

GEX613 – Programação II

HTML



1100/1101 – CIÊNCIA DA COMPUTAÇÃO

Giancarlo Salton & Edimar Junior

O que é HTML?

HyperText Markup Language Tags

Documento HTML

Marcação

Express Server

O que é HTML?

- HTML (Linguagem de Marcação de Hipertexto) é o código que você usa para estruturar uma página web e seu conteúdo.

HyperText Markup Language

- HTML não é uma linguagem de programação, é uma *linguagem de marcação* !

Utilizada para definir estrutura do conteúdo

- HTML consiste de elementos utilizados para delimitar e agrupar diferentes partes do conteúdo para que ele apareça ou atue de uma maneira determinada.
- Cada elemento é definido por uma *tag*

<p>: parágrafos

<a>: links

: fonte negrito

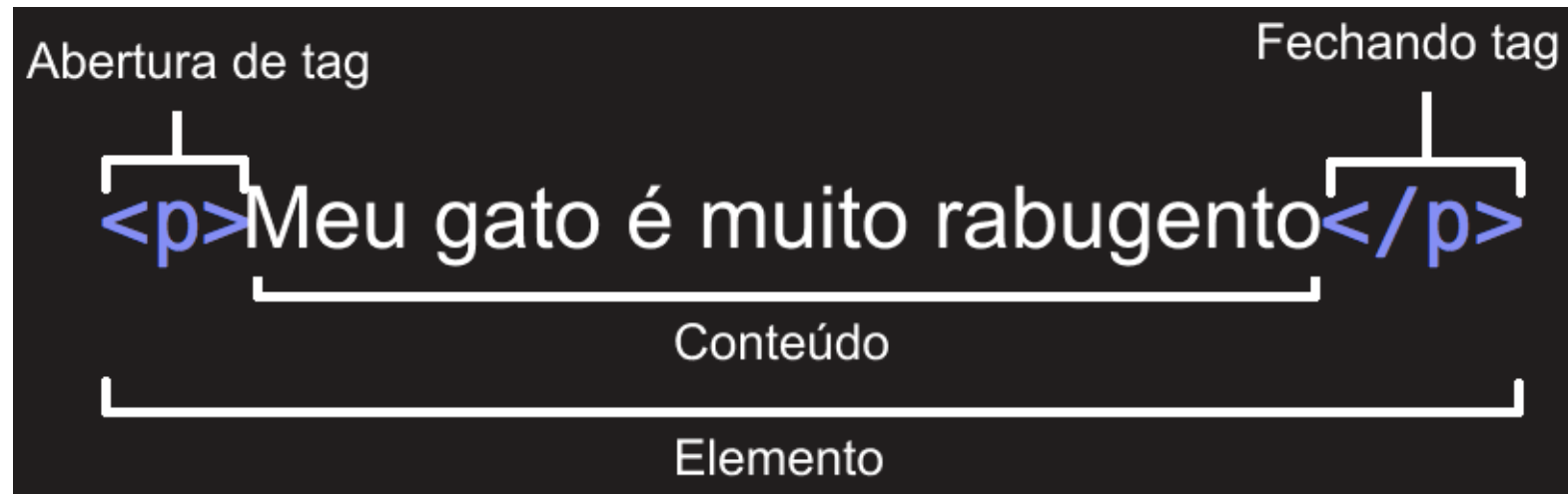
...

