Programação Web

Professor: Euclides Paim

euclidespaim@gmail.com

Programação Web

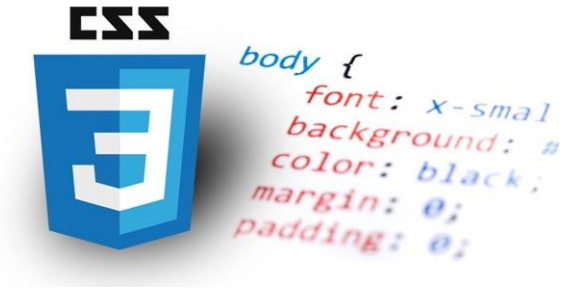
Professor: Euclides Paim

euclidespaim@gmail.com

Desenvolvimento Web I

Sumário

- **Fundamentos de CSS**
  - Posicionamento*;*
  - Transformações 2D;
  - Transformações 3D;
  - Transições.
- **Resumo**



Desenvolvimento Web I

Desenvolvimento Web

CSS Position

- **A propriedade position**
  - A propriedade **`position`** especifica o tipo de método de posicionamento usado para um elemento. Existem cinco valores de posição diferentes:
    - `static`
    - `relative`
    - `fixed`
    - `absolute`
    - `Sticky`

Os elementos são posicionados usando as propriedades *top*, *bottom*, *left*, e *right*. No entanto, essas propriedades **não funcionarão** a menos que a propriedade *position* seja configurada primeiro. Eles também funcionam de forma diferente dependendo do valor da posição.

Desenvolvimento Web

CSS Position

- **`position: static;`**
  - Os elementos HTML são posicionados estáticos por padrão.
  - Os elementos posicionados estáticos não são afetados pelas propriedades superior, inferior, esquerda e direita.
  - Um elemento com *postion: static;* não está posicionado de forma especial; está sempre posicionado de acordo com o fluxo normal da página:

```
.elemento{
  position: static;
  border: 3px solid #73AD21;
}
```

Desenvolvimento Web

CSS Position

- **`position: relative;`**
  - Um elemento com *position: relative;* está posicionado em relação à sua posição normal.
  - Definir as propriedades *top*, *right*, *bottom* e *left* de um elemento relativamente posicionado fará com que ele seja ajustado para longe de sua posição normal.
  - Outro conteúdo não será ajustado para caber em qualquer lacuna deixada pelo elemento.

```
.relative {
  position: relative;
  left: 30px;
  border: 3px solid #73AD21;
}
```

Desenvolvimento Web

CSS Position

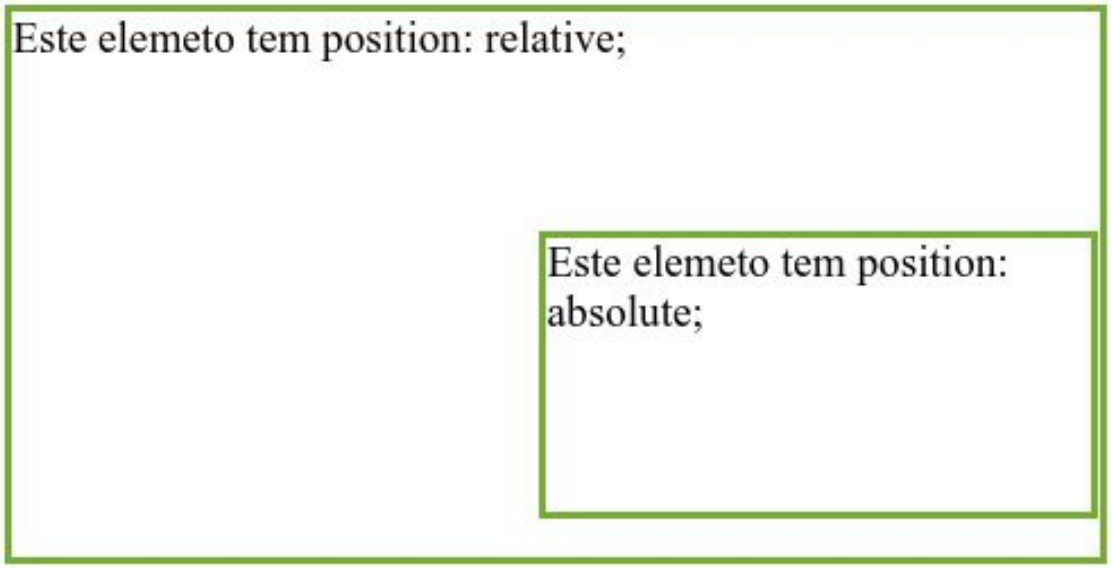
- **`position: fixed;`**
  - Um elemento com *position: fixed;* está posicionado em relação à janela de visualização, o que significa que sempre permanece no mesmo lugar, mesmo se a página for rolada.
  - As propriedades *top*, *right*, *bottom* e *left* serão usadas para posicionar o elemento.
  - Um elemento fixo não deixa uma lacuna na página onde normalmente estaria localizado. Aqui está o CSS usado:

```css
.fixed {
  position: fixed;
  bottom: 0;
  right: 0;
  width: 300px;
  border: 3px solid #73AD21;
}
```

