Programação Web Front-End
Aula 4 - JavaScript

Profa. Rosangela de Fátima Pereira Marquesone
romarquesone@utfpr.edu.br

Proposta: apresentar as funcionalidades da API Web Storage, via JavaScript.

Objetivos: espera-se que após essa aula, você tenha habilidade para compreender os seguintes tópicos:

1. Aprender o que é a API Web Storage
2. Aprender as funcionalidades do localStorage
3. Aprender a utilizar o localStorage em um cadastro de login
4. Atividade prática - Remoção de dados do armazenamento local

Dicas de aprendizado:

- Execute todos os passos com atenção, compreendendo o que está sendo realizado;
- Procure não copiar código, para ter a prática de digitar o código desenvolvido;
- Pergunte quando tiver alguma dúvida;
- Mantenha um histórico dos códigos desenvolvidos, seja no github ou em algum outro meio de armazenamento (e-mails, google drive, etc.);
- Tenha curiosidade e explore os recursos apresentados.

Tópicos anteriores:

- Compreender o que é HTML
- Compreender o que são tags HTML básicas
- Criar um arquivo .html no Visual Studio (VS) Code
- Abrir o arquivo .html em um navegador
- Visualizar o código-fonte de uma página em um navegador
- Inspecionar a página em um navegador
- Utilizar o Live Server no VS Code
- Aprender a utilizar tags semânticas
- Aprender a inserir links
- Aprender a inserir listas
- Aprender a criar uma página com seu Curriculum Vitae (CV) (atividade prática)
- Aprender a inserir figuras
- Aprender a utilizar a tag semântica <figure>
- Inserir figuras em seu Curriculum Vitae (CV) (atividade prática)
- Aprender a criar formulários
- Criar um formulário (atividade prática)
- Descobrir o que é CSS
- Aprender a sintaxe do CSS

- Aprender os tipos de seletores CSS
- Aprender as formas de inclusão de CSS
- Aprender a definir cores
- Aprender a alterar as propriedades de texto
- Aprender o conceito de modelo de caixa do CSS
- Aprender a trabalhar com a margem
- Aprender a trabalhar com a borda
- Aprender a trabalhar com o preenchimento (padding)
- Aprender a usar a propriedade display
- Aprender a utilizar a propriedade float
- Aprender a utilizar a propriedade overflow
- Estruturar páginas por meio do modelo de caixa (atividade prática)
- Aprender o conceito de flex-box
- Aprender as propriedades do elemento pai (flex container)
- Aprender as propriedades dos elementos filhos (flex items)
- Descobrir a história da linguagem JavaScript
- Compreender as formas de uso da linguagem JavaScript
- Conhecer as características da linguagem JavaScript
- Atividade prática com JavaScript
- Aprender a utilizar JavaScript com DOM HTML
- Atividade prática com JavaScript
- Aprender a utilizar o método QuerySelector() e QuerySelectorAll()
- Aprender a utilizar o método classList.add()
- Aprender a utilizar o método classList.remove()
- Criar uma lista de tarefas com HTML, CSS e JavaScript
- Atividade prática com JavaScript - Alteração da lista de tarefas

Passo 1 - Aprender o que é a API Web Storage

Vimos nas aulas anteriores algumas das funcionalidades providas pela API DOM HTML, que possibilita manipular elementos HTML e alterar o layout CSS de uma determinada página. Além dessa API, também aprenderemos nesse tutorial uma funcionalidade muito importante para a construção de uma página: a API Web Storage.

Essa API, criada a partir do HTML 5, provê um conjunto de funcionalidades que permitem realizar o armazenamento dos dados localmente, dentro do navegador de um usuário. Para isso, esse armazenamento tem algumas características importantes, tais como:

- Os objetos armazenados são sempre no formato chave-valor (*key-value*);
- As chaves e os valores são sempre strings.

Para prover essa funcionalidade, a API oferece dois mecanismos de armazenamento: sessionStorage e localStorage.

