<LAB365>

DOM

<LAB365> SENAI

AGENDA | M1S05 - A1

- DOM
  - O que é?
  - Querys
  - Criação e manipulação de HTML
  - Alterando estilos e propriedade pelo JavaScript;

DOM

"O **DOM (Document Object Model)** é a representação estruturada de um **documento HTML** ou XML, permitindo que scripts dinâmicos **acessem e modifiquem** o conteúdo, a estrutura e o estilo de uma página web de forma programática."

Referência adaptada de: Flanagan, David. JavaScript: O Guia Definitivo. Bookman, 2012.

Principais Características do DOM:

- Estrutura em Árvore;
- Manipulação dinâmica;
- Independência de linguagem;
- Padronização.

DOM

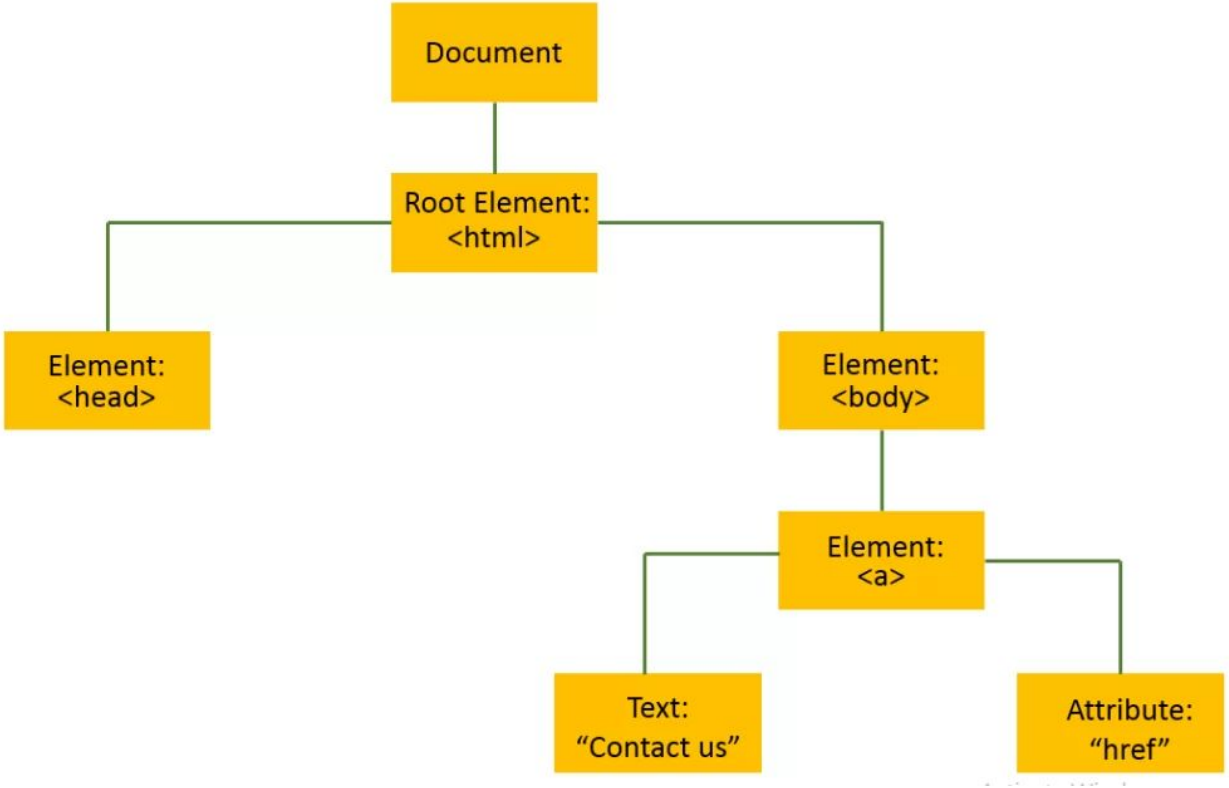
Estrutura em Árvore: O DOM organiza o documento em uma hierarquia de nós, onde o nó raiz é o documento HTML. Cada elemento, atributo e texto é representado como um nó na árvore.

