**SENAI- SERVIÇO NACIONAL DE APRENDIZAGEM INDUSTRIAL**

**GABRIEL SANCHES**

**PAOLA FERNANDA**

**THIAGO PEREIRA**

# ANIMAÇÕES CSS E BOX LAYOUT FLEXÍVEL

**CURITIBA**

**2017**

**GABRIEL SANCHES**

**PAOLA FERNANDA**

**THIAGO PEREIRA**

# ANIMAÇÕES CSS E BOX LAYOUT FLEXÍVEL

Trabalho apresentado à unidade curricular de Programação de Aplicativos do Curso Técnico em Informática, Senai Boqueirão como requisito para obtenção de nota.

Orientador: Prof°: Cezar Jenzura
Coorientador: Prof°: Robson Barbosa

**CURITIBA**

**2017**

**SUMÁRIO**

## LISTA DE FIGURAS

## 1. INTRODUÇÃO

Este trabalho visa avaliar, ensinar e praticar os conhecimentos a respeito da unidade curricular Programação de Aplicativos, que tem como maior finalidade ensinar e aprimorar os conhecimentos em relação aos conteúdos apresentados neste trabalho. Os assuntos abordados envolvem recursos como o CSS e HTML.

## 2. ANIMAÇÕES CSS

O objetivo das animações é criar funções que permitam ao autor executar uma mudança de valor de uma propriedade CSS, de maneira que a mudança de valor ocorra de uma maneira suave, em um espaço de tempo especificado. A diferença entre animações e transições é que as funcionalidades das animações oferecem mecanismos adicionais, que são capazes de exercer um controle mais cuidadoso sobre os valores intermediários da animação.

As funcionalidades das animações é criar uma estrutura que permita definir mudanças de apresentações de forma suave em um determinado tempo.

O principal componente utilizado para as animações é o *keyframes*, responsável pelas regras de animação. Uma animação é uma continuidade de *frames*, onde cada um deles irá representar o estado de animação. Ou seja, o componente *keyframes* foi desenvolvido para determinar os valores da propriedade CSS do elemento animado em determinado tempo de animação.

Mesmo que seja possível criar uma animação usando todos os *frames*, a finalidade principal não é essa, mas expandir as funcionalidades de transições. Isto é, a maior finalidade é usar *frames* quantidade suficiente para atingir efeitos que não são possíveis com o uso de transições.

A construção CSS para a criação e inclusão de um *frame*, com regras de estilização em uma animação, é simples.

Exemplo:

```
@keyframes nome{
    0% {width: 1px; }
    50% {height: 400px; }
    100% {width:900px; }
}
```

A regra CSS, quando mostrada sozinha, não tem efeito. Ela deve ser associada com outras regras CSS que irão definir os valores para as propriedades de animação.

A combinação mostrada, indicado a criar *frames*, deve estar similar com algumas diretrizes básicas.

- A regra CSS deve começar sendo declarada pela orientação *@keyframes,* seguida por uma string que será escolhida pelo autor e que

indicará o nome da animação. Como qualquer regra CSS , irá começar e terminar com sinais de abrir ({) e fechar (}) chaves.

- O ponto de inclusão do *frame* na animação deve ser determinado com o uso de porcentagem (%). A porcentagem é o que indica a quantidade de animação já decorrido. A cada porcentagem declarada, são definidos uma ou mais declarações CSS que irão estilizar propriedades daquele *frame*.
- Se houver somente uma declaração de *frame* em uma animação, não será válida. Para que se possa declarar um ou mais *frames*, deve-se declarar os *frames* 0% e 100%.
- O sinal de porcentagem (%) deve ser incluso obrigatoriamente quando o valor for 0.
- Para que se possa usar uma sintaxe alternativa nos *frames* iniciais e finais, pode-se usar a palavra-chave *from*, no lugar do 0% e a palvra-chave *to*, no lugar do 100%

A regra *@keyframes* ou outras propriedades de animação não são suportadas pelos navegadores Opera 11 e Internet Explorer anteriores ao IE10. Para se utilizar essas propriedades no navegadores Firefox, Safari e Chrome é necessário utilizar os prefixos *–moz-* e *–webkit-*.

## 3. Propriedades

As propriedades utilizadas para animações são:

- *Animation-name*
- *Animation-duration*
- *Animation-timing-function*
- *Animation-iteration-count*
- *Animation-direction*
- *Animation-play-state*
- *Animation-delay*
- *Animation*

### 3.1. Animation-name

A propriedade *Animation-name* visa definir o nome ou uma lista de animações, separados por uma vírgula. Os nomes serão escolhidos livremente pelo

autor para ser inclusa na regra CSS @*keyword.* O valor inicial que será utilizado é o *none* e essa propriedade não é herdável e se utiliza a todos os elementos nível de bloco em inline.