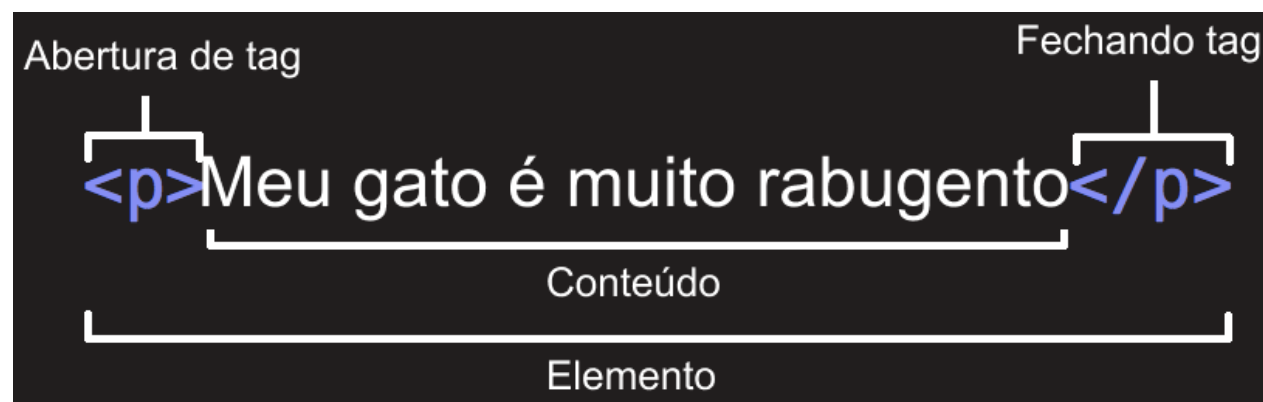
```
GET / HTTP/1.1
Host: developer.mozilla.org
Accept-Language: fr
```

```
HTTP/1.1 200 OK
Date: Sat, 09 Oct 2010 14:28:02 GMT
Server: Apache
Last-Modified: Tue, 01 Dec 2009 20:18:22 GMT
ETag: "51142bc1-7449-479b075b2891b"
Accept-Ranges: bytes
Content-Length: 29769
Content-Type: text/html
```

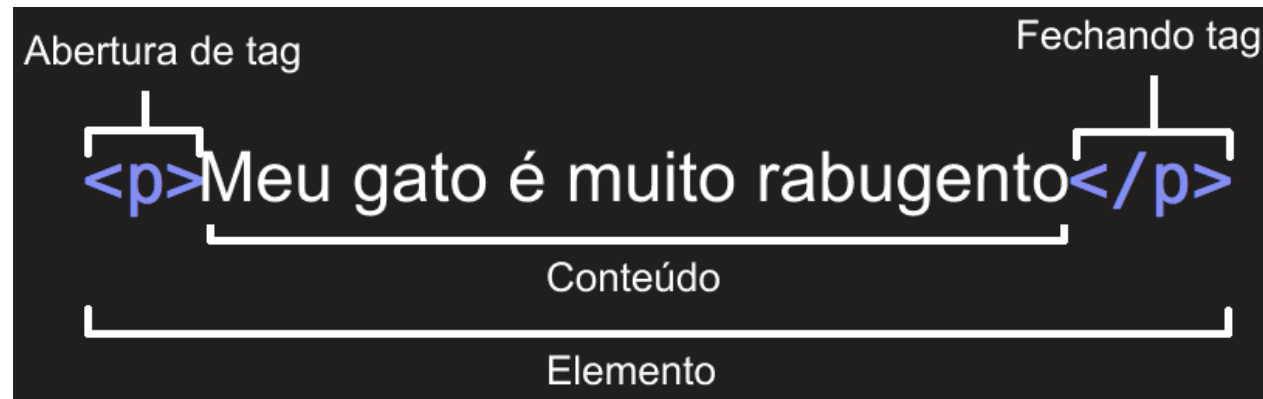
```
<!DOCTYPE html... (here comes the 29769 bytes of the requested web page)
```

