

# HTML Básico

**Graziela Göedert de Souza**

# Cronograma

1. HTML Básico
2. HTML Básico
3. HTML Básico com servidor NodeJs
4. Incluindo nossa aplicação NodeJs no Heroku
5. HTML com um Framework de mercado (Angular), rotas e Conexão com o Banco de dados
6. HTML com um Framework de mercado (Angular), rotas e Conexão com o Banco de dados
7. HTML com um Framework de mercado (Angular). Projeto de conclusão, controle da despesa
8. HTML com um Framework de mercado (Angular). Projeto de conclusão controle da despesa

# Aula 1 - HTML Básico

---

# Introdução

HTML é a sigla de **Hyper Text Markup Language** (**Linguagem de marcação de hipertexto**), ou seja, a linguagem usada para criar páginas Web.

# Introdução

Tags são os comandos usados para a construção de sua página na web, elas podem ser abertas, como `<br>`, ou fechadas, como `<p></p>`.

As tags que possuem "/" sempre servem para fechar(`</p>`) uma tag antes aberta (`<p>`).

# Editores

Podemos estar criando nossas páginas HTMLs com qualquer editor de texto, bastando salvá-las com a extensão **.html** (ex: index.html).

Iremos utilizar no curso o **Visual Studio Code**:

<https://code.visualstudio.com/download>

# Iniciando uma página HTML

O corpo básico de uma página HTML, é composto pelas tags:

**<html></html>**

**<body></body>**

Não há necessidade de as tags estarem em letra maiúscula, ou minúscula, porém organizar as tags, é mais fácil para a localização em um código.

**<html>**

**<body>**

**</body>**

**</html>**

# Construindo uma página HTML

Para se colocar um título na página é necessário ter as tags <head> e <title> conforme abaixo:

```
<html>  
  <head>  
    <title>  
      Título da página  
    </title>  
  </head>  
  <body>  
  </body>  
</html>
```



# Construindo uma página HTML

Na tag body, podemos adicionar as configurações padrões da página:

**bgcolor:** altera a cor de fundo da página HTML.

```
<html>  
  <head>  
    <title>  
      Título da página  
    </title>  
  </head>  
  <body bgcolor="000055">  
  </body>  
</html>
```

# Construindo uma página HTML

Na tag body, podemos adicionar as configurações padrões da página:

**background:** adiciona uma imagem como plano de fundo da página HTML.

```
<html>
  <head>
    <title>
      Título da página
    </title>
  </head>
  <body background="fundo.jpg">
  </body>
</html>
```

# Construindo uma página HTML

Na tag body, podemos adicionar as configurações padrões da página:

**text:** define a cor de texto padrão da página HTML.

```
<html>
```

```
  <head>
```

```
    <title>
```

```
      Título da página
```

```
    </title>
```

```
  </head>
```

```
  <body background="fundo.jpg" text="#CCoooo">
```

```
    Texto de exemplo para verificar a cor
```

```
  </body>
```

```
</html>
```

# Textos - Hs

Há vários tipos de forma de se fazer textos em páginas HTML.

A mais simples delas é com a tag <hx>(x, corresponde a um número de 1 a 6, sendo 1 o maior e seis o menor).

```
<html>
```

```
    <head>
```

```
        <title>
```

**Título da página**

```
    </title>
```

```
</head>
```

```
<body background="fundo.jpg" text="#CCoooo">
```

```
    <H1>Aqui será Exibido o texto com a cor padrão da página, e no maior tamanho possível</H1>
```

```
    <H2>Aqui será exibido um texto menor...</H2>
```

```
    <H3>...aqui será exibido um texto menor...</H3>
```

```
    <H4>...aqui será exibido um texto menor...</H4>
```

```
    <H5>...aqui será exibido um texto menor...</H5>
```

```
    <H6>...E aqui será exibido o menor texto possível com a tag "<Hx>"</H6>
```

```
</body>
```

```
</html>
```

# Textos - Parágrafos

Podemos estar quebrando linha de 2 formas, utilizando a tag <br>, ou parágrafo <p>.

**<html>**

**<head>**

**<title>**

**Título da página**

**</title>**

**</head>**

**<body>**

**O texto vai aqui, e eu uso o <br>**

**“br” para quebrar a linha**

**<p>Quando eu uso o “p”, eu não preciso quebrar linha,</p>pois o parágrafo que começa depois, já é outro**

**</body>**

**</html>**

# Textos - Parágrafos

A tag <p> serve também para configurarmos posição do texto na página:

<html>

<head>

<title>

Título da página

</title>

</head>

<body>

<p align="center">Alinha o texto escrito aqui ao centro da página</p>

</body>

</html>

# Textos - Parágrafos

Os alinhamentos podem ser conforme abaixo:

Comando	Posição
<code>align="left"</code>	Esquerda - Padrão
<code>align="right"</code>	Direita
<code>align="center"</code>	Centro

# Textos - Font

Nele informamos as configurações de formatação dos texto, como letra, tamanho e cor:

```
<html>  
  <head>  
    <title>  
      Título da página  
    </title>  
  </head>  
  <body>  
    <font color="#AA0000" face="arial" size="12">Onde Face é a letra, Size, é o  
tamanho da letra e color é a cor da letra</font>  
  </body>  
</html>
```



# Textos - Estilos de caracteres

Podemos formatar uma **palavra**, *frase* ou **letra**, como **negrito**, *itálico*, sublinhado, <sub>subscrito</sub> e <sup>sobrescrito</sup>.

Formatação	Comando
Negrito	<code>&lt;b&gt;&lt;/b&gt;</code>
Itálico	<code>&lt;i&gt;&lt;/i&gt;</code>
Sublinhado	<code>&lt;u&gt;&lt;/u&gt;</code>
Subscrito	<code>&lt;sub&gt;&lt;/sub&gt;</code>
Sobrescrito	<code>&lt;sup&gt;&lt;/sup&gt;</code>

# Textos - Estilos de caracteres

Podemos formatar uma **palavra**, frase ou **letra**, como **negrito**, **itálico**, **sublinhado**, **subscrito** e **sobrescrito**.

`<html>`

`<body>`

Esse é o texto, e essa palavra está em `<B>Negrito</B>`, essa está em `<I>Itálico</I>`, essa `<U>sublinhada</U>`, essa `<SUB>subscrita</SUB>`, e essa `<SUP>sobrescrita</SUP>`

`</body>`

`</html>`

# Quebras de linha

A tag `<br>` faz a quebra de linha sem acrescentar espaços extras entre linhas. Finaliza a linha de texto e insere automaticamente um outra linha em branco. Não precisa ser finalizada com `</br>`.

```
<html>
```

```
  <body>
```

```
    Esse é o texto irá aparecer em uma linha.
```

```
    <br>
```

```
    Esse outro na linha seguinte.
```

```
  </body>
```

```
</html>
```

# Linhas Horizontais

Desenha uma linha horizontal (<**hr**>) no documento. Não precisa ser finalizada com </hr>.

Atributos:

- **size**: Define a espessura, em **pixels**, da linha.
- **width**: define a largura da linha, o que pode ser feito em **pixels** (número absoluto) ou em **percentual** da tela (com o símbolo de %).
- **align**: alinhamento, os mesmos que vimos anteriormente, pode ser **left**, **right** e **center**.
- **noshade**: Linha sem sombra. O padrão é a linha sombreada, utilizando esse atributo temos uma linha sem sombra.

# Linhas Horizontais

```
<html>
```

```
  <body>
```

**Primeiro exemplo com a linha horizontal**

```
<hr width="100%" align="left" size="2" color="silver"><br>
```

**<center>Segundo exemplo com a linha horizontal</center>**

```
<hr width="70%" align="center" size="3" color="blue"><br>
```

**Terceiro exemplo com a linha horizontal**

```
<hr width="30%" align="center" size="5" color="red" noshade><br>
```

```
</body>
```

```
</html>
```

# Caracteres Especiais

Á	&Aacute;	á	&aacute;	Â	&Acirc
â	&acirc;	À	&Agrave;	à	&agrave;
Å	&Aring;	å	&aring;	Ã	&Atilde;
ã	&atilde;	Ä	&Auml;	ä	&auml;
Æ	&AElig;	æ	&aelig;	É	&Eacute;
é	&eacute;	Ê	&Ecirc;	ê	&ecirc;
È	&Egrave;	è	&egrave;	Ë	&Euml;
ë	&euml;	Ð	&ETH;	ð	&eth;
Í	&Iacute;	í	&iacute;	Î	&Icirc;
î	&icirc;	Ì	&Igrave;	ì	&igrave;
Ĭ	&Iuml;	Ĳ	&iuml;	Ó	&Oacute;
ó	&oacute;	Ô	&Ocirc;	ô	&ocirc;
Õ	&Ograve;	ò	&ograve;	Ø	&Oslash;
ø	&oslash;	Ö	&Otilde;	ö	&otilde;
Ö	&Ouml;	ö	&ouml;	Ú	&Uacute;
ú	&uacute;	Û	&Ucirc;	û	&ucirc;
Û	&Ugrave;	ù	&ugrave;	Ü	&Uuml;
ü	&uuml;	Ç	&Ccedil;	ç	&ccedil;
Ñ	&Ntilde;	ñ	&ntilde;	<	&lt;
>	&gt;	&	&amp;	"	&quot;
®	&reg;	©	&copy;	Ý	&Yacute;
ý	&yacute;	Þ	&THORN;	þ	&thorn;
ß	&szlig;	°	&#186;	ª	&170;
¹	&#185;	²	&#178;	³	&#179;
f	&#131;	†	&#134;	‡	&#135;
‰	&#137;	¢	&#162;	£	&#163;
«	&#171;	±	&#177;	»	&#187;
·	&#183;	¼	&#188;	½	&#189;
¾	&#190;	¿	&#191;	×	&#215;
÷	&#247;	¡	&#161;	¤	&#164;

# Comentário

No HTML podemos acrescentar parte do código comentada, seja para deixar uma mensagem em código, ou apenas para comentar temporariamente parte do nosso código, esses comentário não são apresentados em tela:

```
<html>
  <head>
    <title>
      Título da página
    </title>
  </head>
  <body background="fundo.jpg" text="#CCoooo">
    Texto não comentado.
    <!--<p>Texto de exemplo para verificar a cor</p>--!>
  </body>
</html>
```

# Cores

No HTML, as cores podem ser nomeadas pelo nome (em Inglês) como: **White**, **green**, **blue** e assim por diante.

Podem ser codificadas, por números hexadecimais como: **FFAAoo**, na verdade os códigos são como uma misturas de cores **RGB (Red, Green, Blue)**, então se o código hexadecimal for **oo66oo**, deduz-se que a cor é **verde**, pois os dois primeiros números (que correspondem ao **vermelho** – **RRGGBB**), são **o**, os dois números do meio são **6** e os dois números finais são **o**.

