HTML

RICARDO FROHLICH DA SILVA

FABRÍCIO TONETTO LONDERO

Linguagem HTML

Ao falar de internet é impossível não mencionar a principal linguagem responsável por sua evolução, o *Hypertext Markup Language*, ou simplesmente HTML.

Seu criador foi o matemático Tim Berners Lee, para suprir a necessidade de comunicação entre ele e um grupo de colegas pesquisadores.

Linguagem HTML

Com o crescimento da internet pública, HTML tornou a base para o desenvolvimento de aplicações e recursos para a transmissão de informações na *World Wide Web*.

As primeiras especificações foram publicadas por Berners Lee no ano de 1993, tendo publicada a versão 2.0 no ano de 1995.

Linguagem HTML

A partir do ano de 1996 o desenvolvimento de especificações foi atribuído e mantido por um grupo de empresas fabricantes de software denominada *World Wide Web Consortium*, ou simplesmente W3C.

Pilares estipulados pela W3C

Um esquema de nomes para localização de fontes de informação na Web, esse esquema chama-se URI.

Um Protocolo de acesso para acessar estas fontes, hoje o **HTTP.**

Uma linguagem de Hypertexto, para a fácil navegação entre as fontes de informação: o HTML.

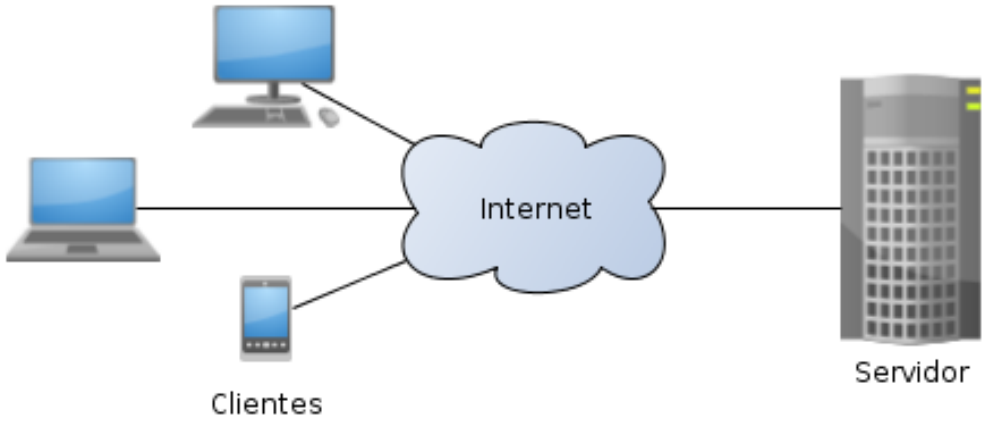
HTML5

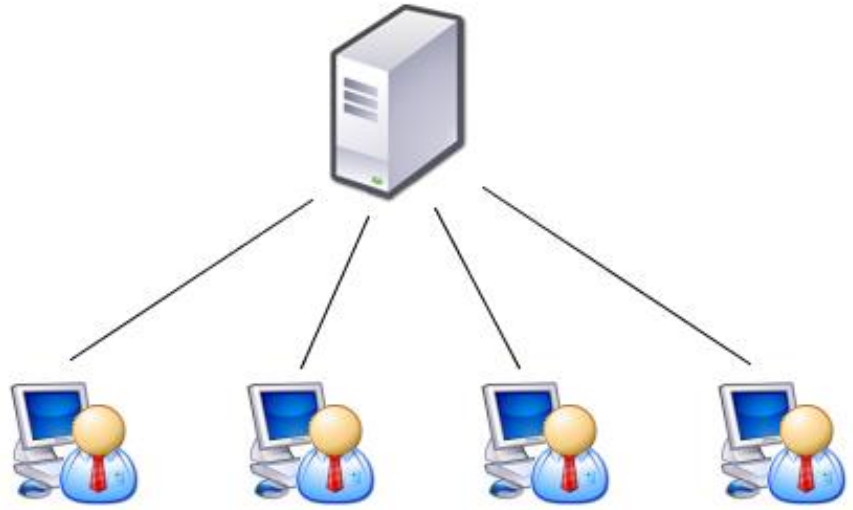
A partir da versão 5 do HTML, começou a ser fornecido ferramentas para CSS e Javscript trabalhe do melhor modo possível, de forma que o website ou a aplicação continue leve e funcional.

Também foi adicionado novas *tags*.

E o mais importante: foi definido um padrão universal do HTML (*tags* e suas funcionalidades)

Arquitetura Cliente-Servidor





Arquitetura Cliente-Servidor

O termo Cliente/Servidor refere-se ao método de distribuição de aplicações computacionais através de muitas plataformas. Tipicamente essas aplicações estão divididas entre um provedor de acesso e uma central de dados e numerosos clientes contendo uma interface gráfica para usuários para acessar e manipular dados.

Arquitetura Cliente-Servidor

Cliente/Servidor geralmente refere-se a um modelo onde dois ou mais computadores interagem de modo que um oferece os serviços aos outros. Este modelo permite aos usuários acessarem informações e serviços de qualquer lugar.

