



Fundamentos de Programação Web

UNIDADE 04

Desenvolvimento Front-End

Olá!

Para finalizar os vários aspectos já trabalhados sobre o desenvolvimento **front-end**, o lado **cliente** da aplicação web, iremos acompanhar a construção de um pequeno site web, acessado localmente, a partir dos arquivos em uma pasta da nossa máquina de trabalho.

Para iniciar este trabalho, precisamos garantir a **portabilidade do conjunto de arquivos** de um **site**. A **portabilidade** é a habilidade de **migrar**, de um equipamento para outro, o conjunto de arquivos de um site, a fim de garantir que as referências aos recursos internos continuarão funcionais.

Por essa razão, devemos utilizar **caminhos relativos** (ex: `img/foto.png`), ou invés de **caminhos absolutos** (ex: `c://apache/htdocs/img/foto.png`).

Para exemplificar a **portabilidade para referências**, observe a estrutura de **pastas e arquivos** a seguir:

```
├── css
│   └── style.css
├── img
│   ├── muffin.png
│   └── CupFlavor1.png
├── 1 script
│   └── script.js
├── index.html
└── news.html
```

De acordo com esta estrutura, quando indicamos no código **HTML** as referências para outros recursos, como arquivos ou imagens do site, devemos utilizar seu **caminho relativo**. Assim, no código do arquivo local **index.html**, as referências aos recursos internos devem ser escritas como:

1. Imagens locais na pasta **img**: ``.
2. Script locais na pasta **script**: `<script src="script/script.js"></script>`
3. Estilo local na pasta **css**: `<link rel="stylesheet" href="css/style.css">`
4. Outros arquivos **HTML** locais: `News`.

Dessa forma, nosso site poderá ser copiado para **diferentes pastas**, em **diferentes equipamentos**, mantendo a exibição e funcionalidade dos seus recursos! Ou seja, nosso site será **portável**. 😊

Na sequência, vamos apresentar alguns recursos que devemos explorar neste pequeno site de front-end, no qual o conteúdo e layout são interativos e trabalhados com Javascript e CSS:

1. Criar um menu principal, para acesso às demais páginas do site.
2. Criar uma página inicial do site.
3. Criar um formulário para obtenção de dados do usuário.
4. Criar página para processar os dados obtidos de um formulário, enviados por GET.

Vejam, nas próximas seções, como podemos construir esse exemplo de site, em preparação para a Atividade Somativa 1.

| Construção: *Site Front-End*

Vamos ver alguns exemplos de como criar uma estrutura e estilo para manter em todas as páginas de um site. Assim, com esse objetivo, iremos deixar nossa página HTML flexível, definindo uma estrutura o mais simples possível, que organize o documento em seções. Observe o exemplo da Figura a seguir.

Exemplo menu em HTML

```
1  <!DOCTYPE html>
2  <html>
3
4  <head>
5      <meta charset="UTF-8">
6      <title>Front-end</title>
7      <link rel="stylesheet" href="css/layout.css" type="text/css">
8  </head>
9
10 <body>
11
12     <!-- Este elemento que define o Menu -->
13     <div class="sidebar">
14         <a class="active" href="index.html">Home</a>
15         <a href="news.html">News</a>
16         <a href="contact.html">Contact</a>
17         <a href="about.html">About</a>
18     </div>
19
```

Observe no código HTML:

Temos 2 seções principais, delimitadas pelo par de *tags* `<div></div>`. Apenas definimos classes que nos auxiliarão a criar um menu dinâmico, com as classes CSS:

- `class="sidebar"` (para criar uma barra lateral para menu)
- `class="content"` (para manter o conteúdo da página)

Observem que a página `"index.html"` é a página atual, logo ela indica isso usando a classe CSS: `class="active"`

Exemplo menu em CSS

```
5     background-color: #f1f1f1;
6     position: fixed;
7     height: 100%;
8     overflow: auto;
9 }
10
11 .sidebar a {
12     display: block;
13     color: black;
14     padding: 16px;
15     text-decoration: none;
16 }
17 .sidebar a.active {
18     background-color: rgb(81, 92, 104);
19     color: white;
20 }
21 .sidebar a:hover:not(.active) {
22     background-color: rgb(158, 183, 209);
23     color: white;
24 }
```

Observe no código CSS:

`.sidebar` é um elemento de largura fixa em `width: 200px`, com fundo cinza, `background-color: #f1f1f1`, e que ocupa o comprimento completo da janela em que está.

