

ENSINO MÉDIO INTEGRADO: INFORMÁTICA

Disciplina de Desenvolvimento Web

Aula 01: HTML - Principais Tags e Formulário

Gil Eduardo de Andrade

Conceitos Preliminares

Arquitetura Web

A infra-estrutura da Internet é baseada no modelo cliente x servidor, onde os clientes efetuam requisições junto aos servidores no intuito obter determinados tipos de dados. Tecnologias como HTML e CSS encontram-se no lado do cliente (*client side*), pois podem ser interpretadas e renderizadas pelo navegador executado no computador do usuário – não há necessidade de conexão com um servidor remoto. Já tecnologias como PHP encontram-se no lado do servidor (*server side*) pois sua interpretação é feita remotamente, visto que o navegador não é capaz disso. Para tal é necessário que o cliente efetue requisições a um servidor (*Apache*, por exemplo), que após interpretar o script PHP retorna como resultado dados que podem ser manipulados e exibidos pelo navegador. As requisições e respostas geradas precisam obedecer a um padrão, para que ambos os lados possam trocar informações e compreender o que está sendo requisitado e respondido. Por esse motivo o protocolo HTTP é utilizado.

Protocolo e Requisição HTTP

A sigla HTTP refere-se a Protocolo de Transferência de Hipertexto (*Hyper Text Transfer Protocol*). Tal protocolo define regras e padrões que permitem que clientes e servidores Web possam se comunicar adequadamente, efetuando requisições e obtendo suas respectivas respostas. Por exemplo, quando um cliente acessa uma determinada URL (endereço web) através do navegador, uma requisição será efetuada a um determinado servidor, tendo como procedimentos, de modo mais simplista:

1. **Navegador** (cliente) efetua uma conexão com o servidor e envia uma solicitação HTTP para a página da web especificada;

2. **Servidor** recebe e verifica a solicitação, sendo adequada, o servidor devolve como resposta os dados para página especificada e um código indicando que a solicitação foi atendida corretamente. Caso o servidor, por algum motivo, não consiga atender a solicitação, enviará uma mensagem de erro juntamente com um código que permite sua identificação;
3. **Navegador** recebe a resposta do servidor (“página / código” ou “mensagem de erro / código”), e a conexão é finalizada;
4. **Navegador** analisa a resposta, caso indique que a solicitação foi atendida adequadamente o navegador adota os procedimentos necessários para exibir a página especificada na solicitação;

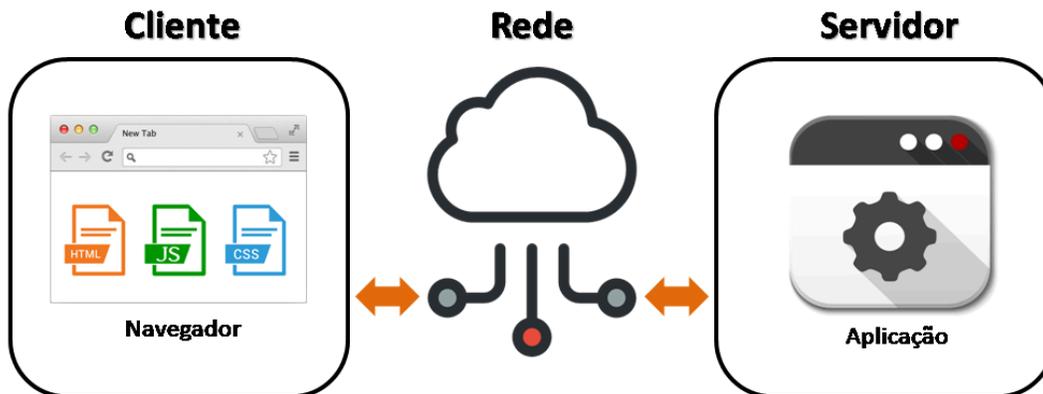


Figura 1: Navegadores Web e Protocolo HTTP.

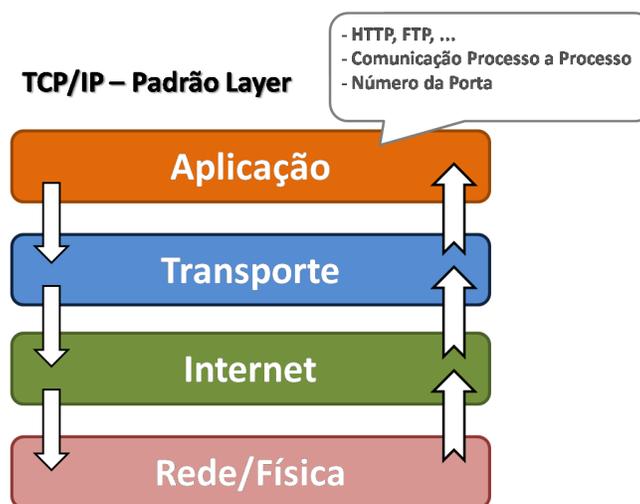


Figura 2: Protocolo TCP/IP - Comunicação processo a processo.

As requisições efetuadas por um cliente são denominadas *HTTP Request*. Tais requisições são recebidas e processadas pelos servidores (como o Apache), que devolvem como resposta, tanto os conteúdos das páginas web solicitadas pelo usuário, quanto códigos que permitem ao cliente identificar se a requisição ocorreu como esperado. A seguir temos uma lista resumida com os principais códigos de resposta emitidos por um servidor quando uma requisição HTTP é recebida:

- **1XX:** Informativo – solicitação foi aceita ou continua em andamento;
- **2XX:** Confirmação – solicitação foi concluída ou entendida;
- **3XX:** Redirecionamento – um ou mais procedimentos são necessários para atender a solicitação;
- **4XX:** Erro/Cliente – solicitação não pode ser atendida ou contém erro de sintaxe;
- **5XX:** Erro/Servidor – servidor falhou durante o atendimento da solicitação;

A lista completa pode ser encontrada em: <https://httpstatuses.com/>

O HTTP possui um conjunto de métodos que podem ser utilizados quando uma solicitação é efetuada por um determinado cliente. Esses métodos definem o modo como os parâmetros são enviados quando uma requisição é efetuada ao servidor. Dentre esse conjunto de métodos destacam-se:

- **GET:** é o método padrão utilizado ao efetuar uma solicitação, nele os parâmetros são passados juntamente com cabeçalho da requisição HTTP, sendo possível vê-los na URI. Exemplo:
- www.pagina.com.br/cadastrar/cpf=00000000001
- **POST:** é um dos métodos que podem ser especificados no momento em que uma solicitação é efetuada. Diferentemente do método *GET*, o *POST* permite que os parâmetros sejam passados junto ao corpo da requisição, não sendo mais visíveis na URI. Exemplo:
- www.pagina.com.br/cadastrar

A utilização prática dos métodos será estudada ao longo desta aula.

Linguagem HTML5 (<https://developer.mozilla.org/pt-BR/docs/Web/Guide/HTML/HTML5>)

HTML (Linguagem de Marcação de HiperTexto) é o bloco de construção mais básico da web. Define o significado e a estrutura do conteúdo da web. Outras tecnologias, além do HTML, também costumam ser utilizadas para descrever a aparência/apresentação (CSS) ou a funcionalidade/comportamento (JavaScript) de uma página da web.

"Hipertexto" refere-se aos links que conectam páginas da Web entre si, seja dentro de um único site ou entre sites. Links são um aspecto fundamental da web. Ao carregar conteúdo na Internet e vinculá-lo a páginas criadas por outras pessoas, você se torna um participante ativo na *world wide web*.

O HTML usa "Marcação" para anotar texto, imagem e outros conteúdos para exibição em um navegador da Web. A marcação HTML inclui "elementos" especiais, como:

- `<head>`
- `<title>`
- `<body>`
- `<header>`
- `<footer>`
- `<nav>`
- *Entre outros.*

Um elemento HTML é definido no formato de "tags", que consistem em colocar o nome do elemento entre os sinais "<" e ">". O nome de um elemento dentro da tag é insensível a letras maiúsculas e minúsculas. Em outras palavras, o elemento pode ser escrito em letras maiúsculas ou minúsculas. Por exemplo, a tag `<title>` pode ser escrita como `<Title>`, `<TITLE>` ou de qualquer outra forma.