Para escrever uma cor, é mais fácil usar-se códigos, e mais fácil ainda usar os códigos hexadecimais. Na tag **body**, **bgcolor**, define a cor do plano de fundo da página e **text** a cor do texto padrão na página, na tag **font color** define a cor do texto. Para se escrever uma cor codificada, usa-se **#** antes do código, e dentro das aspas, por exemplo, "**#550000**"



# Listas - Numeradas

Servem para organizar assuntos em tópicos, números e menus:

`<ol>` é a tag que define lista ordenada, ou seja, numerada, e `<li>` é a tag que define os itens da lista.

```
<html>
```

```
  <body>
```

```
    <ol>
```

```
      <li>Introdução
```

```
      <li>Dedicatória
```

```
      <li>Capítulo 1
```

```
    </ol>
```

```
  </body>
```

```
</html>
```

# Listas - Tópicos

Servem para organizar assuntos em tópicos, números e menus:

`<ul>` é a tag que define lista não ordenada, ou seja, tópicos, e `<li>` é a tag que define os itens da lista.

```
<html>
```

```
  <body>
```

```
    <ul>
```

```
      <li>Introdução
```

```
      <li>Dedicatória
```

```
      <li>Capítulo 1
```

```
    </ul>
```

```
  </body>
```

```
</html>
```

# Listas - Tópicos

Servem para organizar assuntos em tópicos, números e menus:

**<dl>** é a tag que define lista não ordenada, ou seja, tópicos, **<dt>** é a tag que define os itens da lista e **<dd>** define uma descrição detalhada para os itens que estiverem imediatamente acima do **dd**.

```
<html>
```

```
    <body>
```

```
        <dl>
```

```
            <dt>Firefox</dt>
```

```
            <dt>Mozilla Firefox</dt>
```

```
            <dt>Fx</dt>
```

```
            <dd>A free, open source, cross-platform, graphical web browser  
            developed by the Mozilla Corporation and hundreds of volunteers.</dd>
```

```
        </dl>
```

```
    </body>
```

```
</html>
```

# Tabelas - Table

As tabelas são ótimos formatos para apresentar informações.

**<table>**: é utilizada para a representação de dados tabulares. A estrutura e o conteúdo da tabela devem ficar dentro das Tags **<table>****</table>**

**<caption>**: especifica o título de uma tabela. Por exemplo:

**<caption>**Notas da primeira avaliação**</caption>**

**<td>**: especifica a célula de dados de uma tabela. Por se tratar de dados comuns (e não cabeçalhos), essas células possuem seu conteúdo escrito em fonte normal, sem nenhum destaque e alinhamento à esquerda. Assim como o **th**, pode-se construir células em branco, usando o elemento **td**, como no exemplo a seguir:

**<td>**Células de dados**</td>**

# Tabelas - Table

`<tr>`: indica o início de uma linha na tabela. Cada linha da tabela pode conter várias células, e, portanto, é necessário que se faça uso de uma marcação que indique exatamente o ponto de quebra de uma linha e início de outra. Toda linha deve terminar com um `</tr>`

# Tabelas - Table

```
<html>
  <body>
    <table>
      <caption>Nota da primeira avaliação</caption>
      <td>Notas/Alunos</td>
      <th>Eduardo</th>
      <th>Ana Lúcia</th>
      <th>Adréa</th>
    <tr>
      <th>Notas</th>
      <td>8,0</td>
      <td>9,3</td>
      <td>7,8</td>
    <tr>
      <th>Nº de Inscrição</th>
      <td>123456</td>
      <td>456789</td>
      <td>789456</td>
    </table>
  </body>
</html>
```

# Tabelas - Table - Border

As marcações das tabelas podem apresentar resultados diferentes, se acompanhadas de alguns atributos. Os principais são:

**border:** É um atributo opcional, onde caso esteja presente, a tabela será formatada com linhas de borda.

Ele pode receber um valor que vai estabelecer a espessura (além da existência) da linha de borda da tabela (**border="valor"**). Se o valor atribuído for o(zero), o **border** funciona exatamente como o caso padrão, sem o border. Dessa maneira, é possível colocar tabelas em maior destaque, atribuindo um valor maior que 1 para o **border**.

# Tabelas - Table - Border

```
<html>
  <body>
    <table border>
      <caption>Nota da primeira avaliação</caption>
        <td>Notas/Alunos</td>
        <th>Eduardo</th>
        <th>Ana Lúcia</th>
        <th>Adréa</th>
      <tr>
        <th>Notas</th>
        <td>8,0</td>
        <td>9,3</td>
        <td>7,8</td>
      <tr>
        <th>Nº de Inscrição</th>
        <td>123456</td>
        <td>456789</td>
        <td>789456</td>
      </table>
    </body>
  </html>
```



# Tabelas - Table - Border

```
<html>
  <body>
    <table border=5>
      <tr>
        <td>teste</td>
        <td>teste2</td>
        <td>teste3</td>
      </tr>
      <tr>
        <td>teste4</td>
        <td>teste5</td>
        <td>teste6</td>
      </tr>
    </table>
  </body>
</html>
```

# Tabelas - Table - Align

**align:** Esse atributo pode ser aplicado a **th**, **td**, **tr** e controla o alinhamento do texto dentro de uma célula, com relação as bordas laterais da tabela. Quando aplicado a **tr**, ele define o alinhamento de toda uma linha da tabela.

O exemplo abaixo mostra como o **align** aceita os valores **left**, **center** e **right**:

# Tabelas - Table - Align

```
<html>
  <body>
    <table border>
      <tr>
        <td>teste</td>
        <td>teste2</td>
        <td>teste3</td>
      </tr>
      <tr>
        <td align="center">Centro</td>
        <td align="left">Esqueda</td>
        <td align="righ">Direita</td>
      </tr>
    </table>
  </body>
</html>
```

# Tabelas - Table - VAlign

**valign:** Esse atributo pode ser aplicado a **th**, **td** e define o alinhamento do texto em relação as bordas superiores e inferiores.

Aceita os valores **top**, **middle**, e **bottom**, para alinhar **na parte de cima**, **no meio** e **na parte de baixo**, respectivamente.

# Tabelas - Table - VAlign

```
<html>  
  <body>  
    <table border>  
      <tr>  
        <td>Teste de alinhamento</td>  
        <td valign="top">Top</td>  
        <td valign="middle">Middle</td>  
        <td valign="bottom">Bottom</td>  
      </tr>  
    </table>  
  </body>  
</html>
```

# Tabelas - Table - Cellspacing

**cellspacing:** Esse atributo compreende a distância entre células e linhas.

Deve ser adicionado dentro da tag <table>.

Como padrão dos navegadores a distância é 2 pixels.

```
<html>
```

```
  <body>
```

```
    <h3>Exemplo com cellspacing</h3>
```

```
    <table border="1" width="80%" align="center" cellspacing="6">
```

```
      <tr>
```

```
        <td width="33%" align="center">Coluna 1</td>
```

```
        <td width="33%" align="center">Coluna 2</td>
```

```
        <td width="34%" align="center">Coluna 3</td>
```

```
      </tr>
```

```
    </table>
```

```
  </body>
```

```
</html>
```

# Tabelas - Table - Cellpadding

**cellpadding:** Esse atributo é utilizado para formatar o espaço entre o conteúdo de uma célula e suas bordas em todos os sentidos

É aplicado dentro da tag **<table>**

```
<html>
```

```
  <body>
```

```
    <table border="1" width="80%" align="center" cellpadding="6">
```

```
      <tr>
```

```
        <td width="33%" align="center">Coluna 1</td>
```

```
        <td width="33%" align="center">Coluna 2</td>
```

```
        <td width="34%" align="center">Coluna 3</td>
```

```
      </tr>
```

```
    </table>
```

```
  </body>
```

```
</html>
```

# Tabelas - Table - rowspan

**rowspan:** Define quantas linhas uma mesma célula pode abranger. Por padrão, na maioria dos navegadores cada célula adicionada em uma tabela corresponde a uma linha.

Pode ser aplicado dentro das tags `<th>` ou `<td>`, proporcionando o mesmo efeito.



# Tabelas - Table - rowspan

```
<html>
  <body>
    <table border="1" width="80%" align="center" cellpadding="7">
      <tr>
        <td width="33%" rowspan="3">Coluna 1 - Linha 1, 2 e 3</td>
        <td width="33%" align="center">Coluna 2 - Linha 1</td>
        <td width="34%" align="center">Coluna 3 - Linha 1</td>
      </tr>
      <tr>
        <td width="33%" align="center">Coluna 2 - Linha 2</td>
        <td width="34%" align="center">Coluna 3 - Linha 2</td>
      </tr>
      <tr>
        <td width="33%" align="center">Coluna 2 - Linha 3</td>
        <td width="34%" align="center">Coluna 3 - Linha 3</td>
      </tr>
      <tr>
        <td width="33%">Coluna 1 - Linha 4</td>
        <td width="33%" align="center">Coluna 2 - Linha 4</td>
        <td width="34%" align="center">Coluna 3 - Linha 4</td>
      </tr>
    </table>
  </body>
</html>
```

# Tabelas - Table - colspan

**colspan:** Define quantas colunas uma mesma célula pode abranger. Por padrão, na maioria dos navegadores cada célula adicionada em uma tabela corresponde a uma linha.

Pode ser aplicado dentro das tags `<th>` ou `<td>`, proporcionando o mesmo efeito.

# Tabelas - Table - colspan

```
<html>
  <body>
    <table border="1" width="80%" align="center" cellpadding="7">
      <tr>
        <td width="33%" colspan="3">Célula com 3 colunas</td>
        <td width="33%" align="center">Coluna 4</td>
        <td width="34%" align="center">Coluna 5</td>
      </tr>
      <tr>
        <td width="33%" align="center">Coluna 1</td>
        <td width="34%" align="center">Coluna 2</td>
        <td width="34%" align="center">Coluna 3</td>
        <td width="34%" align="center">Coluna 4</td>
        <td width="34%" align="center">Coluna 5</td>
      </tr>
    </table>
  </body>
</html>
```

# Tabelas - Table - Largura da célula

**width:** Para alterar a largura de uma célula da tabela basta acrescentar o parâmetro width dentro da tag **<td>**.

```
<html>
  <body>
    <table border="2" width="80%">
      <tr>
        <td width="100">Width=100</td>
        <td align="middle" width="200">width=200</td>
      </tr>
    </table>
  </body>
</html>
```

# Tabelas - Table - Cor de fundo das células

**bgcolor:** Outro atributo que podemos ter nas tabelas é mudar a sua cor de fundo, isto se torna particularmente útil quando se quer dar destaque a uma célula em especial.

```
<html>
  <body>
    <table>
      <tr>
        <td bgcolor="red">Vermelho</td>
        <td bgcolor="blue">Azul</td>
        <td bgcolor="red">Vermelho</td>
      </tr>
      <tr>
        <td bgcolor="blue">Azul</td>
        <td bgcolor="red">Vermelho</td>
        <td bgcolor="blue">Azul</td>
      </tr>
    </table>
  </body>
</html>
```

# Exercícios

1) Crie um arquivo para cada tabela a seguir:

<b>Tabela</b>	<b>Hora da saída</b>	<b>Hora de chegada</b>	<b>Classe do ônibus</b>	<b>Tarifa normal</b>	<b>Frequência</b>
<b>Cidade A</b>	06:00	14:00	Convencional	20,00	Diária
<b>Cidade B</b>	12:00	15:00	Convencional	10,00	Domingos
<b>Cidade C</b>	14:00	17:00	Leito	15,00	Diária
<b>Cidade D</b>	16:00	17:00	Convencional	10,00	Diária
<b>Cidade E</b>	18:30	20:00	Executivo	30,00	Domingos
<b>Cidade F</b>	20:00	23:00	Convencional	80,00	Sábados
<b>Cidade G</b>	21:00	23:30	Convencional	26,00	Diária
<b>Cidade H</b>	22:00	23:00	Executivo	16,00	Diária
Horários de ônibus					

# Exercícios

## Tabela de Serviços Mecânicos

Nacionais	Veículos	Serviço	Tempo Previsto	Valor
	Caminhonete S10	Suspensão	3 horas e 25 minutos	350,00
	Caminhonete Ranger	Freio	3 horas e 45 minutos	200,00
	Caminhonete Hilux	Troca de Bateria	30 minutos	25,00
	Caminhonete D10	Alinhamento	1 hora e 40 minutos	220,00

# Exercícios

Venha Morar em Nossos Belos Residenciais - Viver Serra						
<b>Apartamentos</b>			<b>Localização</b>	Laranjeiras - 4 min.	Manguinhos- 2 min.	Vitoria- 8 min.
2 Quartos	3 Quartos	4 Quartos	<b>Corretor On-Line</b>			
1 Suíte	2 Suíte	3 Suítes	3333-2333	9999-9969	8888-8868	8111-1169
<b>Lazer</b>	Piscina, Salão de Festas, Quadra Poliesportivas		Pronto Para Morar			<b>LANÇAMENTO</b>



# Imagens

Para inserir imagens, basta que ela seja de preferência GIF, ou JPG, não é obrigatório, mas recomenda-se o uso de imagens GIF, apenas para botões e ícones, pois ela possui uma definição de apenas 256 cores e o uso de imagens JPG, para a inserção de imagens fotográficas, pois ela possui uma definição de 16,7 milhões de cores.