Manipulação Dinâmica: Através do DOM, é possível adicionar, remover ou modificar elementos e conteúdo da página web em tempo real, sem precisar recarregar a página.

Independência de Linguagem: Embora seja comumente usado com JavaScript, o DOM é uma API que pode ser implementada em qualquer linguagem de programação.

Padronização: O DOM é padronizado pelo W3C (World Wide Web Consortium), o que garante consistência e compatibilidade entre diferentes navegadores.

DOM


Document
Root Element:
<html>
Element:
<head>
Element:
<body>
Element:
<a>
Text:
“Contact us”
Attribute:
“href”

DOM

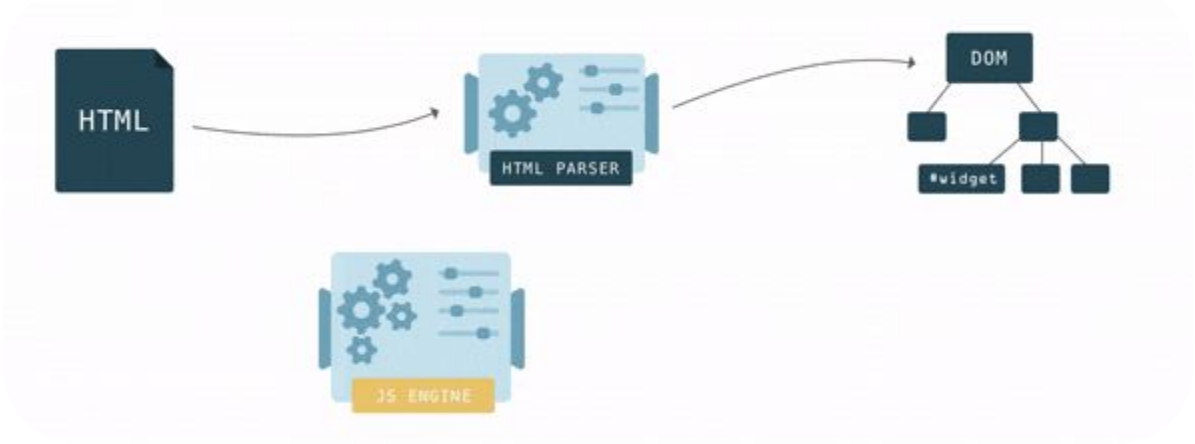
Componentes principais

- **Document**: Representa o documento HTML inteiro. É o ponto de entrada para acessar todos os elementos.
- **Element**: Representa um elemento HTML (como , <p>, <h1>, etc.).
- **Node**: É a base genérica para todos os objetos no DOM (elementos, atributos, textos, etc.).
- **Event**: Permite a interação do usuário com a página, como cliques, movimentos do mouse, teclas pressionadas, etc.

DOM: QUERYS

Querys no DOM referem-se ao uso de **métodos e técnicas** para **selecionar** elementos específicos em um documento HTML/XML, permitindo manipulá-los ou interagir com eles via JavaScript.

Essas seleções são feitas utilizando métodos como querySelector, querySelectorAll, getElementById, entre outros.