HyperText Markup Language Tags

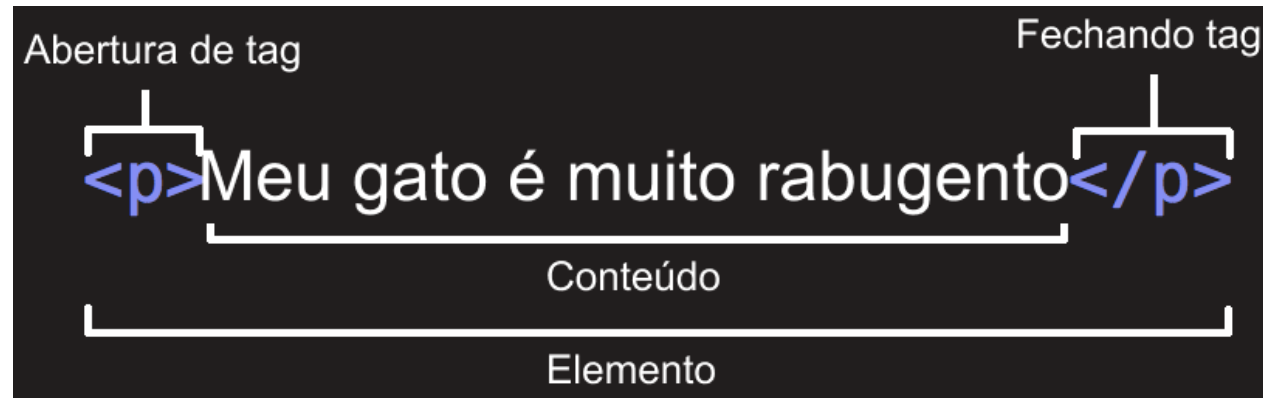




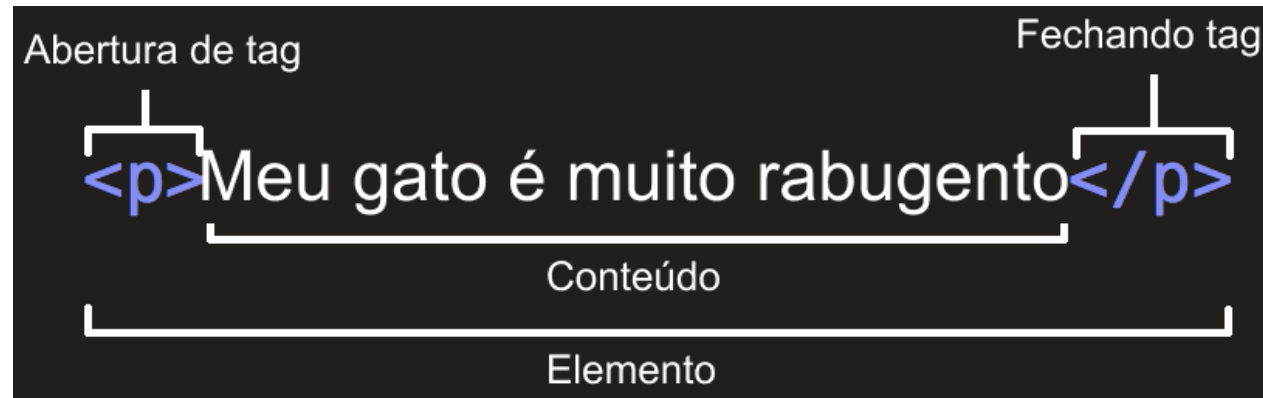
A **tag de abertura**: Consiste no nome do elemento (no caso, `p`), envolvido em **parênteses angulares** de abertura e fechamento. Isso demonstra onde o elemento começa, ou onde seu efeito se inicia — nesse caso, onde é o começo do parágrafo.



A **tag de fechamento**: Isso é a mesma coisa que a tag de abertura, exceto que inclui uma barra antes do nome do elemento. Isso demonstra onde o elemento acaba — nesse caso, onde é o fim do parágrafo. Esquecer de incluir uma tag de fechamento é um dos erros mais comuns de iniciantes e pode levar a resultados estranhos.



O **conteúdo**: Esse é o conteúdo do elemento, que nesse caso é apenas texto.



O **elemento**: A tag de abertura, a de fechamento, e o conteúdo formam o elemento.

Atributo

```
<p class="editor-note">Meu gato é muito rabugento</p>
```

Atributos contém informação extra sobre o elemento que você não quer que apareça no conteúdo real. Aqui, `class` é o nome do atributo e `editor-note` é o valor do atributo.

Atributo

```
<p class="editor-note">Meu gato é muito rabugento</p>
```

1. Um espaço entre ele e o nome do elemento (ou o atributo anterior, se o elemento já tiver um).
2. O nome do atributo, seguido por um sinal de igual.
3. Aspas de abertura e fechamento, envolvendo todo o valor do atributo.

```
<p>Meu gatinho é <strong>muito</strong> mal humorado.</p>
```

```
<p>Meu gatinho é <strong>muito mal humorado.</p></strong>
```

Você pode colocar
elementos dentro de outros
elementos também – isso
é chamado
de **aninhamento**.

