

```
grep: function( elems, callback, invert ) {
    var callbackInverse,
        matches = [],
        i = 0,
        length = elems.length,
        callbackExpect = !invert;
```

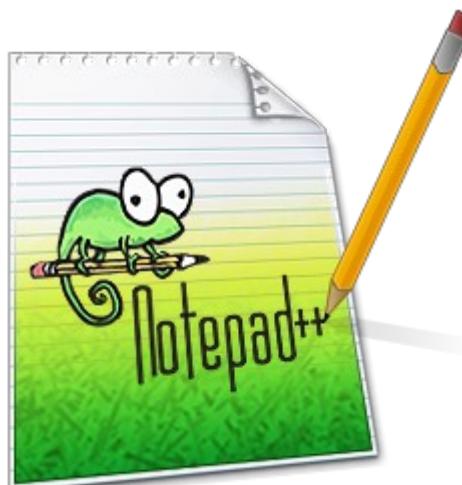
Mini-curso de introdução às tecnologias HTML e CSS



```
grep: function( elems, callback, invert ) {  
    var callbackInverse,  
        matches = [],  
        i = 0;  
    if( !elems.length )  
        return callback();  
    callbackInverse = !invert;  
    for( ; i < elems.length; i++ )  
        matches.push( elem );  
    if( invert )  
        matches.reverse();  
    callback( matches );  
};
```

Qual editor / IDE utilizaremos?

- Os exemplos serão feitos utilizando o bom e velho (nem tão velho) NotePad++



```
.ui-helper-hidden-accessible {  
border: 0;  
clip: rect(0 0 0 0);  
height: 1px;  
margin: -1px;  
overflow: hidden;  
padding: 0;  
position: absolute;  
width: 1px;}  
  
Instituto de Computação  
UFMT 2018
```

```
grep: function( elems, callback, invert ) {  
    var callbackInverse,  
        matches = [],  
        i = 0,  
        length = elems.length,  
        callbackExpect = !invert;
```

Qual editor / IDE utilizar?

- Todavia, encorajo vocês a experimentarem Editores/IDEs voltado para o desenvolvimento **frontend**:



Adobe Brackets



Sublime Text



Microsoft Visual Studio Code



```
grep: function( elems, callback, invert ) {
  var callbackInverse,
      matches = [],
      i = 0,
      length = elems.length,
      callbackExpect = !invert;
```

Outras Ferramentas



PASTEBIN



Ferramenta para colar e compartilhar trechos específicos de código
<https://pastebin.com/>

Ferramenta para executar código HTML, CSS e JavaScript sem necessidade de um editor
<https://jsfiddle.net/>



Semelhante ao JsFiddle
<https://codepen.io/>

```
grep: function( elems, callback, invert ) {
    var callbackInverse,
        matches = [],
        i = 0,
        length = elems.length,
        callbackExpect = !invert;
```

HTML

- O que é HTML?

- **HTML (Hyper Text Markup Language)** é uma linguagem para descrever páginas/documentos web
- HTML **não** é uma linguagem de programação (**procedural**), **mas sim** uma linguagem de marcação
 - Uma linguagem de marcação é um conjunto de marcadores: tags (elementos/nós)
 - Os marcadores são utilizados para descrever as páginas/documents web

```
grep: function( elems, callback, invert ) {  
    var callbackInverse,  
        matches = [],  
        i = 0,  
        length = elems.length,  
        callbackExpect = !invert;
```

Especificação

- A especificado da linguagem HTML é mantido pelo W3C:
 - World Wide Web Consortium
 - **HTML5 A vocabulary and associated APIs for HTML and XHTML**, Editores: Hickson I., Berjon R., Faulkner S, Leithead T., Navara, E. D., O'Connor E., Pfeiffer, S. Disponível em: <http://www.w3.org/TR/html5/>

Nota: O W3C, consórcio que conta com a participação de importantes empresas relacionadas a web (ex: Mozilla, Google, Microsoft) definem as especificações da linguagem HTML que os navegadores devem seguir.

```
grep: function( elems, callback, invert ) {
    var callbackInverse,
        matches = [],
        i = 0,
        length = elems.length,
        callbackExpect = !invert;
```

Outros Materiais

- Tutoriais / Cursos
 - W3Schools, **HTML(5) Tutorial**. Disponível em:
<http://www.w3schools.com/html/>
 - W3C Brasil, **Curso de HTML5**. Disponível em:
<http://www.w3c.br/Cursos/CursoHTML5>
- Fóruns, espaços para dúvidas e discussões, etc:
 - **HTML5ROCKS**: <http://www.html5rocks.com/en/>
 - **Stackoverflow**: <http://stackoverflow.com/>

```
grep: function( elems, callback, invert ) {
    var callbackInverse,
        matches = [],
        i = 0,
        length = elems.length,
        callbackExpect = !invert;
```

Outros Materiais

- Especificações dos Navegadores
 - MDN – Mozilla Developers Network
 - <https://developer.mozilla.org/pt-BR/docs/Web/HTML>
 - MSDN – Microsoft Developers Network
 - [https://msdn.microsoft.com/en-us/library/hh772721\(v=vs.85\).aspx](https://msdn.microsoft.com/en-us/library/hh772721(v=vs.85).aspx)

```
.ui-helper-hidden-accessible {
border: 0;
clip: rect(0 0 0 0);
height: 1px;
margin: -1px;
overflow: hidden;
padding: 0;
position: absolute;
width: 1px;}
```

```
grep: function( elems, callback, invert ) {
    var callbackInverse,
        matches = [],
        i = 0,
        length = elems.length,
        callbackExpect = !invert;
```

Outros Materiais

- W3Schools

The screenshot shows the homepage of w3schools.com. At the top, there's a green logo with 'w3schools' and the tagline 'the world's largest web development site' followed by 'educate yourself! beginners and experts'. A search bar is on the right. Below the header, there are six main categories with sub-links:

- HTML/CSS**
 - » Learn HTML
 - » Learn HTML5
 - » Learn CSS
 - » Learn CSS3
 - » Learn Bootstrap
- JavaScript**
 - » Learn JavaScript
 - » Learn jQuery
 - » Learn jQueryMobile
 - » Learn AppML
 - » Learn AngularJS
 - » Learn AJAX
 - » Learn JSON
- HTML Graphics**
 - » Learn Canvas
 - » Learn SVG
 - » Learn Google Maps
- w3 HTML**
 - HTML Tutorial
 - HTML Tag Reference
- w3 CSS**
 - CSS Tutorial
 - CSS Reference
- w3 JavaScript**
 - JavaScript Tutorial
 - JavaScript Reference
- w3 SQL**
 - SQL Tutorial
 - SQL Reference
- w3 PHP**
 - PHP Tutorial
 - PHP Reference
- w3 JQuery**
 - JQuery Tutorial
 - JQuery Reference

```
grep: function( elems, callback, invert ) {
    var callbackInverse,
        matches = [],
        i = 0,
        length = elems.length,
        callbackExpect = !invert;
```

W3Schools

- Não tem relação com o W3C
- Foi muito criticada por alguns anos:
 - Por induzir pessoas a pensarem que fazia parte do W3C
 - Por algum tempo o conteúdo esteve desatualizado
- Nos últimos anos o conteúdo foi atualizado
 - Fornece um guia rápido para consultas diretas

.ui-helper-hidden-accessible {
border: 0;
clip: rect(0 0 0 0);
height: 1px;
margin: -1px;
overflow: hidden;
padding: 0;
position: absolute;
width: 1px;}

```
grep: function( elems, callback, invert ) {
  var callbackInverse,
      matches = [],
      i = 0,
      length = elems.length,
      callbackExpect = !invert;
```

Voltando...

- Marcadores HTML (conhecidos por *tags*)
 - As tags HTML são palavras-chave circundadas por < > (“maior que” e “menor que”), ex: <html>
 - As tags HTML geralmente aparecem em pares como (abertura e fechamento), exm: <html> e </html>
 - O processo de iniciar e finalizar *tags* também é conhecido por abrir e fechar *tags*

```
.ui-helper-hidden-accessible {
border: 0;
clip: rect(0 0 0 0);
height: 1px;
margin: -1px;
overflow: hidden;
padding: 0;
position: absolute;
width: 1px;
```

```
grep: function( elems, callback, invert ) {
    var callbackInverse,
        matches = [],
        i = 0,
        elemLen = elems.length,
        callbackExpect = !invert;
```

Primeiro Documento HTML

```
<!DOCTYPE html>
<html>
    <head>
        <title>Primeiro HTML</title>
    </head>
    <body>
        <p>Olá!</p>
    </body>
</html>
```

Save o arquivo com a extensão .html, clique duas vezes ou abra à partir do seu navegador preferido!

.ui-helper-hidden-accessible {
border: 0;
clip: rect(0 0 0 0);
height: 1px;
margin: -1px;
overflow: hidden;
padding: 0;
position: absolute;
width: 1px;}

Instituto de Computação
UFMT 2018

Mais tags

```
grep: function( elems, callback, invert ) {  
  var callbackInverse,  
      matches = [],  
      i = 0,  
      length = elems.length,  
      callbackExpect = !invert;
```

- <p>** Definição de parágrafos de texto / imagens
-
** Adiciona uma quebra de linha ao texto / conteúdo
- <h1>, <h2>, <h...>, <h6>** Título de um trecho / seção
- ** Define uma palavra, texto ou trecho como “importante”
- ** Enfatiza uma palavra, texto ou trecho.
- <mark>** Destaca (highlight) uma palavra, texto ou trecho.

```
grep: function( elems, callback, invert ) {
  var callbackInverse,
      matches = [],
      i = 0,
      length = elems.length,
      callbackExpect = !invert;
```

Tags Descontinuadas

- Com a quinta versão da linguagem HTML (HTML5) algumas *tags* foram descontinuadas, entre elas: ***font***, ***center***, ***big***, entre outras.
 - Um dos motivos é que a formatação/estilização deve estar em folhas de estilo (CSS) e não nos elementos HTML
- Entretanto, *tags* como *b*, *i*, *u* e *br* **não** foram descontinuadas, algumas ganharam novos usos (não oficiais), ex: <i> para ícones

https://developer.mozilla.org/en-US/docs/Web/HTML/Element#Obsolete_and_deprecated_elements

<https://www.w3.org/International/questions/qa-b-and-i-tags>

```
grep: function( elems, callback, invert ) {
    var callbackInverse,
        matches = [],
        i = 0,
        length = elems.length,
        callbackExpect = !invert;
```

Outro Exemplo

```
<!DOCTYPE html>
<html>
...
<h2>Título da seção</h2>
<p>
<strong>Lorem ipsum dolor sit amet</strong>, consectetur adipiscing elit. Sed
purus orci, accumsan eget egestas viverra, <mark>feugiat et tellus</mark>.
<br />
Donec pellentesque vulputate eros, <em>a cursus velit interdum</em> a. Donec
mattis nunc elementum felis tempor,      sed consectetur lectus mattis.
</p>
<hr />
<p>
<mark>Pellentesque habitant morbi tristique senectus et netus et malesuada
fames ac turpis egestas</mark>. Etiam      volutpat neque accumsan rhoncus
fermentum.
<br />
<ui-helper-hidden-accessible>
    Quisque odio quam, vehicula ac elementum et, <em>dapibus sit amet erat.</em>
    border</p>;
    clip: rect(0 0 0 0);
    height: 1px;
    margin: -1px;
    overflow: hidden;
    padding: 0;
    position: absolute;
    width: 1px;
    </ui-helper-hidden-accessible>

```

```
grep: function( elems, callback, invert ) {
    var callbackInverse,
        matches = [],
        i = 0,
        length = elems.length,
        callbackExpect = !invert;
```

Resultado Esperado

Título da seção

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed purus orci, accumsan eget egestas viverra, feugiat et tellus.

Donec pellentesque vulputate eros, a cursus velit interdum a. Donec mattis nunc elementum felis tempor, sed consectetur lectus mattis.

Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Etiam volutpat neque accumsan rhoncus fermentum.

Quisque odio quam, vehicula ac elementum et, dapibus sit amet erat.

```
.ui-helper-hidden-accessible {
border: 0;
clip: rect(0 0 0 0);
height: 1px;
margin: -1px;
overflow: hidden;
padding: 0;
position: absolute;
width: 1px;}
```

```
grep: function( elems, callback, invert ) {  
    var callbackInverse,  
        matches = [],  
        i = 0;  
    length = elems.length;  
    callbackExpect = !invert;
```

Layout de documentos HTML

- Por padrão utilizaremos o layout do **slide 18** para documentos HTML, por sem mais intuitivo e didático. Porém, aquele não é o único layout válido, por exemplo, o layout a seguir é considerado válido:

```
<!DOCTYPE html>  
<meta charset="UTF-8">  
<title>Minimal HTML Template</title>  
<p>Insert content here.</p>
```

<https://dev.w3.org/html5/html-author/#basic-templates>

.ui-helper-hidden-accessible {
border: 0;
clip: rect(0 0 0 0);
height: 1px;
margin: -1px;
overflow: hidden;
padding: 0;
position: absolute;
width: 1px;

Utilidades

- “***Lorem Ipsum***”: Texto em latim utilizado para preencher espaços e simular o resultado esperado em trabalhos com mídias digitais.
 - Gerador: <http://br.lipsum.com/>
- **Entidades HTML**: Conjunto de caracteres que representam caracteres específicos geralmente “difíceis” de escrevermos. Podemos utilizar o **nome** da entidade, ex: ou o código

<https://dev.w3.org/html5/html-author/charref>

```
grep: function( elems, callback, invert ) {
    var callbackInverse,
        matches = [],
        i = 0,
        length = elems.length,
        callbackExpect = !invert;
```

Utilidades

- Codificação:

- É importante observar a **codificação do arquivo (físico)** e **explicitá-la no documento HTML**, ex:
 - <meta charset="UTF-8">
- Geralmente a codificação utilizada por padrão é UTF-8. Outra codificação muito utilizada é a ISO-8859-1 (Latin 1)

```
.ui-helper-hidden-accessible {
border: 0;
clip: rect(0 0 0 0);
height: 1px;
margin: -1px;
overflow: hidden;
padding: 0;
position: absolute;
width: 1px;
```

```
grep: function( elems, callback, invert ) {  
    var callbackInverse,  
        matches = [],  
        i = 0,  
        length = elems.length,  
        callbackExpect = !invert;
```

Metatags

- A tag `<title>` é uma **metatag** ou seja, uma tag para definirmos informações do próprio documento. Muitas outras informações, como a codificação do documento podem ser definidas pela **metatag** `<meta>`. Padrão:

```
<meta name="nome" content="valor" />
```

- Exemplos:

```
<meta name="author" content="Jivago Medeiros" />
```

```
<meta name="description" content="Mini-curso" />
```

```
<meta name="keywords" content="html css javascript" />
```

https://developer.mozilla.org/pt-PT/docs/utilizando_meta_tags

```
grep: function( elems, callback, invert ) {
  var callbackInverse,
      matches = [],
      i = 0,
      length = elems.length,
      callbackExpect = !invert;
```

Utilidades

- Comentários:

- Comentários em HTML são colocados entre as *tags* `<!-- ... -->`, exemplo:

```
<!-- Eu sou um comentário -->
<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit.</p>
```

```
.ui-helper-hidden-accessible {
border: 0;
clip: rect(0 0 0 0);
height: 1px;
margin: -1px;
overflow: hidden;
padding: 0;
position: absolute;
width: 1px;}
```