DOM: QUERYS

Principais métodos para realizar querys no DOM:

- **querySelector**: Seleciona o primeiro elemento que corresponde a um seletor CSS.
- **querySelectorAll**: Seleciona todos os elementos que correspondem a um seletor CSS, retornando uma NodeList (uma coleção de nós).
- **getElementById**: Seleciona um elemento pelo seu ID.
  Como IDs são únicos, esse método retorna apenas um elemento.

```
// Seleciona o primeiro elemento com a classe "destaque"
const elemento = document.querySelector('.destaque');

// Seleciona o primeiro parágrafo (<p>) no documento
const paragrafo = document.querySelector('p');

// Seleciona todos os elementos com a classe "item"
const itens = document.querySelectorAll('.item');

// Seleciona o elemento com o ID "cabecalho"
const cabecalho = document.getElementById('cabecalho');
```

DOM: QUERYS

Principais métodos para realizar querys no DOM:

- **getElementsByClassName**: Seleciona todos os elementos que possuem uma determinada classe, retornando uma HTMLCollection.
- **getElementsByTagName**: Seleciona todos os elementos com uma determinada tag (por exemplo, , <p>, etc.), retornando uma HTMLCollection.
- **matches**: Verifica se um elemento corresponde a um seletor CSS específico. Retorna true ou false.

```
// Seleciona todos os elementos com a classe "botao"
const botoes = document.getElementsByClassName('botao');

// Seleciona todos os parágrafos (<p>) no documento
const paragrafos = document.getElementsByTagName('p');

const elemento = document.querySelector('.destaque');
if (elemento.matches('.destaque.ativo')) {
    console.log('O elemento corresponde ao seletor.');
}
```

DOM: QUERYS

Você pode usar seletores CSS complexos para realizar querys no DOM, como:

- Seletores de atributo;
- Seletores de pseudo-classes;
- Seletores combinados.

```
const linksExternos = document.querySelectorAll('a[target="_blank"]');

const primeiroParagrafo = document.querySelector('p:first-child');

const elemento = document.querySelector('div.destaque > p');
```

MODIFICANDO HTML COM JAVASCRIPT

Agora que vimos sobre a querys em js, podemos usar a mesma para **modificar elementos** já existentes em nosso código, assim podendo manipular seu conteúdo, atributos e estilos.

Para a **alteração de conteúdos** temos os três principais propriedades:

- **textContent**: para alterar o texto;
- **innerHTML**: para alteração de HTML;
- **value**: para valores de inputs.

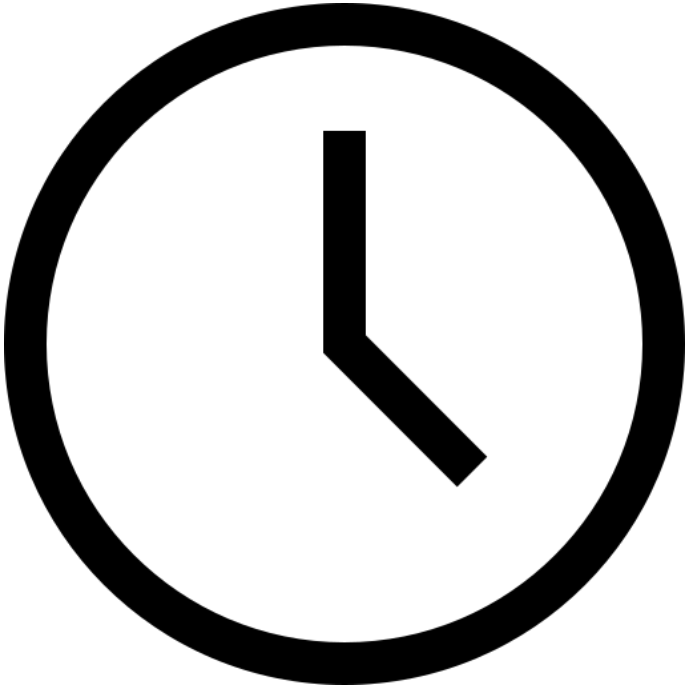
```
// Alterar texto interno
document.getElementById('titulo').textContent = 'Novo Título';

// Alterar HTML interno
document.querySelector('.container').innerHTML = '<p>Conteúdo atualizado</p>';

// Alterar valor de inputs
document.getElementById('email').value = 'usuario@exemplo.com';
```

INTERVALO!