A construção para essa propriedade pode-se ser observada nos exemplos:

```
@keyframes alongar{
    0% {width:100px; }
    50% {heigth:200px; }
    100% {width:900px; }
}
@keyframes aumentar{
    0% {font-size:14px; }
    25% {font-size :18px; }
    50% {font-size:22px; }
    75% {font-size:26px; }
    100% {font-size:30px; }
}
Seletor {animation-name: alongar;}
Seletor {animation-name: aumentar;}
Seletor {animation-name: alongar, aumentar;}
```

Exemplo para que possa usar a sintaxe alternative com palavras chaves:

```
@keyframes alongar{
    from {width:100px; }
    50% {heigth:200px; }
    to {width:900px; }
}
@keyframes aumentar{
    from {font-size:14px; }
    25% {font-size :18px; }
    50% {font-size:22px; }
    75% {font-size:26px; }
    to {font-size:30px; }
}
```

Exemplo para que se possa aplicar as regras CSS nos navegadores:

```
@-moz- keyframes alongar{
    from {width:100px; }
    50% {heigth:200px; }
    to {width:900px; }
}
 @-moz- keyframes aumentar{
    from {font-size:14px; }
    25% {font-size :18px; }
    50% {font-size:22px; }
    75% {font-size:26px; }
    to {font-size:30px; }
}
Seletor {-moz-animation-name: alongar;}
Seletor {-moz-animation-name: aumentar;}
Seletor {-moz-animation-name: alongar, aumentar;}
```

### 3.2. Animation-duration

A finalidade desta propriedade é definir o tempo para a animação completar um ciclo. O valor da propriedade é o segundos, o que irá emitir a duração de um ciclo completo da animação. O valor inicial para a duração dos ciclos é zero, e números negativos serão considerados zero.

A sintaxe para essa propriedade é:

```
Seletor {
    Animation-name: pular;
    Animation-duration: 8s;
}
Seletor {
    Animation-name: saltar, quicar;
    Animation-duration: 12s, 10s;
}
Seletor {
    Animation-name: subir, aumentar, cair;
    Animation-duration: 10s, 5s;
}
```

Para a aplicação em navegadores é:

```
Seletor {
    Animation-name: pular;
    -moz-animation-duration: 8s;
}
Seletor {
    Animation-name: saltar, quicar;
    -moz-animation-duration: 12s, 10s;
}
Seletor {
    Animation-name: subir, aumentar, cair;
    -moz-animation-duration: 10s, 5s;
}
```

### 3.3. Animation-timing-function

Está propriedade irá definir a função que descreve como serão calculados os valores medianos de um ciclo de animação. Os valores possíveis para essa propriedade são as palavras-chaves: *ease, linear, ease-in, ease-out* a função *cubic-bezier* ().

O valor inicial utilizado nesta propriedade é a *ease.*

A construção para essa propriedade é:

```
Seletor {animation-timing-function: ease;}
Seletor { animation-timing-function: linear;}
Seletor { animation-timing-function: ease-in;}
Seletor { animation-timing-function: ease-out;}
Seletor { animation-timing-function: ease-in-out;}
Seletor { animation-timing-function: cubic-bezier;}
```

E a aplicação para navegadores é:

```
Seletor {-moz-animation-timing-function: ease;}
Seletor {-moz-animation-timing-function: linear;}
Seletor {-moz- animation-timing-function: ease-in;}
Seletor {-moz- animation-timing-function: ease-out;}
Seletor {-moz- animation-timing-function: ease-in-out;}
```

*Seletor {-moz- animation-timing-function: cubic-bezier;}*

### 3.4. Animation-iteration-count

A propriedade *animation-iteration-count* define a quantidade de ciclos da animação. O valor inicial é 1, o valor igual a infinito irá fazer com que a animação nunca pare. Números negativos são considerados 0, e números decimais são considerados válidos.

A sintaxe para essa propriedade é:

*Seletor { animation-iteration-count: 4;}*

*Seletor { animation-iteration-count: 6,8;}*

*Seletor { animation-iteration-count: infinite;}*

E o prefixo utilizado para a aplicação em navegadores é:

*Seletor {-moz-animation-iteration-count: 4;}*

*Seletor {-moz-animation-iteration-count: 6,8;}*

*Seletor {-moz-animation-iteration-count: infinite;}*

### 3.5. Animation-direction

Esta animação irá definir a ocorrência da animação da animação em ordem inversa, ou seja, quando há a intercalação da animação, conforme o ciclo ímpar ou par. Nos ciclos ímpar a animação será normal, e ciclos par a animação é na ordem inversa. Essa propriedade só irá funcionar quando o valor declarado for maior ou igual a dois. É possível usar os valores *normal* e *alternate*.

Para que essa propriedade:

*Seletor { animation-direction: normal;}*

*Seletor { animation-direction: alternate;}*

E para que essa propriedade possa ser usada em navegadores:

*Seletor {-moz-animation-direction: normal;}*

*Seletor {-moz-animation-direction: alternate;}*

### 3.6. Animation-play-state

Essa propriedade irá definir se a animação está parada ou em andamento. Os valores utilizados para essa propriedade são *running* e *paused.*

O valor inicial é *running.* Quando a animação é colocada em *paused* ela irá parar no estado em que se encontra.

