Layout

Position

Por padrão todos os elementos são **static**

**Relative:** Dentro do relative temos 5 propriedades

* Left
* Right
* Top
* Bottom
* Z-index

Todos recebem valores para terem um espaçamento

**Absolute:** Libera as mesmas propriedades do relative, mas em absolute são eliminados os espaços e os objetos estilizados são elevados a uma nova camada.

Absolutes são absolutos ao pai

**Fixed:** Ele cria um contexto, ele fixa o elemento a página, então mesmo que haja um scroll da página ele permanecera no mesmo lugar, mas seguindo o scroll

**Sticky**: Um hibrido entre relative e fixed, faz com que os elementos possam ser editados em uma posição na página, mas ao scrollar os elementos ficam fixos na parte superior da tela

Flexbox

Permite posicionar os elementos dentro de um caixa

Alinhamento, direcionamento, ordenar e tamanhos

**Flex-direction:** Qual a direção do flex horizontal e vertical

Row ou column

**Justify-content:** Manipula o espaço entre os elementos

Space-between

Center

**Align-items:** Alinha os elementos

Center

Grid

Pode ser flexível ou fixo

Cria espaços para os elementos filhos habitarem

**Grid-template-areas**: Cria um template das posições do layout

Grid-template-area:

“header header”

“main aside”

“footer footer”

Após isto coloque dentro dos elementos um dos nomes dentro do template

Grid-area: header

**Fr**: Significa flexível, elementos com este tamanho se daptaram ao tamanho da página

Grid-template-columns ou rows: recebe valores de comprimento para delimitar o tamanho das linhas e colunas

Float

**Float** é uma propriedade CSS em que podemos mover elementos com valores relacionada a direção desejada, exemplo:

Float: left;

**Clear** é uma propriedade relacionada ao float que limpa espaços flutuantes ao redor, mudando assim a posição dos elementos. Os valores aceitos pelo clear são: left, right e both, lembrando que isto não é usado no elemento com float e sim nos elemento ao redor dele.

\*Podem ocorrer certos problemas com a utilização de floats e wrapper’s, para isto a 2 formas de resolver a com “overflow: auto” (que é uma gambiarra) que é suportado por todos os navegadores e a de usar um “display: flow-root” no elemento wrapper, mas não é suportada por todos os navegadores

Multiple-columns layout

Neste tipo de layout podemos criar páginas com colunas. As propriedades básicas são column-count e column-witdh, uma não precisa da outra para funcionar.

**Column-count**: recebe valores referente a quantidade de colunas que você queira e por padrão todas tem valores flexíveis

**Column-width**: recebe valores para configurar o tamanho de todas as colunas

**Column-gap:** recebe valores para configurar o tamanho entre as colunas

**Column-rules:** recebe estilizações para serem usadas nos espaços entre as colunas

Exemplo: column-rules: 4px dotted blue;

A outras propriedades relacionadas a “column-rules” como a column-rules-color, -style e -width

**Column-span:** serve para indicar qual o raio de ocupação de um elemento referente as colunas, um dos Keywords é none (padrão) e all

Algumas vezes os conteúdos de uma coluna podem vazar para outra, ocasionando problemas. Uma forma de resolver, isto é, usando a propriedade “break-inside” que fragmenta as colunas, deixando-as e seus conteúdos com proporções mais coesas

A propriedade “break-inside” aceita somente estes valores auto [|](https://developer.mozilla.org/en-US/docs/Web/CSS/Value_definition_syntax#single_bar) avoid [|](https://developer.mozilla.org/en-US/docs/Web/CSS/Value_definition_syntax#single_bar) avoid-page [|](https://developer.mozilla.org/en-US/docs/Web/CSS/Value_definition_syntax#single_bar) avoid-column