Finalizamos o nosso primeiro período de hoje. Que tal descansar um pouco?!

Nos vemos em 20 minutos.

Início: 20:20
Retorno: 20:40



MODIFICANDO HTML COM JAVASCRIPT

Seguimos para **modificação de atributos**, neles podemos estar alterando um atributo qualquer, o nosso CSS e id.

```
// Alterar um atributo qualquer
document.getElementById('minha-imagem').setAttribute('src', 'nova-imagem.jpg');

// Alterar classe CSS
const elemento = document.querySelector('.box');
elemento.className = 'box destaque'; // Substitui todas as classes
elemento.classList.add('ativo');     // Adiciona uma classe
elemento.classList.remove('inativo'); // Remove uma classe
elemento.classList.toggle('visivel'); // Alterna a classe

// Alterar ID
elemento.id = 'novo-id';
```

MODIFICANDO HTML COM JAVASCRIPT

Também podemos estar manipulando os estilos diretamente:

```
const div = document.getElementById('minha-div');

// Alteração direta
div.style.backgroundColor = '#ff0000';
div.style.padding = '20px';
div.style.display = 'flex';
```

CRIANDO HTML COM JAVASCRIPT

Uma das grandes ferramentas que temos em mãos é a **criação de html pelo JavaScript**, já vimos a manipulação, mas ainda existe a possibilidade de criar novos elementos, para isso usaremos o método createElement() da classe document.

```
// Criar um novo elemento div
const novaDiv = document.createElement('div');

// Criar um parágrafo
const novoParagrafo = document.createElement('p');
```

CRIANDO HTML COM JAVASCRIPT

Na sequência precisamos **adicionar conteúdo** a nossos elementos e para isso podemos usar o método que aprendemos anteriormente.

No processo agora teremos de **adicionar** os elementos que criamos no nosso DOM, usaremos para isso:

- **appendChild()**: Adiciona como último filho
- **insertBefore()**: Insere antes de um elemento
- **insertAdjacent()**: método mais moderno, com passagem de parâmetros podemos controlar onde fazer a inserção (beforebegin, afterbegin, beforeend, afterend)

CRIANDO HTML COM JAVASCRIPT

```
novaDiv.innerHTML = '<p>Conteúdo da nova div</p>';

document.body.appendChild(novaDiv);
// ou a um elemento específico
const container = document.getElementById('container');
container.appendChild(novaDiv);

const elementoReferencia = document.getElementById('container');
document.body.insertBefore(novaDiv, elementoReferencia);
```

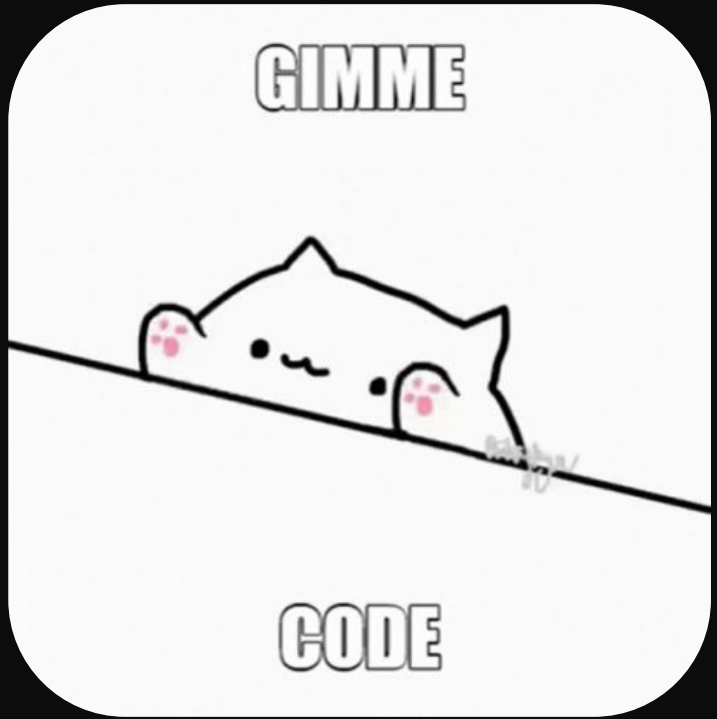
TREINANDO NOSSAS HABILIDADES!