HTML Semântico

O HTML Semântico é uma forma de escrever código HTML, descrevendo o significado e a estrutura do conteúdo de uma página da web, em vez de apenas sua aparência visual. Em outras palavras, o HTML Semântico ajuda a dar sentido ao conteúdo, tornando-o mais fácil de entender para navegadores, mecanismos de busca e desenvolvedores.

Benefícios do HTML Semântico:

- **Acessibilidade:** facilita a navegação para pessoas com deficiência visual, que utilizam leitores de tela para interpretar o conteúdo da página.
- **SEO (Otimização para Mecanismos de Busca):** ajuda os mecanismos de busca a entender o conteúdo da página, o que pode melhorar o ranking nos resultados de pesquisa.
- **Manutenção:** torna o código mais organizado e fácil de manter, facilitando a colaboração entre desenvolvedores.
- **Compatibilidade:** garante que a página seja exibida corretamente em diferentes dispositivos e navegadores.

Principais tags semânticas:

- **<header>:** define o cabeçalho da página ou de uma seção.
- **<nav>:** define a área de navegação do site.
- **<main>:** define o conteúdo principal da página.
- **<article>:** define um conteúdo independente, como um post de blog ou um artigo de notícia.
- **<section>:** define uma seção temática dentro da página.
- **<aside>:** define um conteúdo lateral, como uma barra lateral ou um bloco de informações adicionais.
- **<footer>:** define o rodapé da página ou de uma seção.

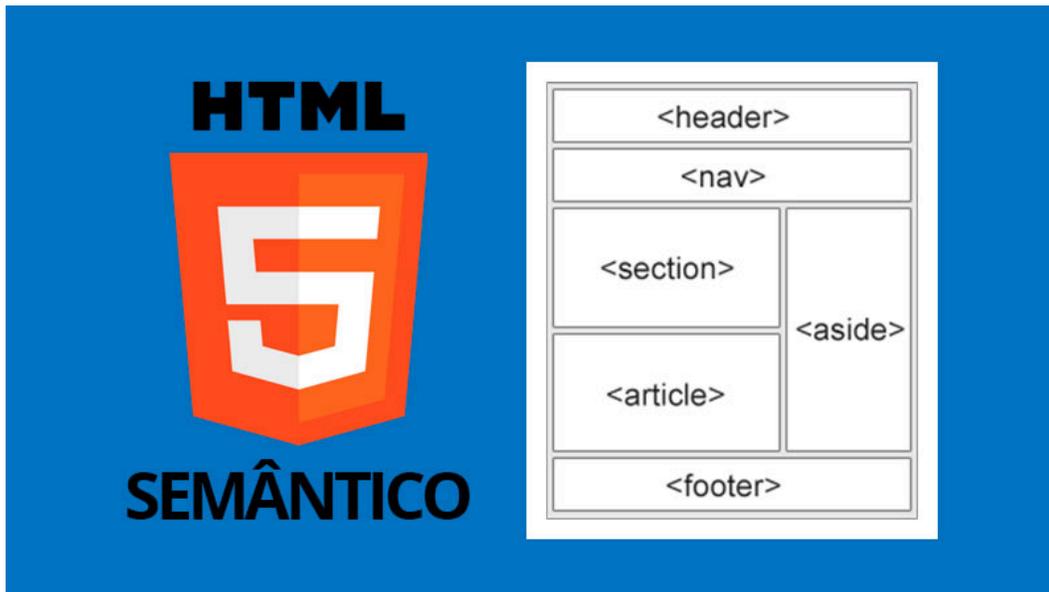


Figura 2: HTML Semântico - Principais tags.



Codificação – Linguagem HTML₅

(<https://www.w3schools.com/html/>)

Anatomia de uma <tag>

A anatomia de uma tag segue o seguinte padrão **<nome_tag>**, podendo ela possuir corpo ou não, ou seja, possuir elementos filhos. Observe a Figura 3.

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Título da Página</title>
</head>
<body>
  <input type="text"/>
```

```
</body>  
</html>
```

Anatomia de uma tag HTML - com e sem corpo.
(Arquivo-fonte: anatomia_tag.html)

Observe que as tags `<html>`, `<head>`, `<title>` e `<body>` possuem elementos filhos, ou seja, elas possuem outros elementos que compõem o seu corpo. O corpo de uma tag é o conteúdo colocado entre a sua abertura (`<title>`) e seu fechamento (`</title>`). Perceba que o fechamento de uma tag que possui corpo segue o mesmo padrão de sua abertura com a adição do “/” antes do seu nome (`<title>`Título da Página`</title>`). Por outro lado, algumas tags podem não possuir corpo, como é o caso das tags `<meta>` e `<input>` no exemplo da Figura 3. Nesses casos não há elementos filhos que compõem seu corpo, e por isso elas não possuem uma tag de fechamento. O fechamento é efetuado na própria tag de abertura (`<input type=“text” />`). O HTML₅ permite também que algumas tags sejam escritas sem a necessidade de indicar seu fechamento (uso da “/”), caso da tag `<meta>` (`<meta charset=“UTF-8”>`) no exemplo anterior.

Estrutura Básica - Documento HTML

A estrutura básica de um documento HTML₅ é comumente composta pelas tags `<!DOCTYPE html>`: informa ao navegador a versão HTML que está sendo utilizada; `<html></html>`: especifica o início e o fim do documento, onde serão colocados todos os elementos da página; `<head></head>`: especifica/delimita o cabeçalho do documento, onde é possível indicar ao navegador configurações importantes (documentos de estilo, links externos) para renderização da página; `<title></title>`: especifica o título da página, texto que será exibido na janela, aba ou guia do navegador; `<body></body>`: especifica/delimita o corpo do documento, dentro dela são inseridos todos os elementos da página que serão visíveis ao usuário.

```
<!DOCTYPE html>
```

```
<html>
  <head>
    <title>Document</title>
  </head>
  <body>
  </body>
</html>
```

Estrutura básica de um documento HTML - Codificação.
(Arquivo-fonte: padrao.html)

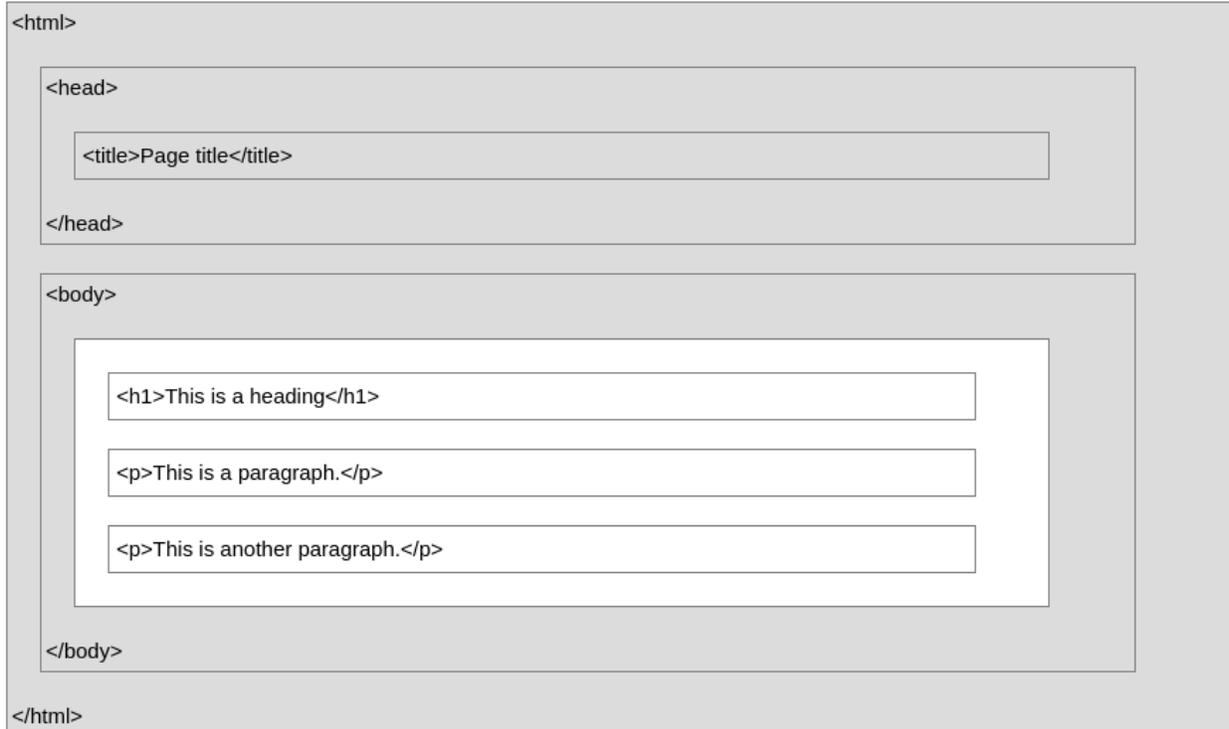


Figura 3: Estrutura básica de um documento HTML - Layout.

A estrutura básica de um arquivo HTML₅ pode ser criada, ao utilizarmos o VSCode, através do comando de atalho: **! + TAB**