A construção para essa propriedade é:

*Seletor { animation-play-state: running;}*

*Seletor { animation-play-state: paused;}*

A sintaxe usada em navegadores:

*Seletor { -moz-animation-play-state: running;}*

*Seletor { -moz-animation-play-state: paused;}*

### 3.7. Animation-delay

A *animation-delay* tem como propriedade definir um tempo de espera para o início da animação que será contado quando uma ação iniciar a animação. O valor desta propriedade é número de segundos que irá expressar o tempo de espera.

O valor inicial é o zero, e números negativos são aceitos.

A sintaxe usada nesta propriedade é:

*Seletor { animation-delay: 4s;}*

*Seletor { animation-delay: -4s;}*

E a sintaxe utilizada para esta propriedade em navegadores é:

*Seletor { -moz-animation-delay: 4s;}*

*Seletor { -moz-animation-delay: -4s;}*

### 3.8. Animation

Está propriedade é o formato abreviado de declarar todas as propriedades para a animação. Os valores devem ser declarados na ordem, separados por espaço em branco.

E a sintaxe usada é:

*Seletor { animation: rodar 4s ease-out 1s 8 alternate;}*

*Seletor { transition: subir 8s linear 1s infinite normal;}*

*Seletor { transition: virar 12s ease-in-out;}*

*Seletor { transition: colorir 6s ease-in infinite, aumentar 14s ease 2s 4 alternate;}*

## 4. Box Layout flexível

Em outubro de 2011, os navegadores não suportavam as funcionalidades do Box Layout flexível. Porém, os navegadores Firefox, Safari, Chrome e IE10 possuem suporte com o uso de prefixos proprietários *-moz-, -webkit-* e *-ms-.*

Os prefixos foram criados antes de 2010. Em 22 de março de 2011, 22 de março de 2012 e 12 de junho de 2012, foram lançados novas versões do Rascunho de Trabalho, alterando assim a sintaxe de propriedades e valores. E acrescentando também novas funcionalidades.

O W3C descreve a finalidade de Box Layout Flexível da seguinte forma:

> Essa especificação descreve um Box Model CSS otimizado para desenho de interfaces. Nesse modelo de layout de boxes flexíveis os boxes filhos de um container flexível podem ser flexíveis em ambas as dimensões, dispostos na vertical ou na horizontal e podem preencher eventuais espaços vagos no container, seja por expansão ou contração. Para criar um layout, é possível manipular facilmente o alinhamento dos boxes bem como aninhar boxes horizontais em boxes verticais e vice-versa. (Silva, Maurício, 2011,pág.390).

O Box Layout flexível tem algumas características, por exemplo:

- Podem ser ordenados em qualquer posição.
- Podem ser organizados em linhas ao longo de um eixo principal ou em múltiplas linhas em um eixo secundário.
- É capaz de ter sua ordem de colocação invertida ou reordenada sem alterações na ordem em que aparecem em HTML
- Pode ser flexível em várias direções, ou seja, são adaptáveis aos espaços dispostos
- Podem ser alinhados relativamente a outros boxes e contêiner.
- É capaz de ser dinamicamente ligados ou separados ao longo do eixo principal sem alterar o eixo secundário

O Box Layout flexível tem seus cálculos baseados em flow-directions que irão determinar as direções e as suas dimensões.

## 5. Propriedades

### 5.1. Display

Está propriedade visa definir o contexto para Box Layout flexível. Os valores possíveis são: *flex* e *inline-flex.* Sabe-se que o valor inicial desta propriedade depende de um elemento, por exemplo: elementos como *div* têm a sua propriedade com um valor inicial igual a *block*, enquanto que elementos como *span* tem como propriedade o valor igual a *inline.*

Os valores desta propriedade, quando aplicados a algum elemento, definem como um container flexbox, ou seja, todos os elementos herdeiros do contaneir comportam-se de maneira que seguem o modelo flex layout. Para que os elementos sigam o mesmo modelo em uma página web, esses valores devem ser declarados para a propriedade *display* do elemento *body.*

### 5.2. Flex-direction

Essa propriedade é destinada a definir o sentido que os elementos-filhos serão ordenados no elemento conteiner. Os valores possíveis são: *row, column, row-reverse* e *column-reverse.* O valor inicial é *row*

| Efeito | Descrição |
|---|---|
| *Row* | Os boxes são dispostos em linha, segundo o *main axis* (na horizontal). |
| *Column* | Os boxes são dispostos como boxes nível de bloco, segundo o *cross axis* (na vertical). |
| *Row-reverse* | Os boxes em linha são dispostos na ordem inversa do código. |
| *Column-reverse* | Os boxes nível de bloco são dispostos na ordem inversa do código. |

(Silva, Maurício,2011,pág.396)

### 5.3. Flex-direction:row

Está propriedade irá definir a direção horizontal para a organização dos boxes, segundo o *main axis* e da esquerda para a direita. Essa é a direção padrão para a organização de flexboxes.