Para esse treino teremos um arquivo html base (disponibilizados via github no dia da aula).

Vamos focar em criar uma receita de pudim de leite, para fazer isso precisaremos de uma estrutura com:

Titulo, Subtitulo (Ingredientes e passos);
Imagem do Pudim;
Lista de ingredientes e lista de passos.

Utilize o JS para manipular o DOM e criar nossa receita, não modifique o html!!!



<LAB365>

<LAB365>

JavaScript: Eventos do usuário

AGENDA | M1S05 - A2

- Vivência
- Eventos
  - O que são?
  - Eventos de Mouse
  - Eventos de teclado

VIVÊNCIA

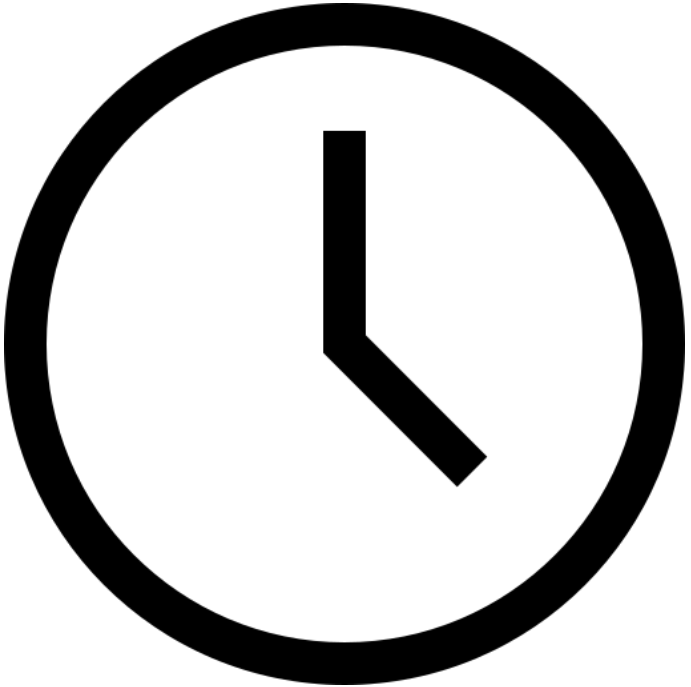
Momento

INTERVALO!

Finalizamos o nosso primeiro período de hoje. Que tal descansar um pouco?!

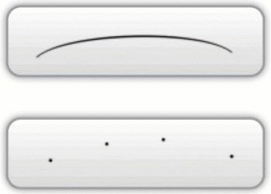
Nos vemos em 20 minutos.

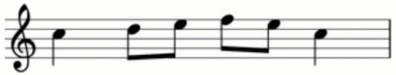
Início: 20:30
Retorno: 20:50



EVENTOS

Eventos em JavaScript são **ações ou ocorrências** que acontecem no **navegador** e podem ser detectadas e manipuladas pelo código. Eles permitem que a página web reaja a **interações** do **usuário** (como cliques, digitação ou movimentos do mouse) ou a **mudanças** no **sistema** (como carregamento da página ou temporizadores).





EVENTOS DE MOUSE

Os eventos de mouse são fundamentais para criar interatividade em páginas web. Eles permitem detectar quando o usuário clica, move ou passa o mouse sobre elementos.