Arquitetura Cliente-Servidor

Cliente:

- Pode ser denominado como **Front-End** e **WorkStation**;
- é um processo que interage com o usuário através de uma interface gráfica ou não, permitindo consultas ou comandos para recuperação de dados e análise e representando o meio pela qual os resultados são apresentados.

Arquitetura Cliente-Servidor

Cliente:

- É o processo ativo na relação Cliente/Servidor;
- Inicia e termina as conversações com os Servidores, solicitando serviços distribuídos;
- Não se comunica com outros Clientes;

Arquitetura Cliente-Servidor

Servidor:

- Pode ser denominado como **Back-End**;
- Fornece um determinado serviço que fica disponível para todo Cliente que o necessita;
- É o processo reativo na relação Cliente/Servidor;
- Recebe e responde às solicitações dos Clientes;

Arquitetura Cliente-Servidor

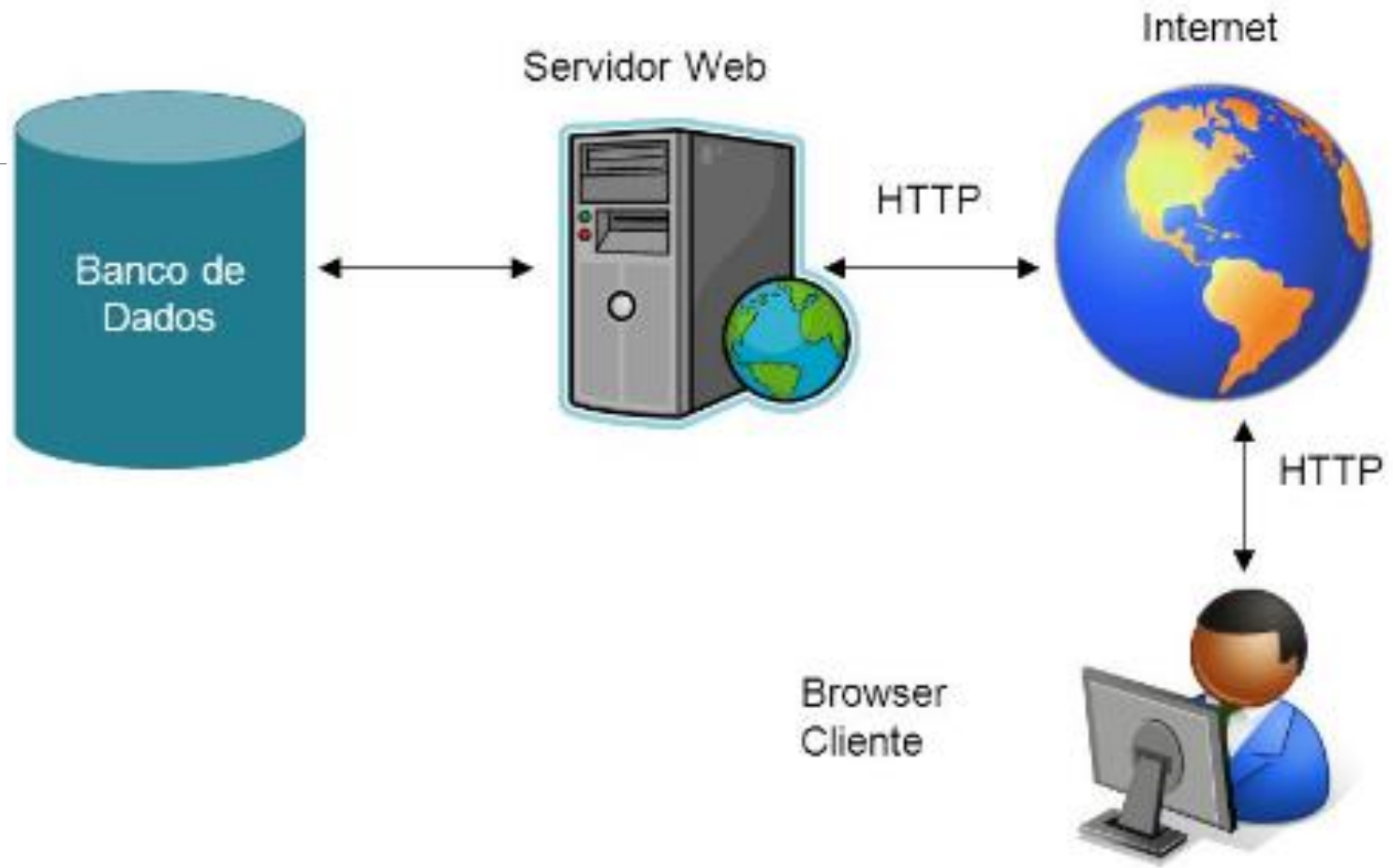
Servidor:

- Não se comunica com outros Servidores enquanto estiver fazendo o papel de Servidor;
- Atende a diversos Clientes simultaneamente;
- Possui execução contínua;

Servidor

Alguns tipos de serviços que um Servidor pode proporcionar são:

- Servidor de Arquivos
- Servidor de Impressora
- Servidor de Banco de Dados
- Servidor de Redes
- Servidor de Fax
- Servidor de Processamento e Imagens
- Servidor de Comunicação
- e etc.