```
<!DOCTYPE html>
<html lang="en">
```

```
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>
</body>
</html>
```

Estrutura padrão de um documento HTML₅ criada via atalho do VS Code.
(Arquivo-fonte: padrao_vscode.html)

A tag `<meta charset="UTF-8">` é utilizada para indicar o formato de codificação de caracteres que será utilizado na criação do documento HTML.

A tag `<meta http-equiv="X-UA-Compatible" content="IE=edge">` é exclusiva do navegador Internet Explorer (introduzido no IE8), utilizada para definir as configurações de renderização do documento HTML₅ no Internet Explorer. Atualmente tornou-se uma tag de pouca importância, podendo ser removida do documento.

A tag `<meta name="viewport" content="width=device-width, init-scale=1.0">` é utilizada para dar suporte ao processo de criação de páginas responsivas. No exemplo, a propriedade `content` indica que o navegador deve considerar como largura do layout da página, a largura do dispositivo (smartphone, computador) usado para visualização do conteúdo (`width`). Além disso, também é configurado o zoom (`init-scale`) que deverá ser utilizado na exibição da página, no exemplo o layout será exibido corretamente em escala 1:1, nenhum zoom será aplicado. Essas especificações são feitas através dos atributos das tags HTML, utilizados para configurar as características dos elementos.

Atributos

Os atributos são usados para configurar e personalizar as tags HTML, possibilitando modificar sua estrutura e/ou funcionalidade. As tags costumam conter seus próprios atributos. Ainda sim, existe um conjunto de atributos que são genéricos, utilizados na maioria das tags HTML:

- **class**: atribui uma classe ao elemento. Utilizado quando temos elementos do HTML que precisam ser agrupados de alguma forma, como, por exemplo, estilos e ações que devem ser aplicados em todos os elementos da classe;
- **id**: atribui um identificador único (ID) ao elemento, permitindo que seja referenciado através deste identificador;
- **style**: permite adicionar estilo a tag / elementos CSS;
- **alt**: define um texto alternativo ao elemento, utilizado quando há problemas para carregar imagens, pelo sintetizador de voz que dão suporte a deficientes visuais, entre outros. Considerando as boas práticas de programação, deve ser utilizado sempre como atributo da tag ``;
- **hidden**: possibilita ocultar o elemento;
- **align**: possibilita definir o alinhamento do elemento - 'right', 'center', 'left' e 'justify';
- **width**: permite definir a largura do elemento;
- **height**: permite definir a altura do elemento.

Elementos Estruturais / Semântica do HTML₅

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
  <meta charset="UTF-8">
  <title>Elementos Estruturais</title>
</head>
<body>
  <header>
```

```
<h1>Documento HTML</h1>
<nav>
  <a href="/">Navegação</a>
</nav>
</header>
<main>
  <section id="conteudo">
    Conteúdo
  </section>
</main>
<footer>
  Rodapé
</footer>
</body>
</html>
```

Elementos Estruturais do HTML₅ - Codificação.
(Arquivo-fonte: elementos_estruturais.html)



Figura 4: Elementos Estruturais do HTML₅ - Layout.

- **<header>** - contêiner que armazena o conteúdo introdutório e/ou os links de navegação da página.
- **<h1>** - define o nível de importância (tamanho) do texto exibido, começando em 1 (<h1> - mais importante) e indo até 6 (<h6> - menos importante);
- **<nav>** - define um conjunto de links para navegação entre páginas;
- **<main>** - especifica o conteúdo principal do documento;
- **<section>** - define uma nova seção para o documento;
- **<footer>** - especifica um rodapé para o documento ou para uma seção;

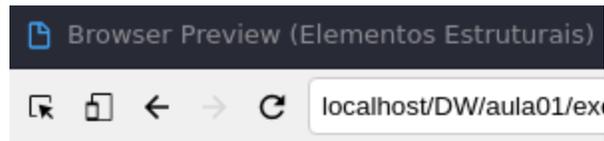
Mais exemplos práticos e um pouco de CSS

Utilizando Seletores CSS

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
  <meta charset="UTF-8">
  <title>Elementos Estruturais</title>
  <style>
    h1 {
      color: red;
    }
    .content {
      font-size: 32px;
      font-weight: bold;
      background-color: #66A;
    }
    .footer {
      font-size: 21px;
      font-style: italic;
    }
    #conteudo {
      margin: 50px;
    }
  </style>
</head>
<body>
  <h1>Elementos Estruturais</h1>
  <div class="content">
    <p>Conteúdo principal</p>
  </div>
  <div class="footer">
    <p>Rodapé</p>
  </div>
  <div id="conteudo">
    <p>Conteúdo principal</p>
  </div>
</body>
</html>
```

```
    }  
  </style>  
</head>  
<body>  
  <header>  
    <h1>Documento HTML</h1>  
    <nav>  
      <a href="/">Navegação</a>  
    </nav>  
  </header>  
  <main>  
    <section id="conteudo">  
      <span class="content">Conteúdo</span>  
    </section>  
  </main>  
  <footer>  
    <span class="footer">Rodapé</span>  
  </footer>  
</body>  
</html>
```

Elementos Estruturais do HTML₅ - Codificação.
(Arquivo-fonte: exemplo01.html)



Documento HTML

[Navegação](#)

Conteúdo

Rodapé

Figura 5: Elementos Estruturais do HTML₅ - Layout.

Seletores CSS e Elementos HTML:

- **<style>**: contém informações sobre o estilo da página (CSS);
- **h1{ }**: seletor de tipo, permite especificar o estilo para um elemento específico do HTML, nesse caso **<h1>**
- **.content{ }** / **.footer{ }**: seletor de classe, permite estilizar um grupo de elementos. No caso do exemplo foi utilizado nas linhas: ** Conteúdo ** e ** Rodapé **
- **#conteudo{ }**: seletor identificador, permite estilizar elementos de acordo com o seu atributo id. No caso do exemplo foi utilizado na linha: **<section id="conteudo">**
- ****: é um container genérico em linha utilizado, normalmente, para agrupar elementos de acordo com uma estilização.

Propriedades e Valores dos Seletores CSS:

- **color: red;** - indica que a cor do texto exibido pelo elemento vinculado ao seletor que contém a propriedade **color** será vermelho;
- **font-size: 32px;** - indica que o tamanho de fonte do texto exibido pelo elemento vinculado ao seletor que contém a propriedade **font-size** será **32px**;
- **font-weight: bold;** - indica que a espessura do texto exibido pelo elemento vinculado ao seletor que contém a propriedade **font-weight** será no formato negrito;
- **background-color: #66A;** - indica que a cor de fundo do elemento vinculado ao seletor que contém a propriedade **background-color** será de acordo com escala RGB **#66A**;
- **font-style: italic;** - indica que o estilo do texto exibido pelo elemento vinculado ao seletor que contém a propriedade **font-style** será no formato itálico;
- **margin: 50px;** - indica que o elemento vinculado ao seletor que contém a propriedade **margin** deve ter uma margem (espaçamento) de 50 px em todas as direções (left, right, top, bottom);

PRINCIPAIS TAGS HTML

A seguir serão apresentadas as principais <tags> utilizadas para construção de documentos HTML. Considerando que existe uma quantidade grande de <tags> disponíveis, serão abordadas apenas as mais relevantes. No decorrer das próximas aulas, sempre que necessário, novas <tags> serão abordadas. Neste contexto, a dinâmica da aula será baseada na apresentação de exemplos práticos de funcionamento dessas <tags>, seguidos da respectiva explicação.

Tags de Texto (<h1>, ..., <h6>, <p>, , <i>, <sup>, <sub>)