### 5.4. Flex-direction:column

Irá definir a direção vertical para a ordenação dos boxes, segundo o *cross axis* e de cima para baixo.

### 5.5. Flex-direction:row-reverse

Essa propriedade define a direção horizontal e o sentido contrário para a disposição dos boxes, segundo *main axis* e da direita para a esquerda.

### 5.6. Flex-direction:column-reverse

Define a direção vertical e o sentido oposto para organização dos boxes, segundo *cross axis* e de baixo para cima

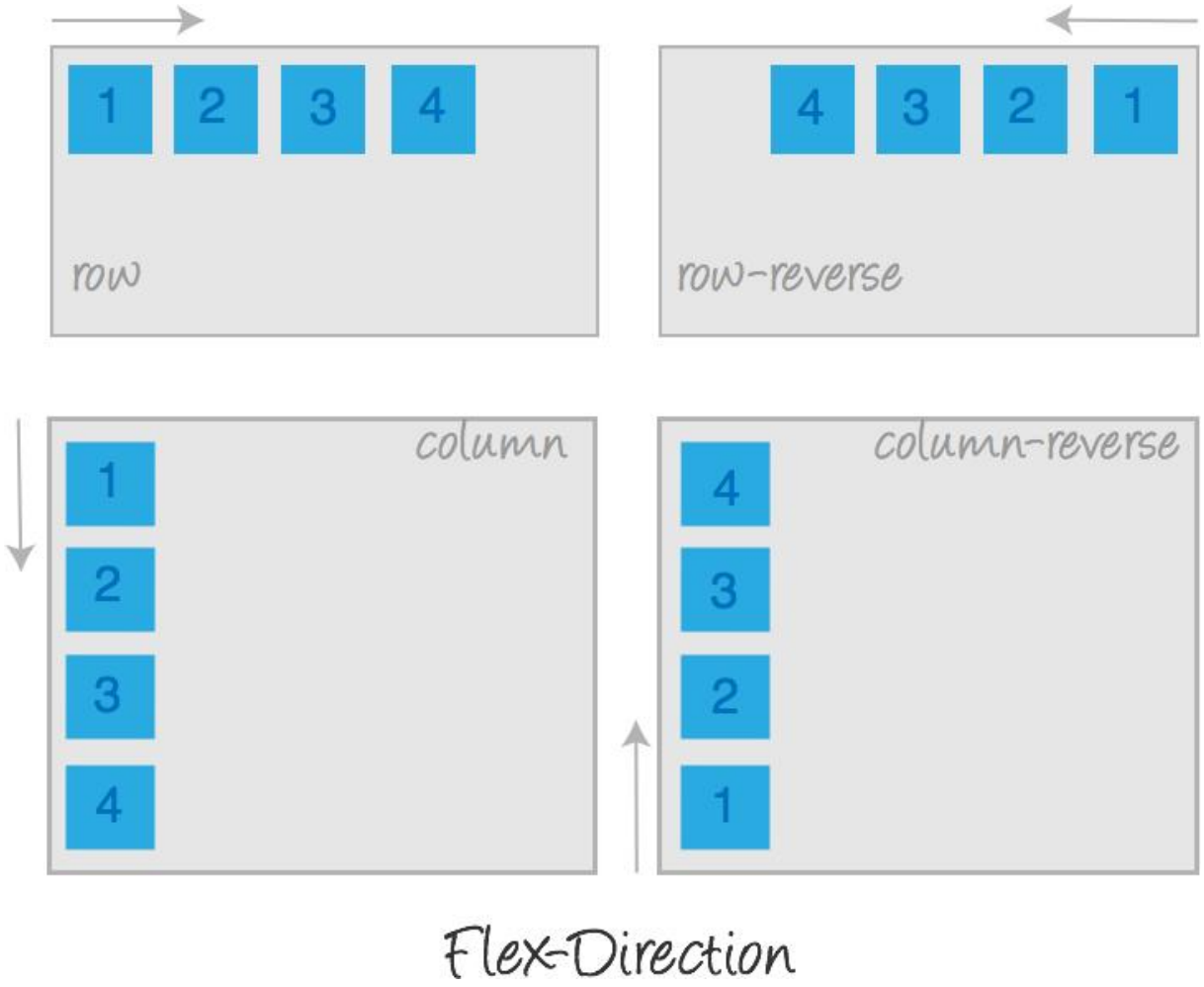


Figura 1-Flex-direction

Fonte: https://tympanus.net/codrops/css_reference/flexbox/

### 5.7. Flex-wrap

Está propriedade é destinada a definição de como será a ordenação de linhas no flex container. Os valores atribuídos para essa propriedade são: *nowrap, wrap* e *wrap-reverse.* O valor inicial é o *nowrap.*

### 5.8. Flex-Flow

É Uma propriedade de declaração única, usa-se junto as propriedades de flex-direction, e flex-wrap, que juntas definem os eixos de um flex container.