`<p>Meu gatinho é muito mal humorado.</p>`

`<p>Meu gatinho é muito mal humorado.</p>`


```
<p>Meu gatinho é <strong>muito</strong> mal humorado.</p>
```

Você precisa, no entanto, certificar-se de que seus elementos estejam adequadamente aninhados.

```
<p>Meu gatinho é <strong>muito mal humorado.</p></strong>
```

```
<p>Meu gatinho é <strong>muito</strong> mal humorado.</p>
```

```
<p>Meu gatinho é <strong>muito mal humorado.</p></strong>
```

Os elementos precisam ser abertos e fechados corretamente para que eles estejam claramente visíveis dentro ou fora um do outro. Se eles se sobrepuserem conforme mostrado acima, seu navegador tentará adivinhar o que você estava tentando dizer, o que pode levar a resultados inesperados.

```

```

Alguns elementos não possuem conteúdo e são chamados de **elementos vazios**. Alguns elementos consistem apenas em uma única tag, que é geralmente usada para inserir/incorporar algo no documento no lugar em que ele é incluído.

Elementos vazios

Considere o elemento `` abaixo que contém dois atributos. Não há tag `` de fechamento e não há conteúdo interno. Isso acontece porque um elemento de imagem não envolve conteúdo para ter efeito em si mesmo

```

```

Alguns elementos não possuem conteúdo e são chamados de **elementos vazios**. Alguns elementos consistem apenas em uma única tag, que é geralmente usada para inserir/incorporar algo no documento no lugar em que ele é incluído.

Documento HTML

```
<!doctype html>
<html>
  <head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width" />
    <title>Minha página de teste</title>
  </head>
  <body>
    
  </body>
</html>
```

```
<!doctype html>
```

`<!doctype html>`
é a parte inicial obrigatória do documento.

```
<html>
```

```
  <head>
```

```
    <meta charset="utf-8" />
```

```
    <meta name="viewport" content="width=device-width" />
```

```
    <title>Minha página de teste</title>
```

```
  </head>
```

```
  <body>
```

```
    
```

```
  </body>
```

```
</html>
```

```
<!doctype html>
```

```
<html>
```

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

Esse elemento envolve todo o conteúdo da página e às vezes é conhecido como o elemento raiz.

```
<head>
```

```
<meta charset="utf-8" />
```

```
<meta name="viewport" content="width=device-width" />
```

```
<title>Minha página de teste</title>
```

```
</head>
```

```
<body>
```

```

```

```
</body>
```

```
</html>
```



```
<!doctype html>
```

```
<html>
```

```
  <head>
```

```
    <meta charset="utf-8" />
```

```
    <meta name="viewport" content="width=device-width" />
```

```
    <title>Minha página de teste</title>
```

```
  </head>
```

```
  <body>
```

```
    
```

```
  </body>
```

```
</html>
```

`<head></head>`

Esse elemento age como um recipiente de tudo o que você deseja incluir em uma página HTML que *não é* o conteúdo que você quer mostrar para quem vê sua página.

```
<!doctype html>
```

```
<html>
```

```
  <head>
```

```
    <meta charset="utf-8" />
```

```
    <meta name="viewport" content="width=device-width" />
```

```
    <title>Minha página de teste</title>
```

```
  </head>
```

```
  <body>
```

```
    
```

```
  </body>
```

```
</html>
```

```
<meta charset="utf-8" />
```

Esse elemento define o conjunto de caracteres que seu documento deve usar para o UTF-8.

```
<!doctype html>
<html>
  <head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width" />
    <title>Minha página de teste</title>
  </head>
  <body>
    
  </body>
</html>
```

`<title></title>`

Ele define o título da sua página, que é o título que aparece na guia do navegador onde sua página é carregada.

