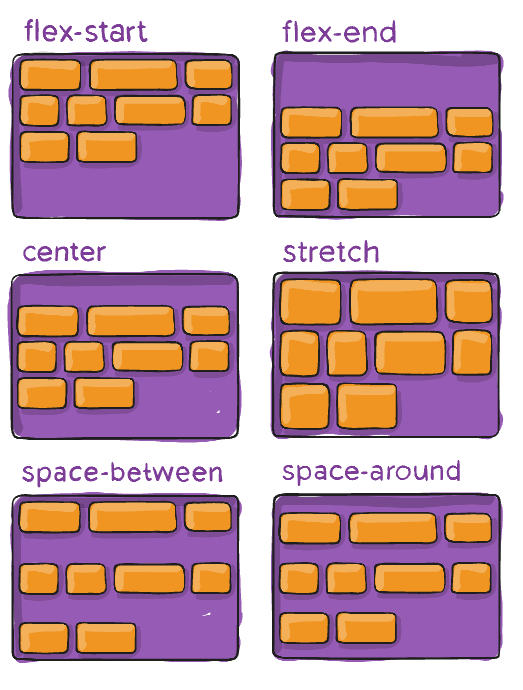
define o comportamento padrão de como flex items são alinhados de acordo com o eixo transversal (cross axis). De certa forma, funciona de forma similar ao justify-content, porém no eixo transversal (perpendicular ao eixo principal).

.flex-container {

align-items: stretch | flex-start | flex-end | center | baseline;

}

Align-content:



Organiza as linhas dentro de um flex container quando há espaço extra no eixo transversal, similar ao modo como justify-content alinha ítens individuais dentro do eixo principal.

**Importante:** Esta propriedade não tem efeito quando há somente uma linha de flex items no container.

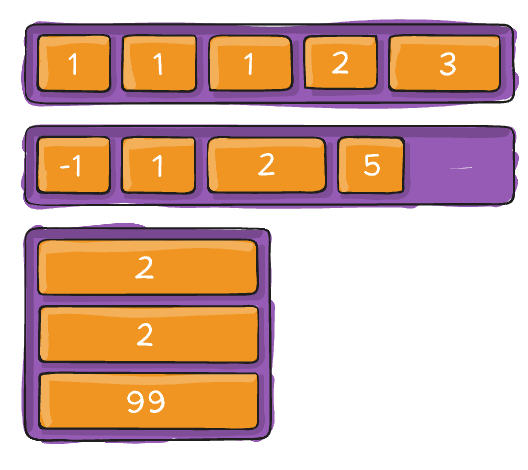
.flex-container {

align-content: flex-start | flex-end | center | space-between | space-around | stretch;

}

* flex-start / start: ítens alinhados com o início do container. O valor (com maior suporte dos navegadores) flex-start se guia pela flex-direction, enquanto start se guia pela direção do writing-mode.
* flex-end / end: ítens alinhados com o final do container. O valor (com maior suporte dos navegadores) flex-end se guia pela flex-direction, enquanto end se guia pela direção do writing-mode.
* center: ítens centralizados no container.
* space-between: ítens distribuídos igualmente; a primeira linha junto ao início do container e a última linha junto ao final do container.
* space-around: ítens distribuídos igualmente com o mesmo espaçamento entre cada linha.
* space-evenly: ítens distribuídos igualmente com o mesmo espaçamento entre eles.
* stretch (padrão): ítens em cada linha esticam para ocupar o espaço remanescente entre elas.

**Propriedades para elementos filhos:**



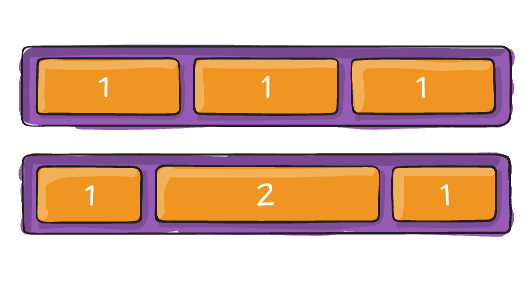
Determina a ordem em que os elementos aparecerão.

.flex-item {

order: <número>; */\* o valor padrão é 0 \*/*

}

Flex-grow



Define a habilidade de um flex item de crescer, caso necessário. O valor dessa propriedade é um valor numérico sem indicação de unidade, que serve para cálculo de proporção. Este valor dita a quantidade de espaço disponível no container que será ocupado pelo item.