**Por Exemplo:** ``, onde **foto.jpg**, é o nome da imagem, ou ``, onde **c:** é o destino da imagem, e **foto.jpg**, é o nome da imagem.

# Imagens

Quando carregamos a imagem ela carrega no seu tamanho real, para corrigir esses detalhes podemos utilizar alguns atributos como tamanho, o alinhamento, a borda, etc:

Atributos	Descrição
<b>src</b>	Atributo obrigatório, devemos especificar o caminho ou a url da imagem
<b>alt</b>	Apresenta um texto quando a imagem estiver indisponível
<b>height</b>	Define uma altura para a imagem, caso seja uma altura maior que a da imagem ela ajusta a imagem distorcendo a qualidade dela
<b>width</b>	Define uma largura para a imagem, caso seja uma largura maior que a da imagem ela ajusta a imagem distorcendo a qualidade dela
<b>align</b>	<b>absmiddle</b> - alinha a imagem com o centro da linha <b>absbottom</b> - alinha a imagem com a parte de baixo da linha <b>Outros</b> - possui os alinhamentos tradicionais como <b>top</b> , <b>left</b> , <b>right</b> , <b>bottom</b> , <b>baseline</b> , além de <b>middle</b> referente aos demais elementos do html a ela dispostos
<b>border</b>	Define uma espessura para a borda, em pixels
<b>hspace</b>	Define os espaços em pixels, à direita e à esquerda da imagem
<b>vspace</b>	Define os espaços em pixels, acima e abaixo da imagem

# Imagens

```
<html>
  <body>
    <table border="1" width="100%" cellpadding="1" cellspacing="1">
      <tr>
        <td>
          </img>
          Este é um exemplo para border e absmiddle
        </td>
        <td>
          </img>
          Este é um teste para width e height - use um editor para ter a imagem no tamanho desejado!
        </td>
        <td>
          Este é um exemplo do atributo top e do atributo alt com link
          <a href="/index.jsp">
            </img>
          </a>
          </img>
        </td>
      </tr>
    </table>
  </body>
</html>
```

# Imagens

Quando carregamos a imagem ela carrega no seu tamanho real, para corrigir esses detalhes podemos utilizar alguns atributos como tamanho, o alinhamento, a borda, etc:

Atributos	Descrição
<b>src</b>	Atributo obrigatório, devemos especificar o caminho ou a url da imagem
<b>alt</b>	Apresenta um texto quando a imagem estiver indisponível
<b>height</b>	Define uma altura para a imagem, caso seja uma altura maior que a da imagem ela ajusta a imagem distorcendo a qualidade dela
<b>width</b>	Define uma largura para a imagem, caso seja uma largura maior que a da imagem ela ajusta a imagem distorcendo a qualidade dela
<b>align</b>	<b>absmiddle</b> - alinha a imagem com o centro da linha <b>absbottom</b> - alinha a imagem com a parte de baixo da linha <b>Outros</b> - possui os alinhamentos tradicionais como <b>top</b> , <b>left</b> , <b>right</b> , <b>bottom</b> , <b>baseline</b> , além de <b>middle</b> referente aos demais elementos do html a ela dispostos
<b>border</b>	Define uma espessura para a borda, em pixels
<b>hspace</b>	Define os espaços em pixels, à direita e à esquerda da imagem
<b>vspace</b>	Define os espaços em pixels, acima e abaixo da imagem

# Imagens

```
<html>
  <body>
    <table border="1" width="100%" cellpadding="1" cellspacing="1">
      <tr>
        <td>
          </img>
          Este é um exemplo para border e absmiddle
        </td>
        <td>
          </img>
          Este é um teste para width e height - use um editor para ter a imagem no tamanho desejado!
        </td>
        <td>
          Este é um exemplo do atributo top e do atributo alt com link
          <a href="/index.jsp">
            </img>
          </a>
          </img>
        </td>
      </tr>
    </table>
  </body>
</html>
```

# Aula 2 - HTML Básico

---

# Links

A interligação entre documentos não se restringe somente às ligações com outras páginas. Em páginas muito longas onde um assunto tem vários tópicos, podemos utilizar índices onde os links têm a função de interligar os tópicos de um texto e que com apenas um clique em um dos tópicos do índice, o item é exibido.

# Links - Mesmo diretório

Só precisamos especificar o nome do arquivo que será chamado e a sua extensão.

```
<a href="nomeDoArquivo.html">
```

**Texto ou imagem**

```
</a>
```

Onde:

- A: abertura da tag do link;
- href="nomeDoArquivo.html": deve ser informado o nome completo do arquivo que será acessado;
- Texto ou imagem: que servia como link;
- /a: encerra a tag de link;



# Links - Mesmo diretório

```
<html>
  <body>
    <h1><font face="arial" color="orange">MENU</font></h1>
    <hr width="100%" align="left" size="2" color="silver">
    <a href="EstiloTexto.html">Estilo de Texto</a>
    <br>
    <a href="ExemploTitulo.html">Títulos e Subtítulos</a>
    <br>
    <a href="LinhaHorizontal.html">LinhaHorizontal</a>
    <br>
    <a href="Fontes.html">Fontes</a>
    <br>
    <a href="Imagem.html">Imagem</a>
  </body>
</html>
```

## Links - Para outro diretório

Para criar links para uma página localizada e outros diretórios é necessário indicar o caminho completo do arquivo. Para a web isto tem uma forma um pouco diferente do Windows e do Dos:

- A barra utilizada para separar os diretórios é a barra convencional (/);
- O ponto de partida para localizar um arquivo em outro diretório é o atual;
- Para baixar um nível deve utilizar os sinais “../”;

Exemplo:

```
<a href="../matricula/CadastraAluno.html">
```

**Cadastro de Aluno**

```
</a>
```

## Links - Name

O parâmetro **name** serve para marcar um ponto para possíveis desvios. Quando desviamos para um determinado ponto dentro de um documento, indicamos este nome com um “#”

Por exemplo:

`<a name="aqui">Aqui é um ponto para desvios</a>`

`<a href="#aqui">Desvia para o ponto "AQUI"</a>`

# Formulários

Com o html o cliente (navegador) pode interagir com o servidor, preenchendo campos, clicando em botões e passando informações.

O elemento **form**, da linguagem html, é justamente o responsável por tal interação. Ele provê uma maneira agradável e familiar para coletar dados do usuário através da criação de formulários com janelas de entrada de textos, botões, etc.

# Formulários - Construindo formulários com o form

Para fazer formulários, você tem que colocar as tags `<form></form>`.

Todos os outros comandos, devem ficar dentro dessas tags.

# Formulários - Atributos para form

O elemento form pode conter dois atributos que determinaram para onde será mandada a entrada do **form**. Vejam como eles são:

## Get:

Os dados entrados fazem parte do url associado à consulta enviado para o servidor. Os parâmetros são passados na url.

- Desvantagens
  - Limite de caracteres é de 2.000
  - Os dados enviados são visíveis na barra de endereço do navegador
  - O método POST resolve isso
- Vantagem
  - Pode ser utilizado para passagem de parâmetros por link

# Formulários - Atributos para form

## Post:

É o mais utilizado, pois envia cada informação de forma separada da url. Com este método post os dados entrados fazem parte do corpo da mensagem enviada para o servidor e transfere grande quantidade de dados.

Diferente do GET, o POST envia os dados por meio do corpo da mensagem encaminhada ao servidor

- Vantagens

- Não é visível a cadeia de variáveis [http://www.seusite.com.br/recebe\\_dados.php](http://www.seusite.com.br/recebe_dados.php)
- Não limites no tamanho dos dados, sendo mais usado para formulários com grande quantidade de informações
- Enviar outros tipos de dados, não aceitos pelo GET, como imagens ou outros arquivos (usar valor file na opção type da tag input)

- Desvantagens

- Não é possível a passagem de parâmetros

# Formulários - Input

A tag **<input>** especifica uma variedade de campos editáveis dentro de um formulário. Ele pode receber vários atributos que definem o tipo de mecanismo de entrada (botões, janelas de texto, etc), o nome da variável associada com os dados da entrada, o alinhamento e o campo do valor mostrado.

O atributo mais importante do input é o name. Ele associa o valor da entrada do elemento. Por exemplo, quando você for receber os dados, já, processados, irá vir o name = resposta dada pelo visitante. Outro atributo importante é o type. Ele determina o campo de entradas de dados. Veja como se usa este atributo:

```
<input type="text" name="nome">
```



# Formulários - Input - Text

Para mudar o tamanho, da janela padrão, você tem que colocar o comando **size**.

Por exemplo:

```
<input type="text" name="nome" size=8>
```

Outro comando importante é o value. Ele acrescenta uma palavra digitada no comando à janela.

Por exemplo:

```
<input type="text" name="nome" size=8 value="texto">
```

# Formulários - Input - Number

```
<input type="number" name="quantity">
```

Outros atributos importantes são o min e o max que representa os valores mínimos e máximo que podem ser utilizados pelo campo.

Por exemplo:

```
<input type="number" name="quantity" min="1" max="5">
```

# Formulários - Input - Radio

Quando o usuário deve escolher uma resposta e um única alternativa, de um conjunto, utiliza-se o radio button. Um exemplo típico do uso de tais botões é cuja resposta pode ser **sim** ou **não**. Para isso os inputs que correspondem a mesmo grupo devem conter o mesmo **name**.

Exemplos:

```
<input type="radio" name="você gostou essa home page?"  
value="sim">Sim<br>
```

```
<input type="radio" name="você gostou essa home page?"  
value="nao">Não<br>
```

# Formulários - Input - Checkbox

Esse comando é válido quando for permitido mais de uma resposta para a mesma pergunta. Ele prevê outros botões através dos quais mais de uma alternativa poderá ser escolhida.

Exemplos:

```
<input type="checkbox" name="netscape" value="net">Netscape<br>
```

```
<input type="checkbox" name="netscape" value="exp">Internet explorer<br>
```

```
<input type="checkbox" name="netscape" value="mos">Mosaic<br>
```

```
<input type="checkbox" name="netscape" value="hot">Hot Java<br>
```

# Formulários - Input - Submit

Esse é o botão que submete os dados do formulário quando pressionados, ou seja, possibilitam, o envio, dos dados para o script que vai tratá-los.

Exemplos:

```
<input type="submit" name="enviar" value="enviar">
```

# Formulários - Input - Reset

Esse é o botão serve para limpar os dados de todos os campos do formulário, voltando a sua situação inicial.