Evento	Descrição
`click`	Acionado quando o usuário clica (pressiona e solta) em um elemento.
`dblclick`	Ocorre quando o usuário clica duas vezes rapidamente.
`mousedown`	Disparado quando o botão do mouse é pressionado (antes de soltar).
`mouseup`	Acionado quando o botão do mouse é liberado.
`mouseover`	Quando o ponteiro do mouse entra em um elemento.
`mouseout`	Quando o ponteiro do mouse sai de um elemento.
`mousemove`	Acionado enquanto o mouse se move sobre um elemento.
`contextmenu`	Quando o usuário clica com o botão direito (menu de contexto).

EXEMPLO DE CÓDIGO

```
const botao = document.querySelector('button');
const h1 = document.querySelector('h1');

botao.addEventListener('click', function() {
    console.log('Elemento clicado!');
});

botao.addEventListener('dblclick', function() {
    console.log('Duplo clique!');
});

h1.addEventListener('mousedown', function() {
    console.log('Botão pressionado');
});

h1.addEventListener('mouseover', function() {
    console.log('Mouse sobre o elemento');
});
```

EVENTOS DE TECLADO

Os eventos de teclado permitem que você **capture e responda às interações** do usuário com o **teclado**.

Propriedades Importantes do Objeto Event

- **event.key**: Retorna o valor da tecla pressionada (ex: 'a', 'Enter', 'ArrowUp')
- **event.code**: Retorna o código físico da tecla (ex: 'KeyA', 'Enter', 'ArrowUp')
- **event.altKey, event.ctrlKey, event.shiftKey, event.metaKey**: Booleanos que indicam se teclas modificadoras estavam pressionadas

Evento	Descrição
`keydown`	Tecla pressionada (inclui Shift/Ctrl) - repete se mantida
`keyup`	Tecla liberada (sempre após keydown)
`keypress`	(Obsoleto) Teclas que produzem caracteres

```
//Diferença entre key e code
document.addEventListener('keydown', function(event) {
    console.log(`Tecla: ${event.key}, Código: ${event.code}`);
});
```

EXEMPLO DE CÓDIGO

```
const botaoModo = document.getElementById('modoEscuro');
let modoEscuro = false;

// Por CLICK
botaoModo.addEventListener('click', mudarModo);

// Por TECLADO
document.addEventListener('keydown', (e) => {
  if (e.key === 'd' || e.key === 'D') {
    mudarModo();
  }
});

function mudarModo() {
    modoEscuro = !modoEscuro;
    if(modoEscuro) {
        document.body.style.background = '#222';
        console.log(`Modo escuro: ${'ON'}`);
    }
    else{
        document.body.style.background = '#fff';
        console.log(`Modo escuro: ${'OFF'}`);
    }
}
```

<LAB365>

<LAB365>

Eventos de formulário e revisão

<LAB365> SENAI

AGENDA | M1S05 - A3

- Eventos de formulário
- Revisão
- Correções

EVENTOS DE FORMULÁRIO

Os formulários são elementos essenciais em aplicações web, permitindo a **interação do usuário** através de **campos de entrada, seleção, botões e submissão de dados**.

O JavaScript oferece diversos eventos específicos para manipulação de formulários, validação e tratamento de dados.

Evento	Descrição
submit	Disparado quando o formulário é submetido (via botão submit ou Enter)
reset	Acionado quando o formulário é redefinido para seus valores iniciais
change	Ocorre quando o valor de um elemento é alterado e confirmado (perde foco)
input	Disparado continuamente enquanto o valor de um campo é modificado
focus	Acionado quando um elemento recebe foco
blur	Disparado quando um elemento perde o foco
invalid	Ocorre quando um elemento falha na validação

EXEMPLO DE CÓDIGO

```html
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Teste</title>
</head>
<body>

    <form id="form-exemplo">
        <input type="text" id="nome" placeholder="Seu nome">
        <input type="email" id="email" placeholder="Seu e-mail">
        <button type="submit">Enviar</button>
    </form>
    <div id="mensagem"></div>

</body>
</html>
```