Desenvolvimento Web

CSS Position

- **`position: absolute;`**
  - Um elemento com *position: absolute;* é posicionado em relação ao ancestral posicionado mais próximo (em vez de posicionado em relação à janela de visualização, como *fixed*).
  - Contudo se um elemento posicionado de forma absoluta não tiver ancestrais posicionados, ele usará o corpo do documento e se moverá junto com a rolagem da página.



Nota: Um elemento "posicionado" é aquele cuja posição é qualquer uma, exceto estática.

Desenvolvimento Web

CSS Position

- **`position: sticky;`**
  - Um elemento com *position: sticky;* é posicionado com base na posição de rolagem do usuário.
  - Um elemento "aderente" alterna entre "relativo" e "fixo", dependendo da posição de rolagem.
  - Ele é posicionado em relação até que uma determinada posição de deslocamento seja encontrada na *viewport* - então, ele "se fixa" no lugar (como a posição: fixa).
  - Neste exemplo, o elemento aderente adere ao topo da página (*top: 0*), quando atingimos a posição de rolagem.

```
.sticky {
  position: -webkit-sticky; /* Safari */
  position: sticky;
  top: 0;
  background-color: green;
  border: 2px solid #4CAF50;
}
```

Nota: o Internet Explorer não oferece suporte para posicionamento *sticky*. O Safari requer um prefixo *-webkit-* (veja exemplo). Devemos especificar *top, right, bottom* ou *left* para que o posicionamento fixo funcione.

Desenvolvimento Web

CSS Position

- **Elementos sobrepostos**
  - Quando os elementos são posicionados, eles podem se sobrepor a outros elementos.
  - A propriedade **`z-index`** especifica a ordem da pilha de um elemento (qual elemento deve ser colocado na frente ou atrás dos outros).
  - Um elemento pode ter uma ordem de empilhamento positiva ou negativa:

```
img {
  position: absolute;
  left: 0px;
  top: 0px;
  z-index: -1;
}
```

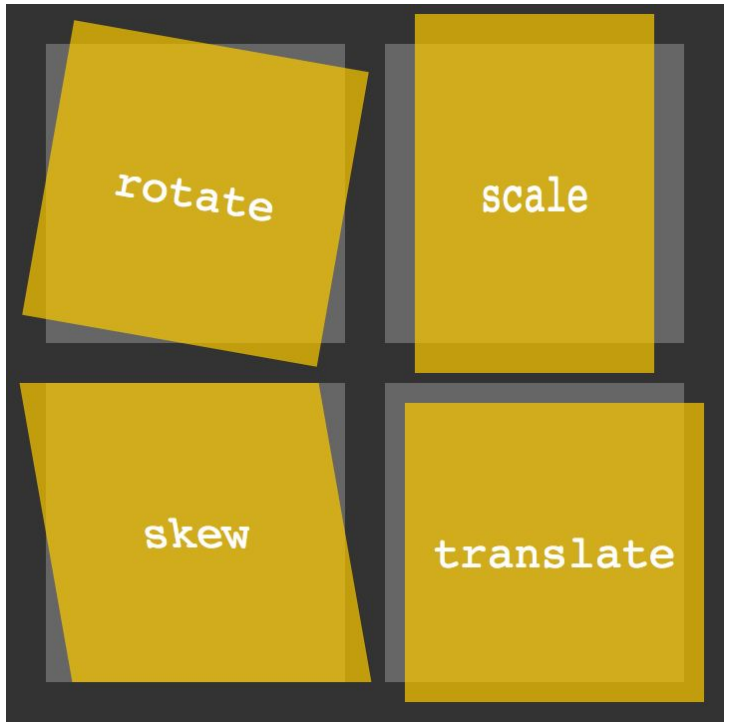
  - Um elemento com uma ordem de pilha maior estará sempre na frente de um elemento com uma ordem de pilha inferior.