Exemplos:

```
<input type="reset" name="limpar" value="Limpar">
```

## Formulários - Textarea

Para se limitar o tamanho do campo, faz-se o uso dos atributos **cols** e **rows** que especificam, respectivamente, o número de colunas e linhas que se deseja mostrar para o usuário. O atributo **name** é obrigatório, e especifica o nome da variável, que será associada à entrada do cliente (navegador).o atributo **value** não é aceito nesse elemento, mas você pode colocar já um texto conforme o exemplo abaixo:

```
<textarea name="nome" cols=20 rows=3>Texto</textarea>
```

# Formulários - Select

Permite definir uma list de opções, com barra de rolagem ou fixa na tela do navegador.

É uma tag que deve ser iniciada com `<select>` e finalizada com `</select>`.

**`<select multiple name="nomeDaLista">`**

**`<option selected value="valor do retorno">Valor Visualizado`**

**`<option value="valor do retorno 2">Valor Visualizado 2`**

**`</select>`**



# Formulários - Select

Onde:

- name: obrigatório, serve para a identificação da lista;
- option: item da lista;
- multiple: com este atributo a lista aparecerá sempre aberta;
- selected: indica o valor padrão da lista;
- value: valor a ser retornado ao servidor.

Exemplo:

```
<select multiple name="estados">
```

```
  <option selected value="SP">São Paulo
```

```
  <option value="RJ">Rio de Janeiro
```

```
  <option value="MG">Minas Gerais
```

```
  <option value="SC">Santa Catarina
```

```
</select>
```

# Css

O que é css?

Css é o mesmo que folhas de estilo. É uma forma de aperfeiçoar seus documentos, o css tem muitos tipos de definição.

No que eles podem nos ajudar?

- Economizar o seu tempo
- Diminuir o tamanho do código de sua página
- Sua página irá carregar mais rápido
- Mais facilidade de manter e fazer alterações na página
- Mais controle no layout da página

# Css - Como criar estilos

Cada estilo que você cria é definido como uma regra css. Cada regra deve utilizar a seguinte sintaxe:

Elemento { atributo1: valor; atributo2: valor ... }

- **Elemento** - descreve o elemento de design ao qual o estilo será aplicado. A mesma tag html mas, sem os sinais de maior e menor.
- **Atributo** - o aspecto específico do elemento que você quer usar como estilo. Deve ser um nome de atributo css válido, como o atributo font-size.
- **Valor** - a configuração aplicada ao atributo. Deve ser uma configuração válida para o atributo em questão, como 20pt (20 pontos) para font-size.
- **Atributo: valor** - à parte declaração da regra. Você pode atribuir múltiplas declarações se desejar separá-los com ponto e vírgula (;). Não coloquem um ponto e vírgula depois da última declaração.

# Css - Como criar estilos

Abaixo todos os títulos de nível 1 (tag <h1>) sejam exibidos em fonte de 36 pontos:

```
h1 { font-size: 36pt }
```

Abaixo todos os títulos de nível 2 (tag <h2>) sejam exibidos em fonte de 24 pontos e cor azul:

```
h2 { font-size: 24pt; color: blue }
```

# Css - Como criar estilos

Você pode inserir quebras de linha e espaços em branco dentro da regra como quiser. Assim, é possível ver mais facilmente todas as declarações e certificar-se de que colocou todos os sinais de ponto e vírgula e colchetes nos lugares corretos.

Por exemplo, aqui está uma regra que diz que os parágrafos aparecerão em fonte Times, 12 pontos, azul e recuados meia polegada a partir da margem esquerda da página:

```
p {  
    font-family: Times;  
    font-size: 12pt;  
    color: blue;  
    margin-left: 0.5in  
}
```

# Css - Tipos de folhas de estilo

Você pode definir regras de css em três lugares. E, por definição, pode utilizar uma combinação dos três métodos nos seus htmls.

- **Externo:** em um documento separado fora de todos os documentos html;
- **Incorporado:** no cabeçalho de um documento html;
- **Inline:** dentro de uma tag html. Cada um destes métodos tem um nome e afeta as páginas html em seu site de um modo diferente.

# Css - Externo

Para definir um conjunto de regras de estilo que você pode facilmente aplicar em alguma página do seu site, é preciso colocar as regras em um arquivo de texto com a extensão **.css**.

Sempre que quiser utilizar esses estilos em uma nova página, basta colocar uma tag >link> no cabeçalho que referencie esse arquivo .css. Veja o exemplo abaixo:

Arquivo **OrgaoColegiado.css**:

```
H4 {  
    font-family: 'Arial';  
    font-size: 14pt;  
    color: blue  
}
```

## Css - Externo

Agora, para utilizar os estilos definidos neste arquivo .css você precisa adicionar a tag a seguir ao cabeçalho da página, onde nome\_do\_arquivo é uma referência absoluta ou relativa ao arquivo .css.

**<link rel="stylesheet" href="OrgaoColegiado.css" type="text/css">**

Obs: Você deve inserir este texto entre as tags <head> ... </head>, e colocar a localização correta do seu arquivo e seu nome.



# Css - Interno

Se você criar um conjunto de estilos que se aplica a uma única página, você pode configurar os estilos exatamente como fizemos no exemplo dos estilos externos, mas em vez de colocar as regras em um arquivo separado, colocaremos as tags `<style type="text/css"> ... </style>` dentro do nosso arquivo html entre as tags `<head> ... </head>`

```
<html>
  <head>
    <title>Exemplo Estilos Incorporados</title>
    <style type="text/css">
      p {
        background-color: #FFFFFF;
        font-family: 'Comic Sans MS';
        font-size: 14pt
      }
    </style>
  </head>
</html>
```

## Css - Inline

Os estilos inline são os que têm menos efeitos. Eles afetam somente a tag atual – não as outras tags na página e tampouco outros documentos. A sintaxe para definir um estilo inline é a seguinte:

```
<a style="color: gree; text-decoration: none" href="http://www.globo.com">
```

Note que em v das tags `<style> ... </style>`, você apenas utiliza um atributo `style` dentro da tag para definir o estilo. E, em vez de colocar as regras de css entre colchetes, você as coloca entre aspas, separando-as com ponto e vírgula como de costume.

# Css - Tags personalizadas

Com as classes de estilo, é possível definir diversas variações de uma única tag. Por exemplo, você poderia fazer um estilo de parágrafo “texto alinhado à direita”, um estilo de parágrafo “texto centralizado” e assim por diante, criando múltiplos temas em torno da tag de parágrafo (<p>).

Você pode definir classes de estilo tanto em folhas de estilo externa como nas internas. (Porém não faz sentido definir uma classes em um estilo inline) A sintaxe é praticamente idêntica à sintaxe normal para os estilos externos e internos, com adição de um ponto e o nome da classe depois do elemento na qual será utilizado o atributo.

**nomeDaClasse {atributo: valor; ...}**

# Css - Tags personalizadas

Exemplo:

```
<style type="text/css">  
    .meusLinks {  
        color: red;  
        text-decoration: none  
    }  
</style>
```

Depois apenas adiciono o atributo class="meusLinks" aos links em que eu desejar que fiquem vermelho e não sublinhados. Veja como ficaria:

```
<a class="meusLinks" href="http://www.globo.com">  
    Globo  
</a>
```

# Css - Div

As tags `<div> ... </div>` podem ser usadas para formatar um grande bloco de texto - uma divisão - abrangendo diversos parágrafos e outros elementos. Isso as torna uma boa opção para definir estilos que afetam grandes seções de um texto em uma página. Veja:

```
<style type="text/css">
```

```
    sidebar {
```

```
        font-family: "Arial;"
```

```
        font-size: 12pt;
```

```
        text-align: right;
```

```
        background-color: #cococo;
```

```
        margin-left: 1in;
```

```
        Margin-right: 1in;
```

```
    }
```

```
</style>
```

# Css - Div

Ao colocar na tag <div> o atributo class, você estará fazendo com que todos os elementos que estejam englobados nesta tag sigam estes padrões:

```
<html>
  <body>
    <div class="sidebar">
      <font>Texto de teste</font>
      <br>
      <p>Parágrafo com texto</p>
      <br>
      <a href="http://www.globo.com">Globo</a>
      <br>
      <a href="http://www.google.com">Google</a>
    </div>
    <div>
      <font>Texto de teste</font>
      <br>
      <p>Parágrafo com texto</p>
      <br>
      <a href="http://www.globo.com">Globo</a>
      <br>
      <a href="http://www.google.com">Google</a>
    </div>
  </body>
</html>
```

# Css - Div

Ao colocar na tag <div> o atributo class, você estará fazendo com que todos os elementos que estejam englobados nesta tag sigam estes padrões:

```
<html>
```

```
  <body>
```

```
    Curso:
```

```
    <ol>
```

```
      <li>Selecione
```

```
      <div class="sidebar">Maemátca</div>
```

```
      <br>
```

```
      <div class="sidebar">Medicina</div>
```

```
      <br>
```

```
      <div>Literatura</div>
```

```
    </ol>
```

```
  </body>
```

```
</html>
```

# Css - Span

As tags `<span> ... </span>` são como as tags `<div> ... </div>` no sentido de que você pode utilizá-las para definir estilo que formatam um bloco de texto. Ao contrário de `<div>`, contudo, que é utilizada para divisões de textos grandes, a tag `<span>` é especializada para bloco de textos menores – que podem ser tão pequenos como um único caractere

```
<style type="text/css">
    hot {
        color: green;
        Text-decoration: underline
    }
</style>
<html>
    <body>
        Para sair de um programa:
        <ol>
            <li>Selecione <span class="hot">A</span>rquivo - <span class="hot">S</span>air</li>
        </ol>
    </body>
</html>
```



# Css - Lista de atributos css

Lista com atributos css para consultar:

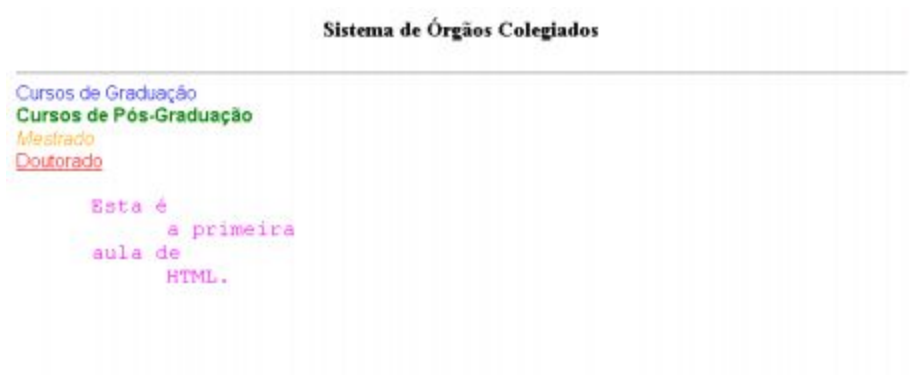
[https://www.oficinadanet.com.br/artigo/1041/atributos\\_css](https://www.oficinadanet.com.br/artigo/1041/atributos_css)

Material complementar:

<http://learnlayout.com/index.html>

# Exercícios

1. Crie uma nova página chamada Exercicio1.html e deixe-a com o seguinte layout:



- a) O cabeçalho “Sistema de Órgãos Colegiados” deve ter tamanho 3 e ficar centralizado.
- b) Após o cabeçalho incluir uma linha horizontal tamanho 2 e cor cinza.
- c) O texto “Curso de Graduação” deve ser Negrito, fonte Arial, tamanho 3 e cor azul.