Internet
Servidor Web
Banco de Dados
HTTP
HTTP
Browser Cliente

Front-end

Responsável pela coleta de informações do cliente e em alguns casos essas informações podem ser processadas.

Exemplos:

- HTML
- CSS
- Javascript
- Ajax
- jQuery

Back-end

Responsável por tudo que é executado no servidor, que envolve o plano de negócio do sistema. Envolve analise dos dados provenientes do Front-End, acesso ao banco de dados, armazenamento e manutenção de arquivos.

Exemplos:

- C#
- Java
- Perl
- PHP
- Python

Fluxo

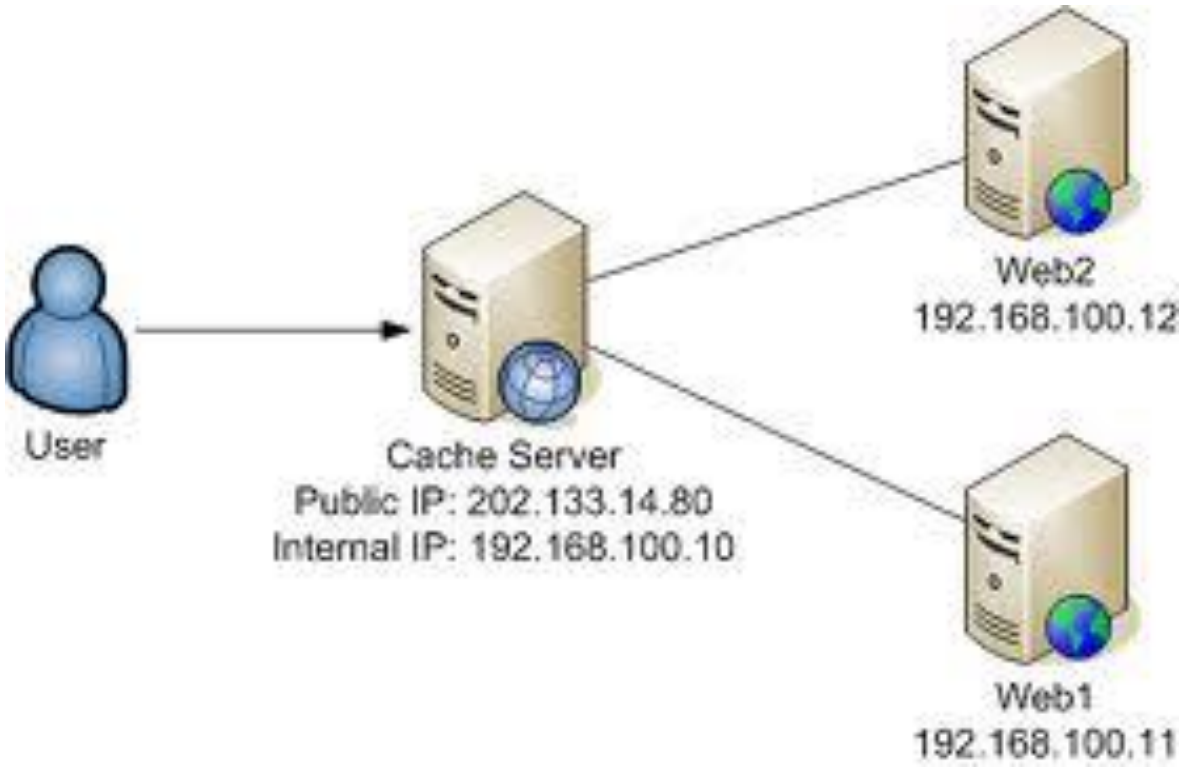
Cliente acesso um site pelo endereço, exemplo: www.globo.com.br

Um servidor DNS recebe a requisição e encaminha para o endereço IP onde o site esta hospedado, no caso: 186.192.90.5

O servidor do site (IP) processa e encaminha o Front-End ao Browser(navegador do cliente)

Cliente recebe a pagina web (html, css, js ...)

Fluxo


User
Cache Server
Public IP: 202.133.14.80
Internal IP: 192.168.100.10
Web2
192.168.100.12
Web1
192.168.100.11

Fluxo


www.globo.com
globo.com
g1
ge
gshow
tech
vídeos
encontre na globo.com
Abstenção é a maior das últimas eleições municipais no país
• Número de mulheres mais do que dobra na Câmara de São Paulo
• PT perde 44,8% das cadeiras nas câmaras; PSDB e PDT crescem
• PMDB é o que mais elegeu prefeitos; PT encolhe 59% nas prefeituras
Após tumulto com
Abstenção é a maior das últimas eleições municipais no país
ACM Neto é 3º eleito prefeito mais rico
'A Lei do Amor': saiba tudo sobre a estreia da nova novela
• Indicada, Grazi se cobra mais
• Novela não teria 1ª fase

Fluxo

No site, o cliente navega entre as paginas, cada interação do usuário, o processo se repete, ele envia uma requisição ao servidor e o servidor responde a essa requisição alterando a pagina do cliente.