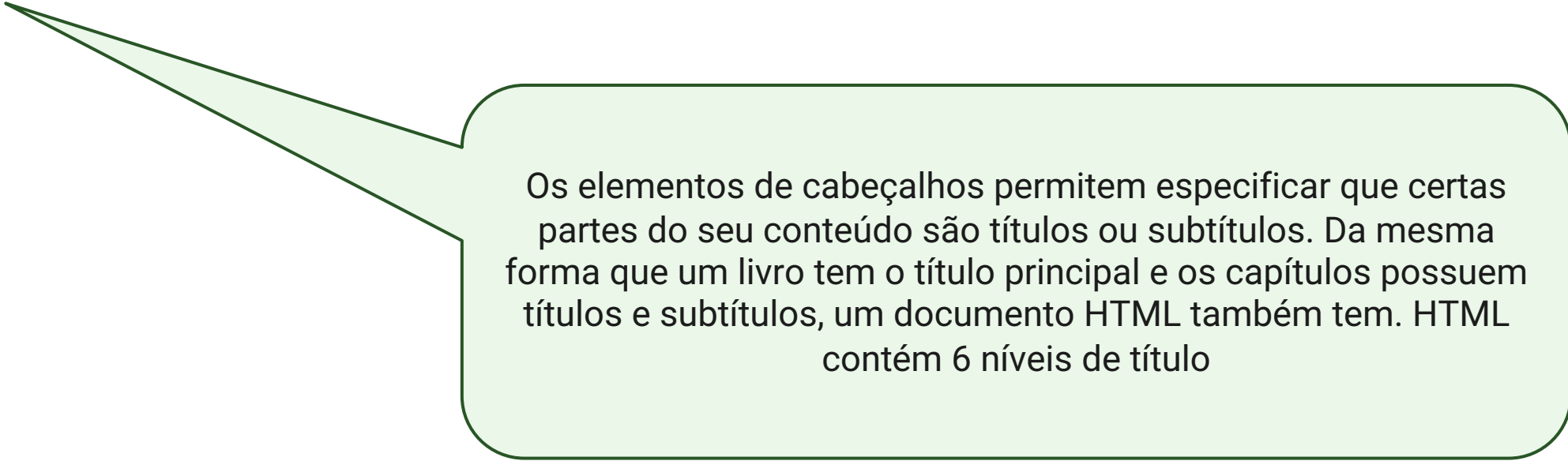
```
<!doctype html>
<html>
  <head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width" />
    <title>Minha página de teste</title>
  </head>
  <body>
    
  </body>
</html>
```

`<body></body>`

Contém *todo* o conteúdo que você quer mostrar ao público que visita sua página, seja texto, imagens, vídeos, jogos, faixas de áudio reproduzíveis ou qualquer outra coisa.

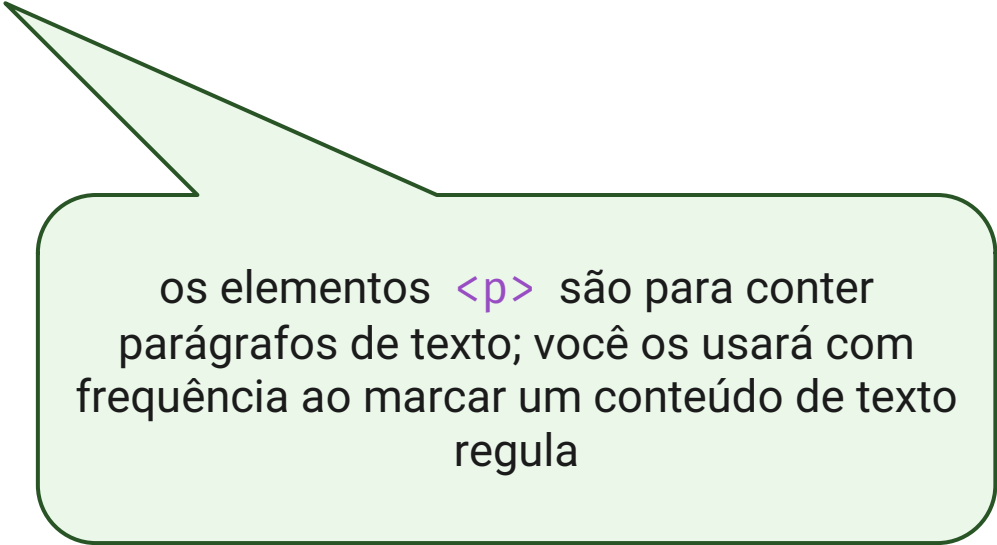
Marcação

```
<h1>Meu título principal</h1>  
<h2>Meu título de alto nível</h2>  
<h3>Meu subtítulo</h3>  
<h4>Meu segundo subtítulo</h4>
```



Os elementos de cabeçalhos permitem especificar que certas partes do seu conteúdo são títulos ou subtítulos. Da mesma forma que um livro tem o título principal e os capítulos possuem títulos e subtítulos, um documento HTML também tem. HTML contém 6 níveis de título

`<p>Este é um parágrafo simples</p>`



os elementos `<p>` são para conter parágrafos de texto; você os usará com frequência ao marcar um conteúdo de texto regula

`<p>Na Mozilla, somos uma comunidade global de</p>`

```
<ul>
  <li>tecnólogos</li>
  <li>pensadores</li>
  <li>construtores</li>
</ul>
```

`<p>trabalhando juntos ...</p>`

Listas de marcação sempre consistem em pelo menos 2 elementos. Os tipos mais comuns de lista são ordenadas e não ordenadas:

1.Listas não ordenadas são para listas onde a ordem dos itens não importa, como uma lista de compras, por exemplo. Essas são envolvidas em um elemento ``.