```
<!-- Cabeçalhos -->
```

```
<h1>Cabeçalho Principal / Nível 1</h1>
<h2>Cabeçalho Nível 2</h2>
<h3>Cabeçalho Nível 3</h3>
<h4>Cabeçalho Nível 4</h4>
<h5>Cabeçalho Nível 5</h5>
<h6>Cabeçalho Nível 6</h6>
<!-- Parágrafo -->
<p>
    Gil Eduardo de Andrade - Desenvolvimento Web
</p>
<!-- Negrito e Itálico -->
<p><b>[Negrito] Aula 02: </b><i>Linguagem HTML [Itálico]</i>.</p>
<!-- Sobrescrito e Subscrito -->
<p> - Equação do 2<sup>o</sup>: x<sup>2</sup> = 4</p>
<p> - Ácido Sulfúrico: H<sub>2</sub>SO<sub>4</sub></p>
<!-- Espaço em branco -->
<p>Espaço      entre      palavras</p>
```

<tags> de texto - exemplo 01.
(Arquivo-fonte: texto01.html)

- **<h1>**, ..., **<h6>**: utilizadas para apresentação de textos de acordo com o nível de importância, sendo o mais importante **<h1>**, e o mesmo importante **<h6>**;
- **<p>**: define um novo parágrafo dentro do documento HTML;
- ****: especifica que o texto colocado em seu corpo deve ser apresentado em negrito;
- **<i>**: especifica que o texto colocado em seu corpo deve ser apresentado em itálico;
- **<sup>**: especifica que o texto colocado em seu corpo deve ser apresentado sobrescrito;

- `<sub>`: especifica que o texto colocado em seu corpo deve ser apresentado subscrito;

Cabeçalho Principal / Nível 1

Cabeçalho Nível 2

Cabeçalho Nível 3

Cabeçalho Nível 4

Cabeçalho Nível 5

Cabeçalho Nível 6

Gil Eduardo de Andrade - Desenvolvimento Web

[Negrito] Aula 02: *Linguagem HTML [Itálico]*.

- Equação do 2º: $x^2 = 4$

- Ácido Sulfúrico: H_2SO_4

Espaço entre palavras

Figura 6: Tags HTML de Texto.

Tags de Texto (`
`, `<hr>`, ``, ``, `<blockquote>`, `<abbr>`, `<cite>`, ...)

```
<!-- Quebra de Linha <br> e Quebra Temática <hr> -->
<p>Nova linha<br>abaixo</p>
<p>Outra nova linha<br/>abaixo</p>
<hr />
<p>Novo tópico após linha horizontal</p>
<hr />
<!-- Marcações semânticas -->
```

```
<p>
  <strong>Atenção!</strong> Área <em>restrita.</em>
</p>
<p>
  <!-- Citação Longa -->
  <blockquote
cite="https://ieeexplore.ieee.org/document/8447368">Delay/Disruption-Tolerant Networks
(DTN) was initially used for Inter-Planetary Networks (IPN)...
  </blockquote>
</p>
<p>
  <!-- Citação Curta -->
  O diretor do campus disse: <q>é com imensa alegria que... </q>
</p>
<!-- Outras tags -->
<p>
  <!-- Abreviatura -->
  O <abbr title="Doutor em Filosofia">PhD</abbr> Michael Greeves escreveu em sua
tese...
</p>
<p>
  <!-- Definição de Título (Trabalho / Obra) -->
  O filme <cite>Avatar</cite> apresentada a história...
</p>
<p>
  <!-- Definição de Termo -->
  <dfn>AJAX</dfn> é uma tecnologia que envolve JavaScript e XML...
</p>
<!-- Dados para Contato -->
<address>
  <p>Rua Antônio Carlos Rodrigues, n<sup>o</sup> 453</p>
</address>
```

```
<p>  
  <!-- Texto Removido / Texto Inserido -->  
  A palavra <del>excesão</del> deve ser escrita <ins>exceção</ins>,  
</p>  
<!-- Texto não mais relevante -->  
<p>Promoção, Notebook Dell, de <s>R$ 4.299,00</s> por R$ 3.099,00.</p>
```

<tags> de texto - exemplo 02.
(Arquivo-fonte: texto02.html)

- **
**: define uma quebra de linha (nova linha);
- **<hr>**: define uma quebra temática - linha horizontal;
- ****: define um texto importante que deve ser destacado;
- ****: define um texto enfatizado (destacado);
- **<blockquote>**: especifica uma seção de citação (longa);
- **<q>**: especifica uma seção de citação (curta);
- **<abbr>**: define uma abreviatura;
- **<dfn>**: especifica um termo que será definido no conteúdo do texto;
- **<cite>**: define o título de um trabalho (livro, poema, música);
- **<address>**: define informações de contato (endereço, e-mail, telefone);
- ****: define um texto removido do documento;
- **<ins>**: define um texto inserido no documento;
- **<s>**: define um texto que não é mais correto ou relevante;

Nova linha
abaixo

Outra nova linha
abaixo

Novo tópico após linha horizontal

Atenção! Área restrita.

Delay/Disruption-Tolerant Networks (DTN) was initially used for Inter-Planetary Networks (IPN)...

O diretor do campus disse: “é com imensa alegria que...”

O Ph.D. Michael Greeves escreveu em sua tese...

O filme *Avatar* apresentada a história...

AJAX é uma tecnologia que envolve JavaScript e XML...

Rua Antônio Carlos Rodrigues, nº 453

A palavra ~~excesão~~ deve ser escrita exceção,

Promoção, Notebook Dell, de ~~R\$ 4.299,00~~ por R\$ 3.099,00.

Figura 7: Tags HTML de Texto.

Tags de Listas (, , , <dl>, <dt>, <dd>)

```
<h1>Listas</h1>
<!-- Lista Ordenada -->
<h2>Classificação</h2>
  <ol>
    <li>Alemanha</li>
    <li>Estados Unidos</li>
    <li>Japão</li>
    <li>China</li>
  </ol>
<!-- Lista Não Ordenada -->
```

```
<h2>Disciplinas</h2>
```

```
<ul>
```

```
<li>Linguagem de Programação</li>
```

```
<li>Orientação a Objeto</li>
```

```
<li>Desenvolvimento Web</li>
```

```
<li>Desenvolvimento Mobile</li>
```

```
</ul>
```

```
<!-- Lista de Definição -->
```

```
<h2>Glossário de Cursos</h2>
```

```
<dl>
```

```
<dt>EMI-INFO</dt>
```

```
<dd>Ensino Médio Integrado - Informática</dd>
```

```
<dt>TADS</dt>
```

```
<dd>Tecnólogo em Análise e Desenvolvimento</dd>
```

```
<dt>TGA</dt>
```

```
<dd>Tecnólogo em Gestão Ambiental</dd>
```

```
</dl>
```

<tags> de listas.
(Arquivo-fonte: listas.html)

- ****: define uma lista ordenada;
- ****: define um item de uma lista;
- ****: define uma lista não ordenada;
- **<dl>**: especifica uma lista de definições;
- **<dt>**: define um termo / nome numa lista de definições;
- **<dd>**: descreve um termo / nome numa lista de definições;

Listas

Classificação

1. Alemanha
2. Estados Unidos
3. Japão
4. China

Disciplinas

- Linguagem de Programação
- Orientação a Objeto
- Desenvolvimento Web
- Desenvolvimento Mobile

Glossário de Cursos

EMI-INFO	Ensino Médio Integrado - Informática
TADS	Tecnólogo em Análise e Desenvolvimento
TGA	Tecnólogo em Gestão Ambiental

Figura 8: Tags HTML de Texto.

Listas Aninhadas

```
<h1>Organização IFPR-Paranaguá</h1>
<ul>
  <li>
    <span>Áreas e Eixos</span>
    <ul>
      <li>
        <span>Informação e Comunicação</span>
```

```
<ul>
  <li>EMI - Informática</li>
  <li>Tecnólogo em Análise e Desenvolvimento</li>
</ul>
</li>
<li>
  <span>Recurso Naturais</span>
  <ul>
    <li>EMI - Meio Ambiente</li>
    <li>Tecnólogo em Gestão Ambiental</li>
  </ul>
</li>
</ul>
</li>
<li>
  <span>Blocos Didáticos</span>
  <ul>
    <li>
      <span>Mecânica</span>
      <ul>
        <li>Laboratório de Manutenção</li>
        <li>Laboratório de Automação</li>
      </ul>
    </li>
  </ul>
</li>
</ul>
```

Listas Aninhadas
(Arquivo-fonte: listas_aninhadas.html)

Organização IFPR-Paranaguá

- Áreas e Eixos
 - Informação e Comunicação
 - EMI - Informática
 - Tecnólogo em Análise e Desenvolvimento
 - Recurso Naturais
 - EMI - Meio Ambiente
 - Tecnólogo em Gestão Ambiental
- Blocos Didáticos
 - Mecânica
 - Laboratório de Manutenção
 - Laboratório de Automação

Figura 9: Listas aninhadas - Layout.

Listas Aninhadas - Árvore (CSS)