Efetuando assim, uma serie de “conversas” entre cliente e servidor.

Fluxo

Em uma das paginas, existe um formulário de cadastro, o usuário preenche todos os campos, mas a data de nascimento foi inserida com um valor errado, exemplo: 22/13/1992. Só que, antes de submeter a requisição ao servidor, o programador elaborou uma validação dos campos via *Javascript* no qual foi identificada a data inválida e o usuário foi alertado. Esse fluxo não “saiu” do Front-End.

Após o usuário efetuar a correção e submeter o formulário novamente, a requisição será enviada ao servidor.

Fluxo

O servidor pode encaminhar paginas HTML com arquivos CSS e JS inteiros, tais como são ao cliente, e seus navegadores (browsers) que efetuam a interpretação destes códigos para o cliente.

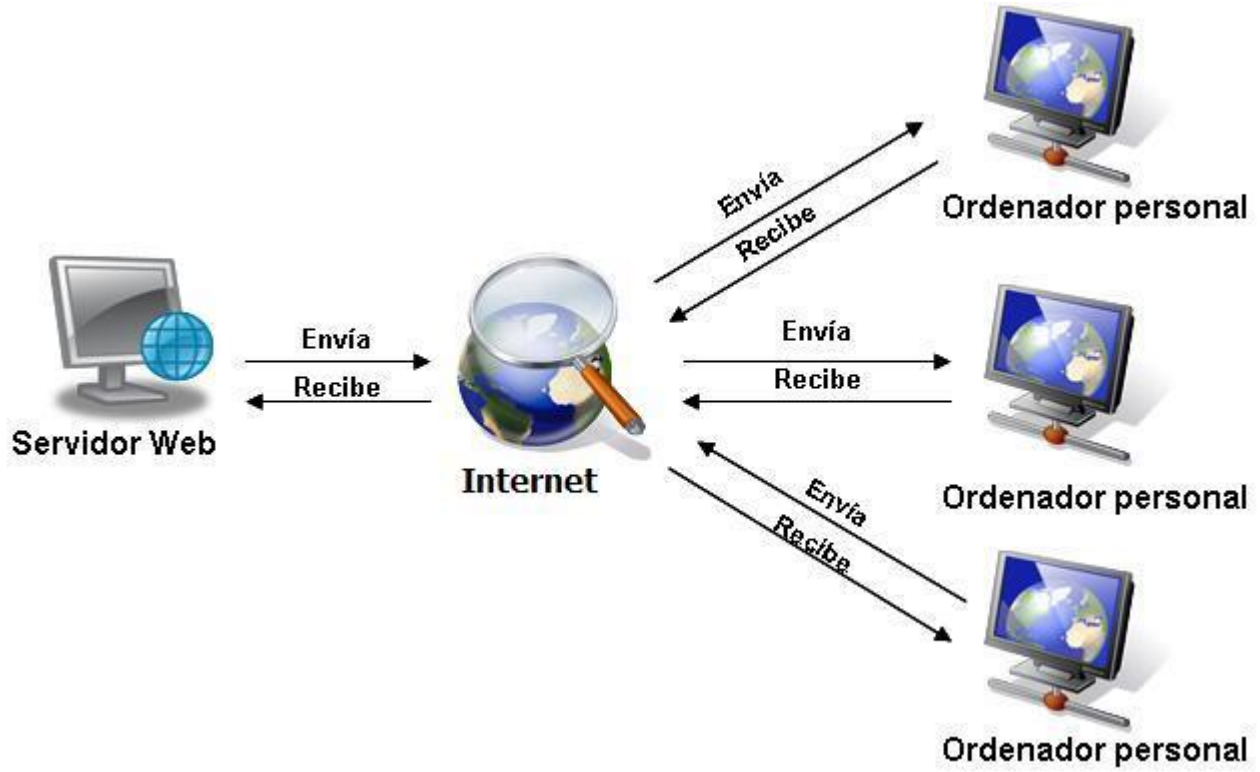
Estas páginas são denominadas ESTÁTICAS.