Os seletores de link `a.active` e `a:hover:not(.active)`, contidos na classe `.sidebar`, são formatados com cores diferentes, para indicar a possibilidade de ação por parte do usuário, simulando botões prontos para acionar uma funcionalidade. No caso, ativam o link selecionado.

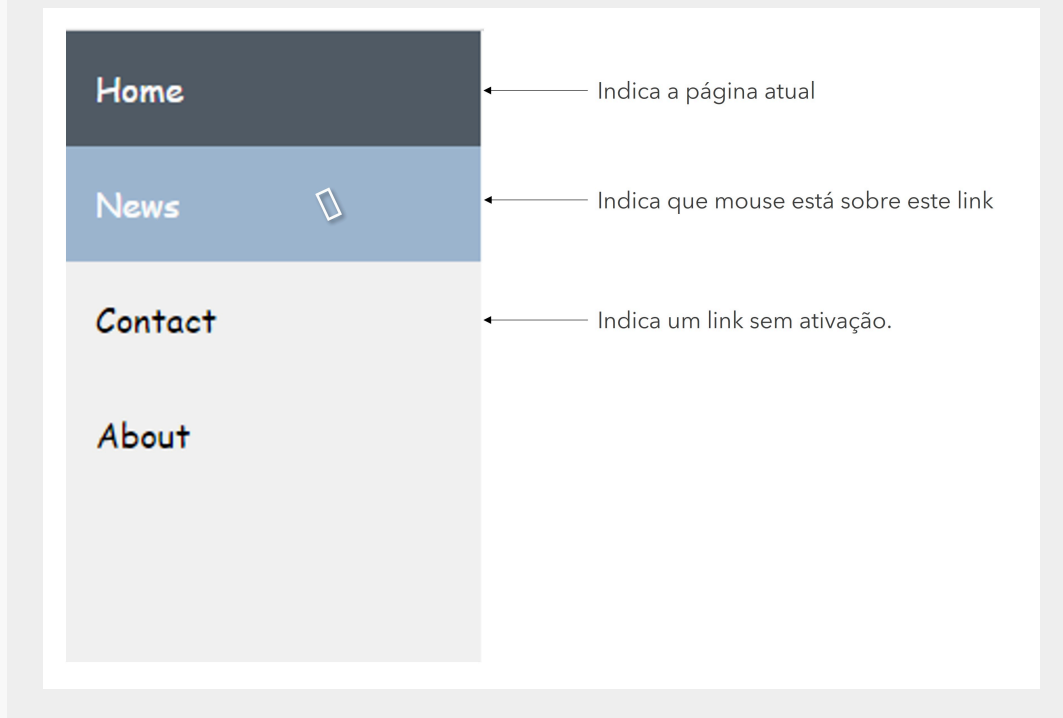


Figura 1: Exemplo de Menu para um site (front-end). Fonte: A autora (2023).



SAIBA MAIS

Menu

Experimente criar uma estrutura de Menu como as sugestões em **CSS Barra de Navegação**, do site W3Schools.

A referência **CSS em Cascata** explica como entender interpretação de mais de um estilo em um seletor.

A referência **CSS Layout Grid** explica como criar uma estrutura para organizar conteúdo em HTML

- **CSS Barra de Navegação:**
https://www.w3schools.com/css/css_navbar.asp
- **Efeito Cascata, Especificidade e Herança no CSS:** explica como entender o a hierarquia de regras CSS, e como elas modificam a apresentação visual dos elementos HTML.
<https://medium.com/jaguaribetech/efeito-cascata-no-css-c55bda0a2ed4>
- **CSS Layout Grid:** módulo que Layout oferece um sistema de layout baseado em grid, com linhas e colunas, facilitando o design de páginas web.
https://www.w3schools.com/css/css_grid.asp

Como incluir arquivos HTML em outros arquivos HTML?