Nota: Se dois elementos posicionados se sobrepõem sem um índice z especificado, o elemento posicionado por último no código HTML será mostrado na parte superior.

Leitura complementar: https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Positioning/Understanding_z_index/Adding_z-index

Desenvolvimento Web

Transformações CSS 2D



Desenvolvimento Web

Transformações CSS

- **Transformações CSS 2D**
  - As transformações CSS permitem mover, girar, dimensionar e inclinar elementos. A seguir vamos estudar a propridade:
    - `transform`

- **Métodos de transformação**
  - Com a propriedade `transform`, podemos usar os seguintes métodos de transformação 2D:
    - `translate()`
    - `rotate()`
    - `scaleX()`
    - `scaleY()`
    - `scale()`
    - `skewX()`
    - `skewY()`
    - `skew()`
    - `matrix()`

- **O método `translate()`**
  - O método `translate()` move um elemento de sua posição atual de acordo com os parâmetros fornecidos para os eixos X e Y.
  - O exemplo a seguir move o elemento 50 pixels para a direita e 100 pixels para baixo de sua posição atual:

```
div {
  transform: translate(50px, 100px);
}
```

- **O método `rotate()`**
  - O método `rotate()` gira um elemento no sentido horário ou anti-horário de acordo com um determinado grau.
  - Usar valores negativos girará o elemento no sentido anti-horário.
  - O exemplo a seguir gira o elemento no sentido horário em 20 graus:

```
div {
  transform: rotate(20deg);
}
```

- **O método `scale()`**
  - O método `scale()` aumenta ou diminui o tamanho de um elemento de acordo com os parâmetros dados para largura e altura.
  - O exemplo a seguir aumenta o elemento para duas vezes de sua largura original e três vezes de sua altura original:

```
div {
  transform: scale(2, 3);
}
```

- **O método `scaleX()`**
  - O método `scaleX()` aumenta ou diminui a largura de um elemento.

- **O método `scaleY()`**
  - O método `scaleY()` aumenta ou diminui a altura de um elemento.

Desenvolvimento Web

Transformações CSS

- **O método `scewX()`**
  - O método `skewX()` inclina um elemento ao longo do eixo X de acordo com o ângulo fornecido.

```
div {
  transform: skewX(20deg);
}
```

- **O método `matrix()`**
  - O método `matrix()` combina todos os métodos de transformação 2D em um.
  - O método matrix () usa seis parâmetros, contendo funções matemáticas, que permitem girar, dimensionar, mover (traduzir) e inclinar elementos.
  - Os parâmetros são os seguintes: *matrix(scaleX(), skewY(), skewX(), scaleY(), translateX(), translateY())*.

```
div {
  transform: matrix(1, -0.3, 0, 1, 0, 0);
}
```

Desenvolvimento Web

Transformações CSS 3D

- **Transformações CSS 3D**
  - CSS também suporta transformações 3D através de métodos da propriedade `transform`. Podemos usar os seguintes métodos de transformação:
    - `rotateX()`
    - `rotateY()`
    - `rotateZ()`

- **O método rotate<u>X</u>()**
  - O método `rotateX()` método gira um elemento em torno de seu eixo X em um determinado grau.

```
#myDiv {
  transform: rotateX(150deg);
}
```

- **O método rotate<u>Y</u>()**
  - O método `rotateY()` método gira um elemento em torno de seu eixo Y em um determinado grau.

```
#myDiv {
  transform: rotateY(150deg);
}
```