EXEMPLO DE CÓDIGO

```
// Elementos
const form = document.getElementById('form-exemplo');
const nomeInput = document.getElementById('nome');
const emailInput = document.getElementById('email');
const mensagemDiv = document.getElementById('mensagem');

// 1. Evento 'input' (enquanto digita)
nomeInput.addEventListener('input', () => {
  mensagemDiv.textContent = `Digitando: ${nomeInput.value}`;
});

// 2. Evento 'change' (quando termina de editar)
emailInput.addEventListener('change', () => {
  mensagemDiv.textContent = `E-mail digitado: ${emailInput.value}`;
});

// 3. Evento 'focus' (quando clica no campo)
nomeInput.addEventListener('focus', () => {
  nomeInput.style.backgroundColor = '#ffffcc';
});

// 4. Evento 'blur' (quando sai do campo)
nomeInput.addEventListener('blur', () => {
  nomeInput.style.backgroundColor = 'white';
});
```

```
// 5. Evento 'submit' (envio do formulário)
form.addEventListener('submit', (e) => {
  e.preventDefault(); // Evita recarregar a página
  mensagemDiv.textContent = `Formulário enviado! Nome: ${nomeInput.value}, E-mail: ${emailInput.value}`;

  // Limpa os campos
  nomeInput.value = '';
  emailInput.value = '';
});
```

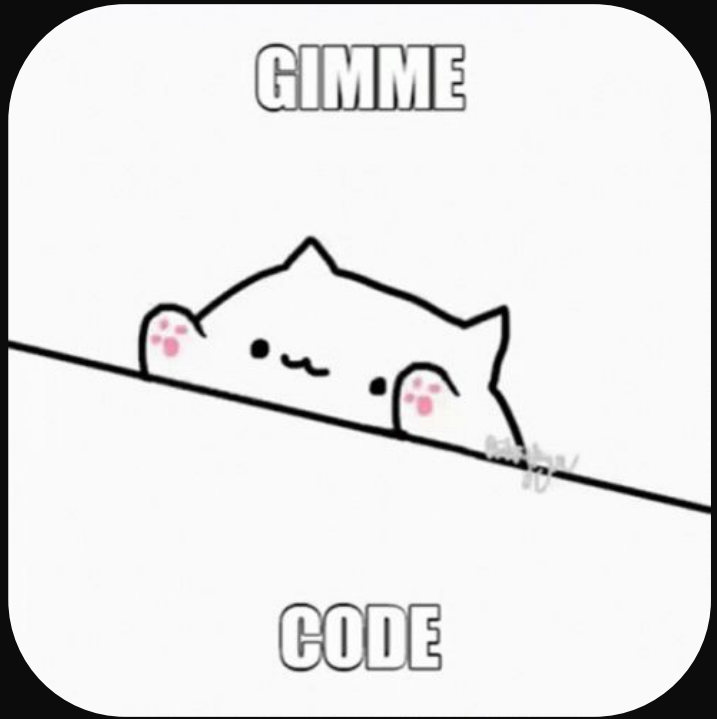
TREINANDO NOSSAS HABILIDADES!

Chegando a essa etapa dos nossos aprendizados já temos conhecimentos para desenvolver algo um pouco mais complexo.

Precisamos de uma tela que nos pedirá o nome e email do usuário, após isso deverá aparecer um campo com assunto, urgência e mensagem.

Esse formulário deve ser enviado e exibido após o envio.

Estilize seus HTML/CSS e desenvolva os eventos para criar esse processo de solicitação de serviço!