**Instituto de Computação
UFMT 2018**

**Futxicaia da Tecnológica
Projeto de Extenção**

```
grep: function( elems, callback, invert ) {
    var callbackInverse,
        matches = [],
        i = 0,
        elemLength = elems.length,
        callbackInvert = !invert;
```

Importante Conjunto de tags

- **Listas**
 - , , , <dl>, <dt>, <dd>
- **Blocos**
 - <p>, <div>,
- **Tabelas**
 - <table>, <thead>, <tbody>, <tfoot>,

```
.ui-helper-hidden-accessible {
border: 0;
clip: rect(0 0 0 0);
height: 1px;
margin: -1px;
overflow: hidden;
padding: 0;
position: absolute;
width: 1px;}
```

```
grep: function( elems, callback, invert ) {
    var callbackInverse,
        matches = [],
        i = 0,
        length = elems.length,
        callbackExpect = !invert;
```

Listas

- Lista não ordenada `` para a lista e `` para os itens
- Listas ordenadas `` para a lista e `` para os itens
- Listas de descrição `<dl>` para a lista, `<dt>` para o título dos itens e `<dd>` para a descrição dos itens

```
.ui-helper-hidden-accessible {
border: 0;
clip: rect(0 0 0 0);
height: 1px;
margin: -1px;
overflow: hidden;
padding: 0;
position: absolute;
width: 1px;}
```

```
grep: function( elems, callback, invert ) {
    var callbackInverse,
        matches = [],
        i = 0,
        length = elems.length,
        callbackExpect = !invert;
```

Atributos

- Atributos especificação “informações extras/mais” para os elementos.
- Sintaxe:
 - **nomeDoAtributo="Valor do Atributo"**
- Exemplo:
 - **<ol type="1">**
 - **<ol type="A">**

.ui-helper-hidden-accessible {
border: 0;
clip: rect(0 0 0 0);
height: 1px;
margin: -1px;
overflow: hidden;
padding: 0;
position: absolute;
width: 1px;

```
grep: function( elems, callback, invert ) {  
    var callbackInverse,  
        matches = [],  
        i = 0,  
        length = elems.length,  
        callbackExpect = !invert;
```

Atributos Globais

- A especificação do HTML5 define uma série de atributos **comuns** a **todos** os elementos, denominados de “Atributos Globais” (*global attributes*):

<http://www.w3.org/TR/html-markup/global-attributes.html>

https://developer.mozilla.org/en-US/docs/Web/HTML/Global_attributes

http://www.w3schools.com/tags/ref_standardattributes.asp

.ui-helper-hidden-accessible {
border: 0;
clip: rect(0 0 0 0);
height: 1px;
margin: -1px;
overflow: hidden;
padding: 0;
position: absolute;
width: 1px;}

Mais importantes nesse momento

- **id** – Atribui uma identificação / nome a um elemento, é esperado que seja único
- **class** – Atribui uma ou várias classes a um elemento de forma a “agrupar” elementos de uma mesma classe
- **style** – Define estilos de formatação *inline* para o elemento
- **title** – Adiciona alguma informação “a mais” sobre o elemento

```
grep: function( elems, callback, invert ) {  
    var callbackInverse,  
        matches = [],  
        i = 0,  
        length = elems.length,  
        callbackExpect = !invert;
```

Voltando... Blocos

- **<div>** Cria blocos do tipo *block*. Que ocupam toda a largura disponível, “bloqueando” outros elementos na mesma “linha”. Muito utilizado para agrupar diferentes elementos em uma determinada região / setor da página com características próprias.
- **** Cria blocos *inline*. blocos que circundam o texto / conteúdo porém não adicionam quebram de linha, permitindo vários blocos em uma mesma linha. Muito utilizado para aplicação de uma formatação / estilo a um determinado trecho de texto.

.ui-helper-hidden-accessible {
border: none;
clip: rect(0 0 0 0);
height: 0;
margin: -1px;
overflow: hidden;
padding: 0;
position: absolute;
width: 1px;

```
grep: function( elems, callback, invert ) {
    var callbackInverse,
        matches = [],
        i = 0,
        length = elems.length,
        callbackExpect = !invert;
```

Tabelas

- Define uma tabela: `<table>`
- Linhas: `<tr>`
- Colunas: `<td>`
- Cabeçalho, corpo e rodapé da tabela: `thead`, `tbody`, `tfoot`

```
.ui-helper-hidden-accessible {
border: 0;
clip: rect(0 0 0 0);
height: 1px;
margin: -1px;
overflow: hidden;
padding: 0;
position: absolute;
width: 1px;}
```

```
grep: function( elems, callback, invert ) {
    var callbackInverse,
        matches = [],
        i = 0,
        length = elems.length,
        callbackExpect = !invert;
```

Imagens

- A especificação da linguagem HTML define uma *tag* para a representação de imagens nos documentos HTML, a tag **img**, sintaxe básica:

```

```

```
.ui-helper-hidden-accessible {
border: 0;
clip: rect(0 0 0 0);
height: 1px;
margin: -1px;
overflow: hidden;
padding: 0;
position: absolute;
width: 1px;
```

**Instituto de Computação
UFMT 2018**

**Futxicaia da Tecnológica
Projeto de Extenção**

```
grep: function( elems, callback, invert ) {
  var callbackInverse,
      matches = [],
      i = 0,
      length = elems.length,
      callbackExpect = !invert;
```

Exemplo

```

```

```
.ui-helper-hidden-accessible {
border: 0;
clip: rect(0 0 0 0);
height: 1px;
margin: -1px;
overflow: hidden;
padding: 0;
position: absolute;
width: 1px;}
```

**Instituto de Computação
UFMT 2018**

**Futxicaia da Tecnológica
Projeto de Extenção**

```
grep: function( elems, callback, invert ) {
    var callbackInverse,
        matches = [],
        i = 0,
        length = elems.length,
        callbackExpect = !invert;
```

Links

- Utilizando a tag `<a>`
 - Ligações para outras páginas / documentos
 - Ligações para a mesma página / documento
 - Exemplo:
 - `Outra Página`
- ou ainda

```
.ui-helper-hidden-accessible {  
border: 0;  
clip: rect(0 0 0 0);  
height: 1px;  
margin: -1px;  
overflow: hidden;  
padding: 0;  
position: absolute;  
width: 1px;}
```

Instituto de Computação
UFMT 2018

Futxicaia da Tecnológica
Projeto de Extensão

```
grep: function( elems, callback, invert ) {  
    var callbackInverse,  
        matches = [],  
        i = 0,  
        length = elems.length,  
        callbackExpect = !invert;
```

Atributo target

- O atributo **target** define “onde” o link será aberto, por exemplo, em uma nova página.
Exemplo:
 - `Outra Página`

```
.ui-helper-hidden-accessible {  
border: 0;  
clip: rect(0 0 0 0);  
height: 1px;  
margin: -1px;  
overflow: hidden;  
padding: 0;  
position: absolute;  
width: 1px;
```

```
grep: function( elems, callback, invert ) {
  var callbackInverse,
      matches = [],
      i = 0,
      length = elems.length,
      callbackExpect = !invert;
```

Observação

- Todos os elementos HTML possuem características padrões, como exibição, visibilidade, formatação de texto, entre outros. De um modo geral, todas essas características podem ser alteradas utilizando CSS.

```
.ui-helper-hidden-accessible {
border: 0;
clip: rect(0 0 0 0);
height: 1px;
margin: -1px;
overflow: hidden;
padding: 0;
position: absolute;
width: 1px;}
```

```
grep: function( elems, callback, invert ) {
    var callbackInverse,
        matches = [],
        i = 0,
        length = elems.length,
        callbackInvert = !invert;
```

Aninhar elementos HTML

- Aninhar elementos HTML não é somente possível como também em muitos casos é necessário, como por exemplo:

```
<div>
  <p>
    <span>
      <strong>Lorem Ipsum</strong>
    </span>
  </p>
</div>
```

```
.ui-helper-hidden-accessible {
border: 0;
clip: rect(0 0 0 0);
height: 1px;
margin: -1px;
overflow: hidden;
padding: 0;
position: absolute;
width: 1px;
```

```
grep: function( elems, callback, invert ) {  
    var callbackInverse,  
        matches = [],  
        i = 0,  
        length = elems.length,  
        callbackInvert = !invert;
```

Aninhar elementos HTML

- Porém, alguns aninhamentos podem provocar comportamentos não esperados, exemplo:

```
<div>  
  <p>  
    <h2>Lorem Ipsum</h2>  
  </p>  
</div>
```

```
.ui-helper-hidden-accessible {  
border: 0;  
clip: rect(0 0 0 0);  
height: 1px;  
margin: -1px;  
overflow: hidden;  
padding: 0;  
position: absolute;  
width: 1px;}
```

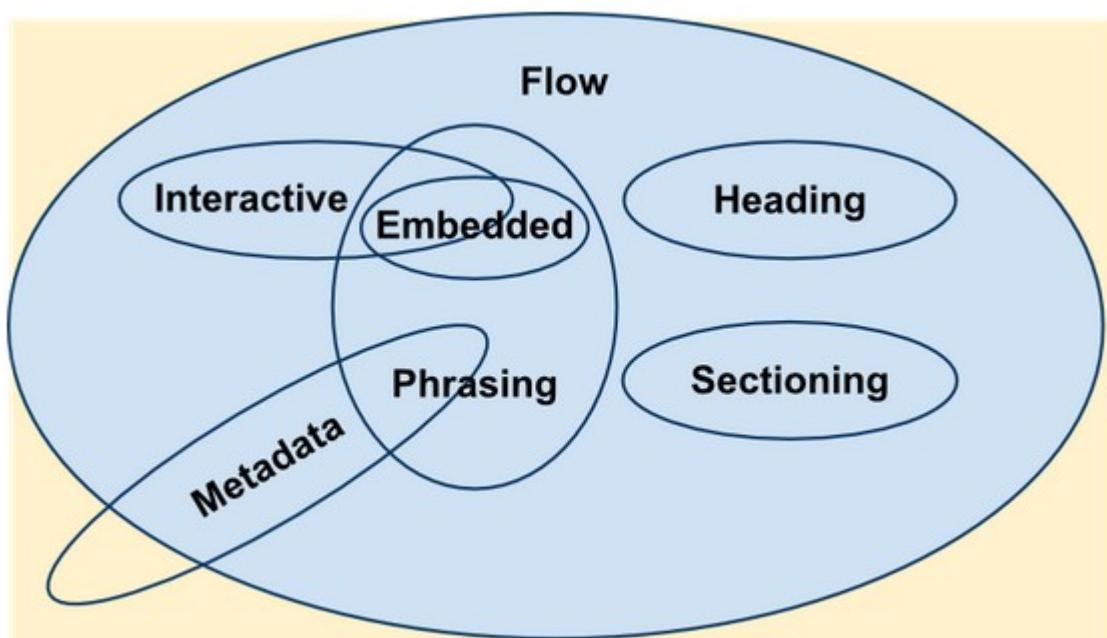
**Instituto de Computação
UFMT 2018**

**Futxicaia da Tecnológica
Projeto de Extenção**

```
grep: function( elems, callback, invert ) {  
    var callbackInverse,  
        matches = [],  
        i = 0,  
        length = elems.length,  
        callbackExpect = !invert;
```

Content Models

- Isso acontece porque cada elemento possui um modelo de conteúdo (*content models*) específico:



https://developer.mozilla.org/en-US/docs/Web/Guide/HTML/Content_categories

<https://www.w3.org/TR/2011/WD-html5-20110525/content-models.html#content-models>

```
grep: function( elems, callback, invert ) {  
    var callbackInverse,  
        matches = [],  
        i = 0;  
    if ( !callback ) {  
        callback = elem.length ?  
            callbackExpect = !invert;
```

Ainda sobre content models

- Na documentação / especificação dos elementos / tags HTML sempre consta a categoria de *content model* que a tag percente e a categoria de *content model* que a tag aceita dentro dela, exemplo da tag `<p>` :

Categorias de conteúdo	Conteúdos de fluxo, conteúdos palpáveis.
Conteúdo permitido	Conteúdo com texto.

<https://developer.mozilla.org/pt-BR/docs/Web/HTML/Element/p>

Nota: As categorias de conteúdo são flexíveis, servem “apenas” como um agrupamento para elementos que compartilham características e comportamentos semelhantes e uma forma de se resolver “questões complicadas”.

```
grep: function( elems, callback, invert ) {
  var callbackInverse,
      matches = [],
      i = 0,
      length = elems.length,
      callbackExpect = !invert;
```

Exercício

- Criar duas páginas HTML:
 - Primeira página:
 - Quatro blocos do tipo div e dentro dos blocos:
 - 1º bloco: três parágrafos com texto
 - 2º bloco: Listas: Uma ordenada, uma não ordenada e uma lista de definição (pelo menos 3 itens em cada)
 - 3º bloco: Uma tabela 3x3 (contendo cabeçalho e rodapé)
 - 4º bloco: Cinco (5) outras tags a sua escolha (br não vale)
 - Link para a segunda página

```
.ui-helper-hidden-accessible {
border: 0;
clip: rect(0 0 0 0);
height: 1px;
margin: -1px;
overflow: hidden;
padding: 0;
position: absolute;
width: 1px;}
```

Instituto de Computação
UFMT 2018

Futxicaiada Tecnológica Projeto d
e Extenção

```
grep: function( elems, callback, invert ) {
  var callbackInverse,
      matches = [],
      i = 0,
      length = elems.length,
      callbackExpect = !invert;
```

Exercício

- Criar duas páginas HTML:
 - Segunda página:
 - Inserir uma imagem (sem copyright)
 - Sugestão: <http://www.freeimages.com/>
 - Inserir na imagem um link para a primeira página
 - Incluir via comentário em HTML e metatags nome completo e e-mail

```
.ui-helper-hidden-accessible {
border: 0;
clip: rect(0 0 0 0);
height: 1px;
margin: -1px;
overflow: hidden;
padding: 0;
position: absolute;
width: 1px;}
```

```
grep: function( elems, callback, invert ) {
    var callbackInverse,
        matches = [],
        i = 0,
        length = elems.length,
        callbackExpect = !invert;
```

Introdução à CSS

```
.ui-helper-hidden-accessible {
border: 0;
clip: rect(0 0 0 0);
height: 1px;
margin: -1px;
overflow: hidden;
padding: 0;
position: absolute;
width: 1px;}
```

**Instituto de Computação
UFMT 2018**

**Futxicaia da Tecnológica
Projeto de Extenção**

```
grep: function( elems, callback, invert ) {
    var callbackInverse,
        matches = [],
        i = 0,
        length = elems.length,
        callbackExpect = !invert;
```

CSS

- CSS (*Cascading Style Sheets*) – Linguagem de folhas de estilo [em cascata]
- **Objetivo:** Separar a estrutura / conteúdo do estilo / formatação em documentos (exemplo, HTML). Definindo e alterando diferentes “folhas de estilo” sempre que necessário.

```
.ui-helper-hidden-accessible {
border: 0;
clip: rect(0 0 0 0);
height: 1px;
margin: -1px;
overflow: hidden;
padding: 0;
position: absolute;
width: 1px;}
```

```
grep: function( elems, callback, invert ) {
  var callbackInverse,
      matches = [],
      i = 0,
      length = elems.length,
      callbackExpect = !invert;
```

Especificação

- **Cascading Style Sheets (CSS).** Editor: ETEMAD, E. J. Disponível em:
<http://www.w3.org/TR/CSS/>

```
.ui-helper-hidden-accessible {
border: 0;
clip: rect(0 0 0 0);
height: 1px;
margin: -1px;
overflow: hidden;
padding: 0;
position: absolute;
width: 1px;}
```

```
grep: function( elems, callback, invert ) {
    var callbackInverse,
        matches = [],
        i = 0,
        length = elems.length,
        callbackExpect = !invert;
```

Outros Materiais

- Tutorias / cursos
 - W3Schools, **CSS3 Introduction**. Disponível em:
http://www.w3schools.com/css/css3_intro.asp
 - W3C Brasil, **Curso de CSS3**. Disponível em:
<http://www.w3c.br/Cursos/CursoCSS3>
- Outros
 - CSS-TRICKS: <https://css-tricks.com/>

```
.ui-helper-hidden-accessible {
border: 0;
clip: rect(0 0 0 0);
height: 1px;
margin: -1px;
overflow: hidden;
padding: 0;
position: absolute;
width: 1px;}
```

```
grep: function( elems, callback, invert ) {  
    var callbackInverse,  
        matches = [],  
        i = 0,  
        length = elems.length,  
        callbackExpect = !invert;
```

CSS

- **História:** A primeira versão (CSS1) foi especificada poucos anos depois da primeira especificação publicada do HTML1 (1993 HTML e 1996 CSS). Obtendo notoriedade em nível global apenas no início da primeira década do ano 2000.

```
.ui-helper-hidden-accessible {  
border: 0;  
clip: rect(0 0 0 0);  
height: 1px;  
margin: -1px;  
overflow: hidden;  
padding: 0;  
position: absolute;  
width: 1px;}
```

```
grep: function( elems, callback, invert ) {
  var callbackInverse,
      matches = [],
      i = 0,
      length = elems.length,
      callbackExpect = !invert;
```

Módulos CSS

- A partir do CSS3, a especificação passou a ser dividida em diferentes módulos que podem ser atualizados de forma independente um dos outros, dando origem a diferentes *levels* (níveis).
- Alguns módulos:
 - Selectors, Color, Background, Box, Media Queries, Namespaces, etc

```
.ui-helper-hidden-accessible {
  border: 0;
  clip: rect(0 0 0 0);
  height: 1px;
  margin: -1px;
  overflow: hidden;
  padding: 0;
  position: absolute;
  width: 1px;}
```

```
grep: function( elems, callback, invert ) {  
    var callbackInverse,  
        matches = [],  
        i = 0,  
        length = elems.length,  
        callbackExpect = !invert;
```

O que é possível formatar CSS?