O mecanismo sessionStorage mantém uma área de armazenamento durante a sessão da página. São características desse mecanismo:

- Armazena dados apenas para uma sessão. Os dados são armazenados até que o navegador (ou guia) seja fechado.
- Os dados nunca são transferidos para o servidor.
- O limite de armazenamento é maior que um cookie (no máximo 5 MB).
- Disponível por meio do objeto Window.sessionStorage.

localStorage faz a mesma coisa, porém sem data de expiração dos dados. Além disso, possui as seguintes características:

- Armazena dados sem data de validade e é limpo apenas por meio de JavaScript ou limpando o cache do navegador/dados armazenados localmente.
- Disponível por meio do objeto Window.localStorage.

Passo 2 - Aprender as funcionalidades do localStorage

O mecanismo localStorage, conforme apresentado no passo anterior, permite armazenar dados de forma persistente no navegador do usuário (não envia os dados ao servidor).

Veja exemplos de algumas de suas aplicações:

- Persistência da atividade anterior do site (exemplo: armazenar o conteúdo de um carrinho de compras de uma sessão anterior).
- Personalizar preferências do site (exemplo: mostrar a escolha do usuário no esquema de cores ou tamanho da fonte).
- Salvar dados para que o download de um site seja mais rápido ou utilizável sem uma conexão de rede.
- Salvar documentos localmente para uso offline.

O objeto Window.localStorage oferece um conjunto de métodos para fazer o gerenciamento do documento, tais como:

- setItem()
- getItem()
- removeItem()
- clear()

Vamos ver os detalhes desses métodos.

setItem()

Esse método é usado para adicionar uma chave ao storage, ou atualizar o valor, caso a chave já exista.

- Sintaxe: setItem(‘chave’, ‘valor’)
    - chave: string que representa a chave do par chave-valor. É usada para acessar o valor posteriormente.
    - valor: valor a ser armazenado. Pode ser de qualquer tipo de dado, mas será convertido em uma string antes de ser armazenado no Local Storage.

Exemplo:

```
localStorage.setItem('nome', 'Maria');
localStorage.setItem('idade', '18');
```

getItem()

Método usado para obter o valor associado a uma chave.

- Sintaxe: getItem(‘chave’)
    - chave: string que representa a chave do par chave-valor que se deseja acessar.

Exemplos:

```
let nome = localStorage.getItem('nome');

console.log(nome); // Saída: 'Maria'
document.getElementById("dados").innerHTML = localStorage.getItem('nome');
```

removeItem()

Método usado para remover um item específico do armazenamento local do navegador.

- Sintaxe: removeItem(‘chave’)
    - chave: string que representa a chave do par chave-valor que se deseja remover.

Exemplos:

```
localStorage.removeItem('nome');

if (localStorage.getItem('nome') !== null) {
   localStorage.removeItem('nome');
}
```

clear()

Método usado para remover todos os itens armazenados no armazenamento local do navegador.

- Sintaxe: clear();

Exemplo:

```
localStorage.clear();
```

Passo 3 - Aprender a utilizar o localStorage em um cadastro de login

Para praticar a utilização do localStorage, utilizaremos o exemplo a seguir, que cria uma página para realizar o cadastro de um usuário e, após isso, permite realizar o login desse.

PRATICANDO: crie uma pasta chamada aula4-js e abra essa pasta no Visual Studio Code. Dentro dessa pasta, crie a seguinte estrutura de arquivos:

- aula-js
    - index.html
    - style.css
    - script.js

Inicialmente, crie o seguinte código para o arquivo index.html:

index.html

```
<!DOCTYPE html>
<html lang="pt-br">
 <head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Cadastro</title>
  <link rel="stylesheet" type="text/css" href="style.css" />
 </head>
 <body>
  <div class="login">
    <form name='form-login'>

      <h1>CADASTRO</h1>
      <label for="name">Email</label>
      <input type="email" id="name" placeholder="Email" required>

      <label for="pw">Senha</label>
      <input type="password" id= "pw" placeholder="Senha" required>

      <ul class="helper-text">
        <li class="length">Deve conter até 8 caracteres.</li>
        <li class="lowercase">Deve conter ao menos uma letra minuscula.</li>
        <li class="uppercase">Deve conter ao menos uma letra maiuscula.</li>
      </ul>
      <input id="rgstr_btn" type="submit" value="Cadastrar" onclick="store()">

    </form>
  </div>

  <div class="login">
    <form name='form-login'>

      <h1>LOGIN</h1>
```

```
        <label for="userName">Email</label>
        <input type="email" id="userName" placeholder="Email" required>

        <label for="userPw">Senha</label>
        <input type="password" id= "userPw" placeholder="Senha" required>

        <input id= "login_btn" type="submit" value="Login" onclick="check()">

    </form>
  </div>
 </body>
 <script src="script.js"></script>
</html>
```

O resultado deve ser similar à figura a seguir:

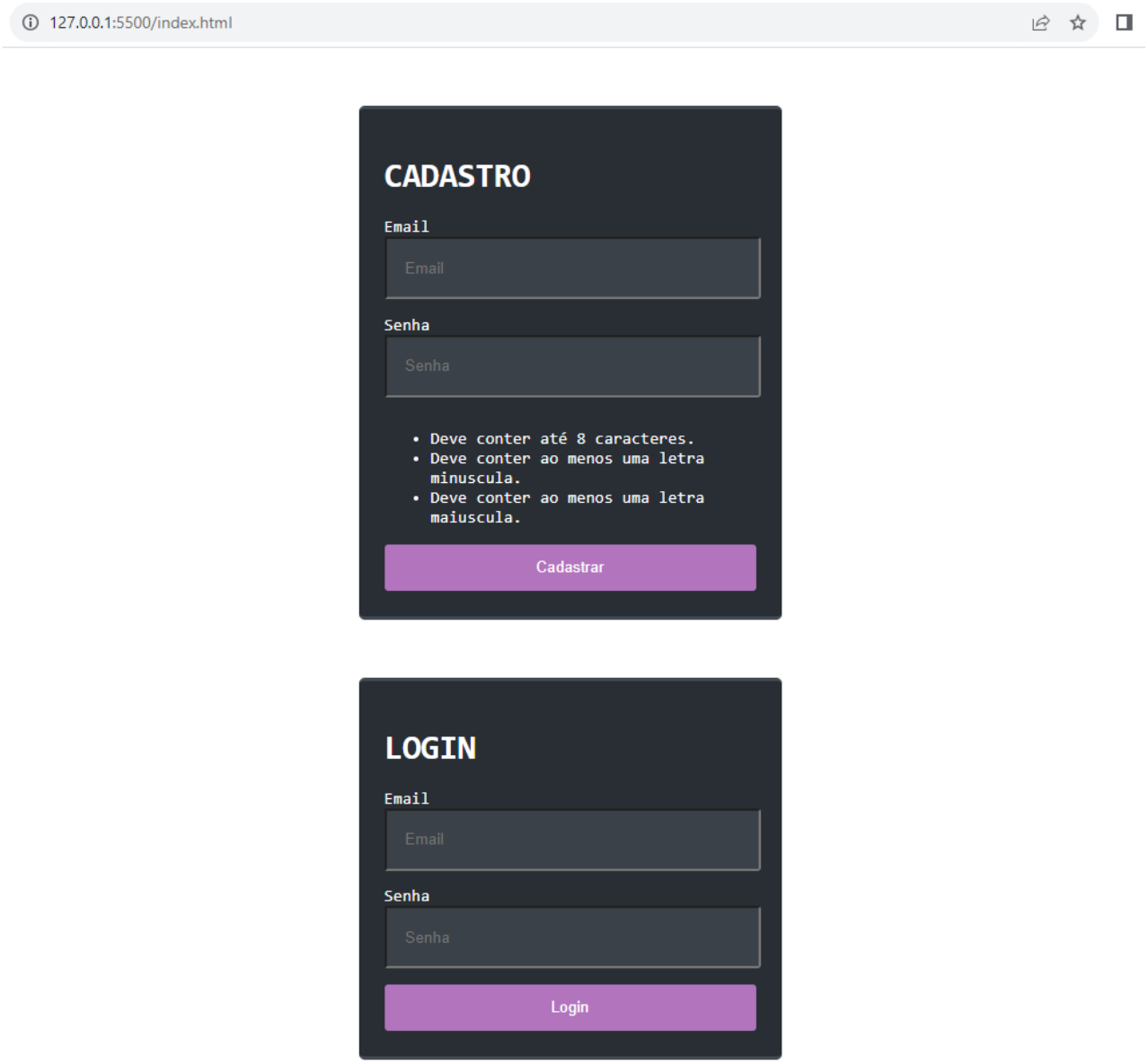
Para aperfeiçoar o estilo da página, adicionaremos o código CSS a seguir, no arquivo style.css.