```
<head>
  <meta charset="UTF-8">
  <title>Listas Aninhadas CSS</title>
  <!-- Folha de Estilos / CSS -->
  <link rel="stylesheet" href="arvore.css">
</head>
<body>
  <h1>Organização IFPR-Paranaguá</h1>
  <!-- Aplica a classe 'arvore' na lista -->
  <ul class="arvore">
```

Listas aninhadas em árvore (CSS)
(Arquivo-fonte: listas_aninhadas_css.html)

Organização IFPR-Paranaguá

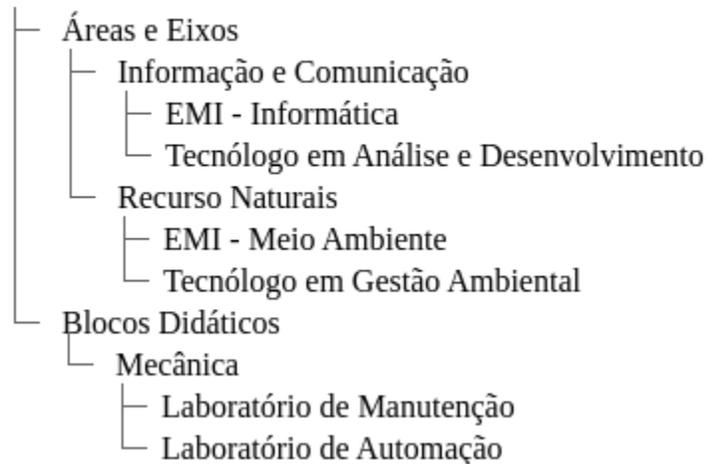


Figura 10: Listas aninhadas em árvores (CSS) - Layout.

Tag de Link (<a>)

```
<style>
  .distante {
    position: absolute;
    top: 1800px;
  }
</style>
<h1>Links</h1>
<!-- URL Absoluta -->
<a id="inicio" href="https://paranagua.ifpr.edu.br/">IFPR-Paranaguá</a><br>
<a href="https://paranagua.ifpr.edu.br" target="_blank">IFPR-Paranaguá (nova
aba)</a><br><br>
<!-- URL Relativa -->
<a href="Links.html">Links</a><br>
<a href="../Listas.html">Árvore</a><br><br>
```

```
<a href="mailto:gil.andrade@ifpr.edu.br">gil.andrade@ifpr.edu.br</a><br><br>
```

```
<div id="noticia" class="distante">
```

```
  <h1>Título da Notícia</h1>
```

```
  <p>Conteúdo da notícia</p>
```

```
  <a href="#inicio">Voltar ao Topo</a>
```

```
</div>
```

```
<a href="#noticia">Ver Notícia</a>
```

<tag> de link.

(Arquivo-fonte: links/exemplo01.html)

- **<a>**: define um hyperlink, utilizado para navegação entre páginas;
- **<div>**: define uma divisão ou uma seção do documento, utilizada como container para elementos HTML. Não é uma tag de link, mas como foi utilizada nos exemplos está sendo descrita;

[IFPR-Paranaguá](#)
[IFPR-Paranaguá \(nova aba\)](#)

[Links](#)
[Árvore](#)

gil.andrade@ifpr.edu.br

[Ver Notícia](#)

Figura 11: <tag> de link.

Tabelas (<table>, <caption>, <thead>, <tr>, <th>, <tbody>, <td>, <tfoot>)

```
<table>
  <caption>Conceitos</caption>
  <thead>
```

```
<tr>
  <th>Aluno</th>
  <th>1<sup>o</sup> Bimestre</th>
  <th>2<sup>o</sup> Bimestre</th>
  <th>Final</th>
</tr>
</thead>
<tbody>
  <tr>
    <td>Carlos Eduardo</td>
    <td>C</td>
    <td>B</td>
    <td>C</td>
  </tr>
  <tr>
    <td>Marina Conte</td>
    <td>A</td>
    <td>A</td>
    <td>A</td>
  </tr>
  <tr>
    <td>Fabiola Moraes</td>
    <td>A</td>
    <td>B</td>
    <td>B</td>
  </tr>
</tbody>
<tfoot>
  <tr>
    <td colspan="3">Média da Turma</td>
    <td>B</td>
  </tr>
</tfoot>
```

```
</tfoot>  
</table>
```

Tabelas (Arquivo-fonte: tabela.html)

- **<table>**: define uma tabela HTML;
- **<caption>**: define uma legenda para uma tabela;
- **<thead>**: serve para agrupar o conteúdo do cabeçalho de uma tabela ;
- **<tr>**: define uma nova linha para uma tabela;
- **<th>**: define as células de cabeçalho da tabela;
- **<tbody>**: serve para agrupar o conteúdo do corpo de uma tabela;
- **<td>**: define uma célula de dados de uma tabela;
- **<tfoot>**: serve para agrupar o conteúdo do rodapé de uma tabela;
- **<td colspan="3">**: serve estender uma célula por várias colunas, neste caso 3 colunas;

Aluno	Conceitos		
	1º Bimestre	2º Bimestre	Final
Carlos Eduardo	C	B	C
Marina Conte	A	A	A
Fabiola Moraes	A	B	B
Média da Turma	B		

Figura 12: Tabela - Layout.

***colspan:** usado para fazer com que uma célula se estenda por várias colunas, definindo quantas colunas uma única célula deve ocupar. Isso é útil para criar células de cabeçalho que abrangem várias colunas de dados.

***rowspan:** usado para fazer com que uma célula se estenda por várias linhas, definindo quantas linhas uma única célula deve ocupar. Isso é útil para criar células que abrangem várias linhas de dados relacionados.

Tabelas (CSS)

```
<style>

  table {
    border-collapse: collapse;
  }

  caption {
    font-size: 21px;
    font-weight: bold;
    color:rgb(184, 128, 26)
  }

  th {
    padding: 10px;
    color:rgb(184, 128, 26);
  }

  td {
    padding: 10px;
  }

  thead tr {
    border-bottom: solid 3px rgb(184, 128, 26);
  }

  tbody tr:hover {
    background-color: #faf3e3;
  }

 tfoot {
    background-color: rgb(224, 159, 37);
    color: #fff;
    font-weight: bold;
  }

</style>
```

Tabela (CSS)

(Arquivo-fonte: tabela_css.html)

Conceitos			
Aluno	1º Bimestre	2º Bimestre	Final
Carlos Eduardo	C	B	C
Marina Conte	A	A	A
Fabiola Moraes	A	B	B
Média da Turma			B

Figura 13: Tabela (CSS) - Layout.

Imagens - HTML (, <picture>)

```
<body>
  <h1>Imagens HTML</h1>
  <p>
    
    

    
  </p>
  <p>
```

```
<picture>
  <source srcset="https://p.kindpng.com/..."
    media="(min-width: 540px)">
  