Essa dúvida surge quando temos padrões para **cabeçalho** e **rodapé** comuns em um site web. Contudo, um documento escrito apenas em **HTML** não tem um “*include file*” direto, como vemos em várias linguagens de programação. De fato, sabemos que **HTML** não é linguagem de programação.

A dificuldade é que **HTML** é uma estrutura bem definida (`<html>
<head> . . . </head><body> . . . </body></html>`), e outro arquivo HTML provavelmente também terá essa mesma estrutura completa. Isso que pode causar erros de interpretação e exibição da página pelo navegador. Logo, o padrão **HTML** não oferece uma marcação (*tag*) para “*include file*”. Existem alternativas para realizar essa inclusão, utilizando linguagens de programação.

Na nossa disciplina, mostraremos como realizar isso com **PHP (Unidade: PHP)**, onde criaremos páginas **HTML dinâmicas**, oferecendo uma alternativa para “*include file*” via “*server-side*” (ou, lado servidor).

NOTA: Para este exercício, **repetiremos** o trecho **HTML de menu** em todas as páginas do nosso *site front-end*. Nas próximas unidades, veremos alternativas com PHP. Abaixo, deixamos algumas referências de como realizar o *include* com Javascript.

IMPORTANTE: as referências na **língua inglesa** podem ser **traduzidas** por completo, na própria janela do navegador. No Google Chrome, temos essa funcionalidade. Veja em “**Mudar idiomas do Chrome e traduzir páginas da Web**”: <https://support.google.com/chrome/answer/173424?hl=pt-BR&co=GENIE.Platform%3DDesktop>

- **How TO - Include HTML:** mostra como incluir HTML com JavaScript e CSS. https://www.w3schools.com/HOWTO/howto_html_include.asp
- **Include another HTML file in a HTML file:** mostra como incluir HTML com Javascript. <https://stackoverflow.com/questions/8988855/include-another-html-file-in-a-html-file>

Exemplo Conteúdo para página inicial - HTML

```
1  <!-- Este elemento que define o Menu -->
2  <div class="sidebar">
3      <a class="active" href="index.html">Home</a>
4      <a href="news.html">News</a>
5      <a href="contact.html">Contact</a>
6      <a href="about.html">About</a>
7  </div>
8
9  <!-- Este elemento que define o conteúdo da página principal -->
10 <div class="content">
11     <h2 class="titleTop">Cupcake Bakery Shop</h2>
12     <h3 class="titleBottom">Menu Tradicional</h3>
13     <div class="first">
14         <div class="menu">
15             
16             <input class="modalButton" type="button" value="Saiba mais..
17         </div>
18         <div class="menu">
19             
```

Observe no código HTML:

O código dá sequência ao HTML do Menu, sendo o conteúdo da página principal definido em: `<div class="content"> ... </div>`

Dentro de `.first` e `.second`, temos seções para organizar imagens de uma loja de cupcakes:

`</>`

```
1  <div class="first">
2      <div class="menu">
3          
4          <input class="modalButton" type="button" value="Saiba mais..."
5      </div>
```

Quando clicamos o botão de cada imagem, acionamos a função Javascript `openModal()` que apresenta a `<div id="knowMore" class="modal">`

A função Javascript, juntamente a definição do evento de click aos elementos, está no arquivo `<script src="script/script.js"></script>`, carregado ao final do HTML.

Exemplo Conteúdo para página inicial - CSS

```
1  .content {
2      padding: 1px 16px;
3      position: inherit;
4      margin-left: 200px;
5      width: 700px;
6      background: #fff;
7      display: grid;
8      overflow: auto;
9      height: 100%;
10 }
11 .content .titleTop {
12     position: relative;
13     max-height: 40px;
14     padding: 10px;
15     background-color: rgb(158, 183, 209);
16     border-radius: 10px;
17     margin-left: 10px;
18     text-align: center;
19     text-transform: uppercase;
```

Observe no código CSS:

`.sidebar` é um elemento de largura fixa em `width: 200px`, com fundo cinza, `background-color: #f1f1f1`, e que ocupa o comprimento completo da janela em que está.

As classes `.titleTop` e `.titleTop`, contidas na classe `.content`, destacam os títulos da página.

Da mesma forma, as classes `.first` e `.second`, também contidas na classe `.content`, separam 2 seções de bloco página (`<div>`), para organizar conteúdo.