2.Listas Ordenadas são para listas onde a ordem dos itens importa, como uma receita. Essas são envolvidas em um elemento ``. Cada item dentro das listas é posto dentro de um elemento `` (item de lista).


```
<table>
  <tr>
    <th>Nome</th>
    <td>Knocky</td>
    <td>Flor</td>
  </tr>
  <tr>
    <th>Raça</th>
    <td>Jack Russell</td>
    <td>Poodle</td>
  </tr>
  <tr>
    <th>Idade</th>
    <td>16</td>
    <td>9</td>
  </tr>
  <tr>
    <th>Dono</th>
    <td>Irmã</td>
    <td>Du</td>
  </tr>
</table>
```

Conteúdo da tabela fica entre duas tags
<table></table>.

```
<table>
  <tr>
    <th>Nome</th>
    <td>Knocky</td>
    <td>Flor</td>
  </tr>
  <tr>
    <th>Raça</th>
    <td>Jack Russell</td>
    <td>Poodle</td>
  </tr>
  <tr>
    <th>Idade</th>
    <td>16</td>
    <td>9</td>
  </tr>
  <tr>
    <th>Dono</th>
    <td>Irmã</td>
    <td>Du</td>
  </tr>
</table>
```

As tags `<tr></tr>` definem uma linha da tabela.

```
<table>
  <tr>
    <th>Nome</th>
    <td>Knocky</td>
    <td>Flor</td>
  </tr>
  <tr>
    <th>Raça</th>
    <td>Jack Russell</td>
    <td>Poodle</td>
  </tr>
  <tr>
    <th>Idade</th>
    <td>16</td>
    <td>9</td>
  </tr>
  <tr>
    <th>Dono</th>
    <td>Irmã</td>
    <td>Du</td>
  </tr>
</table>
```

As tags `<th></th>` definem um cabeçalho de linha ou de coluna da tabela.

```
<table>
  <tr>
    <th>Nome</th>
    <td>Knocky</td>
    <td>Flor</td>
  </tr>
  <tr>
    <th>Raça</th>
    <td>Jack Russell</td>
    <td>Poodle</td>
  </tr>
  <tr>
    <th>Idade</th>
    <td>16</td>
    <td>9</td>
  </tr>
  <tr>
    <th>Dono</th>
    <td>Irmã</td>
    <td>Du</td>
  </tr>
</table>
```