- Tipo da fonte, cor da fonte, estilo da fonte, bordas (tipo, cor, tamanho), fundo (cor, imagem, comportamento). Posição / posicionamento / tamanho / comportamento / visibilidade / fluides / sombra dos elementos. Elementos / itens / campos dos formulários. Etc, etc, etc...

...basicamente, “tudo” em um documento HTML

```
.ui-helper-hidden-accessible {  
border: 0;  
clip: rect(0 0 0 0);  
height: 1px;  
margin: -1px;  
overflow: hidden;  
padding: 0;  
position: absolute;  
width: 1px;}
```

```
grep: function( elems, callback, invert ) {  
    var callbackInverse,  
        matches = [],  
        i = 0,  
        length = elems.length,  
        callbackExpect = !invert;
```

Sintaxe

```
seletor, seletores {  
    propriedade: valor1;  
    outra-propriedade: valor1 valor2 valor3;  
}
```

```
.ui-helper-hidden-accessible {  
border: 0;  
clip: rect(0 0 0 0);  
height: 1px;  
margin: -1px;  
overflow: hidden;  
padding: 0;  
position: absolute;  
width: 1px;
```

**Instituto de Computação
UFMT 2018**

**Futxicaia da Tecnológica
Projeto de Extenção**

```
grep: function( elems, callback, invert ) {  
    var callbackInverse,  
        matches = [],  
        i = 0,  
        length = elems.length,  
        callbackExpect = !invert;
```

Exemplos

```
p {  
    color: #060;  
    text-align: center;  
}
```

```
.ui-helper-hidden-accessible {  
border: 0;  
clip: rect(0 0 0 0);  
height: 1px;  
margin: -1px;  
overflow: hidden;  
padding: 0;  
position: absolute;  
width: 1px;
```

```
grep: function( elems, callback, invert ) {  
    var callbackInverse,  
        matches = [],  
        i = 0,  
        length = elems.length,  
        callbackExpect = !invert;
```

Seletores

- **Seletores:** Definem (selecionam) quais os conjuntos de elementos receberão as propriedades.

Aplica o estilo / formatação (`color` e `text-align`) a todos os **elementos** do tipo `<p>`



```
p {  
    color: #060;  
    text-align: center;  
}
```

```
.ui-helper-hidden-accessible {  
border: 0;  
clip: rect(0 0 0 0);  
height: 1px;  
margin: -1px;  
overflow: hidden;  
padding: 0;  
position: absolute;  
width: 1px;
```

Instituto de Computação
UFMT 2018

```
grep: function( elems, callback, invert ) {  
    var callbackInverse,  
        matches = [],  
        i = 0,  
        length = elems.length,  
        callbackExpect = !invert;
```

Seletores básios

- Nome da *tag / elemento*
 - Aplica o estilo a todos os elementos daquela *tag*
- *#id* do Elemento
 - Aplica o estilo somente ao elemento que possui aquele *#id*
- .*nome da Classe*
 - Aplica a todas os elementos da classe (atributo *class*)

```
.ui-helper-hidden-accessible {  
border: 0; class:  
clip: rect(0 0 0 0);  
height: 1px;  
margin: -1px;  
overflow: hidden;  
padding: 0;  
position: absolute;  
width: 1px;
```

```
grep: function( elems, callback, invert ) {
  var callbackInverse,
    matches = [],
    i = 0,
    length = elems.length,
    callbackExpect = !invert;
```

Exemplos

```
h1 {
  background-color: #000;
  color: #FFF;
  font-size: 32px;
  text-decoration: underline;
}

#conteudo {
  background-color: #CCC;
  border-bottom: 4px double #000;
}
```

```
.paragrafo-maior {
  font-size: 18px;
  font-weight: bold;
```

```
grep: function( elems, callback, invert ) {  
    var callbackInverse,  
        matches = [],  
        i = 0,  
        length = elems.length,  
        callbackExpect = !invert;
```

Sobre cores em CSS...

- Quando trabalhamos com **cores em CSS**, seja cor da fonte (color), cor de fundo (background-color), cor da borda (border-color) entre outras, há **três formas de definí-las**:
 - Pelo nome (em inglês) da cor, ex:
 - **black**, **blue**, **brown**, **gold**, **gray**, **green**, **olive**, **pink**, **silver**, **snow**, **violet**, **white**, **yellow**, entre muitas outras:

https://www.w3schools.com/colors/colors_names.asp

```
grep: function( elems, callback, invert ) {  
    var callbackInverse,  
        matches = [],  
        i = 0,  
        length = elems.length,  
        callbackExpect = !invert;
```

Sobre cores em CSS...

- Definindo a quantidade de cada cor **R G B**:
 - Podemos definir a "quantidade" de cada cor tanto em **HEXA** decimal quanto em **Decimal**, em ambos, o valor de cada cor vai de 0 à 255 (ou FF em hexadecimal)
 - Para "montarmos" um cor em hexadecimal utilizamos cerquilha (hashtag) "#", ex:
 - background-color: #**1F822D**;
 - Quantidade de vermelho: **1F**
 - Quantidade de verde: **82**
 - Quantidade de azul: **2D**

Importante: Podemos definir cores em hexadecimal com apenas três dígitos, exemplo: **#333** Nesse caso, o 3 seria duplicado em cada cor, tendo o mesmo valor que **#333333**. Cores com um ou dois dígitos não são válidas, com quatro e cinco segue a mesma lógica que para três

```
grep: function( elems, callback, invert ) {  
    var callbackInverse,  
        matches = [],  
        i = 0,  
        length = elems.length,  
        callbackExpect = !invert;
```

Sobre cores em CSS...

- Para definirmos uma cor em decimal devemos utilizar a função **rgb()**, exemplo:
 - background-color: **rgb(31,130,45)**;
 - Quantidade de vermelho: **31**
 - Quantidade de verde: **130**
 - Quantidade de azul: **45**
- Há uma variação da função `rgb()`, a `rgba()`, onde, além de definirmos a quantidade de vermelho, verde e azul, definimos o "alpha" (a opacidade / transparência da cor) utilizando um número que vai de 0 a 1, exemplo:

```
.ui-helper-hidden-accessible {  
border: 0; – background-color: rgba(31,130,45,0.65);  
clip: rect(0 0 0 0);  
height: 1px;  
margin: -1px;  
overflow: hidden;  
padding: 0; position: absolute;  
width: 1px;
```

```
grep: function( elems, callback, invert ) {
    var callbackInverse,
        matches = [],
        i = 0,
        length = elems.length,
        callbackExpect = !invert;
```

Sobre cores em CSS...

- Por fim, além de definirmos cores pelo **nome**, pela **quantidade de RGB** (seja em decimal ou hexadecimal) também podemos definir cores pela posição da Matiz (**Hue**), intensidade de Saturação (**Saturation**) e Luminosidade (**Lightness**) utilizando a função **hsl()**, que é usada de maneira semelhante a **rgb()** e também possui a variação **hsla()**.

```
.ui-helper-hidden-accessible {
border: 0;
clip: rect(0 0 0 0);
height: 1px;
margin: -1px;
overflow: hidden;
padding: 0;
position: absolute;
width: 1px;}
```

```
grep: function( elems, callback, invert ) {  
    var callbackInverse,  
        matches = [],  
        i = 0,  
        length = elems.length,  
        callbackExpect = !invert;
```

Sobre cores em CSS...

- Matiz (Hue): posição, em graus (0 a 360) na roda de cores. 0 (ou 360) é vermelho, 120 é verde e 240 azul.
- Saturação: porcentagem (0 a 100) da intensidade da cor. O valor 100 é a cor “inteira”.
- Luminosidade: porcentagem (0 a 100) de preto/branco na cor, 0 é preto 100 é branco.

- Exemplo:
- **hs1(128,76,51);**

```
.ui-helper-hidden-accessible {  
border: 0;  
clip: rect(0 0 0 0);  
height: 1px;  
margin: -1px;  
overflow: hidden;  
padding: 0;  
position: absolute;  
width: 1px;
```

Sobre medidas em CSS...

Unidades Absolutas

Nome	Unidade	Descrição
Centímetros	cm	Define o tamanho/largura/altura de um objeto, elemento, font, etc em centímetros.
Milímetros	mm	Define o tamanho/largura/altura de um objeto, elemento, font, etc em milímetros.
Polegadas	in	Define o tamanho/largura/altura de um objeto, elemento, font, etc em polegadas.
Pontos	pt	Define o tamanho/largura/altura de um objeto, elemento, font, etc em pontos (Unidade muito utilizada na impressão de textos e materiais, 72pt = 1in).

```
grep: function( elems, callback, invert ) {
  var callbackInverse,
      matches = [],
      i = 0,
      length = elems.length,
      callbackExpect = !invert;
```

Sobre medidas em CSS...

Unidades Relativas

Nome	Unidade	Descrição
Pixels*	px	Número de pixels relativo a resolução do dispositivo de exibição.
Porcentagem	%	O tamanho/largura/altura de um objeto ou elemento é definido em porcentagem, exemplo: 10%, 25%, etc.
	em	Tamanho em relação a fonte atual. Ex: um elemento herda uma fonte com tamanho 20px e possui border-width: 2em, significa que a borda terá o tamanho de 40px (duas vezes 20px).

.ui-helper-hidden-accessible {

*Pixels muitas vezes é colocado como uma unidade de medida relativa, pois, o tamanho dos elementos definidos utilizando pixels acabam sendo **proporcionais a resolução da tela.**

```
grep: function( elems, callback, invert ) {
    var callbackInverse,
        matches = [],
        i = 0,
        length = elems.length,
        callbackExpect = !invert;
```

Sobre medidas em CSS...

Unidades Relativas

Nome	Unidade	Descrição
Viewport Height	vh	Unidade relativa à altura da viewport do documento (vai de 1 a 100, 1vh = 1% da altura da viewport)
Viewport Width	vw	Unidade relativa à largura da viewport do documento (vai de 1 a 100, 1vw = 1% da largura da viewport)

Importante: Revisaremos as unidades relativas quando estivermos falando de layouts responsivos. Por hora, utilizaremos, principalmente, pixels, por ser uma unidade de medida mais simples e consequentemente didática.

.ui-helper-hidden-accessible {

border: 0; clip: rect(0 0 0 0); height: 1px; margin: -1px; overflow: hidden;

padding: 0; position: absolute; width: 1px;

**Instituto de Computação
UFMT 2018**

**Futxicaia da Tecnológica
Projeto de Extensão**

```
grep: function( elems, callback, invert ) {
    var callbackInverse,
        matches = [],
        i = 0,
        length = elems.length,
        callbackExpect = !invert;
```

Combinar seletores

- É possível limitar o seletor de classe e ID pela tag, ex:

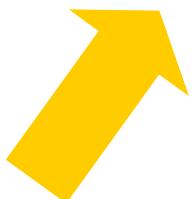
```
div#conteudo {
    background-color: #CCC;
    border-bottom: 4px double #000;
}
```

```
p.paragrafo-maior {
    font-size: 18px;
    font-weight: bold;
```

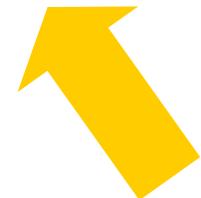
Seletores para elementos aninhados

- Se adicionarmos um **espaço** entre os seletores, a semântica "muda completamente":

```
div#conteudo {  
    ...  
}
```



```
div #conteudo {  
    ...  
}
```



Seleciona elemento **div** com o **id conteudo**:

```
<div id="conteudo">  
.ui-helper-hidden-accessible {  
border: 0;  
clip: rect(0 0 0 0);  
height: 1px;  
margin: -1px;  
overflow: hidden;  
padding: 0;  
position: absolute;  
width: 1px;
```

Seleciona **qualquer** elemento com **id conteudo** que tenha um elemento **div** como **pai** (direto ou indireto):

```
<div>  
    <p id="conteudo">
```

Seletores para elementos aninhados

- É possível aplicar estilo a elementos dentro de outros elementos, exemplo:

```
div#conteudo p.paragrafo-maior strong {  
    color: #F00;  
    font-weight: bold;  
}
```

Nesse caso, todos os p.paragrafo-maior que estiverem dentro do div#conteudo (direta ou indiretamente) receberão as propriedades.

Seletores para elementos aninhados

- Se quisermos aplicar o estilo apenas aos elementos que sejam filhos imediatos utilizamos >:

```
div#conteudo > p.paragrafo-maior {  
    color: #F00;  
    font-weight: bold;  
}
```

.ui-helper-hidden-accessible {
border: 1px solid black;
clip: rect(0 0 0 0);
height: 1px;
margin: -1px;
overflow: hidden;
padding: 0;
width: 1px;

Lista rápida com alguns seletores:

http://www.w3schools.com/cssref/css_selectors.asp

```
grep: function( elems, callback, invert ) {
  var callbackInverse,
      matches = [],
      i = 0,
      notCh = elems.length,
      callbackExpect = !invert;
```

Onde definir os estilos?