Se todos os ítens tiverem flex-grow definido em 1, o espaço remanescente no container será distribuído de forma igual entre todos. Se um dos ítens tem o valor de 2, vai ocupar o dobro de espaço no container com relação aos outros (ou pelo menos vai tentar fazer isso).

.flex-item {

flex-grow: <numero>; */\* o valor default(padrão) é 0 \*/*

}

**flex-shrink**

Define a habilidade de um flex item de encolher, caso necessário.

.flex-item {

flex-shrink: <número>; */\* o valor padrão é 0 \*/*

}

Flex-basis:

Define o tamanho padrão para um elemento antes que o espaço remanescente do container seja distribuído. Pode ser um comprimento (por exemplo, 20%, 5rem, etc) ou uma palavra-chave. A palavra-chave auto significa "observe minhas propriedades de altura ou largura" (o que era feito pela palavra-chave main-size, que foi depreciada). A palavra-chave content significa "estabeleça o tamanho com base no conteúdo interno do ítem" - essa palavra-chave ainda não tem muito suporte, então não é fácil de ser testada, assim como suas relacionadas: max-content, min-content e fit-content.

.flex-item {

flex-basis: flex-basis: | auto; */\* o valor padrão é auto \*/*

}

Com o valor de 0, o espaço extra ao redor do conteúdo não é considerado. Com o valor de auto, o espaço extra é distribuído com base no valor de flex-grox do ítem.

### ****flex****

Esta é a propriedade shorthand para flex-grow, flex-shrink e flex-basis, combinadas. O segundo e terceiro parâmetros (flex-shrink e flex-basis) são opcionais. O padrão é 0 1 auto, mas se você definir com apenas um número, é equivalente a 0 1.

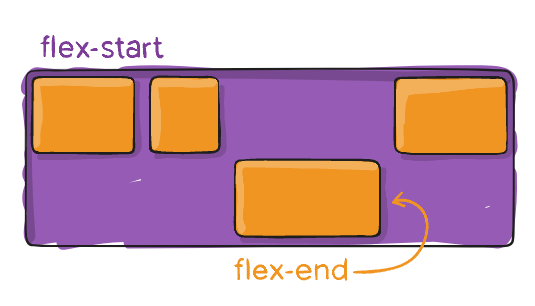
.item {

flex: none | [ <'flex-grow'> <'flex-shrink'>? || <'flex-basis'> ]

}

**É recomendado que você utilize esta propriedade shorthand** ao invés de definir cada uma das propriedades em separado. O shorthand define os outros valores de forma inteligente.

Align-self



Permite que o alinhamento padrão (ou o que estiver definido por align-items) seja sobrescrito para ítens individuais.

Por favor veja a explicação da propriedade align-items para entender quais são os possíveis valores.

.item {

align-self: auto | flex-start | flex-end | center | baseline | stretch;

}

Atalho para fazer comentário selecionando no CSS: CTRL + K + C

Propriedade importante para o flexbox: GAP

**Documentações**

<https://css-tricks.com/snippets/css/a-guide-to-flexbox/>

<https://developer.mozilla.org/pt-BR/docs/Learn/CSS/CSS_layout/Flexbox>

<https://www.w3schools.com/css/css3_box-sizing.asp>

<https://www.figma.com/file/fLQYKVNuB4kqnvO1mnNMSp/Portfolio---Curso-2?type=design&node-id=1-11&mode=design&t=bijK3dOs4aXyuTI0-0>

[https://www.alura.com.br/artigos/guia-de-unidades-no-css?\_gl=1\*ztrqi0\*\_ga\*NjY2OTY4MjIyLjE2OTQzMDUwNjI.\*\_ga\_1EPWSW3PCS\*MTcwNzgzMzQ4NC4xNC4xLjE3MDc4MzYzNTIuMC4wLjA.\*\_fplc\*eUs1MEVDY0Q1NjRIN2d2TGdxdyUyQndKUllhbnRhTW53R0swJTJGWHZjRzZ2WTdZdVVEbUtVSjRyTHdXMWVxbkU0OWVEWVpsU0FqMUlzbWdnRlUyTnJJdEhZem5tMVFzeGhVcVR5NThtcXZqeVcyY2haYWZSNTVXYmlwMzE5Z0IlMkJRJTNEJTNE](https://www.alura.com.br/artigos/guia-de-unidades-no-css?_gl=1*ztrqi0*_ga*NjY2OTY4MjIyLjE2OTQzMDUwNjI.*_ga_1EPWSW3PCS*MTcwNzgzMzQ4NC4xNC4xLjE3MDc4MzYzNTIuMC4wLjA.*_fplc*eUs1MEVDY0Q1NjRIN2d2TGdxdyUyQndKUllhbnRhTW53R0swJTJGWHZjRzZ2WTdZdVVEbUtVSjRyTHdXMWVxbkU0OWVEWVpsU0FqMUlzbWdnRlUyTnJJdEhZem5tMVFzeGhVcVR5NThtcXZqeVcyY2haYWZSNTVXYmlwMzE5Z0IlMkJRJTNEJTNE)