# Exercícios

- d) O texto “Curso de Pós Graduação” deve ser Negrito, fonte Arial, tamanho 3 e cor verde.
- e) O texto “Mestrado” deve ser Itálico, fonte Arial, tamanho 3 e cor laranja.
- f) O texto “Doutorado” deve ser Sublinhado, fonte Arial, tamanho 3 e cor vermelha.
- g) O texto “Esta é a primeira aula de html” deve ser um texto pré formatado (colocam o formato dela direto no html) como mostrado na figura, fonte Arial, tamanho 5 e cor Magenta.

# Exercícios

2. Crie uma página chamada Exercicio2.html e deixe-a com o seguinte layout:

Manutenção do Órgão Colegiado

Unidade	FACULDADE DE ENGENHARIA DE ILHA SOLTEIRA
Tipo Colegiado	CONGREGAÇÃO

Órgão Colegiado

Nome	CONGREGAÇÃO DA FACULDADE DE ILHA SOLTEIRA		
Sigla	CFIS	Status	ATIVADO
Data Ativação	15/10/2000	Data Desativação	
Órgão de Origem		Buscar	
Órgão Superior		Buscar	
Documentação	<div>Órgão criado com o objetivo de aprovar normas da unidade.</div>		
Salvar		Excluir	

Retornar

# Exercícios

3. Com base no exercício 2, criar um arquivo css e alterar o layout para o mais próximo do a seguir:

## Manutenção do Órgão Colegiado

Unidade	FACULDADE DE ENGENHARIA DE ILHA SOLTEIRA
Tipo Colegiado	CONGREGAÇÃO

### Órgão Colegiado

Nome	CONGREGAÇÃO DA FACULDADE DE ILHA SOLTEIRA		
Sigla	CFIS	Status	ATIVADO
Data Ativação	15/10/2000	Data Desativação	
Órgão de Origem		Buscar	
Órgão Superior		Buscar	
Documentação	Órgão criado com o objetivo de aprovar normas da unidade.		

Salvar

Excluir

# HTML Básico com servidor NodeJS

---

# JavaScript

Também chamada de JS, é a linguagem de criação de scripts para a Web.

É utilizado por bilhões de páginas para:

- Adicionar funcionalidades;
- Verificar formulários;
- Comunicar com servidores;
- E muitos mais.

Java e JavaScript são “coisas” completamente distintas e desconexas.

JavaScript reside dentro de documentos HTML e pode prover diferentes níveis de interatividades não suportados pelo HTML sozinho.

# JavaScript

Diferenças chaves em relação ao Java:

- Java é uma linguagem de programação;
- JavaScript é uma linguagem de script;
- Aplicativos Java são executados pela máquina virtual Java;
- Scripts JavaScript são executados pelos browsers;
- Java é compilado e interpretado;
- JavaScript é apenas interpretado pelos browsers;



# JavaScript

Atualmente, o maior mantedor da linguagem é a Fundação Mozilla.

Encontramos ótimos materiais e tutoriais sobre JavaScript na W3School, mas também encontramos referência completa do JavaScript no site do Mozilla:

<https://developer.mozilla.org/en/docs/JavaScript>

# JavaScript - tag

Para inserir códigos JavaScript, iremos fazê-lo em uma Tag HTML apropriada:

Assim como o css, podemos ter o javascript de 2 formas:

Externo em arquivo separado (importamos o arquivo conforme abaixo):

```
<script src="meuScript.js"></script>
```

Interno dentro do nosso próprio html:

```
<script>
```

```
...
```

```
</script>
```

# JavaScript

Vamos criar uma nova página.

Dentro da seção <body> insira o trecho:

```
<script>  
    document.write(“Hello World!”);  
</script>
```

# JavaScript - document

Propriedade de Exemplo:

- **title**: Define ou retorna o título da página;
- **url**: Retorna o url completo da página;

Métodos de exemplo:

- **write()** - Escreve texto no documento;
- **writeln()** - Escreve uma linha de texto no documento;

# JavaScript - document

```
<html>
  <head>
    <script>
      document.title="JavaScript Hello World!"
    </script>
  </head>
  <body>
    <script>
      document.write("<h2>" + document.title + "</h2>")
    </script>
  </body>
</html>
```

# JavaScript - Eventos - Clique em um botão

É possível disparar scripts a partir de diversos tipos de eventos.

```
<html>
```

```
  <body>
```

```
    <button type="button" onclick="alert('Bem vindo!')">Clique
```

```
    Aqui!</button>
```

```
  </body>
```

```
</html>
```

# JavaScript - Eventos - Clique em um botão

É possível disparar scripts a partir de diversos tipos de eventos.

```
<html>
  <head>
    <script>
      Function minhaFuncao() {
        var x = document.getElementById("paragrafo");
        x.innerHTML="Hello!";
      }
    </script>
  </head>
  <body>
    <p id="paragrafo">Olá!</p>
    <button type="button" onclick="minhaFuncao()">Clique Aqui!</button>
  </body>
</html>
```

# JavaScript - Alterando um atributo

```
<html>
  <head>
    <script>
      Function trocaImagem() {
        var elemento=document.getElementById("myImage");
        if (elemento.src.match("bulbon")) {
          elemento.src="pic_bulboff.gif";
        } else {
          elemento.src="pic_bulbon.gif";
        }
      }
    </script>
  </head>
  <body>
    <p></p>
    <p>Clique na lâmpada para ligar/desligar a luz</p>
  </body>
</html>
```



# JavaScript - Variáveis

JavaScript é uma linguagem de tipagem dinâmica e fraca:

- Não é necessário declarar o tipo de uma variável;
- Números são todos reais de 64bits;
- A variável irá “alterar” o seu tipo de dado conforme os valores forem atribuídos:
  - Tipo e dado dinâmico:

`var x; //x é indefinido`

`x = 5; //x é um número`

`x = “John”; //x é uma string`

`x = true; //x é um valor lógico`

`x = null; //x é indefinido`

# JavaScript - Recebendo uma entrada

```
<html>
  <head>
    <script>
      Function soma() {
        var elemento1=parseInt(document.getElementById("v1").value);
        var elemento2=parseInt(document.getElementById("v2").value);
        document.getElementById("res").innerHTML = "Resposta: " + (elemento1 +
          elemento2);
      }
    </script>
  </head>
  <body>
    <input type="number" name="v1" id="v1" min="1" max="100">
    <input type="number" name="v2" id="v2" min="1" max="100">
    <p id="res">Resposta: </p>
    <button type="button" onclick="soma()">Soma</button>
  </body>
</html>
```

# JavaScript - Validação de Formulários

Ao desenvolver aplicativos para a internet, dados serão informados pelo usuário. Antes de enviar estes dados ao servidor, é possível validar/verificar se eles tem coerência em relação ao que é solicitado:

- O usuário esqueceu campos em branco?
- O e-mail digitado é válido?
- A data digitada é válida?
- Num campo numérico, foi digitado um número?

# JavaScript - Validação de Formulários

```
<html>
  <head>
    <script>
      Function validarForm() {
        var val=document.getElementById("valido");
        try {
          var x = document.forms["meuForm"]["nome"].value;/
          if (x == null || x == "") {
            Throw "O Nome deve ser preenchido!"
          }
          var y = document.forms["meuForm"]["email"].value;
          var atpos = y.indexOf("@");
          var dotpos = y.lastIndexOf(".");
          if (atpos < 1 || dotpos < atpos + 2 || dotpos + 2 >= y.length) {
            throw "Digite um e-mail válido!"
          }
          return true;
        } catch (err) {
          val.style.color = "#FF0000";
          val.innerHTML = "Erro: " + err;
          return false;
        }
      }
    </script>
  </head>
  <body>
    <form name="meuForm" onsubmit="return validarForm();" method="post">
      Nome: <input type="text" name="nome">
      e-mail: <input type="text" name="email">
      <button type="submit">Enviar</button>
    </form>
    <p id="valido">Preencha o formulário e clique em Enviar</p>
  </body>
</html>
```

# JavaScript - Adicionando elementos

- É possível adicionar novos elementos html;
- Qualquer tipo de elemento, definindo qualquer propriedade;
- Tudo através do JavaScript;

# JavaScript - Adicionando elementos

```
<html>
  <body id="corpo">
    <h1>Adicionar Elementos</h1>
    <p>Digite o texto: <input type="text" id="texto"></p>
    <p><button onclick="adicionar()">Adicionar</button></p>
    <script>
      function adicionar() {
        var texto = document.getElementById("texto").value;
        var para = document.createElement("p");
        para.innerHTML = texto;
        var corpo = document.getElementById("corpo");
        corpo.appendChild(para);
      }
    </script>
  </body>
</html>
```

# JavaScript - Removendo elementos

- É possível remover elementos html;
- Qualquer tipo de elemento, com a condição de que conheçamos também o seu pai;
- Tudo através do JavaScript;

# JavaScript - Removendo elementos

```
<html>
  <body id="corpo">
    <h1>Remover Elementos</h1>
    <p id="texto">Texto que será removido</p>
    <button onclick="remover()">Remover!!!!</button>
    <script>
      function remover() {
        var pai = document.getElementById("corpo").value;
        var filho = document.getElementById("texto").value;
        pai.removeChild(filho);
      }
    </script>
  </body>
</html>
```



# JavaScript - Exercícios

1) Crie um formulário de inscrição de candidatos em uma faculdade que possua, entre outros pontos relevantes, os seguintes itens:

- Nome (Campos obrigatórios);
- Telefone (Campos obrigatórios);
- Endereço completo do aluno (Campos obrigatório);
- Filiação;
- Curso (dentre os oferecidos pela faculdade, uma combo) (Campo obrigatório);
- Estado civil;
- Idade (Campo obrigatório);
- Renda Familiar (Campo obrigatório);
- Escolha de bolsa;
- Justificativa da bolsa (caso selecionado uma bolsa) (Campos obrigatório caso selecionado uma bolsa);

**Não se esqueçam de realizar a validação de todos os campos obrigatórios.**

# JavaScript - Exercícios

- 2) Crie uma calculadora, na qual terão:
- Campo numérico valor 1. (Obrigatório)
  - Campo select com as operações possíveis. (Obrigatório)
  - Campo numérico valor 2. (Obrigatório)
  - Apresentar um campo texto com o resultado.

**Não se esqueçam de realizar a validação de todos os campos obrigatórios e calcular os resultados.**

# Javascript

Abaixo um pdf com informações mais completas de javascript:

<https://www.caelum.com.br/download/caelum-html-css-javascript.pdf>

[http://conteudo.icmc.usp.br/CMS/Arquivos/arquivos\\_enviados/BIBLIOTECA\\_113\\_ND\\_72.pdf](http://conteudo.icmc.usp.br/CMS/Arquivos/arquivos_enviados/BIBLIOTECA_113_ND_72.pdf)

# Incluindo nossa aplicação NodeJs no Heroku

---

# Bootstrap

**BootStrap** é um framework CSS muito popular para estilização de sites e aplicativos móveis. Criado pela equipe do Twitter como uma forma de padronizar a forma com que seus desenvolvedores programassem.