- Basicamente, os estilos CSS podem ser definidos em três "locais" diferentes:
 - Diretamente no corpo do documento HTML, entre as tags **<style></style>**
 - Em um arquivo externo (geralmente .css), sendo o arquivo incluído no documento HTML por meio da tag **<link>**
 - E "*inline*": diretamente no elemento/tag HTML por meio do atributo **style**

```
grep: function( elems, callback, invert ) {
    var callbackInverse,
        matches = [],
        i = 0,
        length = elems.length,
        callbackExpect = !invert;
```

Arquivo Externo

Seja o arquivo **estilo.css** com o seguinte conteúdo:

estilo.css

```
h1 {
    background-color: #000;
    color: #FFF;
    font-size: 32px;
    text-decoration: underline;
}

#conteudo {
    background-color: #CCC;
    border-bottom: 4px double #000;
}

.paragrafo-maior {
    font-size: 18px;
    font-weight: bold;
}
```

```
.ui-helper-hidden-accessible {
border: 0;
clip: rect(0 0 0 0);
height: 1px;
margin: -1px;
overflow: hidden;
padding: 0;
position: absolute;
width: 1px;}
```

Instituto de Computação
UFMT 2018

```
grep: function( elems, callback, invert ) {
    var callbackInverse,
        matches = [],
        i = 0,
        length = elems.length,
        callbackExpect = !invert;
```

Arquivo Externo

Incluímos a folha de estilos definida em um arquivo **externo** da seguinte forma:

```
<html>
  <head>
    <link rel="stylesheet" type="text/css" href="estilo.css" />
  </head>
  <body>
    ...
  </body>
</html>
```



Usamos o atributo **href** para definirmos o local/caminho e o nome do arquivo .css que queremos incluir em nosso documento HTML

```
.ui-helper-hidden-accessible {
border: 0;
clip: rect(0 0 0 0);
height: 1px;
margin: -1px;
overflow: hidden;
padding: 0;
position: absolute;
width: 1px;
}
```

**Instituto de Computação
UFMT 2018**

```
grep: function( elems, callback, invert ) {
    var callbackInverse,
        matches = [],
        i = 0,
        length = elems.length,
        callbackExpect = !invert;
```

Estilo inline (em linha)

- Podemos aplicar estilo diretamente aos elementos utilizando o atributo **style**, é o que chamamos de atribuição *inline*. Exemplo:

```
<strong style="color: #F00; font-weight: normal;">
    Lorem ipsum
</strong>
```

.ui-helper-hidden-accessible {
border: 0;
clip: rect(0 0 0 0);
height: 1px;
margin: -1px;
overflow: hidden;
padding: 0;
position: absolute;
width: 1px;}

```
grep: function( elems, callback, invert ) {
    var callbackInverse,
        matches = [],
        i = 0,
        elem = elems.length,
        callbackExpect = !invert;
```

Precedência de Seletores

- No caso de **vários seletores** “apontarem” para o **mesmo elemento**, o que acontece?

```
.ui-helper-hidden-accessible {
border: 0;
clip: rect(0 0 0 0);
height: 1px;
margin: -1px;
overflow: hidden;
padding: 0;
position: absolute;
width: 1px;}
```

```
grep: function( elems, callback, invert ) {  
    var callbackInverse,  
        matches = [],  
        i = 0,  
        length = elems.length,  
        callbackExpect = !invert;
```

Exemplo

```
p {  
    background-color: #CCC;  
}
```

```
p.paragrafo {  
    background-color: #0CC;  
}
```

```
p#um {  
    background-color: #922;  
}
```

Qual será a cor de fundo do parágrafo?

```
<p id="um" class="paragrafo">  
    Donec pellentesque vulputate eros  
</p>
```

```
.ui-helper-hidden-accessible {  
border: 0;  
clip: rect(0 0 0 0);  
height: 1px;  
margin: -1px;  
overflow: hidden;  
padding: 0;  
position: absolute;  
width: 1px;
```

```
grep: function( elems, callback, invert ) {  
    var callbackInverse,  
        matches = [],  
        i = 0,  
        length = elems.length,  
        callbackExpect = !invert;
```

E se...

```
p {  
    background-color: #CCC;  
}
```

```
p.paragrafo {  
    background-color: #0CC;  
}
```

```
p#um {  
    background-color: #922;  
}
```

E agora?

```
.ui-helper-hidden-accessible {  
    border: 0;  
    clip: rect(0 0 0 0);  
    height: 1px;  
    margin: -1px;  
    overflow: hidden;  
    padding: 0;  
    position: absolute;  
    width: 1px;  
  


>  
    Donec pellentesque vulputate eros  
    </p>


```

```
grep: function( elems, callback, invert ) {
    var callbackInverse,
        matches = [],
        i = 0,
        elem = elems.length,
        callbackExpect = !invert;
```

Precedência de Seletores

- Para saber o resultado, analisa-se qual seletor tem maior precedência. Do maior para o menor, temos:
 - **#id**
 - **.classe**
 - **elemento**
 - Ainda, o estilo definido *inline (style)* tem maior prioridade que os seletores.

```
.ui-helper-hidden-accessible {
border: 0;
clip: rect(0 0 0 0);
height: 1px;
margin: -1px;
overflow: hidden;
padding: 0;
position: absolute;
width: 1px;}
```

```
grep: function( elems, callback, invert ) {
    var callbackInverse,
        matches = [],
        i = 0,
        elem = elems.length,
        callbackExpect = !invert;
```

Precedência de Seletores

- Importante: podemos "alterar" a precedência de seletores utilizando o operador ***!important***, exemplo:

```
p.paragrafo {
    background-color: #0CC !important;
}
```

```
p#um {
    background-color: #922;
}
```

```
.ui-helper-hidden-accessible {
border: 0;
clip: rect(0 0 0 0);
height: 1px;
margin: -1px;
overflow: hidden;
padding: 0;
position: absolute;
width: 1px;
```

Nota: Nessa caso, seria aplicado o **estilo da classe** e não do id

```
grep: function( elems, callback, invert ) {
    var callbackInverse,
        matches = [],
        i = 0,
        elem = elems.length,
        callbackExpect = !invert;
```

Propriedades importantes

- Propriedades muito utilizadas:
 - Padding – "margem interna" do elemento
 - Margin – "margem externa" do elemento
 - Display – Define o comportamento em relação a exibição do elemento na página

```
.ui-helper-hidden-accessible {
border: 0;
clip: rect(0 0 0 0);
height: 1px;
margin: -1px;
overflow: hidden;
padding: 0;
position: absolute;
width: 1px;}
```

```
grep: function( elems, callback, invert ) {  
    var callbackInverse,  
        matches = [],  
        i = 0,  
        start = elems.length,  
        callbackExpect = !invert;
```

Um pouco sobre **border**

border é uma propriedade muitas vezes definida de forma composta, por exemplo:

1) **border**: *1px solid #FF0000*;

Porém, é possível definir a borda de um elemento utilizando três propriedades:

2) **border-width**: *1px*;
border-style: *solid*;
border-color: *#FF0000*;

```
.ui-helper-hidden-accessible {  
border: 0;  
clip: rect(0 0 0 0);  
height: 1px;  
margin: -1px;  
overflow: hidden;  
padding: 0;  
position: absolute;  
width: 1px;}
```

Nota: Os trechos de CSS 1 e 2 são semanticamente equivalentes (produzem o mesmo efeito)

Um pouco sobre border

Os dois exemplos do slide anterior aplicam as propriedades referentes a borda para todos os 4 lados de um elemento: superior, direito, inferior, esquerdo. Podemos definir, um estilo personalizado para cada um dos lados de um elemento utilizando `border-top`, `border-right`, `border-bottom` e `border-left`, exemplo:

```
border-top: 4px solid #FF0000;  
border-right: 2px dotted #0000FF;  
border-bottom: 4px dashed #b6bbff;  
border-left: none;
```

Assim como fizemos na definição de todas as bordas de uma vez utilizando a propriedade `border`, também é possível definir tamanho, estilo e cor para a borda de cada lado, utilizando, por exemplo, `border-top-width`, `border-top-style` e `border-top-color`.

Um pouco sobre **background**

A propriedade `background` é utilizada para definir estilos referente ao fundo (`background`) dos elementos. Podendo essas propriedades serem definidas de forma composta, como por exemplo:

```
background: #d1d4ff url('imagem.png') no-repeat scroll left center;
```

Que, individualmente equivale as propriedades:

- `background-color`
- `background-image`
- `background-repeat`
- `background-attachment`
- `background-position`

```
grep: function( elems, callback, invert ) {  
    var callbackInverse,  
        matches = [],  
        i = 0,  
        length = elems.length,  
        callbackExpect = !invert;
```

Pseudo-classes

- Pseudo-classe é uma **palavra chave** adicionada **junto aos seletores** CSS com o objetivo de permitir a **definição de estilos para condições (casos) específicas**, como por exemplo, quando um elemento estiver **ativo**, quando o **cursor do mouse** estiver sobre um elemento, entre outros.

Sintaxe:

```
seletor:pseudo-classe {  
    propriedade: valor;  
}
```

```
.ui-helper-hidden-accessible {  
border: 0;  
clip: rect(0 0 0 0);  
height: 1px;  
margin: -1px;  
overflow: hidden;  
padding: 0;  
position: absolute;  
width: 1px;}
```

```
grep: function( elems, callback, invert ) {
    var callbackInverse,
        matches = [],
        i = 0,
        length = elems.length,
        callbackExpect = !invert;
```

Pseudo-classes

- Exemplo:

```
div:hover {
    background-color: #008800;
}
```

<https://developer.mozilla.org/pt-BR/docs/Web/CSS/Pseudo-classes>

https://www.w3schools.com/css/css_pseudo_classes.asp

```
grep: function( elems, callback, invert ) {  
    var callbackInverse,  
        matches = [],  
        i = 0,  
        length = elems.length,  
        callbackExpect = !invert;
```

Pseudo-classes

- Algumas pseudo-classes interessantes:

:hover

Instante em que o cursor do mouse é passado sobre um elemento

:link

Pseudo classe referente aos links “dentro” de um elemento

:visited

Links já visitados

:active

Links ativos e o instante após clicar e antes de “soltar” o cursor do mouse sobre um link / botão

```
.ui-helper-hidden-accessible {  
border: 0;  
clip: rect(0 0 0 0);  
height: 1px;  
margin: -1px;  
overflow: hidden;  
padding: 0;  
position: absolute;  
width: 1px;}
```

Importante: para um funcionamento adequado (onde as propriedades não se sobrescrevam de maneira indesejada) a ordem indicada é: *:link* – *:visited* – *:hover* – *:active*
<https://developer.mozilla.org/pt-BR/docs/Web/CSS/:active>

```
grep: function( elems, callback, invert ) {
    var callbackInverse,
        matches = [],
        i = 0,
        length = elems.length,
        callbackExpect = !invert;
```

Exemplo

```
body { color: black }
a:link { color: CornflowerBlue }
a:visited { color: DarkSlateBlue }
a:hover { color: DarkOrchid }
a:active { color: GreenYellow }
```

```
<body>
  <p>
    Exemplo do comportamento do link alterado
    utilizando pseudo-classes
    <a href="pseudo-classes.html">Pseudo Classes</a>.
```

```
.ui-helper-hidden-accessible {
  border: 0;
  clip: rect(0 0 0 0);
  height: 1px;
  margin: -1px;
  overflow: hidden;
  padding: 0;
  position: absolute;
  width: 1px;
```

```
grep: function( elems, callback, invert ) {  
    var callbackInverse,  
        matches = [],  
        i = 0,  
        length = elems.length,  
        callbackExpect = !invert;
```

Pseudo-classes

- ...mais pseudo-classes interessantes:

`:first-child()`

Faz referência somente aos elementos que sejam o **primeiro filho** do seu (elemento) pai

`:last-of-type()`

Faz referência somente aos elementos que sejam o **último filho** do seu (elemento) pai

`:nth-child(n)`

Faz referência somente aos elementos que sejam ***n*-ésimo filho** filho do seu (elemento) pai

`:not(selector)`

Faz referência a “todos” os elementos com **exceção** dos elementos definidos pelo **seletor**

.ui-helper-hidden-accessible {
border: 0;
clip: rect(0 0 0 0);
height: 1px;
margin: -1px;
overflow: hidden;
padding: 0;
position: absolute;
width: 1px;

```
grep: function( elems, callback, invert ) {
    var callbackInverse,
        matches = [],
        i = 0,
        length = elems.length,
        callbackExpect = !invert;
```

exemplo

```
li:not(.do-contra) {
    color: Gainsboro;
}

li:first-child {
    color: green;
}

li:last-child {
    color: DarkRed;
}

li:nth-child(3) {
    color: DarkViolet;
}

.ui-helper-hidden-accessible {
border: 1px solid black;
clip: rect(0 0 0 0);
height: 1px;
margin: -1px;
overflow: hidden;
padding: 0;
position: absolute;
width: 1px;}
```

```
<ul>
<li>Lorem Ipsum</li>
<li>Lorem Ipsum</li>
<li>Lorem Ipsum</li>
<li>Lorem Ipsum</li>
<li class="do-contra">Lorem Ipsum</li>
<li>Lorem Ipsum</li>
</ul>
```

```
grep: function( elems, callback, invert ) {  
    var callbackInverse,  
        matches = [],  
        i = 0,  
        length = elems.length,  
        callbackExpect = !invert;
```

pseudo-elementos

- Os pseudo-elementos são **semelhantes** às **pseudo-classes**, porém, uma vez que as pseudo-classes nos permite aplicar estilos a “casos” específicos, os **pseudo-elementos** nos permite **aplicar estilos** a **partes específicas** de um elemento que **não** são/estão **explícitas** por meio de *tags*.

```
seletor::pseudo-elemento {  
    propriedade: valor;  
}
```

```
.ui-helper-hidden-accessible {  
border: 0;  
clip: rect(0 0 0 0);  
height: 1px;  
margin: 0;  
overflow: hidden;  
padding: 0;  
position: absolute;  
width: 1px;
```

Importante: a partir do CSS 2.1, padronizou-se o uso de dois pontos simples `:` para pseudo-classes e duplo `::` para pseudo-elementos.

```
grep: function( elems, callback, invert ) {
    var callbackInverse,
        matches = [],
        i = 0,
        length = elems.length,
        callbackExpect = !invert;
```

Exemplo

Considere o seguinte trecho HTML...

```
<p>Lorem ipsum dolor sit amet,  
consectetur adipiscing elit. Ut  
vestibulum sit amet massa vel pharetra.  
Maecenas purus lacus, feugiat convallis  
mauris vitae, laoreet tincidunt diam.  
Curabitur hendrerit, erat ac pretium  
venenatis, quam nisi consectetur diam,  
quis bibendum enim nunc eget lorem.  
Fusce sit amet est in massa venenatis  
euismod.</p>
```

É uma marcação simples, que define um parágrafo. Se observamos, **não existem elementos/tags**, referentes por exemplo, à **primeira linha** ou **primeira letra** do parágrafo. Todavia, **podemos aplicar estilos** tanto à “primeira linha”, como também à “primeira letra” do parágrafo **utilizando**, respectivamente, **os pseudo-elementos** `::first-line` e `::first-letter`.

Exemplo:

```
p::first-line {  
    color: red;  
}  
  
p::first-letter {  
    font-size: 20px;  
    color: blue;  
}
```

```
grep: function( elems, callback, invert ) {
  var callbackInverse,
      matches = [],
      i = 0,
      l = elems.length,
      callbackExpect = !invert;
```

Outros pseudo-elementos...