style.css

```
body {
  background: white;
  color: white;
  font-family: monospace;
  font-size: 1.1em;
  margin: 0;
}

after {
  clear: both;
}

.login {
  margin: 50px auto;
  width: 320px;
}
```

```
.login form {
  margin: auto;
  padding: 22px 22px 22px 22px;
  width: 100%;
  border-radius: 5px;
  background: #282e33;
  border-top: 3px solid #434a52;
  border-bottom: 3px solid #434a52;
}

.login form span {
  background-color: #363b41;
  border-radius: 3px 0px 0px 3px;
  border-right: 3px solid #434a52;
  color: #606468;
  display: block;
  float: left;
  line-height: 50px;
  text-align: center;
  width: 50px;
  height: 50px;
}

.login form input[type="email"] {
  background-color: #3b4148;
  border-radius: 0px 3px 3px 0px;
  color: #a9a9a9;
  margin-bottom: 1em;
  padding: 0 16px;
  width: 90%;
  height: 50px;
}

.login form input[type="password"] {
  background-color: #3b4148;
  border-radius: 0px 3px 3px 0px;
  color: #a9a9a9;
  margin-bottom: 1em;
  padding: 0 16px;
  width: 90%;
  height: 50px;
}

.login form input[type="submit"] {
  background: #b174bd;
  border: 0;
  width: 100%;
  height: 40px;
  border-radius: 3px;
  color: white;
  cursor: pointer;
}
```

Após isso, ao abrir a página no navegador, via Live Server, o resultado deverá ser algo similar à figura a seguir:



Por fim, utilizaremos o JavaScript para realizar a validação e o armazenamento local dos dados do cadastro, para, posteriormente, realizar o login. Para isso, insira o seguinte código no arquivo script.js:

script.js

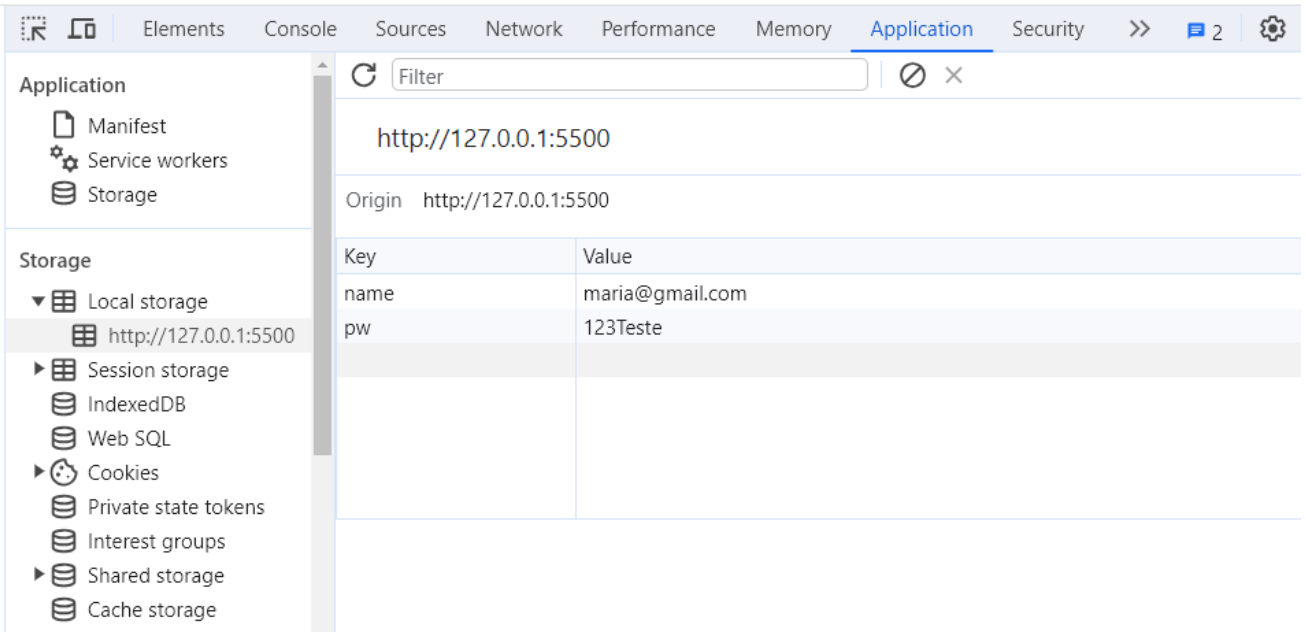
```
function store(){
   var name = document.getElementById('name');
   var pw = document.getElementById('pw');
   var lowerCaseLetters = /[a-z]/g;
   var upperCaseLetters = /[A-Z]/g;
   var numbers = /[0-9]/g;

   if(name.value.length == 0){
      alert('Informe um email');

   }else if(pw.value.length == 0){
      alert('Informe uma senha');

   }else if(name.value.length == 0 && pw.value.length == 0){
      alert('Informe um e-mail e uma senha');
```

```
    }else if(pw.value.length > 8){
        alert('Máximo de 8 caracteres');

    }else if(!pw.value.match(numbers)){
        alert('Deve conter 1 numero');

    }else if(!pw.value.match(upperCaseLetters)){
        alert('Deve conter uma letra maíuscula');

    }else if(!pw.value.match(lowerCaseLetters)){
        alert('Deve conter uma letra minúscula');

    }else{
        localStorage.setItem('name', name.value);
        localStorage.setItem('pw', pw.value);
        alert('Sua conta foi criada');
    }
}

function check(){
    var storedName = localStorage.getItem('name');
    var storedPw = localStorage.getItem('pw');

    var userName = document.getElementById('userName');
    var userPw = document.getElementById('userPw');

    if(userName.value == storedName && userPw.value == storedPw){
        alert('Login realizado.');
    }else{
        alert('Erro ao fazer login');
    }
}
```