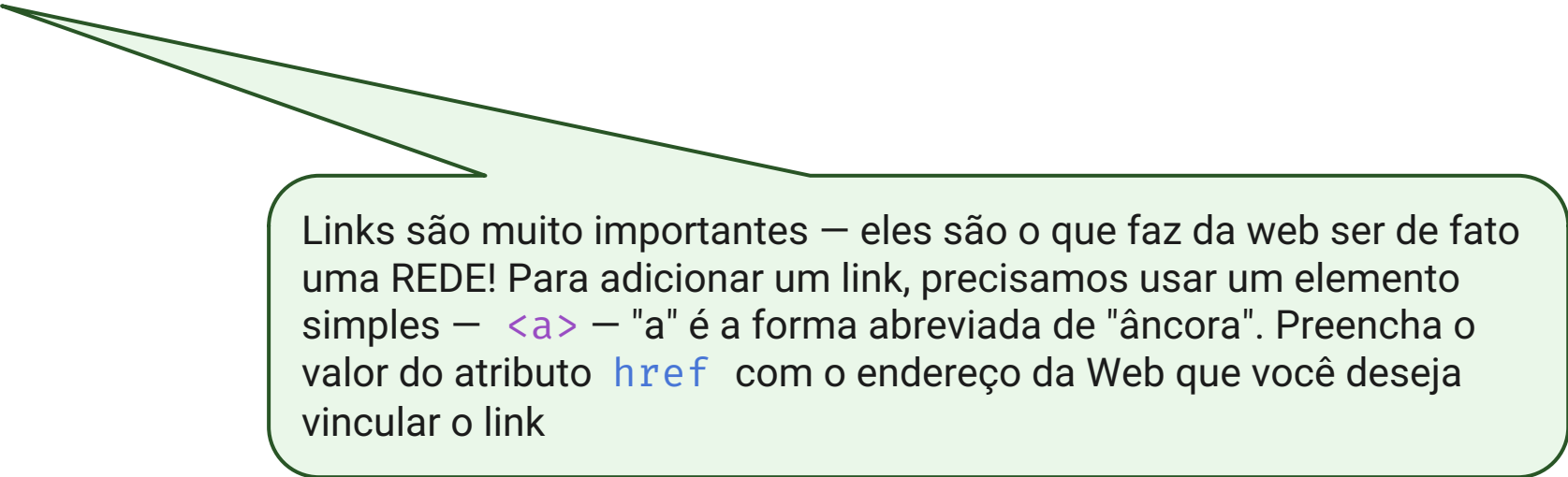
As tags `<td></td>` definem uma célula da tabela.

```
<table>
  <tr>
    <th>Nome</th>
    <td>Knocky</td>
    <td>Flor</td>
  </tr>
  <tr>
    <th>Raça</th>
    <td>Jack Russell</td>
    <td>Poodle</td>
  </tr>
  <tr>
    <th>Idade</th>
    <td>16</td>
    <td>9</td>
  </tr>
  <tr>
    <th>Dono</th>
    <td>Irmã</td>
    <td>Du</td>
  </tr>
</table>
```

Dica: para facilitar o desenvolvimento, você pode utilizar o link abaixo para gerar visualmente as tabelas:

https://www.tablesgenerator.com/html_tables

```
<a href="https://www.mozilla.org/pt-BR/about/manifesto/">Mozilla Manifesto</a>
```



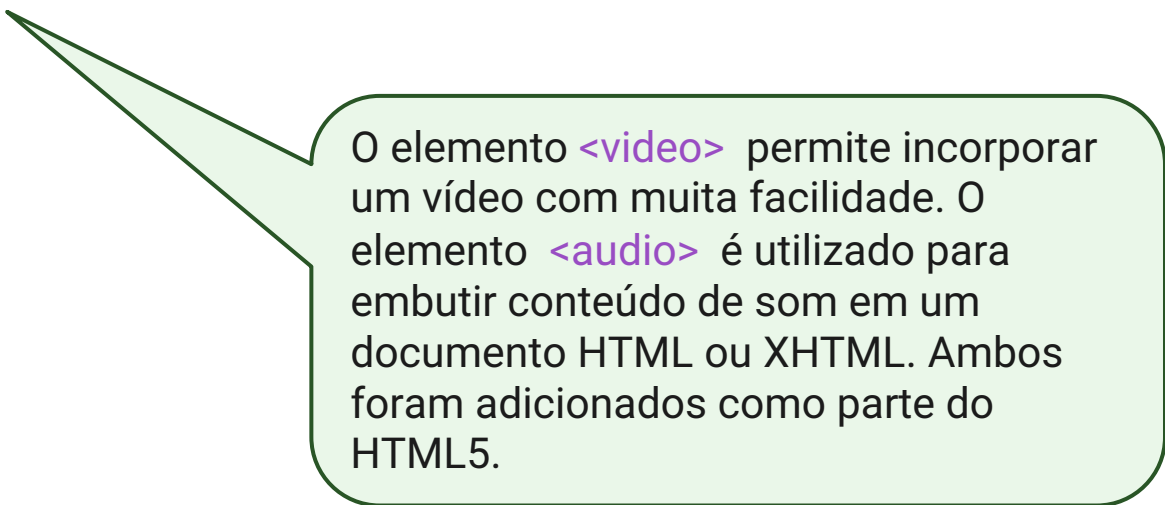
Links são muito importantes – eles são o que faz da web ser de fato uma REDE! Para adicionar um link, precisamos usar um elemento simples – `<a>` – "a" é a forma abreviada de "âncora". Preencha o valor do atributo `href` com o endereço da Web que você deseja vincular o link

```