</picture>
</p>
</body>
```

Trabalhando com imagens HTML (Arquivo-fonte: imagem.html)

- ****: utilizada para inserir uma imagem numa página HTML. Possui como principais atributos:
 - **src** - especifica o caminho onde encontra-se a imagem que deve ser exibida.
 - **width** - define a largura com a qual a imagem deve ser exibida.
 - **height** - define a altura com a qual a imagem deve ser exibida.
 - **alt** - possibilita definir um texto como alternativa a imagem, caso a mesma não possa ser exibida.
 - **title** - define um título para imagem, que é exibido quando o mouse está parado sobre ela.
- **<picture>**: proporciona uma maior flexibilidade para manipulação de imagens no HTML. A tag <picture> costuma ser utilizada para construção de interfaces responsivas, onde, ao invés de posicionar um conjunto de imagens uma embaixo da outra, quando a largura da janela diminui, elas podem ser projetadas para preencher de forma mais eficiente o conteúdo da janela. Possui duas tags principais:
 - **<source>**: nessa tag é especificada a imagem que será exibida pelo navegador de acordo com a largura da janela especificada. Quando a largura da janela é menor que o valor mínimo especificado, a imagem da tag <source> é substituída pela exibição da imagem especificada na tag . Possui como principais atributos:

- **srcset** - especifica o caminho onde encontra-se a imagem que deve ser exibida inicialmente .
 - **media** - indica a condição utilizada para definir se a imagem especificada deve ser exibida ou substituída.
- ****: nessa tag é especificada a imagem que deve ser exibida, em substituição a imagem da tag **<source>**, quando uma largura mínima da janela é alcançada.

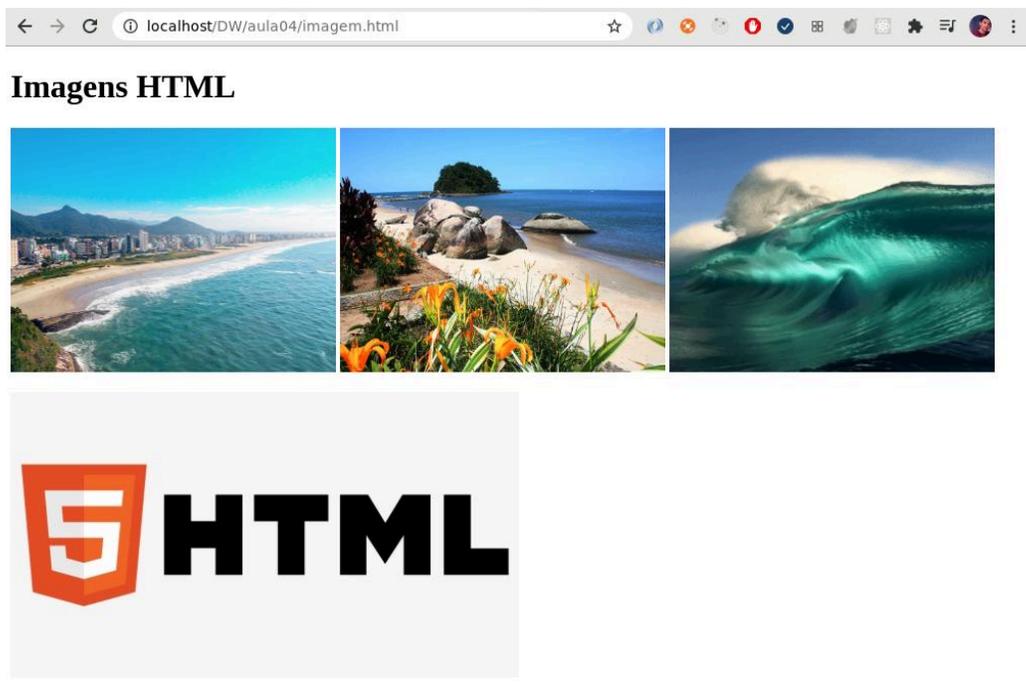


Figura 14: Trabalhando com imagens HTML - Layout / janela maximizada.

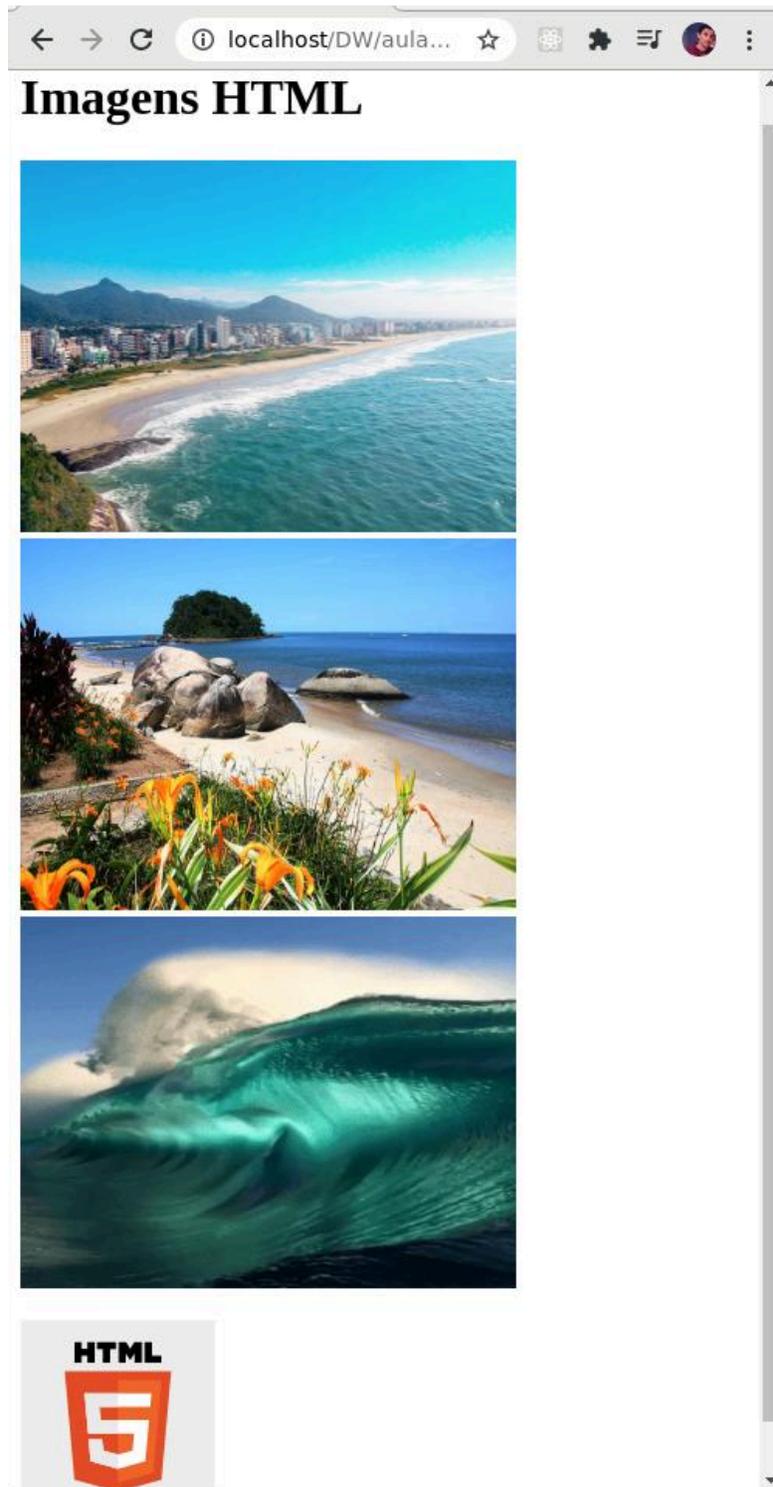


Figura 15: Trabalhando com imagens HTML - Layout / janela com largura reduzida.

Elemento <div> - HTML

O elemento **<div>** é usado como um contêiner para outros elementos HTML, sendo, por padrão, um elemento de bloco, o que significa que ele ocupa toda a largura disponível e efetuando quebras de linha antes e depois.