<https://cursos.alura.com.br/course/html-css-classes-posicionamento-flexbox/task/147896>

[https://www.alura.com.br/artigos/css-border-estilizando-bordas-elementos-css?\_gl=1\*g2vsy8\*\_ga\*NjY2OTY4MjIyLjE2OTQzMDUwNjI.\*\_ga\_1EPWSW3PCS\*MTcwNzg3MTc1OS4xNi4xLjE3MDc4NzI5NDkuMC4wLjA.\*\_fplc\*eUs1MEVDY0Q1NjRIN2d2TGdxdyUyQndKUllhbnRhTW53R0swJTJGWHZjRzZ2WTdZdVVEbUtVSjRyTHdXMWVxbkU0OWVEWVpsU0FqMUlzbWdnRlUyTnJJdEhZem5tMVFzeGhVcVR5NThtcXZqeVcyY2haYWZSNTVXYmlwMzE5Z0IlMkJRJTNEJTNE](https://www.alura.com.br/artigos/css-border-estilizando-bordas-elementos-css?_gl=1*g2vsy8*_ga*NjY2OTY4MjIyLjE2OTQzMDUwNjI.*_ga_1EPWSW3PCS*MTcwNzg3MTc1OS4xNi4xLjE3MDc4NzI5NDkuMC4wLjA.*_fplc*eUs1MEVDY0Q1NjRIN2d2TGdxdyUyQndKUllhbnRhTW53R0swJTJGWHZjRzZ2WTdZdVVEbUtVSjRyTHdXMWVxbkU0OWVEWVpsU0FqMUlzbWdnRlUyTnJJdEhZem5tMVFzeGhVcVR5NThtcXZqeVcyY2haYWZSNTVXYmlwMzE5Z0IlMkJRJTNEJTNE)

<https://css-tricks.com/almanac/properties/g/gap/>

[https://www.alura.com.br/artigos/css-guia-do-flexbox?\_gl=1\*zdaofb\*\_ga\*NjY2OTY4MjIyLjE2OTQzMDUwNjI.\*\_ga\_1EPWSW3PCS\*MTcwNzg3MTc1OS4xNi4xLjE3MDc4NzM0OTYuMC4wLjA.\*\_fplc\*eUs1MEVDY0Q1NjRIN2d2TGdxdyUyQndKUllhbnRhTW53R0swJTJGWHZjRzZ2WTdZdVVEbUtVSjRyTHdXMWVxbkU0OWVEWVpsU0FqMUlzbWdnRlUyTnJJdEhZem5tMVFzeGhVcVR5NThtcXZqeVcyY2haYWZSNTVXYmlwMzE5Z0IlMkJRJTNEJTNE](https://www.alura.com.br/artigos/css-guia-do-flexbox?_gl=1*zdaofb*_ga*NjY2OTY4MjIyLjE2OTQzMDUwNjI.*_ga_1EPWSW3PCS*MTcwNzg3MTc1OS4xNi4xLjE3MDc4NzM0OTYuMC4wLjA.*_fplc*eUs1MEVDY0Q1NjRIN2d2TGdxdyUyQndKUllhbnRhTW53R0swJTJGWHZjRzZ2WTdZdVVEbUtVSjRyTHdXMWVxbkU0OWVEWVpsU0FqMUlzbWdnRlUyTnJJdEhZem5tMVFzeGhVcVR5NThtcXZqeVcyY2haYWZSNTVXYmlwMzE5Z0IlMkJRJTNEJTNE)

https://cursos.alura.com.br/course/html-css-classes-posicionamento-flexbox/task/147896