- Entre outros, temos dois pseudo-elementos muito interessantes, o **::before** e o **::after**, que são utilizados para, principalmente, adicionarmos conteúdo (por meio da propriedade **content**), respectivamente, antes e depois dos elementos.
 - Podemos, por exemplo, adicionar contadores e valor de atributos por meio das funções **counter()** e **attr()**

<https://developer.mozilla.org/en-US/docs/Web/CSS/::after>

<https://developer.mozilla.org/en-US/docs/Web/CSS/::before>

<https://developer.mozilla.org/pt-BR/docs/Web/CSS/content>

https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Lists_and_Counters/Using_CSS_counters

```
grep: function( elems, callback, invert ) {
    var callbackInverse,
        matches = [],
        i = 0,
        length = elems.length,
        callbackExpect = !invert;
```

Exercício

- Aplique estilo a todos elementos da primeira página do exercício sobre HTML:
 - Deve haver pelo menos um seletor de elemento, um de classe e um id.
 - Aplicar **propriedades de borda e fundo** a cada um dos 4 "blocos" para **diferenciá-los / destacá-los**
 - Aplicar estilos de formatação de texto

```
.ui-helper-hidden-accessible {
border: 0;
clip: rect(0 0 0 0);
height: 1px;
margin: -1px;
overflow: hidden;
padding: 0;
position: absolute;
width: 1px;}
```

```
grep: function( elems, callback, invert ) {
    var callbackInverse,
        matches = [],
        i = 0,
        length = elems.length,
        callbackExpect = !invert;
```

Posicionamento de elementos usando CSS

```
.ui-helper-hidden-accessible {
border: 0;
clip: rect(0 0 0 0);
height: 1px;
margin: -1px;
overflow: hidden;
padding: 0;
position: absolute;
width: 1px;}
```

Instituto de Computação
UFMT 2018

Futxicaia da Tecnológica
Projeto de Extenção

```
grep: function( elems, callback, invert ) {
    var callbackInverse,
        matches = [],
        i = 0,
        elemLength = elems.length,
        callbackExpect = !invert;
```

Posicionamento de Elementos

- Posicionar elementos utilizando CSS é uma tarefa rotineira e importante para todo desenvolvedor web
- Diferentes propriedades são utilizadas em CSS para definirmos o posicionamento dos elementos, a citar: margin, position, float, top, bottom, left, right.

```
.ui-helper-hidden-accessible {
border: 0;
clip: rect(0 0 0 0);
height: 1px;
margin: -1px;
overflow: hidden;
padding: 0;
position: absolute;
width: 1px;}
```

```
grep: function( elems, callback, invert ) {  
    var callbackInverse,  
        matches = [],  
        i = 0,  
        length = elems.length,  
        callbackExpect = !invert;
```

Propriedade *margin*

- A propriedade margin define a distância do elemento em relação ao elemento pai.

Pode ser definida de forma composta, exemplo:

margin: 8px 2px 4px 8px;

- Nesse caso, estaríamos definindo 8px para a **margem superior**, 2px para a margem **direita**, 4px para a **margem inferior** e 8px para a margem **direita**.

```
.ui-helper-hidden-accessible {  
border: 0;  
clip: rect(0 0 0 0);  
height: 1px;  
margin: -1px;  
overflow: hidden;  
padding: 0;  
position: absolute;  
width: 1px;
```

```
grep: function( elems, callback, invert ) {
    var callbackInverse,
        matches = [],
        i = 0,
        length = elems.length,
        callbackExpect = !invert;
```

Propriedade *margin*

- Ou, também, definida individualmente para cada margem do elemento:

```
margin-top: 8px;
margin-right: 2px;
margin-bottom: 4px;
margin-left: 8px;
```

```
.ui-helper-hidden-accessible {
border: 0;
clip: rect(0 0 0 0);
height: 1px;
margin: -1px;
overflow: hidden;
padding: 0;
position: absolute;
width: 1px;}
```

```
grep: function( elems, callback, invert ) {  
    var callbackInverse,  
        matches = [],  
        i = 0,  
        length = elems.length,  
        callbackExpect = !invert;
```

Propriedade *margin*

- Além da definição composta da propriedade margin apresentada no **slide 3**, a propriedade também aceita as seguintes definições:

`margin: 8px;`

`margin: 8px 4px;`

`margin: 8px 4px 2px;`

- **Primeiro caso:** 8px é atribuído para **todas** as margens
- **Segundo caso:** 8px é atribuído as margens **superiores e inferiores** e 4px as margens **direita e esquerda**.
- **Terceiro caso:** 8px é atribuído a margem superior, 4px as margens **direita e esquerda** e 2px a margem inferior.

```
grep: function( elems, callback, invert ) {  
    var callbackInverse,  
        matches = [],  
        i = 0,  
        len = elems.length,  
        callbackExpect = !invert;
```

Tamanho dos elementos

- Agora que conhecemos as propriedades **border**, **padding** e **margin** podemos entender bem o tamanho dos elementos em CSS.
 - Basicamente, definimos a **largura** (ou o tamanho na horizontal) utilizando a propriedade **width** e a **altura** (ou o tamanho na vertical) utilizando a propriedade **height**, exemplo:

```
div#bloco {  
    width: 150px;  
    height: 300px;
```

Todavia, outras propriedades como **border**, **padding** e **margin** também podem influenciar no tamanho final do elemento que será renderizado pelo navegador

```
grep: function( elems, callback, invert ) {
    var callbackInverse,
        matches = [],
        i = 0,
        len = elems.length,
        callbackExpect = !invert;
```

Tamanho dos elementos

- **Exemplo.** Seja um elemento com as seguintes propriedades:

```
div#bloco {
    border: 2px solid #ccc;
    margin: 20px;
    padding: 10px;
    width: 300px;
    height: 150px;
}
```

Se perguntarmos “qual o tamanho do elemento que recebe essas propriedades?”, a primeira vista a resposta padrão seria $300px \times 150px$. Porém, as margens (interna e externa) e as bordas **por padrão**, influenciam no tamanho do elemento. Dessa forma, temos $324px \times 174px$.

```
grep: function( elems, callback, invert ) {  
    var callbackInverse,  
        matches = [],  
        i = 0,  
        len = elems.length,  
        callbackExpect = !invert;
```

Tamanho dos elementos

- Se **não quisermos** que outras medidas como margem e borda influenciem no tamanho dos elementos, podemos utilizar a **propriedade *box-sizing*** com o valor **border-box** (o padrão é **content-box**):

```
div#bloco {  
    border : 2px solid #ccc;  
    box-sizing: border-box;  
    padding: 10px;  
    width: 300px;  
    height: 150px;  
}  
.ui-helper-hidden-accessible {  
border: 0;  
clip: rect(0 0 0 0);  
height: 1px;  
margin: -1px;  
overflow: hidden;  
padding: 0;  
position: absolute;  
width: 1px;
```

```
grep: function( elems, callback, invert ) {
    var callbackInverse,
        matches = [],
        i = 0,
        length = elems.length,
        callbackExpect = !invert;
```

Propriedade *position*

- Define o tipo de posicionamento que o elemento terá.
 - Aplica-se a praticamente todos os objetos HTML
 - O valor padrão para todos os elementos é **static**
 - Os demais valores possíveis são: **absolute**, **relative**, **fixed**.

```
.ui-helper-hidden-accessible {
border: 0;
clip: rect(0 0 0 0);
height: 1px;
margin: -1px;
overflow: hidden;
padding: 0;
position: absolute;
width: 1px;}
```

```
grep: function( elems, callback, invert ) {
    var callbackInverse,
        matches = [],
        i = 0,
        elemLength = elems.length,
        callbackExpect = !invert;
```

position: **absolute**

- O elemento é posicionado de maneira “absoluta” (fora do fluxo de elementos) por meio das propriedades: top, right, bottom, left.
 - Que definem as distâncias superior, direita, inferior e esquerda em relação ao elemento pai.

```
.ui-helper-hidden-accessible {
border: 0;
clip: rect(0 0 0 0);
height: 1px;
margin: -1px;
overflow: hidden;
padding: 0;
position: absolute;
width: 1px;}
```

```
grep: function( elems, callback, invert ) {
    var callbackInverse,
        matches = [],
        i = 0,
        elemMatch = elems.length,
        callbackExpect = !invert;
```

position: *absolute*

- Exemplo:

```
<style>
  div#um {
    height: 160px;
    width: 160px;
    border: 1px solid #FF0000;
    background-color: #FFB6B6;
    position: absolute;
    left: 150px;
    top: 15px;
  }
</style>
<div id="um">
  &nbsp;
</div>
```

```
.ui-helper-hidden-accessible {
border: 0;
clip: rect(0 0 0 0);
height: 1px;
margin: -1px;
overflow: hidden;
padding: 0;
position: absolute;
width: 1px;
```

```
grep: function( elems, callback, invert ) {
    var callbackInverse,
        matches = [],
        i = 0,
        elemLength = elems.length,
        callbackExpect = !invert;
```

position: *absolute*

Resultado esperado

`.ui-helper-hidden-accessible {
 border: 0;
 clip: rect(0 0 0 0);
 height: 1px;
 margin: -1px;
 overflow: hidden;
 padding: 0;
 position: absolute;
 width: 1px;`



Ello. Etiam laoreet metus nec feugiat cursus.
quis molestie nunc neque eget mauris. Etiam f
ne. Nunc iaculis, diam vel sagittis pharetra, le

`.ui-helper-hidden-accessible {
 border: 0;
 clip: rect(0 0 0 0);
 height: 1px;
 margin: -1px;
 overflow: hidden;
 padding: 0;
 position: absolute;
 width: 1px;`

Instituto de Computação
UFMT 2018

Futxicaia da Tecnológica
Projeto de Extenção

```
grep: function( elems, callback, invert ) {
    var callbackInverse,
        matches = [],
        i = 0,
        elemLength = elems.length,
        callbackExpect = !invert;
```

position: **absolute**

- A primeira vista pode-se não notar a diferença em relação ao uso da propriedade margin, porém, é importante saber que:
 - As propriedades top, right, bottom, left não possuem efeito em elementos com positon: static (valor padrão)
 - E principalmente: Um elemento com posição absoluta se “desprende” do fluxo normal de elementos, sobrepondo-se sobre os demais.

```
.ui-helper-hidden-accessible {
border: 0;
clip: rect(0 0 0 0);
height: 1px;
margin: -1px;
overflow: hidden;
padding: 0;
position: absolute;
width: 1px;
```

```
grep: function( elems, callback, invert ) {
    var callbackInverse,
        matches = [],
        i = 0,
        elemMatch = elems.length,
        callbackExpect = !invert;
```

position: absolute

Exemplo:

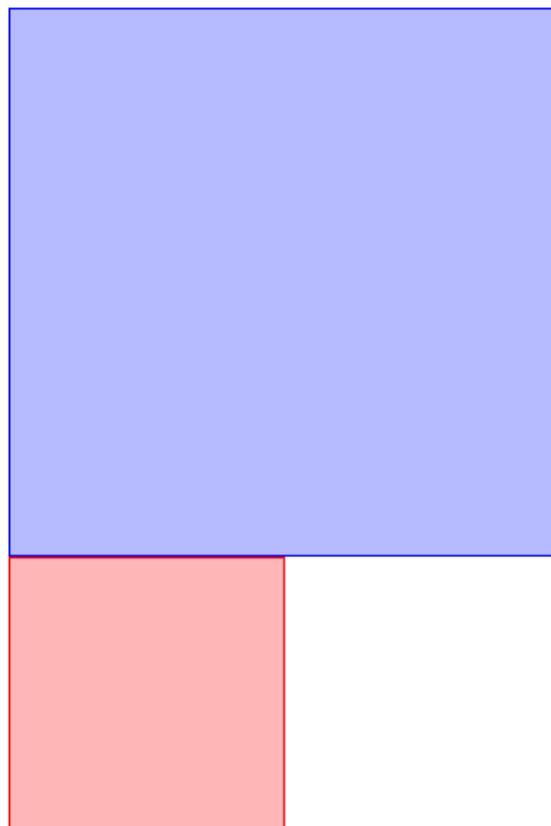
```
<style>
    div#um {
        background-color: #b6bbff;
        height: 300px;
        width: 300px;
        border: 1px solid #0000FF;
        position: static;
    }
    div#dois {
        background-color: #ffb6b6;
        height: 150px;
        width: 150px;
        border: 1px solid #FF0000;
        position: static;
        left: 80px;
        top: 120px;
    }
</style>
<div id="um">&nbsp;</div>
<div id="dois">&nbsp;</div>
```

.ui-helper-hidden-accessible
border: 0;
clip: rect(0 0 0 0);
height: 1px;
margin: -1px;
overflow: hidden;
padding: 0;
position: absolute;
width: 1px;

```
grep: function( elems, callback, invert ) {
    var callbackInverse,
        matches = [],
        i = 0,
        elemMatch = elems.length,
        callbackExpect = !invert;
```

position: absolute

Resultado esperado:



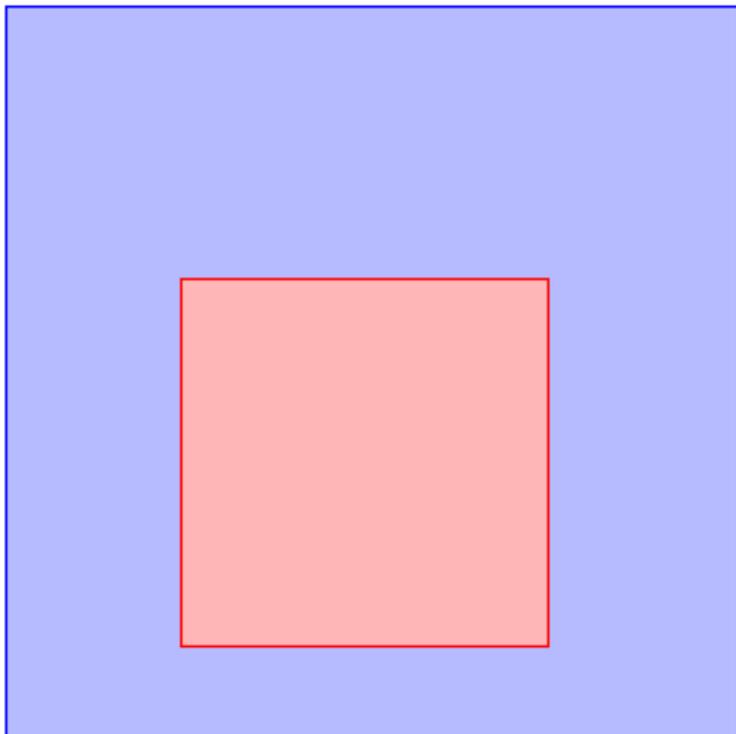
```
.ui-helper-hidden-accessible {
border: 0;
clip: rect(0 0 0 0);
height: 1px;
margin: -1px;
overflow: hidden;
padding: 0;
position: absolute;
width: 1px;}
```

**Instituto de Computação
UFMT 2018**

**Futxicaia da Tecnológica
Projeto de Extenção**

position: *absolute*

Alterando a position do div.dois para absolute, o resultado é consideravelmente diferente:



Importante: Na realidade, o div.dois **não** está “dentro” do div.um e sim “sobre”.

```
grep: function( elems, callback, invert ) {  
    var callbackInverse,  
        matches = [],  
        i = 0,  
        length = elems.length,  
        callbackExpect = !invert;
```

z-index

- A propriedade z-index é utilizada para definir a disposição de um elemento sobre outro. Em suma, qual estará afrente, e qual estará atrás.

```
.ui-helper-hidden-accessible {  
border: 0;  
clip: rect(0 0 0 0);  
height: 1px;  
margin: -1px;  
overflow: hidden;  
padding: 0;  
position: absolute;  
width: 1px;}
```

```
grep: function( elems, callback, invert ) {
    var callbackInverse,
        matches = [],
        i = 0,
        length = elems.length,
        callbackExpect = !invert;
```

z-index

Vamos adicionar mais um elemento ao nosso exemplo:

```
<style>
  div#tres {
    background-color: #c9ffd8;
    height: 50px;
    width: 350px;
    border: 1px solid #005719;
    position: absolute;
    left: 0px;
    top: 160px;
  }
</style>
<div id="tres">
  &nbsp;
</div>
```