O valor inicial de flex-flow é a concatenação dos valores iniciais de *flex-direction* e *flex-wrap*

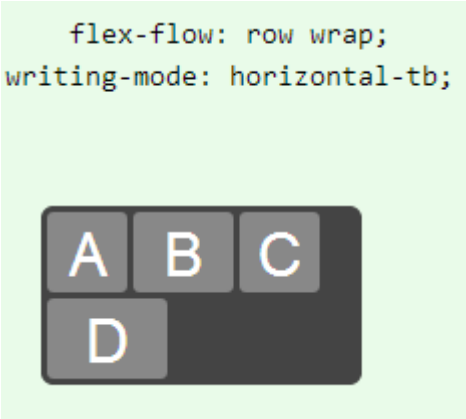


Figura 2- Flex-flow

Fonte http://desenvolvimentoparaweb.com/css/flexbox/#section_align-content

Todo conteúdo dentro do container esta para a horizontal.

### 5.9. Order

A propriedade order é usada para alterar a ordem. Flex items são, por padrão, mostrados e leiautados na mesma ordem em que aparecem no código-fonte do documento.

Ela controla a ordem em que flex itens aparecem num flex container quando é colocado em grupos ordinais, aonde se atribui um valor, exemplo:<integer>, aonde define a qual grupo ordinal o flex pertence. Itens com o mesmo grupos são lidos na ordem que aparecem na camada do conteúdo. O valor inicial de Order é 0, e pode ser definido um valor negativo.

### 5.10. Flex

É  a declaração unica das propiedades Flex-grow, Flex-shrink e Flex-basis.

Quando um elemento é um item flex, flex é usado para determinar o Main Size do elemento. Se um elemento não é um flex item, flex não apresenta quaisquer efeitos.

O valor inicial é 0 1 auto. As propriedades flex-grow e flex-shrink são opcionais e podem ser omitidas da declaração de flex.

Quando o valor de flex-grow é omitido, é como se este fosse 1. Perceba que este 1 não é o valor inicial da propriedade flex-grow! É somente o valor usado quando flex-grow é omitido da propriedade de declaração única.

Quando a propriedade flex-shrink é omitido, é como se esta tivesse o valor de 1. Lembre-se de que o fator de flex-shrink é multiplicado pelo flex-basis quando distribui espaço negativo — veja flex-shrink.( http://desenvolvimentoparaweb.com/css/flexbox/#section_flex  20/05/2017)

### 5.11. Aling-Self

Flex itens podem ser alinhados no Cross Axis da atual linha do flex container, similarmente a justify-content, mas na direção perpendicular. align-items especifica o alinhamento padrão de todos os flex itens do container; a propriedade align-self permite sobrescrever esse alinhamento padrão para itens individuais