Exemplo Conteúdo para página inicial - JS


```

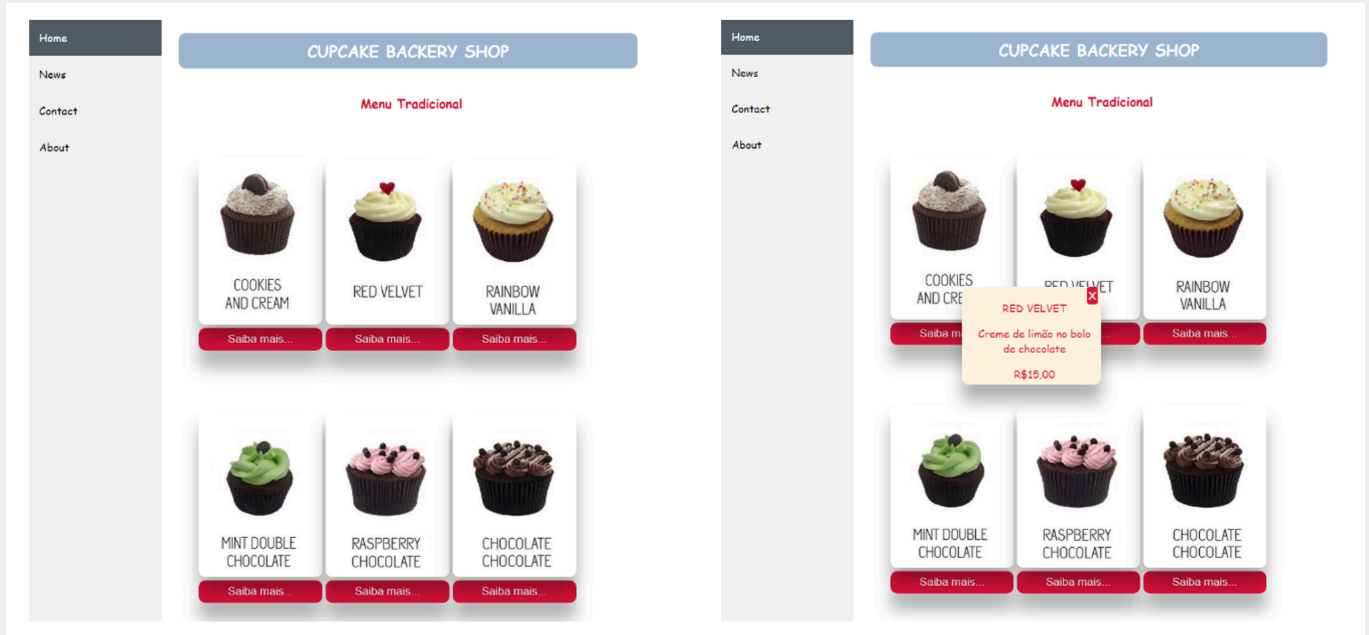
1  function openModal(cupcake) {
2      var txt1 = document.getElementById("txt1");
3      var txt2 = document.getElementById("txt2");
4      switch (cupcake) {
5          case "CupFlavor1":
6              txt1.innerHTML = "Cookies And Cream";
7              txt2.innerHTML =
8                  "Creme de Negresco no bolo de chocolate.";
9              txt3.innerHTML = "R$15,00";
10             break;
11             /* .. continuar igual para as 6 imagens */
12         default:
13             txt1.innerHTML = "Cookies And Cream";
14             txt2.innerHTML = "Cookies And Cream";txt1.innerHTML = "Delicioso c
15             txt3.innerHTML = "R$15,00";
16         }
17         document.getElementById("knowMore").style.display = "inline-block";
18     }
19
20     function closeModal() {
21         document.getElementById("knowMore").style.display = 'none';
22     }

```

Observe no código javascript:

Apenas apresenta o Modal (caixa de diálogo ou janela pop-up exibida na página atual), com informações sobre a imagem em questão (ex.: `src="img/CupFlavor5.png"`, etc.).

No Javascript, também está a função para fechar o Modal.



Página **Index.html** **COM** e **SEM** modal aberto.

Figura 2: Exemplo de página inicial para um site (front-end). Fonte: A autora (2023).