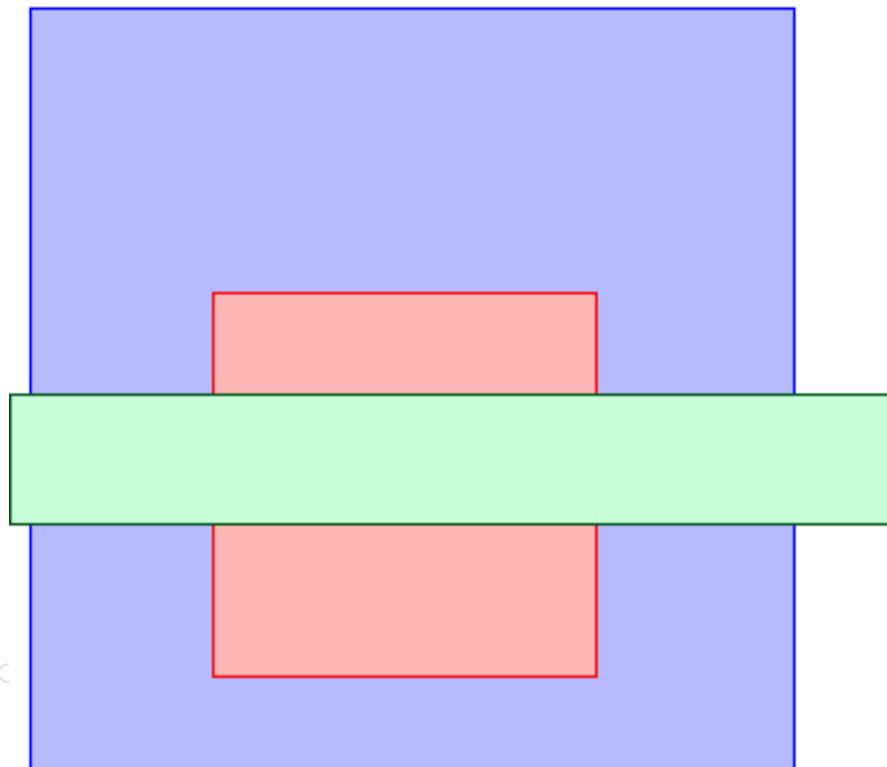
```
.ui-helper-hidden-accessible {
border: 0;
clip: rect(0 0 0 0);
height: 1px;
margin: -1px;
overflow: hidden;
padding: 0;
position: absolute;
width: 1px;
```

**Instituto de Computação
UFMT 2018**

```
grep: function( elems, callback, invert ) {
  var callbackInverse,
      matches = [],
      i = 0,
      length = elems.length,
      callbackExpect = !invert;
```

z-index

Resultado:

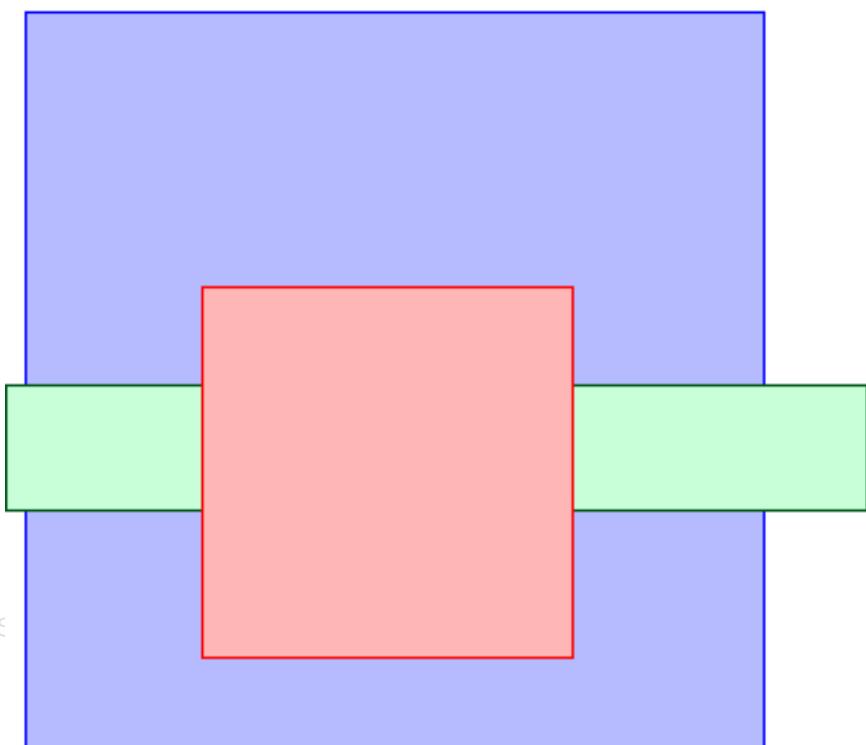


```
.ui-helper-hidden-accessible
border: 0;
clip: rect(0 0 0 0);
height: 1px;
margin: -1px;
overflow: hidden;
padding: 0;
position: absolute;
width: 1px;
```

```
grep: function( elems, callback, invert ) {  
    var callbackInverse,  
        matches = [],  
        i = 0,  
        length = elems.length,  
        callbackExpect = !invert;
```

z-index

Se adicionarmos a propriedade `z-index` para o `div.tres` com o **valor 1** e para o `div.dois` com o **valor 2**, o resultado será:



```
.ui-helper-hidden-accessible  
border: 0;  
clip: rect(0 0 0 0);  
height: 1px;  
margin: -1px;  
overflow: hidden;  
padding: 0;  
Instituto de Computação  
UFMT 2018  
position: absolute;  
width: 1px;
```

```
grep: function( elems, callback, invert ) {
    var callbackInverse,
        matches = [],
        i = 0,
        length = elems.length,
        callbackExpect = !invert;
```

position: **absolute**

- Uma característica importante da propriedade **absolute** (relative também) é que os **aninhamentos** que criamos entre os elementos são mantidos / respeitados

```
.ui-helper-hidden-accessible {
border: 0;
clip: rect(0 0 0 0);
height: 1px;
margin: -1px;
overflow: hidden;
padding: 0;
position: absolute;
width: 1px;}
```

```
grep: function( elems, callback, invert ) {
    var callbackInverse,
        matches = [],
        i = 0,
        elemMatch = elems.length,
        callbackExpect = !invert;
```

position: *absolute*

Exemplo de elementos com **position: absolute** aninhados:

CSS

```
div#um {
    background-color: #ffb6b6;
    height: 150px;
    width: 150px;
    border: 1px solid #FF0000;
    position: absolute;
    left: 40px;
    top: 40px;
}
div#dois {
    background-color: #c9ffd8;
    height: 50px;
    width: 50px;
    border: 1px solid #005719;
    position: absolute;
    right: 0px;
    bottom: 0px;
```

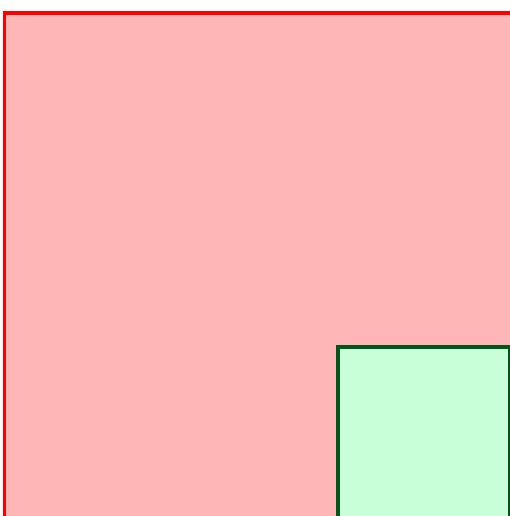
HTML

```
<div id="um">
    <div id="dois">
        &nbsp;
    </div>
</div>
```

```
grep: function( elems, callback, invert ) {  
    var callbackInverse,  
        matches = [],  
        i = 0,  
        elemLength = elems.length,  
        callbackExpect = !invert;
```

position: absolute

Resultado:



A “lógica” ainda é a mesma: As propriedades top, right, bottom, left são em relação aos elementos pai. No caso do div.dois o elemento pai é o div.um e no caso do div.um o elemento pai é o body.

Importante: No caso de elementos aninhados, a propriedade z-index não tem funcionalidade.

```
.ui-helper-hidden-accessible {  
border: 0;  
clip: rect(0 0 0 0);  
height: 1px;  
margin: -1px;  
overflow: hidden;  
padding: 0;  
position: absolute;  
width: 1px;}
```

```
grep: function( elems, callback, invert ) {
    var callbackInverse,
        matches = [],
        i = 0,
        length = elems.length,
        callbackExpect = !invert;
```

position: **relative**

- O valor **relative** para a propriedade position adiciona um comportamento análogo ao position **absolute**.
 - Todavia, as propriedades referentes a posicionamento (top, right, bottom, left) são **relativas** a posição do elemento no fluxo normal de elementos na página.

```
.ui-helper-hidden-accessible {
border: 0;
clip: rect(0 0 0 0);
height: 1px;
margin: -1px;
overflow: hidden;
padding: 0;
position: absolute;
width: 1px;}
```

```
grep: function( elems, callback, invert ) {
  var callbackInverse,
      matches = [],
      i = 0,
      length = elems.length,
      callbackExpect = !invert;
```

position: *relative*

1. HTML

```
<div id="um">
  &nbsp;
</div>
<div id="dois">
  &nbsp;
</div>
```

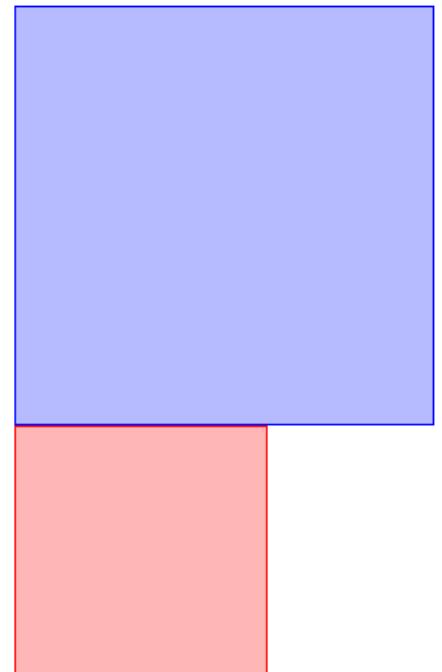
2. CSS

```
<style>
  div#um {
    background-color: #b6bbff;
    height: 250px;
    width: 250px;
    border: 1px solid #0000FF;
    position: static;
  }
  div#dois {
    background-color: #ffb6b6;
    height: 150px;
    width: 150px;
    border: 1px solid #FF0000;
    position: static;
  }
</style>
```

```
.ui-helper-hidden-accessible {
border: 0;
clip: rect(0 0 0 0);
height: 1px;
margin: -1px;
overflow: hidden;
padding: 0;
position: absolute;
width: 1px;
```

Instituto de Computação
UFMT 2018

3. Resultado



```
grep: function( elems, callback, invert ) {
  var callbackInverse,
      matches = [],
      i = 0,
      length = elems.length,
      callbackExpect = !invert;
```

position: *relative*

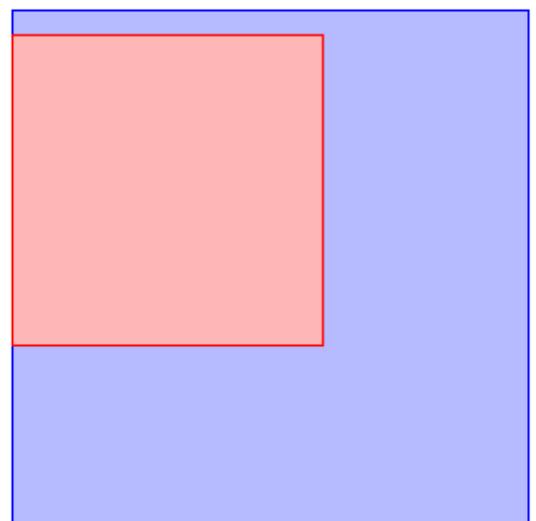
1. HTML

```
<div id="um">
  &nbsp;
</div>
<div id="dois">
  &nbsp;
</div>
```

2. CSS

```
<style>
  div#um {
    background-color: #b6bbff;
    height: 250px;
    width: 250px;
    border: 1px solid #0000FF;
    position: static;
  }
  div#dois {
    background-color: #ffb6b6;
    height: 150px;
    width: 150px;
    border: 1px solid #FF0000;
    position: absolute;
    top: 20px;
  }
</style> {
```

3. Resultado



```
grep: function( elems, callback, invert ) {
  var callbackInverse,
      matches = [],
      i = 0,
      length = elems.length,
      callbackExpect = !invert;
```

position: *relative*

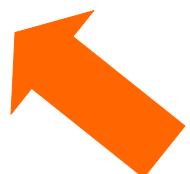
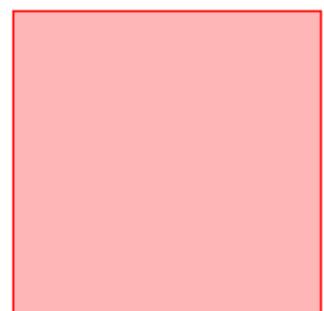
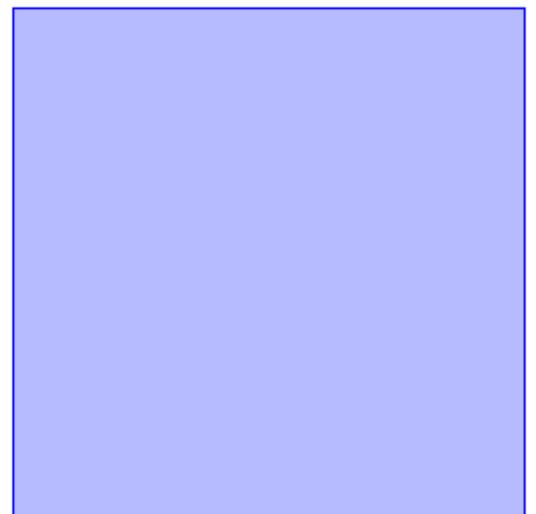
1. HTML

```
<div id="um">
  &nbsp;
</div>
<div id="dois">
  &nbsp;
</div>
```

2. CSS

```
<style>
  div#um {
    background-color: #b6bbff;
    height: 250px;
    width: 250px;
    border: 1px solid #0000FF;
    position: static;
  }
  div#dois {
    background-color: #ffb6b6;
    height: 150px;
    width: 150px;
    border: 1px solid #FF0000;
    position: relative;
    top: 20px;
  }
</style>
```

3. Resultado



```
grep: function( elems, callback, invert ) {
    var callbackInverse,
        matches = [],
        i = 0,
        length = elems.length,
        callbackExpect = !invert;
```

*position: **fixed***

- O valor **fixed** para a propriedade position torna o elemento fixo à uma determinada posição na página mesmo quando a barra de rolagem é acionada.
 - A posição fixa do elemento é dada pelas propriedades top, right, bottom, left, cuja os valores acompanham a rolagem da barra de rolagem.

```
.ui-helper-hidden-accessible {
border: 0;
clip: rect(0 0 0 0);
height: 1px;
margin: -1px;
overflow: hidden;
padding: 0;
position: absolute;
width: 1px;}
```

```
grep: function( elems, callback, invert ) {
  var callbackInverse,
      matches = [],
      i = 0,
      length = elems.length,
      callbackExpect = !invert;
```

float

- Faz com que elementos flutuem para direita (right) ou para esquerda (left).
 - Essa propriedade é muito útil quando queremos alinhar horizontalmente elementos que são exibidos como **block**, como por exemplo, divs.

```
.ui-helper-hidden-accessible {
border: 0;
clip: rect(0 0 0 0);
height: 1px;
margin: -1px;
overflow: hidden;
padding: 0;
position: absolute;
width: 1px;}
```