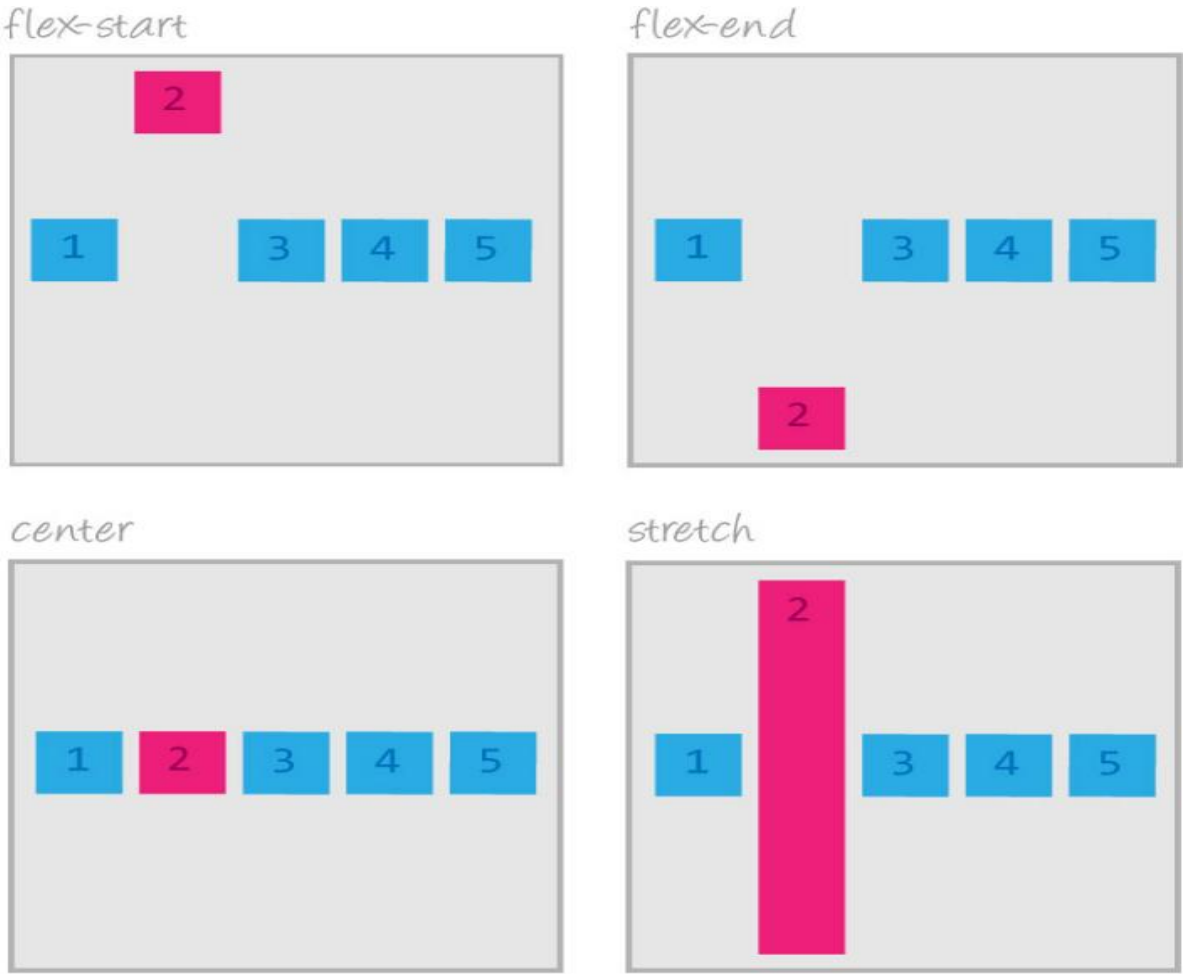


Figura 3- Flex

Fonte http://desenvolvimentoparaweb.com/css/flexbox/

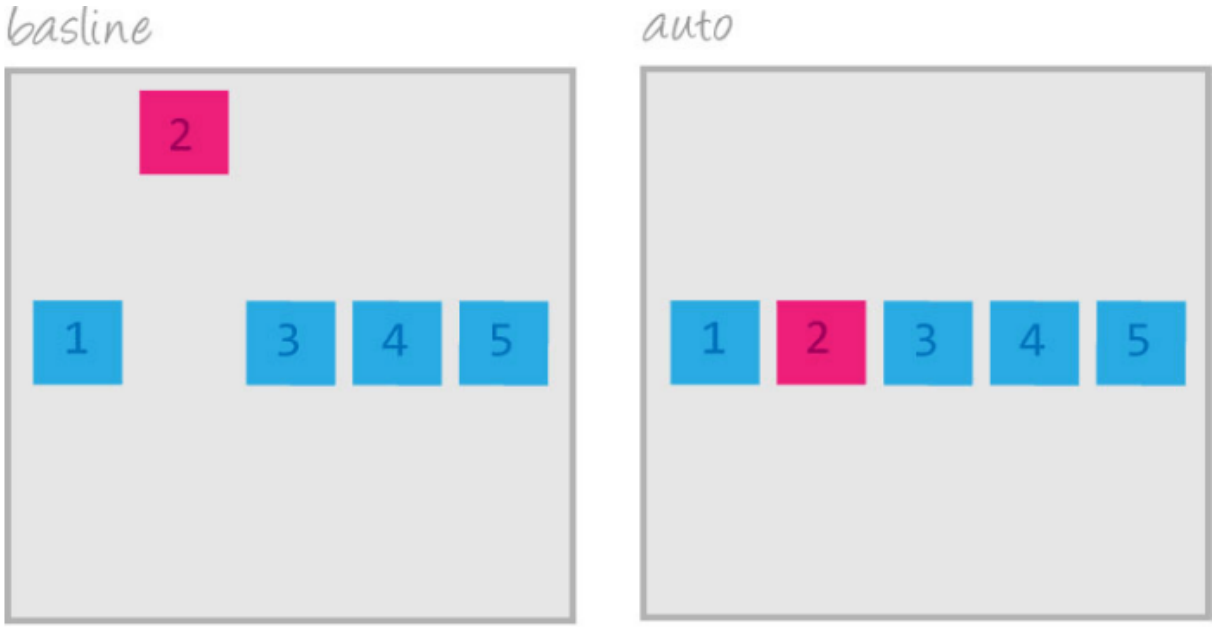


Figura 4- Justify-content

Fonte http://desenvolvimentoparaweb.com/css/flexbox/

## 5.12. Justify-content

Está propriedade alinha os Flex itens através do *Main Axis* da linha atual do Flex container. Isso é feito depois que todas as medidas flexíveis e margens automáticas já tiverem sido definidas. Também controla o  alinhamento de itens quando acontece o overflow.(“estouram as linhas ou o box container”).