<http://getbootstrap.com/getting-started/>

# Bootstrap

```
<html>
```

```
  <head>
```

```
    <script src="https://code.jquery.com/jquery-3.3.1.slim.min.js"></script>
```

```
    <link rel="stylesheet"
```

```
href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css">
```

```
    <script
```

```
src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js"
```

```
></script>
```

```
  </head>
```

```
</html>
```

# Bootstrap

Todo conteúdo de uma página estilizada com BS deve conter uma DIV com uma classe container. Essa DIV container passará a ser a DIV principal da sua página e nela que irei acrescentar todo o conteúdo do site.

```
<body>  
  <div class="container">  
    </div>  
</body>
```

# Bootstrap

A DIV Container pode ser dividida em linhas usando a classe ROW.

```
<body>  
  <div class="container">  
    <div class="row">  
      Linha 1  
    </div>  
    <div class="row">  
      Linha 2  
    </div>  
  </div>  
</body>
```



# Bootstrap

As linhas (rows) também podem ser divididas em colunas usando a classe COL-MD, mas, deve-se observar que o máximo de colunas que consigo trabalhar são 12.

```
<body>
  <div class="container">
    <div class="row">
      <p>Linha dentro de um container.</p>
      <div class="col-md-3">
        Coluna 1 Linha 1
      </div>
      <div class="col-md-3">
        Coluna 2 linha 1
      </div>
      <div class="col-md-3">
        Coluna 3 linha 1
      </div>
    </div>
    <div class="row">
      Linha 2
    </div>
  </div>
</body>
```

# Bootstrap - Exemplo de um e-commerce usando bootStrap

Seguindo os 4 passos mostrados acima, vamos criar uma página simples de e-commerce usando a estrutura de DIVs:

<https://github.com/grazielags/cursoHtml1/blob/master/Aula3/eCommerce.html>

# Bootstrap - Ícones do Bootstrap

BS apresenta uma boa coleção de ícones, também chamados de *Glyphicons*. toda a coleção de Glyphicons podem ser consultada aqui;

<https://getbootstrap.com/docs/3.3/components/>

Para acrescentar um ícone de um carrinho de compras no botão usamos o exemplo mostrado na página, através de uma tag SPAN:

```
<button class="btn btn-danger" type="button"> <span class="glyphicon glyphicon-shopping-cart"></span> Comprar </button>
```

# Bootstrap - Bordas nas imagens com Bootstrap

Pequenos detalhes realmente fazem a diferença. O BS oferece algumas classes de estilização conforme exemplos mostrados aqui:

<http://getbootstrap.com/css/#images>

Neste exemplo, vou usar a class `img-thumbnail` para criar uma borda envolta da imagem, o código ficará da seguinte forma:

```

```

# Bootstrap

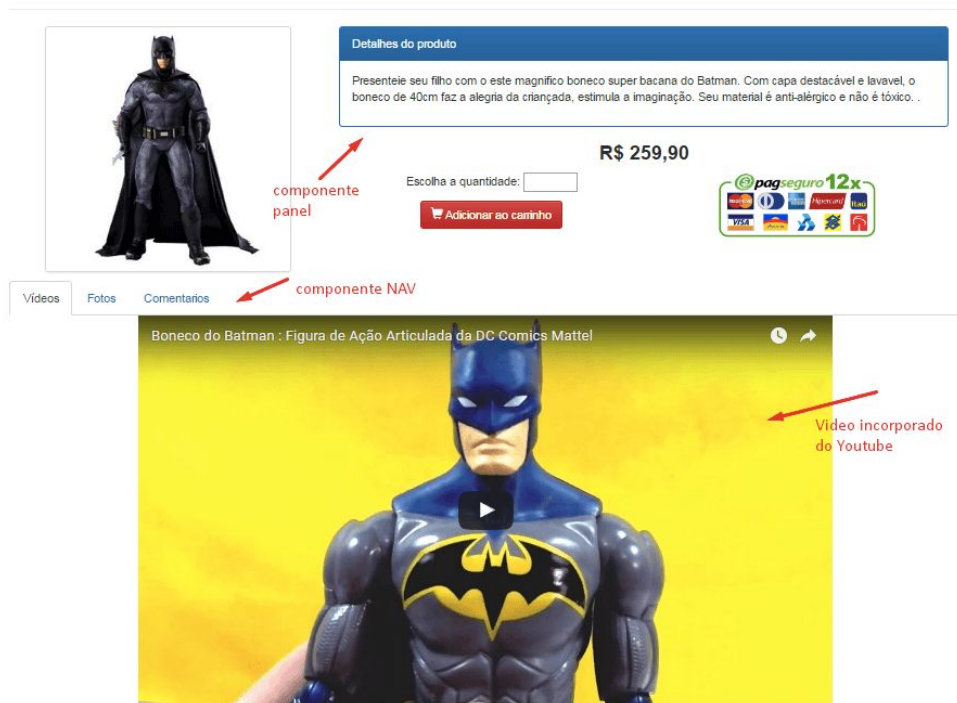
Abaixo a documentação do bootstrap para uma lida um pouco mais detalhada:

<https://getbootstrap.com/docs/4.3/layout/overview/>

# Bootstrap - Exercício

Seguindo o exemplo usado em aula, vamos criar agora uma página para produto com o visual semelhante a este:

Super boneco do Batman



# Bootstrap - Exercício

aba fotos:

Usando as classes *img*, *img-responsive* e *img-thumbnail*

Vídeos Fotos Comentários



# Bootstrap - Exercício


aba comentários:

Vídeos

Fotos

Comentarios

---




Anderson

★★★★☆

Nossa! Show de boneco....

---




Bruce

★★★★★

Esse boneco ficou louco d+... comprei 10 exemplares

---



Coringa

★★★★☆

Meu Deus.... ficou bonitinho, mas dá medo!



# Bootstrap - Exercício

O que deve ser feito?

1. Criar uma página semelhante a mostrada acima.
2. Utilizar o Google WebFonts.
3. Estilizar a página com BootStrap.
4. Utilizar as classes de DIV container, row e col-md para criar o layout da página.
5. Utilizar glapicon no botão de compra.
6. Pesquisar na documentação do Bootstrap como utilizar o componente panel e implementar na sua página. Veja um exemplo aqui:  
[https://www.w3schools.com/bootstrap/bootstrap\\_panels.asp](https://www.w3schools.com/bootstrap/bootstrap_panels.asp)
7. Pesquisar na documentação do Bootstrap como utilizar o componente nav e implementar na sua página. Veja um exemplo aqui:  
[http://www.w3schools.com/bootstrap/tryit.asp?filename=trybs\\_tabs\\_dynamic&sticked=h](http://www.w3schools.com/bootstrap/tryit.asp?filename=trybs_tabs_dynamic&sticked=h)
8. Incorporar um vídeo do Youtube na página. Basta clicar em compartilhar ➤ incorporar, lá copie o html e cole no seu site.

# Como praticar desenvolvimento Frontend?

<http://gabsferreira.com/maneiras-de-se-praticar-frontend/>

# NodeJS

Para instalar o NodeJS é tudo bem simples, basta você entrar nesse link:

<https://nodejs.org/en/download>

Em seguida fazer o download para o seu respectivo sistema operacional e seguir o processo de instalação.

# NodeJS

Começaremos criando um arquivo chamado server.js. Inicialmente nosso arquivo server.js ficará assim:

```
const http = require('http')  
const port = 5000  
const ip = 'localhost'  
  
const server = http.createServer((req, res) => {  
  console.log('Recebendo uma request!')  
  res.end('<h1>Aqui fica o que vamos enviar para o navegador como resposta!</h1>')  
})  
  
server.listen(port, ip, () => {  
  console.log(`Servidor rodando em http://${ip}:${port}`)  
  console.log('Para derrubar o servidor: ctrl + c');  
})
```

# NodeJS

Agora temos um arquivo JS com os comandos necessários para criar o nosso servidor HTTP, no nosso próximo passo precisamos fazer com que a nossa plataforma NodeJS execute nosso código (server.js). Para conseguirmos fazer isso, abra o seu terminal e dentro dele execute os seguintes comandos:

Acesse o diretório no qual foi criado a pasta do exemplo:

**cd app**

**node server.js**

Mensagem abaixo quando o servidor estiver rodando:

**Servidor rodando em <http://localhost:5000>**

**Para derrubar o servidor: ctrl + c**

# NodeJS

Abra o seu navegador e acesse o <http://localhost:5000>

Se tudo estiver ok, a mensagem a seguir será apresentada no terminal ao acessar a url:

```
res.end('<h1>Aqui fica o que vamos enviar para o navegador como resposta!</h1>')
```

# NodeJS - Respondendo a rotas

Toda vez que o nosso servidor HTTP recebe uma requisição a função de callback que passamos para o método `createServer` é executada, por isso vamos colocar as condições para respostas diferentes conforme o endereço da requisição. Vamos começar criando uma resposta para a nossa home (`http://localhost:5000/`):

`server.js`:

```
const http = require('http')
const port = 5000
const ip = 'localhost'
```

```
const server = http.createServer((req, res) => {
  if (req.url === '/') {
    res.end('<h1>Home</h1>')
  }

  res.end('<h1>URL sem resposta definida!</h1>')
})
```

```
server.listen(port, ip, () => {
  console.log(`Servidor rodando em http://${ip}:${port}`)
  console.log('Para derrubar o servidor: ctrl + c');
})
```

# NodeJS - Respondendo a rotas

Já vimos que toda vez que o nosso servidor recebe uma requisição HTTP ele executa o callback que passamos para o `createServer`, outra coisa que é bem interessante é que quando o NodeJS cai em um `res.end` tudo que está a seguir dele continua a ser executado e a resposta já foi enviada, isso não é bom e nem ruim é apenas algo que precisamos lembrar quando estamos desenvolvendo.



# NodeJS - Respondendo a rotas

Server.js:

```
const http = require('http')
```

```
const port = 5000
```

```
const ip = 'localhost'
```

```
const server = http.createServer((req, res) => {  
  if (req.url === '/') {  
    res.end('<h1>Home</h1>')  
  }  
  if (req.url === '/inscreva-se') {  
    res.end('<h1>Inscreva-se</h1>')  
  }  
  if (req.url === '/local') {  
    res.end('<h1>Local</h1>')  
  }  
  if (req.url === '/contato') {  
    res.end('<h1>Contato</h1>')  
  }  
  res.end('<h1>URL sem resposta definida!</h1>')  
})
```

```
server.listen(port, ip, () => {  
  console.log(`Servidor rodando em http://${ip}:${port}`)  
  console.log('Para derrubar o servidor: ctrl + c');  
})
```

# NodeJS - Respondendo a rotas

Conseguimos criar uma resposta para as 4 rotas que nos propomos no começo do post. Apesar do código estar bem legível, eu não faria com if, nesse caso eu utilizaria um Array:

server.js:

```
const http = require('http')
```

```
const port = 3000
```

```
const ip = 'localhost'
```

```
const server = http.createServer((req, res) => {  
  const responses = []  
  responses['/'] = '<h1>Home</h1>'  
  responses['/inscreva-se'] = '<h1>Inscreva-se</h1>'  
  responses['/local'] = '<h1>Local</h1>'  
  responses['/contato'] = '<h1>Contato</h1>'  
  responses['/naoExiste'] = '<h1>URL sem resposta definida!</h1>'  
  res.end(responses[req.url] || responses['/naoExiste'])  
})
```

```
server.listen(port, ip, () => {  
  console.log(`Servidor rodando em http://${ip}:${port}`)  
  console.log('Para derrubar o servidor: ctrl + c');  
})
```

# NodeJS

Tanto o código com if ou array, são códigos complicados de se manter. Por esse motivo e outros que a comunidade de JavaScript começou a criar frameworks para cuidar das rotas. Nos próximos posts vamos ver como criar rotas com os frameworks:

1. Restify
2. ExpressJS
3. HapiJS
4. RoaJS

Vamos utilizar o ExpressJS.