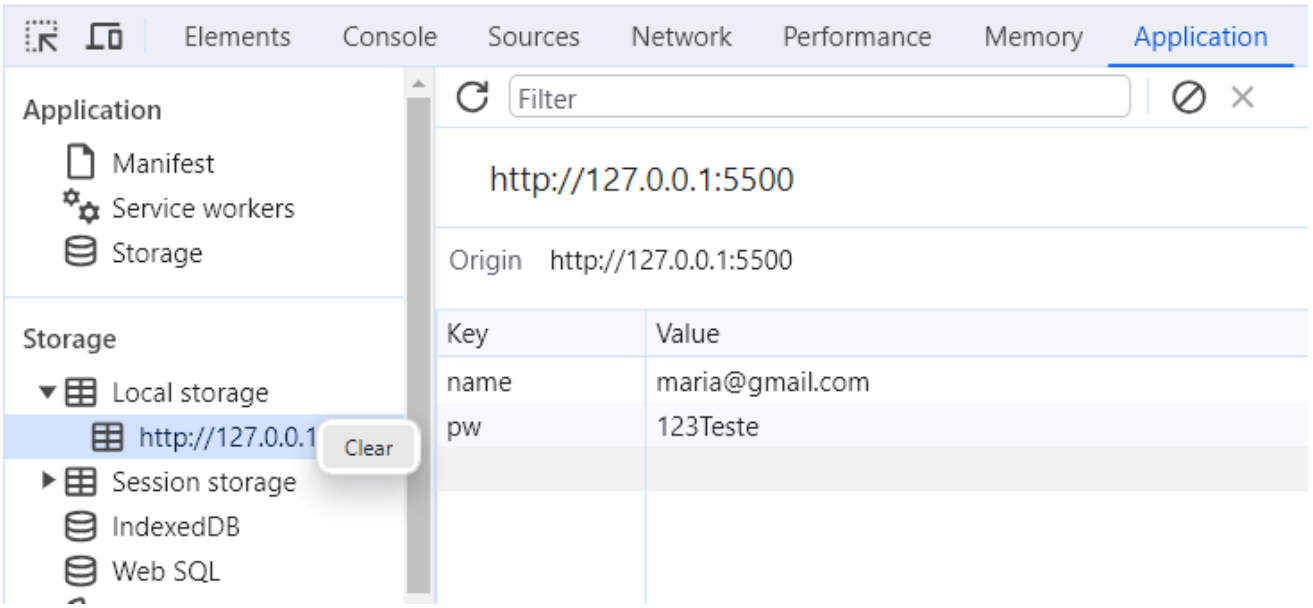
Perceba que o código possui duas funções, a função store(), que é acionada ao evento de clicar no botão de cadastro, e a função check(), acionada ao evento de clicar no botão de login.

Na função store(), após realizar a validação dos dados nos campos de cadastro, o localStorage é utilizado para realizar o armazenamento do nome e da senha informada, nas chaves name e pw, respectivamente. Isso é realizado a partir do método localStorage.setItem. Você pode também verificar se os dados foram armazenados localmente por meio da ferramenta inspecionar, disponível nos navegadores.

Para isso, no navegador Google Chrome, o acesso ao armazenamento local pode ser acessado na aba Application, e, dentro dela, em Local storage, conforme a figura à seguir:



A partir desta ferramenta, você pode também inserir um novo par chave/valor ou limpar todos os dados armazenados, clicando com o botão direito do mouse sobre o endereço da página e selecionando a opção clear, conforme a figura a seguir.



A função check(), por sua vez, utiliza o método localStorage.getItem(), para verificar se as chaves name e pw estão presentes no armazenamento local, obtendo os valores respectivos a cada chave.

Passo 4 - Atividade prática - Remoção de dados do armazenamento local

Para continuar a praticar o armazenamento local, realize as seguintes atividades:

- A partir do código apresentado no passo anterior, faça uma inclusão de um código que verifique no evento onclick() do cadastro de um usuário se há um usuário com o e-mail gmail.com. Caso haja, remova o e-mail e a senha do usuário armazenado localmente.

- Altere o código anterior, para que, ao carregar a página, seja verificado se há um armazenamento com algum e-mail do usuário. Caso haja, apresente um alerta com a seguinte mensagem: "olá, você já possui o seguinte e-mail já cadastrado: <e-mail>".

Considerações finais

Caso tenha chegado até aqui, você conseguiu completar o conteúdo do quarto tutorial sobre JavaScript. A partir desses recursos, você passa a compreender cada vez mais a base para o desenvolvimento de funcionalidades via JavaScript. Nas aulas seguintes veremos ainda mais funcionalidades para tornar as páginas ainda mais dinâmicas.

Bom estudo!