```

Para adicionar uma imagem, precisamos usar um elemento simples `` preenchendo o valor do atributo `src` com o caminho para o arquivo de imagem (que pode ser uma URL web). Incluímos também um atributo `alt` (*alternative*). Neste atributo, você especifica um texto descritivo para usuários que não podem ver a imagem, tanto por serem deficientes visuais como se algo der errado ao carregar a página.

```
<video src="rabbit320.mp4" controls >
  <p>
    Your browser doesn't support HTML5 video. Here is a
    <a href="rabbit320.mp4">link to the video</a> instead.
  </p>
</video>
```



O elemento `<video>` permite incorporar um vídeo com muita facilidade. O elemento `<audio>` é utilizado para embutir conteúdo de som em um documento HTML ou XHTML. Ambos foram adicionados como parte do HTML5.


```
<video
  controls
  width="400"
  height="400"
  autoplay
  loop
  muted
  poster="poster.png">
  <source src="rabbit320.mp4" type="video/mp4" />
  <source src="rabbit320.webm" type="video/webm" />
  <p>
    Your browser doesn't support HTML5 video. Here is a
    <a href="rabbit320.mp4">link to the video</a> instead.
  </p>
</video>
```

Existem muitas opções para melhor controlar os elementos e . Veja o link abaixo para as opções disponíveis e algumas boas práticas.

Express Server

```
const express = require('express');
const bodyParser = require('body-parser');

const app = express();

app.use(bodyParser.json());
app.listen(3001, () => console.log('Servidor na porta 3001.'));

app.get('/', (request, response) => {
  response.sendFile(`_${__dirname}/index.html`);
});
```

```
const express = require('express');  
const bodyParser = require('body-parser');  
  
const app = express();  
  
app.use(bodyParser.json());  
app.listen(3001, () => console.log('Servidor na porta 3001.'));  
  
app.get('/', (request, response) => {  
  response.sendFile(`${__dirname}/index.html`);  
});
```

Como visto anteriormente, estas primeiras cinco linhas são requeridas para que o servidor funcione da forma correta.

```
const express = require('express');
const bodyParser = require('body-parser');

const app = express();

app.use(bodyParser.json());
app.listen(3001, () => console.log('Servidor na porta 3001.'));

app.get('/', (request, response) => {
  response.sendFile(`_${__dirname}/index.html`);
});
```

Para obter uma página HTML via servidor, fazemos uma request do tipo GET e respondemos com o envio de um arquivo.

```
const express = require('express');
const bodyParser = require('body-parser');

const app = express();

app.use(bodyParser.json());
app.listen(3001, () => console.log('Servidor na porta 3001.'));

app.get('/', (request, response) => {
  response.sendFile(`_${__dirname}/index.html`);
});
```

Neste caso, utilizamos a função `sendFile` do objeto `response` para o envio de arquivos que podem ser renderizados no navegador.

```
const express = require('express');
const bodyParser = require('body-parser');

const app = express();

app.use(bodyParser.json());
app.listen(3001, () => console.log('Servidor na porta 3001.'));

app.get('/', (request, response) => {
  response.sendFile(`_${__dirname}/index.html`);
});
```

A variável `__dirname` é uma variável especial exposta pelo `express` e aponta para o diretório raiz onde o servidor está rodando.

```
const express = require('express');  
const bodyParser = require('body-parser');  
  
const app = express();  
  
app.use(bodyParser.json());  
app.listen(3001, () => console.log('Servidor na porta 3001.'));  
  
app.get('/', (request, response) => {  
  response.sendFile(`_${__dirname}/index.html`);  
});
```

Note que no caso acima enviamos o arquivo `index.html`, mas poderíamos ter enviado qualquer arquivo desde que seu nome obedeça as regras.


```
const express = require('express');
const bodyParser = require('body-parser');

const app = express();

app.use(bodyParser.json());
app.listen(3001, () => console.log('Servidor na porta 3001.'));

app.get('/', (request, response) => {
  response.sendFile(`_${__dirname}/index.html`);
});
```

Via de regra, quando fazemos uma requisição do tipo GET na URL raiz, enviamos o arquivo `index.html`. Isso se deve aos primórdios da Web quando servidores procuravam explicitamente por este arquivo.