# NodeJS - ExpressJS

Vamos criar 3 rotas com a seguinte estrutura:

Na nossa primeira rota só mostraremos uma página com um texto escrito Home dentro de uma tag h1 — `http://localhost:5000/` (GET);

Na segunda rota teremos a exibição de um formulário com 2 campos (email e mensagem) — `http://localhost:5000/contato` (GET);

Na terceira rota utilizaremos o mesmo path (`/contato`) mas preparada para responder ao método POST — `http://localhost:5000/contato` (POST).

# NodeJS - ExpressJS

No arquivo server.js faremos o código que será responsável por criar o nosso servidor HTTP:

```
let express = require('express')
let app = express()
let port = 5000

app.listen(port, () => {
  console.log(`Servidor rodando em http://localhost:${port}`)
  console.log('Para derrubar o servidor: ctrl + c');
})
```

Abra o terminal e digite o seguinte comando:  
**node server.js**

# NodeJS - ExpressJS

Na primeira linha estamos importando um module com o nome express que não instalamos no projeto. Felizmente quando instalamos o NodeJS temos de graça um gerenciador de pacote chamado npm (Node Package Manager) e com ele fica muito simples instalar as dependências do nosso projeto.

# NodeJS - Instalando e definindo dependências em um projeto NodeJS

Antes de instalar e definirmos o express como dependência do nosso projeto, precisamos criar um arquivo chamado package.json na raiz do nosso projeto que terá algumas informações do nosso projeto como:

- Nome;
- Versão;
- Descrição;
- Respósitorio git remoto;
- Autor;
- Licença;
- Dependências;
- e outras coisas...

# NodeJS - Instalando e definindo dependências em um projeto NodeJS

Felizmente não precisamos criar o package.json na mão, podemos utilizar o npm. Para criar o package.json pelo npm, abra o terminal e execute o seguinte comando (isso deve ser feito dentro da pasta app):

```
npm init
```

Agora que temos o nosso package.json criado, podemos instalar o express e definir ele como dependência. Abra novamente o terminal e execute:

```
npm install express --save
```



# NodeJS - Instalando e definindo dependências em um projeto NodeJS

Apenas com essa linha de comando que acabamos de executar no terminal fizemos o download do express para o nosso projeto e, por causa do parâmetro `--save`, ele também foi definido como dependência. Se você abrir o `package.json` verá que temos uma nova chave chamada `dependências` com o `express` como valor.

Se tudo estiver certo basta irmos até o terminal e executar o comando para a nossa plataforma NodeJS executar o `server.js`:

**`node server.js`**

**Obs:** Execute esse comando dentro da pasta `app`.

# NodeJS - Express

Após executar o comando anterior você receberá a seguinte saída no terminal:

Servidor rodando em `http://localhost:5000`

Para derrubar o servidor: `ctrl + c`

Só pra ver se tudo está funcionando, vá até o navegador e entre em <http://localhost:5000>, deverá aparecer no navegador a seguinte mensagem: **Cannot GET /**. Isso acontece porque o express usa essa resposta por padrão para rotas que não foram definidas.

# NodeJS - Express - Rotas

Agora que temos o nosso servidor express funcionando, só precisamos criar as 2 rotas que nos propomos no começo do post:

1. `http://localhost:5000/` (GET)
2. `http://localhost:5000/contato` (GET)

# NodeJS - Express - Rotas

Vamos abrir o nosso **server.js** e criar uma resposta para a primeira rota. A nossa resposta conterá apenas uma tag **h1** com o conteúdo Home:

```
let express = require('express')  
let app = express()  
let port = 5000
```

```
app.get('/', (req, res) => {  
  res.send('<h1>Home</h1>')  
})
```

```
app.get('/contato', (req, res) => {  
  res.send('<h1>Contato</h1>')  
})
```

```
app.listen(port, () => {  
  console.log(`Servidor rodando em http://localhost:${port}`)  
  console.log('Para derrubar o servidor: ctrl + c');  
})
```

# NodeJS - Express - Rotas

```
let express = require('express')
```

```
let app = express()
```

```
let port = 5000
```

```
app.use(express.static('pages'));
```

```
app.listen(port, () => {
```

```
  console.log(`Servidor rodando em http://localhost:${port}`)
```

```
  console.log('Para derrubar o servidor: ctrl + c');
```

```
})
```

Usando o servidor dessa forma, o conteúdo estará estático, dessa forma as páginas serão acessadas como html estáticos, de acordo com as urls que são passadas no componente.

# NodeJS - Express

Todas as páginas **.html** devem estar dentro da pasta **pages**.

# NodeJS - Exercício

1. Criar um site, conforme a imagem abaixo, usando os link abrindo ou outra página, ou mesmo uma outra seção:



ABOUT

EXPERIENCE

EDUCATION

SKILLS

INTERESTS

AWARDS

## CLARENCE TAYLOR

3542 BERRY STREET · CHEYENNE WELLS, CO 80810 · (317) 585-8468 · [NAME@EMAIL.COM](mailto:NAME@EMAIL.COM)

I am experienced in leveraging agile frameworks to provide a robust synopsis for high level overviews. Iterative approaches to corporate strategy foster collaborative thinking to further the overall value proposition.



# **HTML com um Framework de mercado (Angular), rotas e Conexão com o Banco de dados**

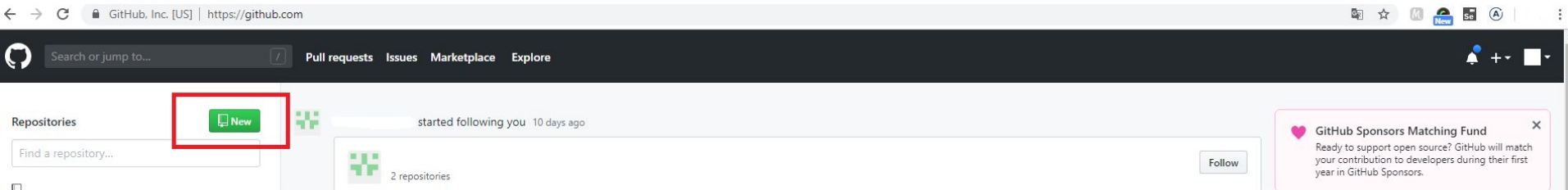
---



# NodeJS - Incluindo nosso site que acabamos de criar no Heruko

Precisamos colocar esse nosso site no github, pois o Heroku irá buscar do nosso repositório do github.

Primeira coisa que devemos fazer é criar um repositório novo lá no github:



# NodeJS - Github

## Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere?

[Import a repository.](#)

Owner

grazielags ▾

Repository name \*

cursoHtmlNode



Great repository names are short and memorable. Need inspiration? How about **friendly-rotary-phone**?

Description (optional)

☒ Public



Anyone can see this repository. You choose who can commit.

☐ Private



You choose who can see and commit to this repository.

Skip this step if you're importing an existing repository.

☐ Initialize this repository with a README

This will let you immediately clone the repository to your computer.

Add .gitignore: None ▾

Add a license: None ▾



Create repository

# NodeJS - Github

grazielags / cursoHtmlNode

Watch 0 Star 0 Fork 0

<> Code Issues 0 Pull requests 0 Projects 0 Wiki Security Insights Settings

### Quick setup — if you've done this kind of thing before

Set up in Desktop or HTTPS SSH `https://github.com/grazielags/cursoHtmlNode.git`

Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

### ...or create a new repository on the command line

```
echo "# cursoHtmlNode" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin https://github.com/grazielags/cursoHtmlNode.git
git push -u origin master
```

### ...or push an existing repository from the command line

```
git remote add origin https://github.com/grazielags/cursoHtmlNode.git
git push -u origin master
```

### ...or import code from another repository

You can initialize this repository with code from a Subversion, Mercurial, or TFS project.

Import code

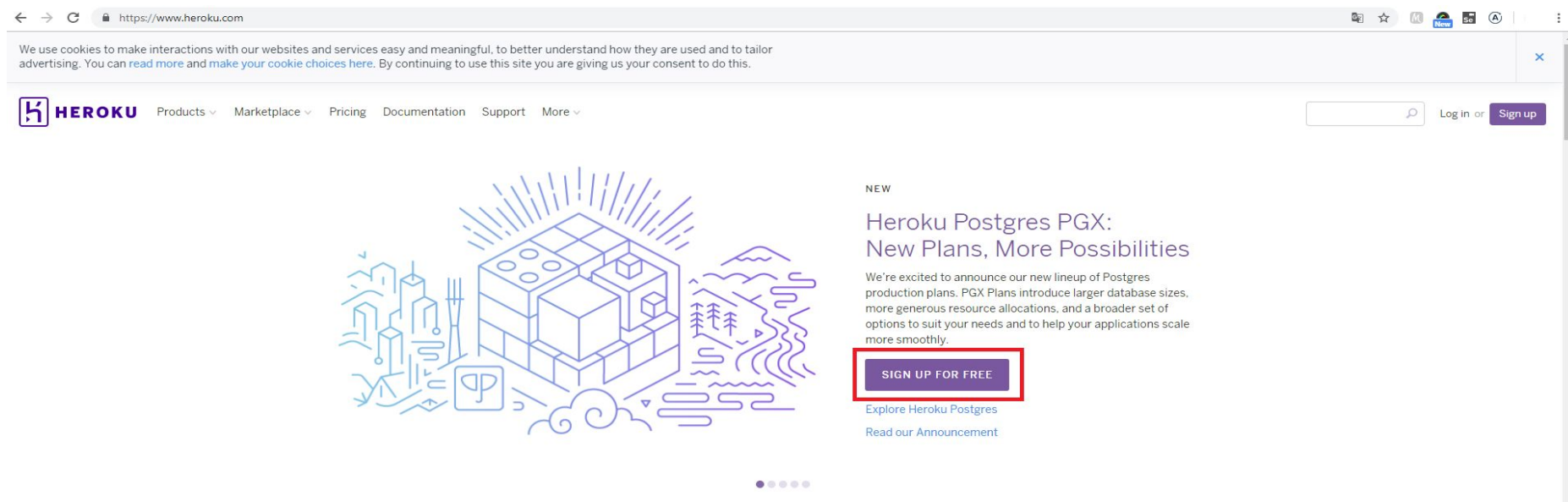
# NodeJS - Criando a conta no Heroku

Precisaremos criar uma conta no heroku, para pode incluir nossa aplicação lá:

Acessem:

<https://www.heroku.com/>

E seguir os passos a seguir:



The screenshot shows the Heroku website homepage. At the top, there is a navigation bar with the Heroku logo and links for Products, Marketplace, Pricing, Documentation, Support, and More. A search bar and 'Log in' or 'Sign up' buttons are also present. The main content area features a large illustration of a cityscape with a central building that has a glowing top, surrounded by trees and clouds. To the right of the illustration, there is a section titled 'NEW Heroku Postgres PGX: New Plans, More Possibilities'. Below this title, there is a paragraph of text and a prominent 'SIGN UP FOR FREE' button, which is highlighted with a red rectangle. At the bottom of the page, there are links to 'Explore Heroku Postgres' and 'Read our Announcement'.

We use cookies to make interactions with our websites and services easy and meaningful, to better understand how they are used and to tailor advertising. You can [read more](#) and [make your cookie choices here](#). By continuing to use this site you are giving us your consent to do this.

**HEROKU** Products Marketplace Pricing Documentation Support More

Log in or [Sign up](#)

**NEW**

## Heroku Postgres PGX: New Plans, More Possibilities


We're excited to announce our new lineup of Postgres production plans. PGX Plans introduce larger database sizes, more generous resource allocations, and a broader set of options to suit your needs and to help your applications scale more smoothly.