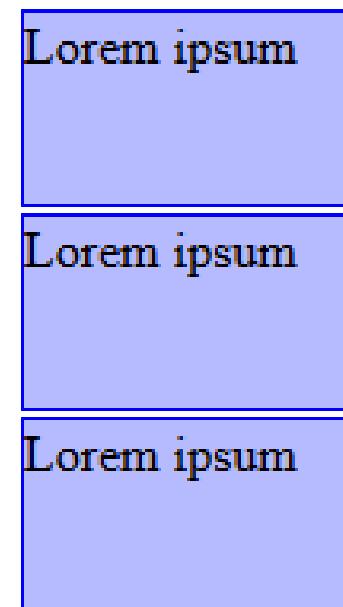
```
grep: function( elems, callback, invert ) {  
    var callbackInverse,  
        matches = [],  
        i = 0,  
        length = elems.length,  
        callbackExpect = !invert;
```

float

```
<style>  
    div.exemplo {  
        background-color: #b6bbff;  
        border: 1px solid #0000FF;  
        height: 60px;  
        width: 100px;  
        margin: 2px;  
    }  
</style>
```

```
<div class="exemplo">Lorem ipsum</div>  
<div class="exemplo">Lorem ipsum</div>  
<div class="exemplo">Lorem ipsum</div>
```

Resultado



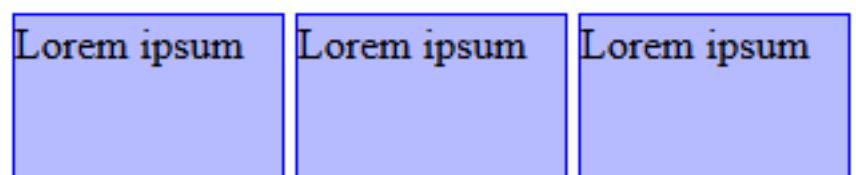
```
.ui-helper-hidden-accessible {  
    border: 0;  
    clip: rect(0 0 0 0);  
    height: 1px;  
    margin: -1px;  
    overflow: hidden;  
    padding: 0;  
    position: absolute;  
    width: 1px;
```

*Comportamento tradicional de elementos do tipo **block***

```
grep: function( elems, callback, invert ) {  
    var callbackInverse,  
        matches = [],  
        i = 0,  
        length = elems.length,  
        callbackExpect = !invert;
```

float

```
<style>  
    div.exemplo {  
        background-color: #b6bbff;  
        border: 1px solid #0000FF;  
        height: 60px;  
        width: 100px;  
        margin: 2px;  
        float: left;  
    }  
</style>
```



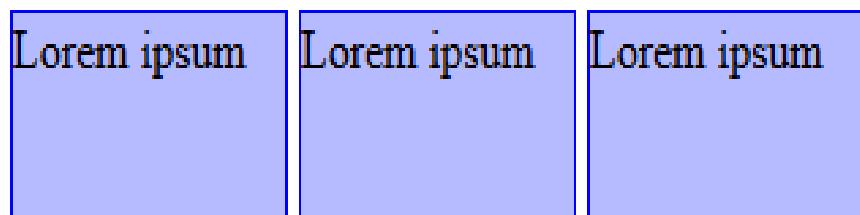
Resultado

```
<div class="exemplo">Lorem ipsum</div>  
<div class="exemplo">Lorem ipsum</div>  
<div class="exemplo">Lorem ipsum</div>  
border: none;  
clip: rect(0 0 0 0);  
height: 1px;  
margin: -1px;  
overflow: hidden;  
padding: 0;  
position: absolute;  
width: 1px;
```

```
grep: function( elems, callback, invert ) {  
    var callbackInverse,  
        matches = [],  
        i = 0,  
        length = elems.length,  
        callbackExpect = !invert;
```

float

Importante: Quando o navegador encontra um elemento com a propriedade **float**, ele “tenta alinhar” os demais elementos na sequência:



...
Lorem ipsum dolor sit amet, consectetur adipiscing et
dolor ultricies elementum. Donec sed felis metus. Nam
habitasse platea dictumst. Class aptent taciti sociosq;
gravida at magna. Nam malesuada turpis a diam acci



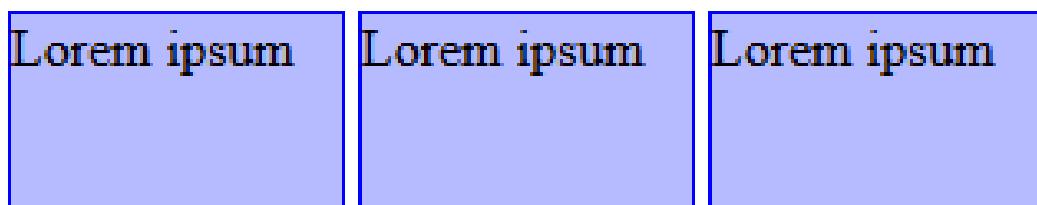
```
.ui-helper-hidden-accessible {  
border: 0;  
clip: rect(0 0 0 0);  
height: 1px;  
margin: -1px;  
overflow: hidden;  
padding: 0;  
position: absolute;  
width: 1px;}
```

```
grep: function( elems, callback, invert ) {  
    var callbackInverse,  
        matches = [],  
        i = 0,  
        length = elems.length,  
        callbackExpect = !invert;
```

float

Para que não aconteça, deve-se adicionar a propriedade clear ao elemento subsequente:

```
<p style="clear: both;">...</p>
```



Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nam metus. Nam ut rhoncus purus, vitae blandit arcu. Proin sollic per inceptos himenaeos. Vivamus lorem est, euismod et iacu auctor.

```
.ui-helper-hidden-accessible:  
border: 0;  
clip: rect(0 0 0 0);  
height: 1px;  
margin: -1px;  
overflow: hidden;  
padding: 0;  
position: absolute;  
width: 1px;
```

Instituto de Computação
UFMT 2018

```
grep: function( elems, callback, invert ) {
    var callbackInverse,
        matches = [],
        i = 0,
        length = elems.length,
        callbackExpect = !invert;
```

inline-block

- Conforme vimos, divs e parágrafos (p) são elementos com display do tipo *block* que "bloqueam" uma "linha" do fluxo do nosso documento, jogando o fluxo para a linha debaixo. Por sua vez, span são blocos do tipo *inline* que não nos permite definir o seu tamanho (largura e altura) mas que por sua vez, são posicionados um ao lado do outro.
- Para unir o "melhor desses dois mundos" existe o valor ***inline-block*** para a propriedade **display**.
- **Blocos do tipo *inline-block*** nos permite alterar sua largura **e altura** e por padrão são posicionados um ao lado do outro.

```
grep: function( elems, callback, invert ) {
    var callbackInverse,
        matches = [],
        i = 0,
        length = elems.length,
        callbackExpect = !invert;
```

table

- Ainda, se desejarmos que os elementos sejam definidos de maneira semelhante a tabelas, podemos utilizar **display: table**, **display: table-row** e **display: table-cell**

```
.ui-helper-hidden-accessible {
border: 0;
clip: rect(0 0 0 0);
height: 1px;
margin: -1px;
overflow: hidden;
padding: 0;
position: absolute;
width: 1px;}
```

```
grep: function( elems, callback, invert ) {
  var callbackInverse,
      matches = [],
      i = 0,
      length = elems.length,
      callbackExpect = !invert;
```

Por fim...

- ...a propriedade display é uma das mais completas e complexas do CSS, principalmente no que tange a exibição e posicionamento de elementos, vale a pena um estudo mais aprofundado dessa propriedade. Obs: é muito importante conhecermos o valor padrão da propriedade para os elementos que utilizamos comumente.

```
.ui-helper-hidden-accessible {
  border: 0;
  clip: rect(0 0 0 0);
  height: 1px;
  margin: -1px;
  overflow: hidden;
  padding: 0;
  position: absolute;
  width: 1px;}
```

```
grep: function( elems, callback, invert ) {  
    var callbackInverse,  
        matches = [],  
        i = 0,  
        length = elems.length,  
        callbackExpect = !invert;
```

Exercício

- Vamos construir um layout de largura fixa (980px) com uma barra para identificação (cabeçalho), menu horizontal, três colunas e rodapé.

```
.ui-helper-hidden-accessible {  
border: 0;  
clip: rect(0 0 0 0);  
height: 1px;  
margin: -1px;  
overflow: hidden;  
padding: 0;  
position: absolute;  
width: 1px;}
```

```
grep: function( elems, callback, invert ) {  
  var callbackInverse,  
      matches = [],  
      i = 0,  
      elem;  
  if( !elems.length,  
      callbackInvert = !invert;  
  ...
```

Ainda sobre posicionamento...

- Está sendo incluída na especificação do HTML/CSS os sistemas (nativos dos navegadores) de **flexbox** e **grid** que visam atender anseios dos desenvolvedores referentes ao posicionamento de elementos, deixo alguns links para quem tiver interesse:

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

<https://css-tricks.com/getting-started-css-grid/>

```
.ui-helper-hidden-accessible {  
border: 1px solid transparent;  
clip: rect(0 0 0 0);  
height: 1px;  
margin: -1px;  
overflow: hidden;  
padding: 0;  
position: absolute;  
width: 1px;
```

Instituto de Computação
UFMT 2018

```
grep: function( elems, callback, invert ) {
    var callbackInverse,
        matches = [],
        i = 0,
        length = elems.length,
        callbackExpect = !invert;
```

Tags Semânticas

- Agora que conhecemos um pouco mais sobre algumas posições de elementos utilizando HTML/CSS, é importante conhecer algumas "tags semânticas" introduzidas no **HTML5**.

```
.ui-helper-hidden-accessible {
border: 0;
clip: rect(0 0 0 0);
height: 1px;
margin: -1px;
overflow: hidden;
padding: 0;
position: absolute;
width: 1px;}
```

```
grep: function( elems, callback, invert ) {
  var callbackInverse,
      matches = [],
      i = 0,
      length = elems.length,
      callbackExpect = !invert;
```

HTML5

- O que conhecemos por HTML5 é a quinta, e atualmente última especificação da linguagem de marcação HTML.
 - É possível encontrar textos/documentos/livros que utilizam o termo HTML5 para se referirem ao conjunto:
 - HTML5+CSS3+JavaScript+Bibliotecas JS.

```
.ui-helper-hidden-accessible {
border: 0;
clip: rect(0 0 0 0);
height: 1px;
margin: -1px;
overflow: hidden;
padding: 0;
position: absolute;
width: 1px;}
```

```
grep: function( elems, callback, invert ) {
    var callbackInverse,
        matches = [],
        i = 0,
        length = elems.length,
        callbackExpect = !invert;
```

HTML5

- O HTML5 introduziu um maior número de “elementos semânticos” à linguagem HTML, como por exemplo:
 - O que seria um “elemento semântico” ?
 - É aquele elemento que possui um significado e consequentemente uma utilização explícita, que devem ser obedecidos, como por exemplo: **section, header, article.**

```
.ui-helper-hidden-accessible {
border: 0;
clip: rect(0 0 0 0);
height: 1px;
margin: -1px;
overflow: hidden;
padding: 0;
position: absolute;
width: 1px;
```

```
grep: function( elems, callback, invert ) {  
    var callbackInverse,  
        matches = [],  
        i = 0,  
        length = elems.length,  
        callbackExpect = !invert;
```

HTML5

- Por que “um maior número” ?
 - Porque já haviam elementos semânticos no HTML4, por exemplo: *form*, *img*, *table*, entre outros.
- Então quais elementos não são semânticos?
 - Elementos como *div* e *span* não possuem nenhuma semântica enquanto elementos como *p*, *b*, *i*, *u* possuem pouca semântica.

```
.ui-helper-hidden-accessible {  
border: 0;  
clip: rect(0 0 0 0);  
height: 1px;  
margin: -1px;  
overflow: hidden;  
padding: 0;  
position: absolute;  
width: 1px;}
```

```
grep: function( elems, callback, invert ) {
    var callbackInverse,
        matches = [],
        i = 0,
        length = elems.length,
        callbackExpect = !invert;
```

HTML5

- Por que deve-se dar preferência a utilização de tags semânticas?
 - Auxiliam na organização e entendimento do documento
 - Facilitam o trabalho de diferentes dispositivos, agentes autônomos e mesmo navegadores no acesso, renderizem e manipulem dos conteúdos e elementos de um documento HTML de acordo com as diferentes necessidades.

```
.ui-helper-hidden-accessible {
border: 0;
clip: rect(0 0 0 0);
height: 1px;
margin: -1px;
overflow: hidden;
padding: 0;
position: absolute;
width: 1px;}
```

```
grep: function( elems, callback, invert ) {
  var callbackInverse,
      matches = [],
      i = 0,
      length = elems.length,
      callbackExpect = !invert;
```

HTML 5

<article> : Define um conteúdo que geralmente contém textos e imagens, como por exemplo, uma postagem, uma notícia, um comentário, etc.
<https://developer.mozilla.org/pt-BR/docs/Web/HTML/Element/article>

<aside> : Define um conteúdo adjacente / ao lado de outro conteúdo.
<https://developer.mozilla.org/pt-BR/docs/Web/HTML/Element/aside>

<details> / **<summary>** : Esse conjunto descreve um conteúdo com comportamento ocultar / mostrar. O conteúdo da tag *summary* será o conteúdo sempre visível. **Importante:** observar a compatibilidade dos navegadores.
<https://developer.mozilla.org/pt-BR/docs/Web/HTML/Element/details>

<figure> / **<figcaption>** : Encapsula imagem e adiciona uma legenda com **figcaption**
<https://developer.mozilla.org/pt-BR/docs/Web/HTML/Element/figura>

```
grep: function( elems, callback, invert ) {
  var callbackInverse,
      matches = [],
      i = 0,
      length = elems.length,
      callbackExpect = !invert;
```

HTML5

<footer> : Conteúdo referente ao rodapé de um documento ou seção.
<https://developer.mozilla.org/pt-BR/docs/Web/HTML/Element/footer>

<header> : Conteúdo referente ao cabeçalho de um documento ou seção.
<https://developer.mozilla.org/pt-BR/docs/Web/HTML/Element/header>

<main> : Tag que “encapsula” o conteúdo principal de um documento HTML.
Importante: observar a compatibilidade dos navegadores.
<https://developer.mozilla.org/pt-BR/docs/Web/HTML/Element/main>

<mark> : Tag para destacar (highlight) um trecho de texto.
<https://developer.mozilla.org/pt-BR/docs/Web/HTML/Element/mark>

<nav> : Define um conjunto de links de navegação (ou menus).
<https://developer.mozilla.org/pt-BR/docs/Web/HTML/Element/nav>

```
grep: function( elems, callback, invert ) {
  var callbackInverse,
      matches = [],
      i = 0,
      length = elems.length,
      callbackExpect = !invert;
```

HTML5

<section> : Define uma seção do documento. Geralmente uma seção encapsula cabeçalhos, menus, artigos, etc.
<https://developer.mozilla.org/pt-BR/docs/Web/HTML/Element/section>

<time> : Informações referentes a data e hora, como por exemplo, a data de criação / publicação de um artigo.
<https://developer.mozilla.org/pt-BR/docs/Web/HTML/Element/time>

<adrees> : Utilizado para fornecer informações de contato. Muitas vezes adicionada a um **article** e/ou no **body**
<https://developer.mozilla.org/pt-BR/docs/Web/HTML/Element/address>

```
.ui-helper-hidden-accessible {
border: 0;
clip: rect(0 0 0 0);
height: 1px;
margin: -1px;
overflow: hidden;
padding: 0;
position: absolute;
width: 1px;
```

```
grep: function( elems, callback, invert ) {  
    var callbackInverse,  
        matches = [],  
        i = 0,  
        length = elems.length,  
        callbackExpect = !invert;
```

HTML5

- A utilização das tags introduzidas pelo HTML5 devem ser capaz de diminuir a “***div-dependência***” que “contaminou” muitos desenvolvedores web.
 - Mas isso não significa que você “nunca mais” irá utilizar tags “não semânticas”, exemplo ***div*** e ***span***.
 - Muitas vezes podem ser úteis para criarmos *containers* ou ainda conseguirmos um posicionamento um posicionamento específicos dos elementos e nosso documento HTML.