SAIBA MAIS

Página inicial

Página **principal**, página **inicial**, página de entrada (**home page** ou **homepage** em inglês) é a página inicial de um site da internet. Apresenta o site, ou seu conteúdo.

Quando usamos um **servidor HTTP** (que retorna arquivos HTML), podemos **omitir** o nome da página inicial quando denominamos esse arquivo como **index.html**. Nesse caso, o **servidor HTTP** identifica o arquivo **index.html** e retorna seu conteúdo.

Ex. URL para uma **homepage** (página inicial)

- Na barra de endereços do navegador, escrevemos:
http://nomesite.com
- No exemplo acima, **omitimos** o nome do arquivo inicial **index.html**

Importante:

Quando **não** usamos um **servidor HTTP**, como é o caso do site exemplo em uma **pasta local** da nossa máquina, precisamos indicar o **index.html** para iniciar a navegação pelo conteúdo do site:

Na barra de endereços do navegador, escrevemos: `file:///C:/Teste-Front/index.html`

Formulário

Vamos solicitar interação com o nosso usuário através de um formulário. Veja que no exemplo, houve uma preocupação em **validar os campos do formulário**, inclusive usando uma “**máscara**” para o **telefone**, com Expressão Regular (**Regex**).

Uma das principais atribuições da **interface com o usuário**, que é o foco no **front-end**, é garantir que os **dados esperados pelo lado servidor** venham no **formato correto**. Por isso, a importância da **validação dos dados** em um formulário.

Conteúdo para página com formulário - HTML e JS

```
1  <!DOCTYPE html>
2  <html lang="pt-BR">
3
4  <head>
5      <meta charset="UTF-8">
6      <title>Front-end</title>
7      <link rel="stylesheet" href="css/layout.css" type="text/css">
8  </head>
9
10 <body>
11
12     <div class="sidebar">
13         <a class="active" href="index.html">Home</a>
14         <a href="news.html">News</a>
15         <a href="contact.html">Contact</a>
16         <a href="about.html">About</a>
17     </div>
18
19     <div class="content">
```

Observe no código HTML:

O código já inclui HTML do Menu, bem como referência ao arquivo CSS externo, sendo o conteúdo da essencialmente do formulário definido em: `<div class="content"> ... </div>`

Temos a criação do formulário, delimitada por `<form method="POST" action="contact.html"> ... </form>`

O `method="POST"` significa que os dados do formulário serão enviados via protocolo HTTP, e não estarão visíveis na URL como no método GET. Os dados serão enviados para o arquivo definido no atributo `"action"`, ou seja, para o próprio arquivo do formulário de contato, o `"contact.html"`. Como ainda não aprendemos a enviar os dados ao servidor, o único indício de que os dados foram realmente enviados, é que a página será recarregada com os campos que foram preenchidos vazios. Na continuidade da disciplina iremos alterar este `"action"` para apontar para um script do lado do servidor, para eventual comunicação com o banco de dados, aí manipularemos o que chamamos de **back-end da aplicação**.

Esses dados apenas serão enviados após o acionamento do botão `<input type=submit>`.

Os campos do formulário estão organizados em uma tabela. Campos do tipo `"texto"` iniciam com a tag, como no exemplo `<input type=text name="Nome" pattern="[A-Za-záâãäåæëèéíóôõúç\s]{8,50}" title="Nome com 8 a 50 letras" required>`, onde os atributos:

- `name="Nome"` identifica unicamente o campo.
- `pattern="[A-Za-záâãäåæëèéíóôõúç\s]{8,50}"` define a expressão regular que controla quais valores são válidos para o campo – no caso: letras maiúsculas e minúsculas, letras acentuadas, espaço em branco (`\s`), e tamanho entre 8 e 50 caracteres.
- `required` indica que o campo tem preenchimento obrigatório.

Perceba que nesta página, o Javascript foi definido dentro da tag `<script>` e não em um arquivo externo como na página principal.