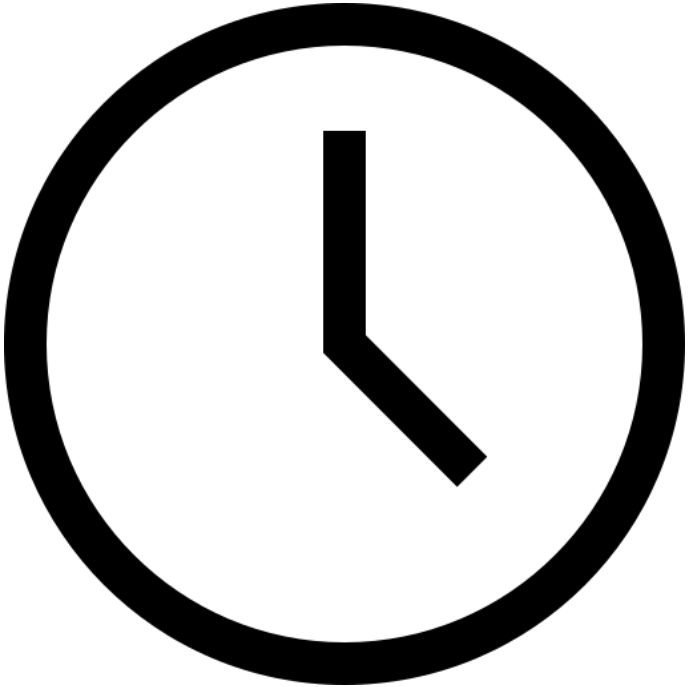


INTERVALO!

Finalizamos o nosso primeiro período de hoje. Que tal descansar um pouco?!

Nos vemos em 20 minutos.

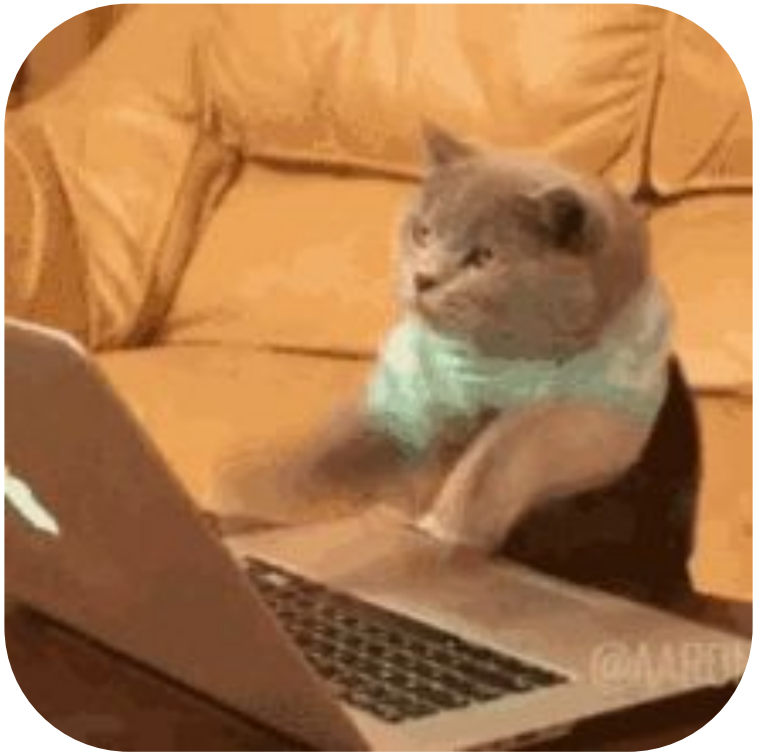
Início: 20:20
Retorno: 20:40



REVISÃO E CORREÇÕES

Nossa revisão será programando!

- HTML
- CSS
- Javascript
  - Estruturas Condicionais
  - Estruturas de repetição
  - DOM: Querys
  - DOM: Manipulação
  - Eventos



VAMOS PARA OS EXERCÍCIOS!

AVALIAÇÃO DOCENTE

O que você está achando das minhas aulas neste conteúdo?

Clique aqui ou escaneie o QRCode ao lado para avaliar minha aula.

Sinta-se à vontade para fornecer uma avaliação sempre que achar necessário.

<LAB365>