Valores Possíveis:

flex-start. Valor inicial. Flex items ficam no começo da linha

flex-end. Flex items ficam no fim da linha

center. Flex items ficam no centro da linha, com igual quantidade de espaço antes e depois (se o espaço de sobra for negativo, os flex items “estouram” em ambas as direções)

space-between. Flex items são igualmente distribuídos na linha (se o espaço de sobra for negativo ou houver somente um flex item na linha, o comportamento é idêntico a flex-start)

space-around. Flex items são igualmente distribuídos na linha com meio-espaços em cada extremidade (se o espaço de sobra for negativo ou houver somente um flex item na linha, o comportamento é idêntico a center http://desenvolvimentoparaweb.com/css/flexbox/#section_flex  20/05/2017)

## 5.13. Aling-Content

Alinha quando  há espaços sobrando no *Cross Axis*, quase a mesma função que o Justify-content faz com itens individuais no *Main Axis*.

Valores possíveis:

flex-start. Linhas são dispostas através de Cross Start do flex container

flex-end. Linhas são dispostas através de Cross End do flex container

center. Linhas são dispostas através do centro do flex container com igual quantidade de espaço entre a extremidade de Cross Start do container e sua primeira e a extremidade de Cross End e sua última linha (se o espaço de sobra for negativo, as linhas “estouram” em ambas as direções)

space-between. Linhas são distribuídas uniformemente no flex container (se o espaço de sobra é negativo, o valor é idêntico a flex-start)

space-around. Linhas são distribuídas uniformemente no flex container com meio-espaços em cada extremidade

stretch. Valor inicial. Linhas esticam para ocupar o espaço restante (se o espaço de sobra é negativo, o valor é idêntico a flex-start, do contrário, o espaço livre é dividido igualmente entre todas as linhas, aumentando o tamanho de seu Cross Size. http://desenvolvimentoparaweb.com/css/flexbox/#section_align-content 20/05/2017)


Figura 5-

Fonte http://desenvolvimentoparaweb.com/css/flexbox/

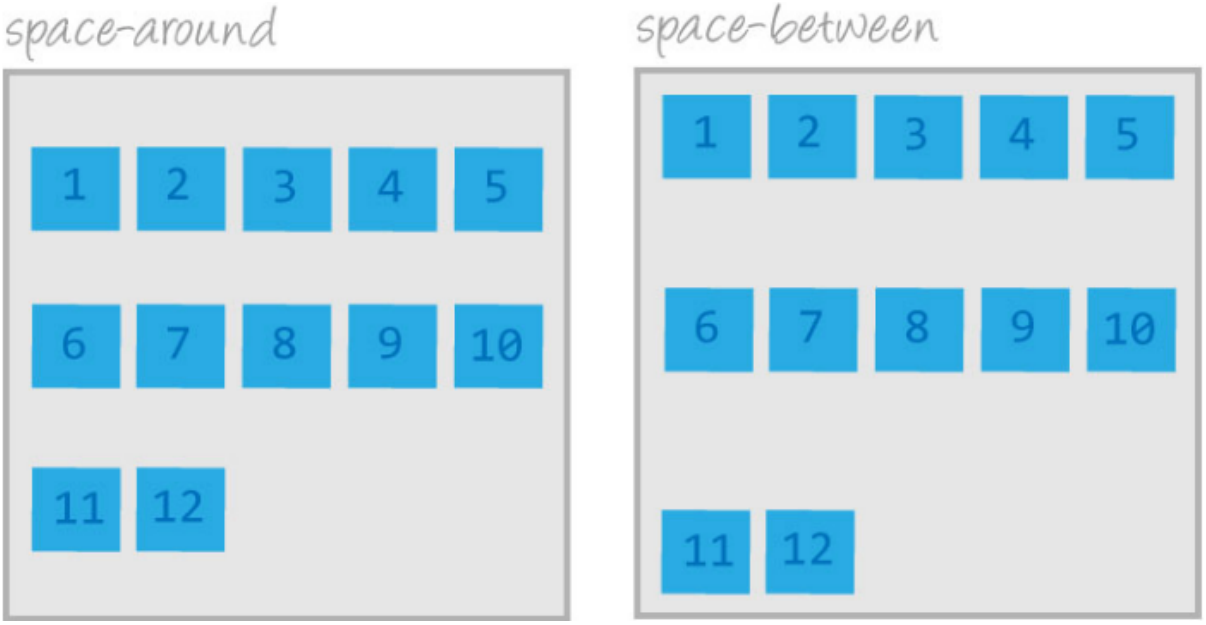


Figura 6-

Fonte http://desenvolvimentoparaweb.com/css/flexbox/

### 5.14. Align-Itens

Em Flexbox, a propriedade align-items é similar a justify-content, mas, ao invés de alinhar os itens através do Main Axis, o faz no Cross Axis.