No JavaScript a function `mascaraTelefone(event)` valida o campo de telefone, com expressão regular, para seguir os padrões:

- `(xx) xxxx-xxxx` ou
- `(xx) xxxxx-xxxx`

A função é “anexada” ao evento de `"keydown"` do campo `"celular"` no código:

```
document.getElementById('celular').addEventListener('keydown', mascaraTelefone);
```



SAIBA MAIS

Regex

As expressões regulares (Regex) são um poderoso recurso para validação de cadeias de caracteres. O **JavaScript** fornece várias funções que utilizam expressões regulares para validar dados. Verifiquem as dicas a seguir sobre como criar e utilizar as Regex.

HTML <input pattern>

O atributo **pattern** especifica uma expressão regular com a qual o valor do elemento **<input>** é verificado no envio do formulário.

Observação: o atributo pattern funciona com os seguintes tipos de entrada <input>:

- `<input type="text">` (texto)
- `<input type="date">` (data)
- `<input type="search">` (pesquisa)
- `<input type="url">`
- `<input type="tel">`
- `<input type="email">`
- `<input type="password">` (senha).

Dicas: Use o atributo global **title** para descrever o padrão de ajuda para o usuário.

- **HTML <input> pattern Attribute.** Exemplos de utilização do atributo pattern. https://www.w3schools.com/tags/att_input_pattern.asp
- **regex101: build, test, and debug regex.** Orienta a criação e teste de expressões regulares, detalhando cada elemento de validação. <https://regex101.com/>
- **Expressões regulares.** Tutorial sobre expressões regulares. https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Guide/Regular_Expressions
- **Expressões Regulares JavaScript.** https://www.w3schools.com/js/js_regexp.asp
- **JavaScript RegExp Reference.** https://www.w3schools.com/jsref/jsref_obj_regexp.asp

```
1  .contact-form {
2      text-align: left;
3  }
4
5  .form-submit {
6      display: inline-block;
7      background: rgb(81, 92, 104);
8      color: #fff;
9      border-bottom: none;
10     width: auto;
11     padding: 15px 39px;
12     border-radius: 5px;
13     -moz-border-radius: 5px;
14     -webkit-border-radius: 5px;
15     -o-border-radius: 5px;
16     -ms-border-radius: 5px;
17     margin-top: 25px;
18     margin-left: 25px;
19     margin-right: 25px;
```

Observe no código CSS:

Os campos do formulário têm formatação de estilo no CSS. Dessa forma, basta trocar os estilo no CSS para alterar completamente a aparência do formulário, sem necessidade de manutenção no HTML.

Conteúdo para página com formulário - JS

```

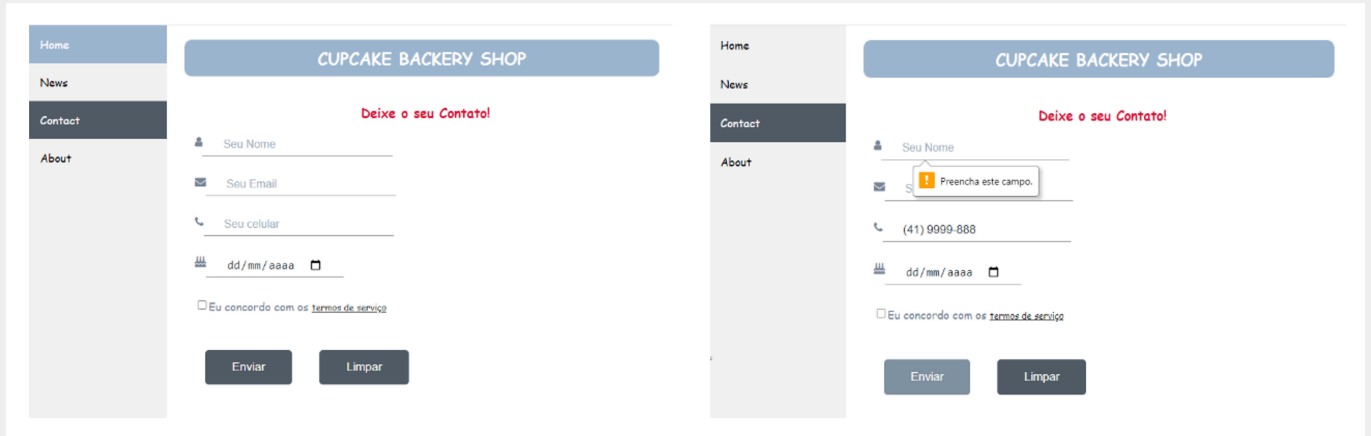
1 function mascaraTelefone(event) {
2     let tecla = event.key;
3     // Regex:
4     // g = não termina verificação enquanto não houver combinação para todos
5     // \D+ = troca qualquer caractere que não seja um dígito por caractere va
6     let telefone = event.target.value.replace(/\D+/g, "");
7
8     // Regex: i = case insensitive
9     // Se Tecla é número, concatena com telefone
10    if (/^[0-9]$/i.test(tecla)) {
11        telefone = telefone + tecla;
12        let tamanho = telefone.length;
13
14        if (tamanho >= 12) { // Se telefone com 12 ou mais caracteres, não faz
15            return false;
16        }
17
18        if (tamanho > 10) {
19            telefone = telefone.replace(/^(\\d\\d)(\\d{5})(\\d{4}).*/ , "($1) $2-$3")
20        } else if (tamanho > 5) {
21            telefone = telefone.replace(/^(\\d\\d)(\\d{4})(\\d{0,4}).*/ , "($1) $2-$3
22        } else if (tamanho > 2) {
23            telefone = telefone.replace(/^(\\d\\d)(\\d{0,5})/ , "($1) $2");
24        } else {
25            telefone = telefone.replace(/^(\\d*)/ , "($1)");
26        }
27
28        event.target.value = telefone;
29    }
30
31    if (!["Backspace", "Delete", "Tab"].includes(tecla)) {
32        return false;
33    }
34 }