HTML5 não é apenas tags semânticas

- Além das tags semânticas e de novas funcionalidades referentes a formulários, também foi incluído no HTML 5 alguns recursos/APIs como:
 - APIs de storage: Web Storage, (**localStorage** e **sessionStorage**), Indexed Database (Indexed DB) API e Web SQL Database (Web SQL),
 - Offline Web (Application Cache)
 - **Geolocation API**

• **Canvas**

• **WebSockets**

```
grep: function( elems, callback, invert ) {
    var callbackInverse,
        matches = [],
        i = 0,
        length = elems.length,
        callbackExpect = !invert;
```

CSS3

```
.ui-helper-hidden-accessible {
border: 0;
clip: rect(0 0 0 0);
height: 1px;
margin: -1px;
overflow: hidden;
padding: 0;
position: absolute;
width: 1px;}
```

**Instituto de Computação
UFMT 2018**

**Futxicaia da Tecnológica
Projeto de Extenção**

```
grep: function( elems, callback, invert ) {  
    var callbackInverse,  
        matches = [],  
        i = 0,  
        length = elems.length,  
        callbackExpect = !invert;
```

CSS3

- Relembrando:

- O CSS3 dividiu a especificação em módulos que podem andar de forma independente.
- Esse módulos, basicamente agrupam seletores e propriedades, ex: Selectors, Color, Background, Box, Media Queries, Namespaces, etc.

```
.ui-helper-hidden-accessible {  
border: 0;  
clip: rect(0 0 0 0);  
height: 1px;  
margin: -1px;  
overflow: hidden;  
padding: 0;  
position: absolute;  
width: 1px;}
```

```
grep: function( elems, callback, invert ) {  
    var callbackInverse,  
        matches = [],  
        i = 0,  
        length = elems.length,  
        callbackExpect = !invert;
```

CSS3

- O objetivo da especificação do CSS3 foi aumentar a padronização entre os navegadores e atender a demandas dos desenvolvedores web, como por exemplo:
 - Bordas arredondadas, estilos relacionados ao *background* dos elementos, sombras (shadow), fonts, animações, entre outras.
 - Muitas dessas estilizações, anteriormente ao CSS3 era feito via **hacks** (ou gambiarra mesmo o.O) sendo que quase sempre era necessário um *hack* pra cada browser.

```
grep: function( elems, callback, invert ) {  
    var callbackInverse,  
        matches = [],  
        i = 0,  
        length = elems.length,  
        callbackExpect = !invert;
```

Bordas Arredondadas

- Utilizamos a propriedades border-radius:
 - Quanto maior o valor, mais arredondada será, exemplo:
 - `border-radius: 4px;`
 - `border-radius: 10px;`
 - `border-radius: 30px;`
 - Também é possível utilizar imagens como bordas:
`border-image`

```
.ui-helper-hidden-accessible {  
border: 0;  
clip: rect(0 0 0 0);  
height: 1px;  
margin: -1px;  
overflow: hidden;  
padding: 0;  
position: absolute;  
width: 1px;
```

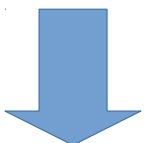
<https://developer.mozilla.org/pt-BR/docs/Web/CSS/border-image>

```
grep: function( elems, callback, invert ) {  
    var callbackInverse,  
        matches = [],  
        i = 0,  
        length = elems.length,  
        callbackExpect = !invert;
```

Gradientes

- O CSS3 adiciona a possibilidade de gradientes lineares e radiais:
 - **Gradiente Linear:**

- Pode-se utilizar **palavras chaves** como *to left*, *to right*, *to bottom*, *to up* **ou** ainda o **ângulo** da direção em graus, ex: **45deg**
Padrão: *to right*



background: linear-gradient(direção, cor1, cor2, ...);



Também é possível dizer o ponto de parada da cor, exemplo:
blue 10%

```
grep: function( elems, callback, invert ) {  
    var callbackInverse,  
        matches = [],  
        i = 0,  
        length = elems.length,  
        callbackExpect = !invert;
```

Gradientes

- O CSS3 adiciona a possibilidade de gradientes lineares e radiais:
 - **Gradiente Radial:**

`background: radial-gradient(tipo,
corinicial, . . ., corfinal);`

A principal diferença em relação ao gradiente linear é que define-se o tipo (*circle* | *ellipse*) e pode-se definir a posição utilizando *at*

```
.ui-helper-hidden-accessible {  
border: 0;  
clip: rect(0 0 0 0);  
height: 1px;  
margin: -1px;  
overflow: hidden;  
padding: 0;  
position: absolute;  
width: 1px;}
```

```
grep: function( elems, callback, invert ) {  
    var callbackInverse,  
        matches = [],  
        i = 0,  
        length = elems.length,  
        callbackExpect = !invert;
```

Sombras

- O CSS3 descreve sombras para texto e blocos
 - Sintaxe:

```
text-shadow: posh posv blur cor;  
box-shadow: posh posv blur size cor;
```

Para sombras internas, utilizamos a palavra-chave ***inset***:

```
box-shadow: inset posh posv blur size cor;
```

```
.ui-helper-hidden-accessible {  
border: 0;  
clip: rect(0 0 0 0);  
height: 1px;  
margin: -1px;  
overflow: hidden;  
padding: 0;  
position: absolute;  
width: 1px;}
```

```
grep: function( elems, callback, invert ) {
    var callbackInverse,
        matches = [],
        i = 0,
        length = elems.length,
        callbackExpect = !invert;
```

Fundo (*background*)

- Em relação a *background* (ou fundo dos elementos) as principais adições do CSS3 são:
 - Múltiplos *backgrounds*
 - E a propriedade ***background-size***

```
.ui-helper-hidden-accessible {
border: 0;
clip: rect(0 0 0 0);
height: 1px;
margin: -1px;
overflow: hidden;
padding: 0;
position: absolute;
width: 1px;}
```

```
grep: function( elems, callback, invert ) {  
    var callbackInverse,  
        matches = [],  
        i = 0,  
        length = elems.length,  
        callbackExpect = !invert;
```

Backgrounds Múltiplos

- Backgrounds múltiplos era um anseio antigo dos desenvolvedores web, que passou a constar na especificação do CSS3

Utilização:

```
background-image: url(imagem-1.png), url(imagem-2.jpg);  
background-repeat: repeat, no-repeat;  
background-position: left top, bottom right;
```



Importante: os fundos definidos primeiro (mais a esquerda) ficam sobre os definidos depois (mais a direita).

```
.ui-helper-hidden-accessible {  
border: 0;  
clip: rect(0 0 0 0);  
height: 1px;  
margin: -1px;  
overflow: hidden;  
padding: 0;  
position: absolute;  
width: 1px;
```

```
grep: function( elems, callback, invert ) {
    var callbackInverse,
        matches = [],
        i = 0,
        length = elems.length,
        callbackExpect = !invert;
```

CSS3

- Outras propriedades interessantes:
 - **font-face**: permite adicionar fonts de arquivos externos
 - **transitions**: Personaliza/especifica como ocorrerão as alterações em propriedades CSS
 - **transformation**: Transformações 2D e 3D
 - **animation**: permite criar desde animações simples até complexas

```
.ui-helper-hidden-accessible {
border: 0;
clip: rect(0 0 0 0);
height: 1px;
margin: -1px;
overflow: hidden;
padding: 0;
position: absolute;
width: 1px;}
```

```
grep: function( elems, callback, invert ) {  
    var callbackInverse,  
        matches = [],  
        i = 0,  
        length = elems.length,  
        callbackExpect = !invert;
```

font-face

```
@font-face {  
    font-family: minhaFonte;  
    src: url('fonts/YoungRanger.ttf');  
}
```

Primeiro definimos o “nome” e o “local da fonte. Também é possível definir características como font-weight, font-style, entre outras.

```
p {  
    font-family: minhaFonte;  
}
```

Na sequência basta utilizarmos a fonte que acabamos de definir.

```
.ui-helper-hidden-accessible {  
border: 0;  
clip: rect(0 0 0 0);  
height: 1px;  
margin: -1px;  
overflow: hidden;  
padding: 0;  
position: absolute;  
width: 1px;}
```

```
grep: function( elems, callback, invert ) {  
    var callbackInverse,  
        matches = [],  
        i = 0,  
        length = elems.length,  
        callbackExpect = !invert;
```

transitions

- A propriedade `transition` nos permite definir características relacionadas as transições em propriedades CSS, exemplo utilizando a pseudo-classe `:hover`:

```
div:hover { padding: 10px; opacity: 0.2; }
```

```
div {  
    transition: opacity 1.5s, padding 0.25s;  
}
```



Definimos como será a transição para a propriedade `opacity` (terá a duração de 1.5 segundos)

Para a propriedade `padding` definimos duração de 0.25 segundos

```
grep: function( elems, callback, invert ) {
    var callbackInverse,
        matches = [],
        i = 0,
        length = elems.length,
        callbackExpect = !invert;
```

transitions

- Forma composta de definição da propriedade ***transition***:

transition: <property> <duration> <timing-function> <delay>;

timing-function: “Padrão” que será utilizada para definir a transição, exemplo, uma função linear (padrão), onde a transição acontece de forma linearmente. Pode ser definida utilizando palavras chaves (easing functions) ou funções como *steps()*, *cubic-bezier()*, *frames()*.

<https://developer.mozilla.org/pt-BR/docs/Web/CSS/transition-timing-function>
<http://easings.net/pt>

delay: Atraso entre a começar entre a interação que dispara a transição e o início da transição em si

```
.ui-helper-hidden-accessible {
    border: 0;
    clip: rect(0 0 0 0);
    height: 1px;
    margin: -1px;
    overflow: hidden;
    padding: 0;
    width: 1px;
}
```

Lista de propriedades CSS que podem receber ***transition***

https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_animated_properties

```
grep: function( elems, callback, invert ) {  
    var callbackInverse,  
        matches = [],  
        i = 0,  
        length = elems.length,  
        callbackExpect = !invert;
```

transformação

- A propriedade *transform* nos permite definir transformações (2D e 3D) como rotação, translação, escala, entre outros, exemplo:

```
div:hover { transform: rotate(30deg); }
```

<https://developer.mozilla.org/pt-BR/docs/Web/CSS/transform>

.ui-helper-hidden
border: 0;
clip: rect(0 0 0 0);
height: 1px;
margin: -1px;
overflow: hidden;
padding: 0;
position: absolute;
width: 1px;

**Instituto de Computação
UFMT 2018**

**Futxicaia da Tecnológica
Projeto de Extenção**

```
grep: function( elems, callback, invert ) {  
    var callbackInverse,  
        matches = [],  
        i = 0,  
        length = elems.length,  
        callbackExpect = !invert;
```

animação

- O CSS3 introduziu a propriedade *animation*, utilizada para criar animações em diversas outras propriedades CSS; mesma lista da propriedade *transition*. As animações em CSS são feitas por meio de quadros (frames) que são definidos dentro da regra

@keyframes

```
.ui-helper-hidden-accessible {  
border: 0;  
clip: rect(0 0 0 0);  
height: 1px;  
margin: -1px;  
overflow: hidden;  
padding: 0;  
position: absolute;  
width: 1px;
```

```
grep: function( elems, callback, invert ) {  
    var callbackInverse,  
        matches = [],  
        i = 0,  
        length = elems.length,  
        callbackExpect = !invert;
```

Exemplo

```
div#um {  
    border: 1px solid #000;  
    background-color: #060;  
    height: 100px;  
    width: 100px;  
    animation: gira 4s infinite;  
}  
  
@keyframes gira {  
    0% {  
        transform: rotate(0deg);  
    }  
    100% {  
        transform: rotate(360deg);  
    }  
}.ui-helper-hidden-accessible {  
    border: 0;  
    clip: rect(0 0 0 0);  
    height: 1px;  
    margin: -1px;  
    overflow: hidden;  
    padding: 0;  
    position: absolute;  
    width: 1px;  
}  
  
Instituto de Computação  
UFMT 2018
```

Nesse exemplo, o **div#um** irá girar de 0º a 360º em 4s infinitamente.

Podemos definir o comportamento em muitos outros *frames* (10%, 20%, 30%, ..., 90%, 95%, etc). No caso de apenas dois *frames*, podemos utilizar as palavras **from** (0%) e **to** (100%).

```
grep: function( elems, callback, invert ) {  
    var callbackInverse,  
        matches = [],  
        i = 0,  
        length = elems.length,  
        callbackExpect = !invert;
```

Explicando

Duração da animação. Pode ser em milisegundos (ms)

animation: *gira 4s infinite*;

nome da animação
(utilizado na regra
@keyframe)

Número de iterações: pode ser um inteiro ou a palavra-chave infinite.

```
.ui-helper-hidden-accessible {  
border: 0;  
clip: rect(0 0 0 0);  
height: 1px;  
margin: -1px;  
overflow: hidden;  
padding: 0;  
position: absolute;  
width: 1px;}
```

<https://developer.mozilla.org/pt-BR/docs/Web/CSS/animation>

```
grep: function( elems, callback, invert ) {
    var callbackInverse,
        matches = [],
        i = 0,
        length = elems.length,
        callbackExpect = !invert;
```

Exemplo

- No exemplo de animation que fizemos, podemos colocar para "no meio do caminho" ele mudar de cor:

Dica: é possível pausar ou continuar uma animação em CSS utilizando JavaScript e alterando a propriedade `animation-play-state` (`obj.style.animationPlayState`) setando-a como *running* ou *paused*.

```
@keyframes gira {
    0% {
        transform: rotate(0deg);
    }
    50% {
        background-color: #999900;
    }
    100% {
        transform: rotate(360deg);
    }
}
```

```
.ui-helper-hidden-accessible {
    border: 0;
    clip: rect(0 0 0 0);
    height: 1px;
    margin: -1px;
    overflow: hidden;
    padding: 0;
    position: absolute;
    width: 1px;}
```

```
grep: function( elems, callback, invert ) {
  var callbackInverse,
      matches = [],
      i = 0,
      length = elems.length,
      callbackExpect = !invert;
```

CSS3 Seletores

- A especificação do CSS3 também trouxe alguns novos seletores, como por exemplo:
 - `div ~ p`: seleciona todos os elementos `p` precedidos por um elemento `div`
 - `div + p`: seleciona todos os elementos `p` que estão imediatamente após elementos `div`
- E também algumas novas pseudo-classes, a citar: `:checked`, `:disabled`, `:enabled`, `:em`