Fluxo

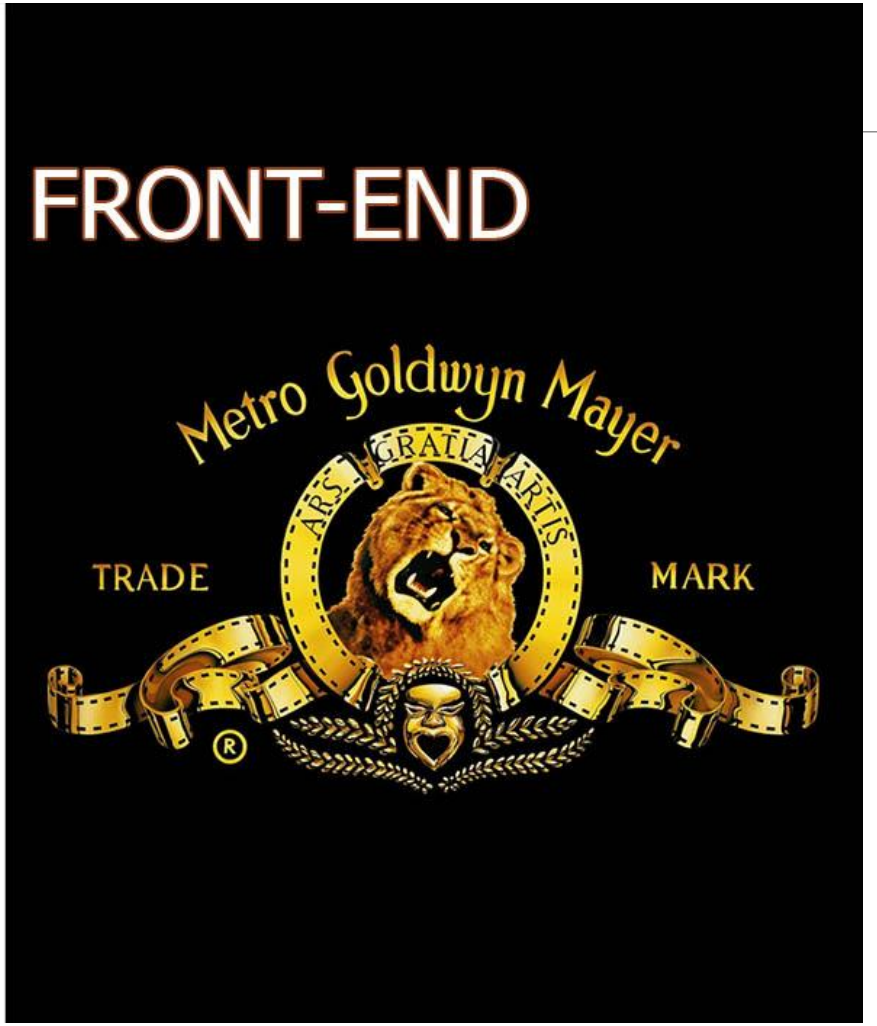
Mas, muitas vezes, toda a pagina ou pequenas partes delas só ficam “prontas” após o servidor processar, como por exemplo, buscar informações do banco de dados ou de arquivos.

Estas páginas são denominadas DINÂMICAS.

Fluxo


Ordenador personal
Envía
Recibe
Envía
Recibe
Servidor Web
Internet
Envía
Recibe
Ordenador personal
Envía
Recibe
Ordenador personal


BACK-END


FRONT-END
Metro Goldwyn Mayer
TRADE
MARK

Hospedagem

Ao publicar um site, devemos pensar qual plataforma iremos utilizar para hospedar nosso site.

Devemos levar em conta, a linguagem utilizada no nosso Back-End.

Hospedagem – Escolha do servidor

Linguagens .Net (C#, VB...)

- IIS (Serviços de Informações da Internet)
- Ambiente Windows

PHP

- Apache
- Windows e Linux

Java

- TomCat
- Windows e Linux

Python

- Apache
- Windows e Linux

HTML

Tags básicas

<html>: define o **início** de um documento HTML e indica ao navegador que **todo conteúdo posterior deve ser tratado como uma série de códigos HTML;**

<head>: define o **cabeçalho** de um documento HTML, que traz informações sobre o documento que está sendo aberto;

<body>: define o conteúdo principal, o **corpo** do documento. Esta é a **parte do documento HTML que é exibida no navegador.** No corpo podem-se definir atributos comuns a toda a página, como cor de fundo, margens, e outras formatações.

<body>

bgcolor: cor do plano de fundo. Deve ser usado o código hexadecimal ou o nome da cor (conforme tabela vista mais adiante), assim como em todos os atributos de cores.

background: URL de uma imagem para ser usada como plano de fundo.

link: cor natural dos links, ou seja, enquanto eles ainda não foram clicados (o padrão é azul).

alink: cor dos links quando são clicados (o padrão é vermelho).

vlink: cor dos links após serem visitados (o padrão é roxo).

text: cor do texto da página.

```
<html>
    <head>
        ...
    </head>
    <body>
        ...
    </body>
</html>
```

Doctype

Para usar os novos elementos de HTML5, devemos informar o *Doctype* corretamente, como segue abaixo:

<!DOCTYPE html>

<html lang=**“pt-br">**

.

.

.

Cabeçalhos

Podem incluir pauta de conteúdos desde o logotipo da cia à caixa de busca. No blog colocaremos apenas o título.