```
<body>
  <h1>HTML "div"</h1>
  <div>
    <h2>Curitiba</h2>
    <p>Curitiba é capital do Paraná.</p>
    <p>Curitiba possui um total, aproximado, de 1.8 milhões de habitantes.</p>
  </div>
  <p>O fundo laranja foi adicionado para destacar o elemento "div".</p>
</body>
```

Elemento <div>
(Arquivo-fonte: div.html)

HTML "div"

Curitiba

Curitiba é capital do Paraná.

Curitiba possui um total, aproximado, de 1.8 milhões de habitantes

O fundo amarelo foi adicionado para destacar o elemento "div".

Figura 16: Elemento <div>.

FORMULÁRIOS HTML

Os formulários HTML são utilizados para coletar as entradas de dados dos usuários. As informações coletadas são enviadas, geralmente, para um servidor de processamento (aplicação back-end).

Conceitos de Front-end e Back-end

Ao abordarmos o desenvolvimento web, constatamos que as tecnologias e linguagens disponíveis podem ser divididas em “**lado-cliente**” (*client-side*) e “**lado-servidor**” (*server-side*). A partir dessa característica surgiram os conceitos de **front-end** e **back-end**, que costumam ser utilizados para referir-se às tecnologias executadas no “*lado-cliente*” e “*lado-servidor*”, respectivamente. As linguagens *server-side* são aquelas interpretadas no “*lado-servidor*”, ou seja, o código-fonte escrito será processado pelo servidor e enviado ao navegador, em resposta a uma requisição HTTP. Já as linguagens *client-side* (interface do usuário) são aquelas interpretadas no “*lado-cliente*”, como, por exemplo, navegador. Sendo assim, desenvolvedores que atuam na construção da interface com o usuário (*client-side*) são denominados desenvolvedores *front-end*. Os profissionais que atuam com o recebimento e processamento de requisições vindas do cliente são denominados desenvolvedores *back-end*.

Node.js e Express

Para que seja possível visualizar como o envio dos dados de um formulário HTML, para o backend, iremos utilizar a plataforma Node.js (back-end) juntamente com o framework Express. O Node.js é um software multiplataforma que possibilita executar código Javascript do lado do servidor (back-end). O Express é um framework web escrito em JavaScript, executado sobre o ambiente Node.js. Dentre suas funcionalidades está a capacidade de gerenciar diferentes requisições HTTP (GET, POST, etc). As duas tecnologias serão estudadas em detalhes em aulas posteriores.

Configurações Iniciais (Lado Servidor)

- Instalação Node (<https://github.com/nvm-sh/nvm>)
 - Ambiente: Linux Mint 20.2 / 64 bits
 - Comandos / Terminal:
 - 1) Instalação do Node Version Manager – nvm
curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.38.0/install.sh | bash
 - 2) Visualizar versões do Node instaladas localmente (**nenhuma até aqui**)
nvm ls
 - 3) Visualizar versões remotas do Node (escolher última versão LTS)
nvm ls-remote
 - 4) Instalação da última versão LTS do NodeJS
nvm install v14.16.0
 - 5) Visualizar versão local instalada do Node (deve aparecer a LTS escolhida)
nvm ls
 - 6) Visualizar versão do nvm instalada (npm – Node Package Manager)
npm -v
 - 7) Atualizar / Futuramente (instalar nova versão / definir como versão padrão)
nvm install Vxx.xx.x
nvm alias default Vxx.xx.x

- Criação da aplicação servidor (Node+Express)
 - Comandos / Terminal:
 - 1) Criação da aplicação Node / arquivo package.json
npm init -y

Obs.: o arquivo package.json é o ponto de partida na criação de um projeto Node, nele são descritas: versão do projeto, dependências de produção, dependências de desenvolvimento, entre outras coisas.
 - 2) Instalação do Express
npm install --save express

Obs.: o parâmetro “install” permite a instalação de um pacote necessário no desenvolvimento da aplicação Node. O parâmetro

“--save” coloca o pacote instalado como dependência dentro do arquivo “package.json”.

Formulário HTML (<form>, <input[type=text]>, <input[type=submit]>)

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
  <meta charset="UTF-8">
  <title>Formulário HTML</title>
</head>
<body>
  <h1>Formulário HTML</h1>
  <form action="http://localhost:3000/alunos" method="POST">
    <input type="text" name="nome" />
    <input type="submit" value="Submeter / Enviar" />
  </form>
</body>
</html>
```

Formulario HTML / front-end.
(Arquivo-fonte: form.html)

- **<form>**: utilizado para criação de formulários HTML. Possui como principais atributos:
 - **action** - define a ação que deve ser efetuada quando o formulário é submetido (enviado).
 - **method** - especifica o método (verbo) HTTP que deve ser utilizado para o envio dos dados do formulário.
- **<input>**: utilizado para criar campos de entrada de dados em formulários HTML. Possui como principais atributos:

- **type** - define o tipo (formato) do campo de entrada (texto, botão, submissão, caixa de seleção, caixa de combinação, etc).
- **name** - define uma string de referência para o campo criado, permitindo a manipulação do dado de entrada vinculado a ele.
- **value** - especifica um valor que será vinculado ao campo de entrada.

Codificação Node - *Server-side* (arquivo server.js)

```
// importa o express
const express = require('express')

// instancia o express e atribui a "app"
const app = express()

// possibilita que express reconheça objetos request
// como strings ou arrays -> req.body
// urlencodded é formato padrão do formulário recebido
app.use(express.urlencoded({extended: true}));

// inidica que o server vai atender uma requisição POST
// para URL "/alunos"
app.post('/alunos', (req, resp) => {

  console.log(req.body)

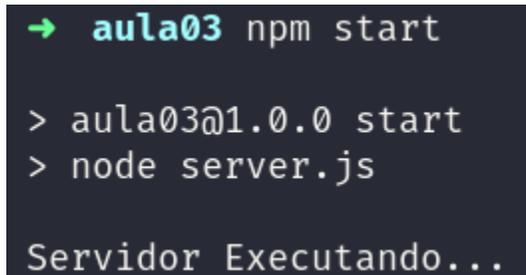
  resp.send('<h1>[OK] Novo Cadastro Recebido!</h1>')
})

// inidica que o server vai atender uma requisição POST
// para URL "/alunos/id"
```

```
app.post('/alunos/:id', (req, resp) => {  
  
  console.log(req.body)  
  
  console.log(req.params.id)  
  
  resp.send('<h1>[OK] Nova Alteração Recebida!</h1>')  
  
})  
  
console.log('Servidor Executando...')  
  
// coloca a aplicação para ouvir a porta 3000  
  
app.listen(3000)
```

Servidor Node / back-end.
(Arquivo-fonte: server.js)

Executando o servidor: *npm start*



```
→ aula03 npm start  
  
> aula03@1.0.0 start  
> node server.js  
  
Servidor Executando...
```

Figura 17: Servidor Node em execução.

Efetando a requisição: *abrir o arquivo form01.html no navegador*



Figuras 18: Abrindo o formulário HTML no navegador.



Figuras 19: Submetendo o formulário (click botão).

```
→ aula03 npm start  
  
> aula03@1.0.0 start  
> node server.js  
  
Servidor Executando...  
{ nome: 'Gabriela' }
```

Figuras 20: Recebendo a requisição, exibindo os dados e respondendo (back-end).

Obs.: o método `express.urlencoded()` converte os dados do formulário para um objeto Javascript (`{nome: 'Gabriela'}`). O “nome” vem do atributo `name=“nome”` utilizado no componente HTML `<input type= “text”>`. Como teste sugere-se comentar a linha com o método

express.urlenconde() e efetuar uma nova submissão, analisando o resultado do `console.log()` no back-end.

Formulário HTML (<textarea>, <input[type=radio]>, <input[type=checkbox]>, <select>)

```
<form action="http://localhost:3000/alunos" method="POST">
  <div class="bloco">
    <label>Nome</label>
    <input type="text" name="nome" />
    <label>Observações</label>
    <textarea name="observacao" cols="30" rows="5"></textarea>
    <label>Curso</label>
    <div class="linha">
      <input type="radio" name="curso" value="info">Informática
      <input type="radio" name="curso" value="mamb">Meio Ambiente
      <input type="radio" name="curso" value="mec">Mecânica
    </div>
    <label>Tópicos Avançados</label>
    <div class="linha">
      <input type="checkbox" name="topicos" value="Sociologia">Sociologia
      <input type="checkbox" name="topicos" value="Física">Física
      <input type="checkbox" name="topicos" value="Matemática">Matemática
      <input type="checkbox" name="topicos" value="Biologia">Biologia
      <input type="checkbox" name="topicos" value="Artes">Artes
    </div>
    <label>Responsável</label>
    <select name="responsavel">
      <option value="pai">Pai</option>
      <option value="mãe">Mãe</option>
    </select>
  </div>
</form>
```

```
<option value="avô">Avô</option>
<option value="avó">Avó</option>
<option value="outro">Outro</option>
</select>
<label>Esportes Praticados</label>
<select name="esportes" multiple="4">
  <option value="Futebol">Futebol</option>
  <option value="Volei">Volei</option>
  <option value="Basquete">Basquete</option>
  <option value="Handebol">Handebol</option>
</select>
<label>Data de Nascimento: </label>
<input type="date" name="data_n">
<label>Hora do Nascimento: </label>
<input type="time" name="hora_n">
<label>Cor Preferida: </label>
<input type="color" name="cor">
<label>Número da Sorte: </label>
<input type="number" name="numero">
<input type="submit" value="Submeter / Enviar" />
</div>
</form>
```

Principais elementos de um formulário HTML / front-end.
(Arquivo-fonte: form02.html)

- **<label>**: define um rótulo para os campos de entrada.
- **<textarea>**: define uma caixa de entrada para um texto que contém várias linhas.