[SIGN UP FOR FREE](#)

[Explore Heroku Postgres](#)


[Read our Announcement](#)

# NodeJS - Criando a conta no Heroku


 **HEROKU**

Already have an account? [Log in](#)


Sign up for free and  
experience Heroku today

 **Free account**

Create apps, connect databases and add-on services, and collaborate on your apps, for free.

 **Your app platform**

A platform for apps, with app management & instant scaling, for development and production.

 **Deploy now**

Go from code to running app in minutes. Deploy, scale, and deliver your app to the world.

First name \*

Last name \*

Email address \*

Company name

Role \*

Role


Country \*

Country

Primary development language \*

Select a language

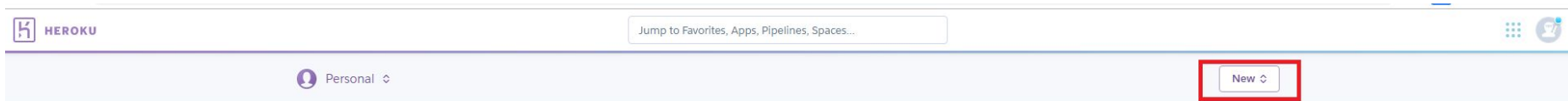
☐ I'm not a robot

  
reCAPTCHA  
[Privacy](#) • [Terms](#)

CREATE FREE ACCOUNT

# NodeJS - Criando a conta no Heroku

Após logados, vocês verão uma tela similar a tela a seguir.  
Clique em New -- > Create new app.




# NodeJS - Criando a conta no Heroku

Deem um nome para a aplicação de vocês, lembrando que esse nome será o nome do link do site também, e deve ser único.



Create New App


App name

curso-html-node-grazi 

curso-html-node-grazi is available

Choose a region

 United States 

 Add to pipeline...

Create app

# NodeJS - Criando a conta no Heroku

Após criada a aplicação, vocês deverão acessar a aba Deploy, após isso selecionar Github (Connect to GitHub).

The screenshot shows the Heroku dashboard for an application named "curso-html-node-grazi". The "Deploy" tab is highlighted with a red box. Below the tabs, there are sections for adding the app to a pipeline, deployment methods, and instructions for deploying using Heroku Git.

**Deploy**

Add this app to a pipeline

Create a new pipeline or choose an existing one and add this app to a stage in it.

Add this app to a stage in a pipeline to enable additional features

Pipelines let you connect multiple apps together and **promote code** between them. [Learn more](#)

Pipelines connected to GitHub can enable **review apps**, and create apps for new pull requests. [Learn more](#)

Choose a pipeline

Deployment method

Heroku Git  
Use Heroku CLI

**GitHub**  
Connect to GitHub

Container Registry  
Use Heroku CLI

Deploy using Heroku Git

Use git in the command line or a GUI tool to deploy this app.

Install the Heroku CLI

Download and install the [Heroku CLI](#).

If you haven't already, log in to your Heroku account and follow the prompts to create a new SSH public key.

```
$ heroku login
```

Create a new Git repository

Initialize a git repository in a new or existing directory.



# NodeJS - Criando a conta no Heroku

Após criada a aplicação, vocês deverão acessar a aba Settings -- > Add buildpack e selecionar NodeJS.

The screenshot shows the Heroku Settings page for an application named 'curso-html-node-grazi'. The 'Settings' tab is selected in the top navigation bar. Below the application name, there is a section for 'Config Vars' with a 'Reveal Config Vars' button. The 'Info' section displays various application details: Region (United States), Stack (heroku-18), Framework (No framework detected), Slug Size (No slug detected), GitHub Repo (grazielags/cursoHtmlNode), and Heroku Git URL (https://git.heroku.com/curso-html-node-grazi.git). At the bottom, the 'Buildpacks' section is visible, featuring an 'Add buildpack' button which is circled in black.

Overview	Resources	Deploy	Metrics	Activity	Access	Settings
----------	-----------	--------	---------	----------	--------	----------

Name: curso-html-node-grazi [Edit](#)

**Config Vars** [Reveal Config Vars](#)

Config vars change the way your app behaves. In addition to creating your own, some addons come with their own.

**Info**

Region	United States
Stack	heroku-18
Framework	No framework detected
Slug Size	No slug detected
GitHub Repo	grazielags/cursoHtmlNode
Heroku Git URL	https://git.heroku.com/curso-html-node-grazi.git

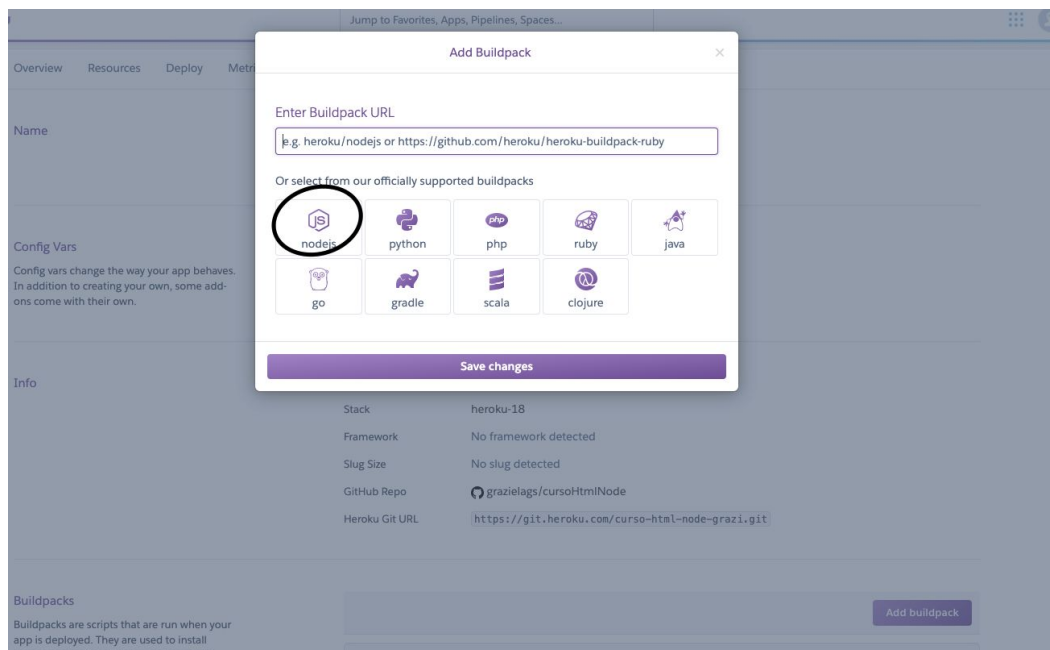
**Buildpacks**

Buildpacks are scripts that are run when your app is deployed. They are used to install dependencies for your app and configure

[Add buildpack](#)

# NodeJS - Criando a conta no Heroku

Após criada a aplicação, vocês deverão acessar a aba Settings --> Add buildpack e selecionar NodeJS.



# NodeJS - Criando a conta no Heroku

Após acessar o GitHub, terá que se conectar na sua conta do github, depois colocar o nome do repositório que criamos lá no github, clicar em Search e após apresentar abaixo, clicar em Connect.

The screenshot shows the Heroku dashboard for an application named 'curso-html-node-grazi'. The 'Deploy' tab is selected. The main section is titled 'Add this app to a pipeline' and includes instructions on how to create a new pipeline or add the app to an existing one. Below this, there are options for deployment methods: Heroku Git (Use Heroku CLI), GitHub (Connect to GitHub), and Container Registry (Use Heroku CLI). The 'Connect to GitHub' section is highlighted with a red box. It contains a search bar with the text 'Search for a repository to connect to'. The search results show a repository named 'grazielags' with a dropdown menu showing 'cursoHtmlNode'. A red box highlights the search bar, the repository name, and the 'Search' button. Below the search results, there is a link to 'Missing a GitHub organization? Ensure Heroku Dashboard has team access.' At the bottom, a red box highlights the repository path 'grazielags/cursoHtmlNode' and a 'Connect' button.

Personal > curso-html-node-grazi

Overview Resources Deploy Metrics Activity Access Settings

Add this app to a pipeline

Create a new pipeline or choose an existing one and add this app to a stage in it.

Add this app to a stage in a pipeline to enable additional features

Pipelines let you connect multiple apps together and promote code between them. [Learn more](#)

Pipelines connected to GitHub can enable review apps, and create apps for new pull requests. [Learn more](#)

Choose a pipeline

Deployment method

Heroku Git Use Heroku CLI

GitHub Connect to GitHub

Container Registry Use Heroku CLI

Connect to GitHub

Connect this app to GitHub to enable code diffs and deploys.

Search for a repository to connect to

grazielags

cursoHtmlNode

Search

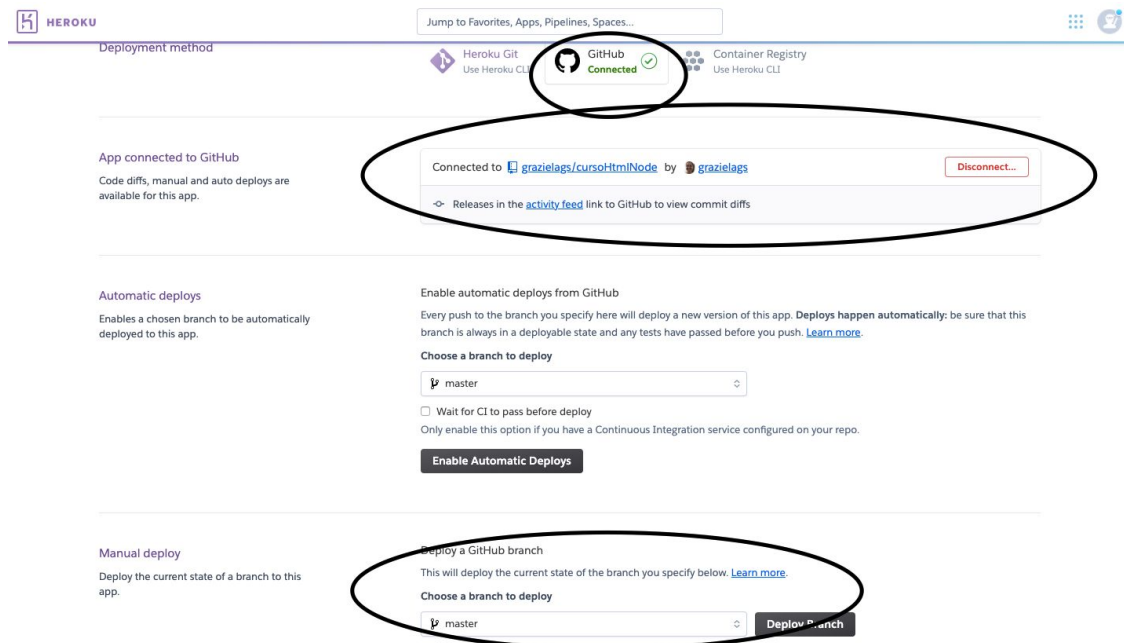
Missing a GitHub organization? [Ensure Heroku Dashboard has team access.](#)

grazielags/cursoHtmlNode

Connect

# NodeJS - Criando a conta no Heroku

Clicar no Github, depois colocar teu usuário, informar o repositório que deseja publicar e após ter se conectado, clicar no Deploy.



# Angular

<https://www.djamware.com/post/5bca67d78oaca7466989441f/angular-7-tutorial-building-crud-web-application>