- Uma página pode conter mais de um cabeçalho, em seção ou artigos, por exemplo. Por isso é interessante usar o atributo ID (falaremos mais sobre ele em breve).

```
<body>
        <header id=“cavecalho">
                <h1>Nosso Blog!</h1>
        </header>
</body>
```

Rodapés

O elemento *footer* define informação de rodapé para um documento ou uma seção adjacente. Por isso também é interessante usar ID para diferenciar uns dos outros.

```
<footer id=“rodape">
        <p>© 2018 Nosso Blog.</p>
</footer>
```

Navegação

A navegação é essencial, por isso foi criada a nova tag para trata-la adequadamente no HTML5. Exemplo:

```
<header id=“cabecalho">
<h1>Nosso Blog!</h1>
    <nav>
        <ul>
            <li><a href="/">Últimas Postagens</a></li>
            <li><a href="archives">Arquivos</a></li>
            <li><a href="contributors">Contribuintes</a></li>
            <li><a href="contact">Contato</a></li>
        </ul>
    </nav>
</header>
```

Seções e Artigos

Seções são as regiões lógicas de uma página

O elemento *section* veio para substituir o uso excessivo de *divs*.

No entanto, não devemos nos deixar levar pelas seções, devemos usá-las para agrupamento lógico do conteúdo.

No caso de um blog, será criada uma seção que conterá todas as postagens.

Todavia, cada postagem não deve estar em uma nova seção, pois tem um tag mais apropriado para isso!

Seções e Artigos

<section id="**posts**">
    CONTEUDO DA POSTAGEM

</section>

Artigos

O tag *article* é o elemento perfeito para descrever o real conteúdo de uma página WEB.

- Cada artigo pode conter cabeçalho, conteúdo e rodapé

Artigos exemplo parte 1

...

```
<section id="posts">
    <article class="post">
        <header>
            <h2>O que é um Blog?</h2>
            <p>Postado por Wikipedia
                <time datetime="2013-12-13T14:39"> em 13 de fevereiro, 2018, às 14h 39min</time>
            </p>
        </header>
```

Artigos exemplo parte 2

```
        <p>Muitos blogs fornecem comentários ou notícias sobre um assunto em particular; outros funcionam mais como diários online. Um blog típico combina texto, imagens e links para outros blogs, páginas da Web e mídias relacionadas a seu tema. A capacidade de leitores deixarem comentários de forma a interagir com o autor e outros leitores é uma parte importante de muitos blogs. </p>
        <footer>
                <p><a href="comments"><i>25 Comentários</i></a> ...</p>
        </footer>
    </article>
</section>
...
```

Conteúdo aparte

Podemos adicionar conteúdo extras a uma postagem, de forma a fazer com que o conteúdo não seja relacionado ao conteúdo principal, para sites de busca serem mais efetivos por exemplo.

SEO -> Search Engine Optimization

Leva as tags em consideração para “subir” um site para o topo de resultados (ranking) de busca , muito importante para páginas de apresentação. OBS: para o tráfego orgânico (diferente do pago).

Conteúdo aparte

<aside>

 Blog: contração do termo inglês web log, "diário da rede“

</aside>

 Um blog é um site cuja estrutura permite a atualização

rápida a partir de acréscimos dos chamados artigos, ou posts.

…

Barras laterais

Também usado para aglomerar links. Faz parte da navegação da página em si.

Para isso, se usa a tag *nav*.

Barras Laterais

```
<section id="sidebar">
    <nav>
        <h3>Arquivos</h3>
        <ul>
            <li><a href="2018/08">Agosto 2018</a></li>
            <li><a href="2018/07">Julho 2018</a></li>
            <li><a href="2018/06">Junho 2018</a></li>
        </ul>
    </nav>
</section>
```

Cabeçalhos de Texto

<h1>Título de nível 1</h1>

<h2>Título de nível 2</h2>

<h3>Título de nível 3</h3>

<h4>Título de nível 4</h4>

<h5>Título de nível 5</h5>

<h6>Título de nível 6</h6>

Alinhando os Cabeçalhos

É possível alterar o alinhamento dos títulos definindo seu **atributo** “align” com um dos seguintes valores: *center*, *left* e *right*;

<h1 align=“center”>Título de nível 1 - centralizado</h1>

<h2 align=“left”>Título de nível 2 - esquerda</h2>

<h3 align=“right”>Título de nível 3 - direita</h3>

Tag - tamanho e cor do texto

Usando a *tag* é possível alterar algumas das características primordiais do texto como o tipo da fonte (Arial, Times New Roman, etc), a cor e o tamanho.

Essas propriedades são alteradas com a definição dos atributos face, color e size, respectivamente, da tag font.

Exemplos com a tag

Arial 3 Azul

Times 4 Verde

Courrier 5 Vermelho

Formatação adicional do texto

Existem algumas tags bastante úteis que permitem aplicar uma formatação a um trecho do texto, apenas adicionando as tags de abertura e fechamento antes e depois do texto a ser formatado, assim como foi visto para a tag <font>.

: marca o texto como em negrito (bold).

: marca o texto como em itálico (italics).

*<u></u>: marca um texto como sublinhado (underlined).

*<s></s> ou <strike></strike>: marca um texto como riscado.