Valores possíveis:

flex-start. Flex items são alinhados através do Cross Start da linha

flex-end. Flex items são alinhados através do Cross End da linha

center. Flex items são alinhados através do centro da linha com igual quantidade de espaços entre a extremidade de Cross Start e o primeiro item da linha e a extremidade de Cross End e o último item da linha (se o espaço de sobra for negativo, os flex items “estouram” em ambas as direções)

baseline. Flex items são alinhados tal como o alinhamento de sua linha de base (baseline)

stretch. Valor inicial. Flex items são esticados através do Cross Start até Cross End, ainda respeitando o que foi definido para min-height/min-width/max-height/max-width(se a altura do flex container é restringida, seu valor pode fazer com que o conteúdo dos flex items “estoure”

http://desenvolvimentoparaweb.com/css/flexbox/#section_align-content 20/05/2017)

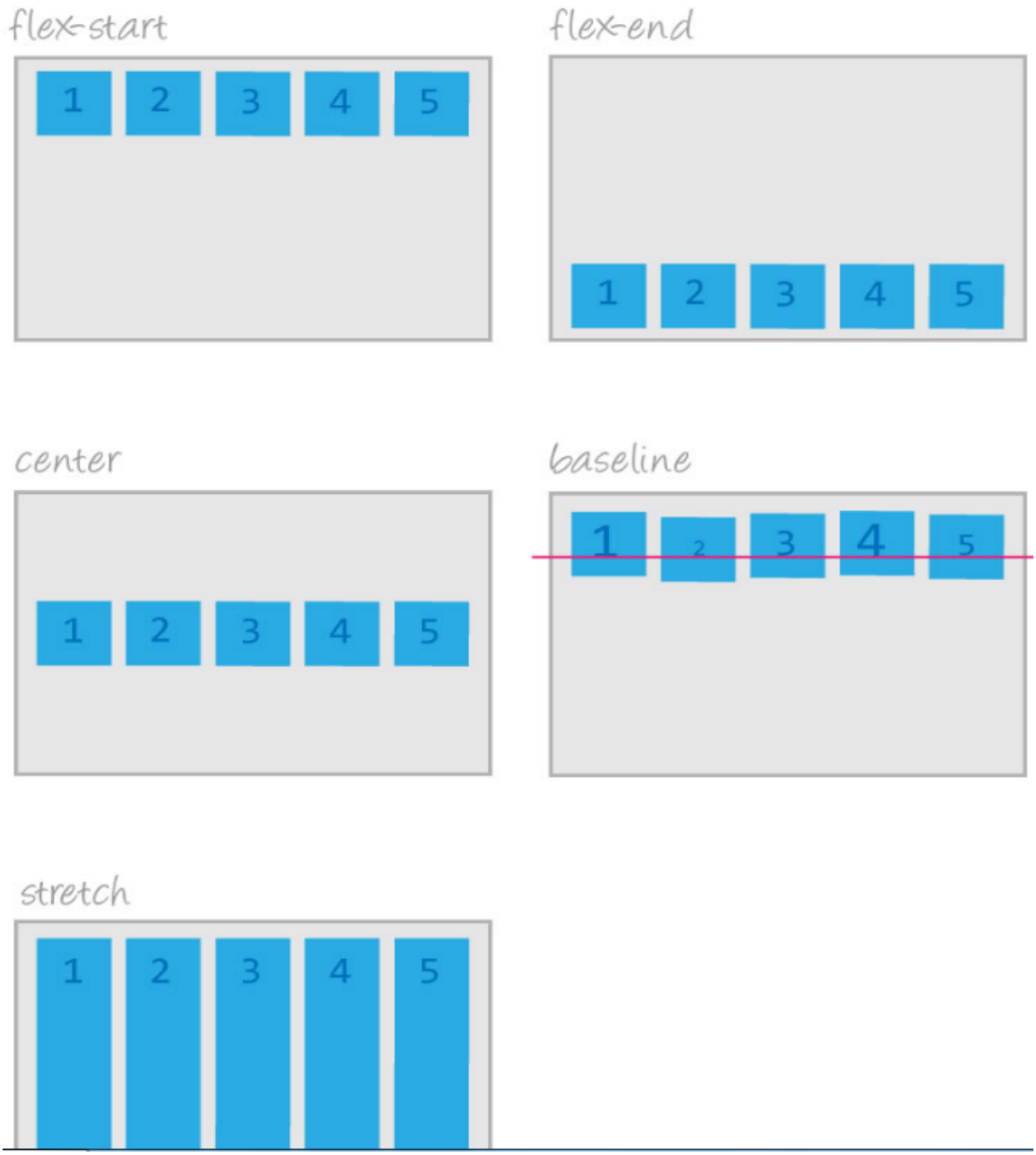


Figura 7- Align-Itens

Fonte: http://desenvolvimentoparaweb.com/css/flexbox/

## 6. Conclusão

Tendo em vista os aspectos apresentados pode-se entender que todos os elementos e propriedades citadas no corpo do texto, são importantes para que se tenha códigos e páginas de qualidade. Muitas propriedades dependem uma da outra, para que realmente códigos tenham uma real funcionalidade.

## 7. Referências

http://desenvolvimentoparaweb.com/css/flexbox/

Livro: CSS3, Maurício Samy Silva