- **O método rotateZ()**
  - O método `rotateZ()` método gira um elemento em torno de seu eixo Z em um determinado grau.

```
#myDiv {
  transform: rotateZ(150deg);
}
```

INSTITUTO FEDERAL
Catarinense
Campus Camboriú

Desenvolvimento Web

Transições CSS

- **Transições CSS**
  - As transições CSS permitem que você altere os valores das propriedades suavemente, ao longo de um determinado período.
  - A seguir vamos aprender sobre as seguintes propriedades:
    - `transition`
    - `transition-delay`
    - `transition-duration`
    - `transition-property`
    - `transition-timing-function`

Desenvolvimento Web

Transições CSS

- **Como usar transições CSS?**
  - Para criar um efeito de transição, devemos especificar duas coisas: a propriedade CSS à qual desejamos adicionar um efeito e a duração do efeito:
  - Se a parte da duração não for especificada, a transição não terá efeito, porque o valor padrão é 0.

```
div {
  width: 100px;
  height: 100px;
  background: red;
  transition: width 2s;
}
```

  - O efeito de transição começará quando a propriedade CSS especificada (largura) alterar o valor. Agora, vamos especificar um novo valor para a propriedade de largura quando o usuário passar o mouse sobre o elemento :

```
div:hover {
  width: 300px;
}
```

- **Curva de velocidade da transição**
  - A propriedade `transition-timing-function` especifica a curva de velocidade do efeito de transição.
  - A propriedade da função de tempo de transição pode ter os seguintes valores:
    - `ease` - especifica um efeito de transição com um início lento, depois rápido e, em seguida, termina lentamente (este é o padrão);
    - `linear` - especifica um efeito de transição com a mesma velocidade do início ao fim;
    - `ease-in` - especifica um efeito de transição com início lento;
    - `ease-out` - especifica um efeito de transição com atenuação de fim lento;
    - `ease-in-out` - especifica um efeito de transição com um início e fim lentos;
    - `cubic-bezier(n, n, n, n)` - permite definir seus próprios valores em uma função cúbica de bezier.

- **Atraso no efeito de transição**
  - A propriedade `transition-delay` especifica um atraso (em segundos) para o efeito de transição. O exemplo a seguir tem um atraso de 1 segundo antes de começar:

```
div {
  transition-delay: 1s;
}
```

- **Transição + Transformação**
  - O exemplo a seguir adiciona um efeito de transição à transformação:

```
div {
  transition: width 2s, height 2s, transform 2s;
}
```

INSTITUTO FEDERAL
Catarinense
Campus Camboriú

Desenvolvimento Web I

Resumo

- **Fundamentos de CSS**
  - Posicionamento;
  - Transformações 2D;
  - Transformações 3D;
  - Transições.
- **Resumo**

INSTITUTO FEDERAL
Catarinense
Campus Camboriú

Referências

Referências Básicas

SILVA, Maurício Samy. CSS3: desenvolva aplicações web profissionais com uso dos poderosos recursos de estilização das CSS3. São Paulo: Novatec, 2012.

SILVA, Maurício Samy. HTML 5: a linguagem de marcação que revolucionou a web. São Paulo: Novatec, 2011.

NIEDERAUER, Juliano. Desenvolvendo websites com PHP: aprenda a criar websites dinâmicos e interativos com PHP e bancos de dados. 2. ed. rev. e atual. São Paulo: Novatec, 2011.

Referências Complementares

FLANAGAN, David. **o guia definitivo**. . O Really. 2012

SILVA, Maurício Samy. **Criando sites com HTML: sites de alta qualidade com HTML e CSS**. . Novatec. 2010

SOARES, Walace. **PHP 5: conceitos, programação e integração com banco de dados**. . Érica. 2010

DALL'OGLIO, Pablo. **PHP: programando com orientação a objetos**. . Novatec. 2009

DEITEL, Paul J. Ajax,. **Rich Internet applications e desenvolvimento Web para programadores**. . Pearson Prentice Hall. 2009

IEPSEN, Edécio Fernandes. **Lógica de Programação e Algoritmos com JavaScript.** Novatec. 2018.

Referências na Internet

https://www.w3schools.com

https://developer.mozilla.org/pt-BR/docs/Web

https://illustrated.dev/advancedjs