- **cols** - define a quantidade de colunas (largura) que a entrada `<textarea>` deve conter.
- **row** - define a quantidade de linhas (altura) que a entrada `<textarea>` deve conter.
- **<div>**: define uma divisão ou seção do documento, utilizada para definir um container de elementos HTML, normalmente estilizados via CSS.
 - **class** - especifica uma ou mais classes de estilos (CSS) que serão aplicadas a `<div>`.
- **<input type="radio">**: utilizado para agrupar opções de entrada onde apenas uma pode ser selecionada:
 - **name** - define uma string de referência para o campo criado, permitindo a manipulação do dado de entrada vinculado a ele.
 - **value** - especifica o valor que será vinculado à opção de entrada selecionada.
- **<input type="checkbox">**: utilizado para agrupar opções de entrada onde várias delas podem ser selecionadas:
 - **name** - define uma string de referência para o campo criado, permitindo a manipulação do dado de entrada vinculado a ele.
 - **value** - especifica o valor que será vinculado à(s) opção(es) de entrada selecionada(s).
- **<select>**: utilizado para agrupar opções de entrada no formato padrão de caixa de combinação, onde, dependendo da configuração, apenas uma opção ou várias opções podem ser selecionadas:
 - **name** - define uma string de referência para o campo criado, permitindo a manipulação do dado de entrada vinculado a ele.
 - **multiple** - define o elemento `<select>` com formato de lista, permitindo que várias opções sejam selecionadas.

Efetuando a requisição: *abrir o arquivo **form02.html** no navegador*

Formulário HTML

Nome

Gabriela

Observações

Possui aptidões para escrita de textos, desenhos técnicos e programação...

Curso

Informática Meio Ambiente Mecânica

Tópicos Avançados

Sociologia Física Matemática Biologia Artes

Responsável

Avô

Esportes Praticados

Futebol
Volei
Basquete
Handebol

Data de Nascimento:

10/02/2003

Hora do Nascimento:

14:40

Cor Preferida:



Número da Sorte:

4

Submeter / Enviar

Figuras 21: Abrindo o formulário HTML no navegador.



Figuras 22: Submetendo o formulário (click botão).

```
{
  nome: 'Gabriela',
  observacao: 'Possui aptidões para escrita de textos, desenhos técnicos e programação...',
  curso: 'info',
  topicos: [ 'Sociologia', 'Matemática', 'Artes' ],
  responsavel: 'avô',
  esportes: [ 'futebol', 'handebol' ],
  data_n: '2003-02-10',
  hora_n: '14:40',
  cor: '#aa1dc3',
  numero: '4'
}
```

Figuras 23: Recebendo a requisição, exibindo os dados e respondendo (back-end).

- **<input type="date">**: utilizado como campo de entrada para definição de uma data:
 - **name** - define uma string de referência para o campo criado, permitindo a manipulação do dado de entrada vinculado a ele.
- **<input type="time">**: utilizado como campo de entrada para definição de um horário:
 - **name** - define uma string de referência para o campo criado, permitindo a manipulação do dado de entrada vinculado a ele.
- **<input type="color">**: utilizado como campo de entrada para definição de uma cor:
 - **name** - define uma string de referência para o campo criado, permitindo a manipulação do dado de entrada vinculado a ele.
- **<input type="number">**: utilizado como campo de entrada para definição de um número:
 - **name** - define uma string de referência para o campo criado, permitindo a manipulação do dado de entrada vinculado a ele.

Formulário HTML - Um Formulário para Vários Actions (<button formaction= "">)

```
<form method="POST">
  <div class="bloco">
    <input type="hidden" name="id" value="2"/>
```

```
<!-- A parte do código que é igual ao da Figura 07 foi removida, visto que já foi explicada anteriormente -->
```

```
<button formaction="http://localhost:3000/alunos">
```

```
    Cadastrar Aluno
```

```
</button>
```

```
<button formaction="http://localhost:3000/alunos">
```

```
    Alterar Aluno
```

```
</button>
```

```
</div>
```

```
</form>
```

Utilizando um mesmo formulário HTML para várias ações / front-end.
(Arquivo-fonte: form03.html)

- **<input type="date">**: utilizado como campo de entrada para definição de uma data:
 - **name** - define uma string de referência para o campo criado, permitindo a manipulação do dado de entrada vinculado a ele.

Codificação Node - *Server-side* (arquivo server.js)

```
// importa o express
const express = require('express')

// instancia o express e atribui a "app"
const app = express()

// possibilita que express reconheça objetos request
// como strings ou arrays -> req.body
// urlencodded é formato padrão do formulário recebido
app.use(express.urlencoded({extended: true}));
```

```
// inidica que o server vai atender uma requisição POST
// para URL "/alunos"
app.post('/alunos', (req, resp) => {
  console.log(req.body)
  resp.send('<h1>[OK] Novo Cadastro Recebido!</h1>')
})

// inidica que o server vai atender uma requisição POST
// para URL "/alunos/id"
app.post('/alunos/:id', (req, resp) => {
  console.log(req.body)
  console.log(req.params.id)
  resp.send('<h1>[OK] Nova Alteração Recebida!</h1>')
})

console.log('Servidor Executando...')

// coloca a aplicação para ouvir a porta 3000
app.listen(3000)
```

Servidor Node / back-end.
(Arquivo-fonte: server.js)

Efetuando as requisições: *abrir o arquivo form03.html no navegador*

Cadastrar Aluno

Alterar Aluno

Figura 13: Efetuando submissões distintas para o mesmo formulário / front-end.
(Arquivo-fonte: aula03/form03.html)

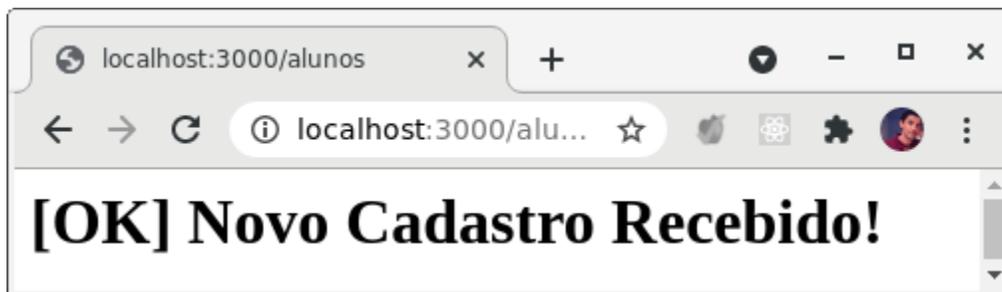


Figura 23: Submetendo o formulário para cadastro (click botão "Cadastrar Aluno").

```
{
  id: '2',
  nome: "Gabriela",
  observacao: 'Qualquer coisa...',
  curso: 'mec',
  topicos: [ 'Física', 'Matemática' ],
  responsavel: 'mãe',
  esportes: [ 'basquete', 'handebol' ],
  data_n: '2005-03-13',
  hora_n: '18:01',
  cor: '#0e4a8b',
  numero: '23'
}
```

Figuras 24: Recebendo a requisição para cadastro, exibindo os dados e respondendo (back-end).

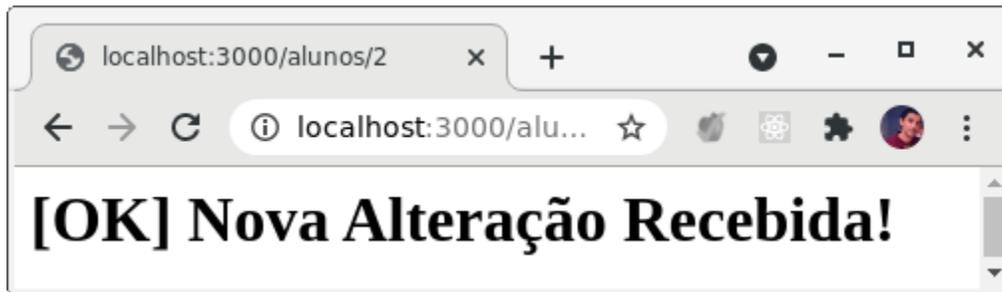


Figura 25: Submetendo o formulário para alteração (click botão “Alterar Aluno”).

```
{
  id: '2',
  nome: "Gabriela",
  observacao: 'Qualquer coisa...',
  curso: 'mec',
  topicos: [ 'Física', 'Matemática' ],
  responsavel: 'mãe',
  esportes: [ 'basquete', 'handebol' ],
  data_n: '2005-03-13',
  hora_n: '18:01',
  cor: '#0e4a8b',
  numero: '23'
}
```

Figuras 26: Recebendo a requisição para alteração, exibindo os dados e respondendo (back-end).