Observação: as tags marcadas com asterisco (*) foram descontinuadas na versão 5 da HTML, ou seja, seu uso ***não é mais indicado****.*

Formatação adicional do texto – cont.

: marca um texto como subscrito.

: marca o texto como sobrescrito.

* : centraliza o texto.

_{subscrito}

^{sobrescrito}

*Observação: as tags marcadas com asterisco (*) foram descontinuadas na versão 5 da HTML, ou seja, seu uso **não é mais indicado**.*

Parágrafos e quebras de linha

 - quebra de linha;

 - sinalização de um parágrafo;

Alinhamento de Parágrafos

Podemos ainda alterar seu alinhamento para atender às diversas situações que surgem no dia-a-dia. Isso pode ser feito com a definição do atributo “align” da tag , com um dos seguintes valores: *left* (alinhado à esquerda), *right* (alinhado à direita), *center* (centralizado) e *justify* (justificado).

Imagens

Inserir **imagens** na página é uma das necessidades mais comuns e, por esse motivo, também é consideravelmente simples de ser feito. Para esse fim existe a tag **<img>,** cujos atributos são mostrados a seguir:

src: caminho completo do arquivo de imagem (incluindo a extensão do arquivo).

alt: texto alternativo para a imagem, geralmente uma descrição da mesma.

width: largura da imagem em pixels.

height: altura da imagem em pixels.

Imagens

Observação: os atributos **src** e **alt** são obrigatórios para a tag <img>. Além disso, a tag deve ser fechada nela mesma, ou seja, a sintaxe mínima é

Inserindo Imagens

Observações sobre as imagens

Caso **não sejam definidas** a largura e/ou a altura, a imagem será adicionada à página em seu **tamanho original**.

Ao alterar apenas uma das dimensões, a outra é redefinida proporcionalmente. Caso se deseje alterar tanto a largura quanto a altura com valores específicos, os dois atributos devem ser definidos.

Criando Links

```
<a href=“https://www.google.com”>Meu Link</a>


<a href="http://www.unifra.br/">
    <img src=“imagem.jpg” alt=“Visitar site"/>
</a>


<a href=“www.terra.com.br">
    <h1>Titulo como link</h1>
</a>
```

Listas

Outra estrutura bastante comum de ser utilizada em páginas web é a **lista**, que serve pra **organizar um conjunto de itens, sequencialmente ou não**. As listas podem ser ordenadas ou não, ou seja, cada item podem ou não ter um número/letra/símbolo que o identifique sequencialmente.

As tags para listas ordenadas e não ordenadas são, respectivamente, e . Entre essas marcações devem ser inseridos os itens, que levam a tag

Usando listas

Listas ordenadas:

```
<ol>
    <li>Primeiro item</li>
    <li>Segundo item</li>
    <li>Terceiro item</li>
</ol>
```

Listas não ordenadas:

```
<ul>
    <li>Primeiro item</li>
    <li>Segundo item</li>
    <li>Terceiro item</li>
</ul>
```

Lista de Definição

É usada para apresentar itens com um título seguido de sua definição. A tag principal desse tipo de lista é a <dl></dl>, enquanto os itens são compostos por duas tags: <dt></td> para o título e <dd></dd> para a definição.

```
<dl>
    <dt>Item 1</dt>
    <dd>Definicao do item 1</dd>
    <dt>Item 2</dt>
    <dd>Definicao do item 2</dd>
    <dt>Item 3</dt>
    <dd>Definicao do item 3</dd>
</dl>
```

Tabelas

As tabelas são criadas sobre a tag base **<table></table>** e é dividida em linhas **<tr></tr>** e colunas **<td></td>.** A ordem de hierarquia é essa: table > tr > td, uma dentro da outra. Ou seja, a tabela é dividida em linhas que, por sua vez, são divididas em colunas.

A tag *table* possui o **atributo** "**border**", que define a borda das células com um número inteiro representando a **espessura**.

Exemplo de Uso de Tabela

```
<table border="1">
        <tr>
            <td>Linha 1, Coluna 1</td>
            <td>Linha 1, Coluna 2</td>
            <td>Linha 1, Coluna 3</td>
        </tr>
        <tr>
            <td>Linha 2, Coluna 1</td>
            <td>Linha 2, Coluna 2</td>
            <td>Linha 2, Coluna 3</td>
        </tr>
</table>
```

Atributos ID e CLASS

Em breve falaremos sobre SELETORES.

Em resumo:

- Como o próprio nome já diz, são formas de “selecionar” tags HTML, por seu tipo ou atributos.
- Toda e qualquer tag pode receber os atributos ID e CLASS, muito utilizados para auxiliar no uso de CSS, Javascript e comunicação entre Back-End e Front-End.

Atributos ID e CLASS

```
<table border="1“ id=“tabela1” class=“modelo1”>
        <tr>
            <td>Linha 1, Coluna 1</td>
            <td>Linha 1, Coluna 2</td>
            <td>Linha 1, Coluna 3</td>
        </tr>
</table>
```


É de conhecimento que existem seletores para as *tags* no HTML, tais como os atributos ID e a CLASS. No qual nos possibilita aplicar, via CSS, o mesmo estilo a diferentes *tag* distribuídas em nossa(s) página(s) HTML.