```

Observe no código javascript:

A function mascaraTelefone(event) valida o campo de telefone, com expressão regular, para seguir um padrão:

(xx) xxxx-xxxx ou (xx) xxxxx-xxxx



Página form.html com validação de campos

Figura 3: Exemplo de página de formulário para um site (front-end). Fonte: A autora (2023).

Agora que você já acompanhou vários trechos dos códigos deste pequeno site exemplo, para execução exclusiva local, verifique na videoaula “Para consolidar: Desenvolvimento Front-End” como podemos depurar os códigos Javascript no navegador para entender seu funcionamento. Também mostramos como acompanhar a aplicação das regras de estilo (CSS), nos diferentes elementos HTML utilizados.

Agora que estudamos um exemplo de como organizar o que já praticamos nas Unidades: HTML, CSS e Javascript, vamos ver como esses vários elementos atuam em um conjunto maior para prover uma aplicação web apenas de front-end.

Veja, em funcionamento, o exemplo da Unidade, apresentado neste vídeo.

Para consolidar: Desenvolvimento front-end



| Atividade Formativa

Desenvolva um site de acordo com as orientações apresentadas na seção anterior. Entregue uma estrutura de documentos web que contemple os seguintes itens:

1. Menu principal, para acesso às demais páginas do site.
2. Página inicial do site.
3. Formulário para obtenção de dados do usuário.
4. Processamento dos dados obtidos de um formulário, enviados por GET.

Ao finalizar esta Atividade Formativa, você já terá uma estrutura de site preparada para realizar a Atividade Somativa 1.

Bom trabalho!

| Conclusão

Olá a todos!

Com esta unidade, mobilizamos os conhecimentos trabalhados nas unidades anteriores, que tratam da parte **visual** e **interativa** (troca informações com o usuário) da nossa aplicação web, denominada **front-end**.

Dessa forma, fizemos nossa **Avaliação Somativa 1**, para verificar o quanto progredimos nesses primeiros Temas de Estudo.

Na próxima unidade, começaremos a praticar programação **back-end** com **PHP**, uma linguagem de programação feita para ser executada em um servidor **HTTP**, que processa páginas **HTML dinâmicas** para envio ao navegador.

Até lá!

| Referências Bibliográficas

MILETTO, E. M.; BERTAGNOLLI, S. C. **Desenvolvimento de software II: introdução ao desenvolvimento web com HTML, CSS, Java Script e PHP**. Porto Alegre: Bookman, 2014.

TERUEL, E. C. **HTML 5: guia prático**. 2. ed. Porto Alegre: Bookman, 2014.

ALVES, W. P. **Desenvolvimento e design de sites**. São Paulo: Erica, 2014.



© PUCPR - Todos os direitos reservados.