Isso nos possibilita grande produtividade e criatividade, mas ainda assim pode não solucionar todos os nossos problemas.

Um menu muito grande, vamos repetir ID e CLASS em cada um dos itens?

O nome ***div*** vem de divisão, e esta *tag* nos permite dividir qualquer trecho de seu código. Isso mesmo, você pode criar um bloco, uma divisão, e dentro deste bloco ter uma imagem, links, textos e o que mais você quiser.

Pode-se aplicar o CSS nesse bloco (div), e tudo que estiver dentro da **div**, seja imagem, texto, link ou o que mais for, vai receber aquelas regras de estilo.

Se você fizer isso usando CLASS, por exemplo, vai ter que colocar esse atributo (seletor) em cada *tag*.

É bem comum dizer que a *tag* é um container para armazenar diversos elementos.

As *divs* são IMPORTANTÍSSIMAS para estruturar, criar uma ordem lógica e organizar um site.

Tu podes cria uma div para o cabeçalho, outra para o conteúdo, outra div para os menus e uma última para o rodapé, então usa o CSS para estilizar cada uma dessas divs separadamente.

Também poderá criar outras divisões dentro destas divs. Ou seja, criar div dentro de div.

Todo, qualquer e quantos elementos do HTML você queira colocar aqui.

Não iremos repetir os **atributos seletores** em cada um dos componentes internos de uma div, mas apenas na div.


```
<body>
        <div id="header">
        </div>

        <div id="content">
        </div>

        <div id="foot">
          <p>© Copyright 2018 UNIFRA</p>
        </div>

</body>
```

```
<!DOCTYPE html>
<html>
    <body>

        <p>Parágrafo fora da div.</p>

        <div style="color:#0000FF">
          <h3>Tag H3 dentro da div</h3>
          <p>Parágrafo dentro da div.</p>
        </div>

        <p>Parágrafo fora da div.</p>

    </body>
</html>
```

Formulários

Em desenvolvimento web é através dos formulários que os usuários (internautas) interagem de forma muita mais dinâmica com os sites. Simplificando ao máximo o conceito, podemos dizer que os formulários recebem os dados digitados pelo usuário e depois, com o auxílio de uma linguagem de programação, esse dados são utilizados e/ou armazenados em um banco de dados.

Formulários

Os formulários fazem então o papel de interface do nosso sistema, no qual sua única finalidade é receber os dados do usuário e repassá-los de forma organizada para outra página contendo uma linguagem de programação, podendo essa linguagem ser em php, java, asp, asp.net etc.

Formulários

Todos os formulários em HTML devem, sem exceção, possuir a tag <form> de abertura e a sua correspondente de fechamento, a tag </form>. Além disso, são imprescindíveis também dois atributos que por enquanto não nos serão uteis, mas é bom conhecermos agora para tornar mais fácil o aprendizado sobre formulários.

```
<form>

        ...

</form>
```

Atributos de um <form>

```
<form name="nome" action="url" method="método" enctype="tipo" target="janela">
    ...
    <!- Esse é um comentário, ignorado pelo navegador, ou seja, não muda nada no
nosso site-->
    ...

</form>
```

Atributos de um <form>

name = "nome" especifica o nome do formulário (opcional).

action = "url" especifica o endereço do programa (CGI) que receberá e tratará os dados dos campos do formulário.

Method = "método" especifica o método como os dados serão enviados ao programa. Os métodos mais usuais são: GET, que envia os dados junto com o endereço POST, que envia os dados separados do endereço
enctype="tipo" especifica a codificação utilizada para o envio dos dados (opcional).

target="janela" especifica a janela que será utilizada para o programa enviar mensagens após processar os dados (opcional).

Method

GET – método que envia as variáveis digitadas pelo usuário pela URL, ou seja, podemos ver as variáveis sendo passadas pela URL da página de destino. Não é muito aconselhável o uso do método GET, pois ele expõe os nomes e os valores das variáveis.

POST – método que envia as variáveis digitadas pelo corpo da página, sendo completamente transparente para o usuário. É o método mais aconselhável.

Campos de entrada de dados

São vários elementos possíveis para trabalharmos com formulários e, entre os mais comuns que aprenderemos nessa aula estão: **campo de texto**, **campo de senha** e **botão de envio**.

Campos de entrada de dados

A tag <input> é uma tag com a qual podemos criar vários tipos de campos de entrada alterando apenas o atributo ***type*** que, como o próprio nome já diz, serve para informar o tipo de campo que desejamos criar.

Text

Para um **input** se comportar como campo de texto, mudamos o atributo **type** para **text**:

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

OBS: Precisamos definer um nome para os **inputs**

Inputs

Button

Checkbox

Color

date

datetime-local

email

File

Hidden

Image

month

number

Password

Radio

range

Reset

Search

Submit

Tel

Text

time

url

week