Interatividade na Web

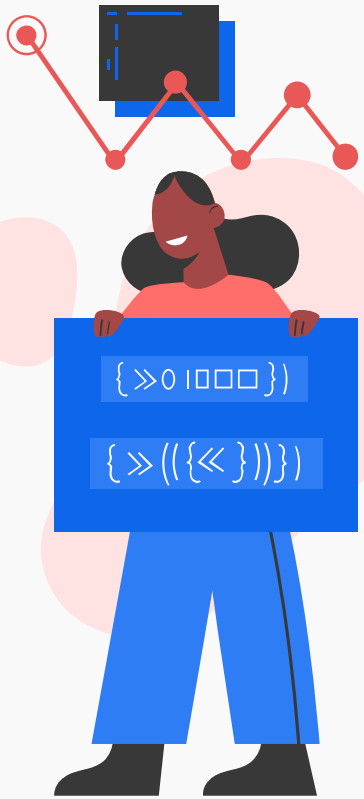
Front End Web

Msc. Lucas G. F. Alves
e-mail: lucas.g.f.alves@gmail.com

Senac

Planejamento de Aula

- Revisão
- CSS Avançado
- Pseudo-Classes
- Pseudo Elementos
- CSS3 box-shadow, text-shadow
- Opacidade



Revisão

Introdução ao Javascript

Para rodar JavaScript em uma página Web, precisamos ter em mente que a execução do código é instantânea.

Para inserirmos um código JavaScript em uma página, é necessário utilizar a tag <script>:

```
<script>
    alert("Olá, Mundo!");
</script>
```

O exemplo acima é um “hello world” em JavaScript utilizando uma função do navegador, a função alert.

É possível adicionar essa tag em qualquer local do documento que a sua renderização ficará suspensa até o término dessa execução.

Introdução ao Javascript

Arquivo Externo.

No arquivo HTML

<script src="scripts/hello.js"></script>

Arquivo externo script/hello.js

alert("Olá, Mundo!");

Com a separação do script em arquivo externo é possível reaproveitar alguma funcionalidade em mais de uma página.

Introdução ao Javascript

Arquivo Externo.

No arquivo HTML

<script src="scripts/hello.js"></script>

Arquivo externo script/hello.js

alert("Olá, Mundo!");

Com a separação do script em arquivo externo é possível reaproveitar alguma funcionalidade em mais de uma página.

Introdução ao Javascript

Console no navegador

Alguns navegadores dão suporte a entrada de comandos pelo console.

Por exemplo:

No Google Chrome o console pode ser acessado por meio do atalho Control + Shift + C;
No Firefox, pelo atalho Control + Shift + K.

Experimente executar um alert no console e veja o resultado:

```
alert("interagindo com o console!");
```

Introdução ao Javascript

Sintaxe básica

Operações matemáticas. Teste algumas contas digitando diretamente no console:

```
> 12 + 13
25
> 14 * 3
42
> 10 - 4
6
> 25 / 5
5
> 23 % 2
1
```

Introdução ao Javascript

Variáveis

Para armazenar um valor para uso posterior, podemos criar uma variável:

```
var curso = "FWeb-23";

alert(curso);
```

No exemplo acima, foi guardo o valor FWeb-23 na variável curso. A partir desse ponto, é possível utilizar a variável para obter o valor que guardamos nela.

Introdução ao Javascript

Tipos

O JavaScript dá suporte aos tipos **String** (letras e palavras), **Number** (números inteiros, decimais), **Boolean** (verdadeiro ou falso) entre outros.

```
var texto = "Uma String deve ser envolvida em aspas simples ou duplas.";
var numero = 2012;
var verdade = true;
```

Outro tipo de informação que é considerado um tipo no JavaScript é o **Array**, onde podemos armazenar uma série de informações de tipos diferentes:

```
var pessoas = ["João", "José", "Maria", "Sebastião", "Antônio"];
```

Introdução ao Javascript

Tipos - Array

O JavaScript é utilizado para interagir com os elementos da página. Para fazer alguma coisa com cada elemento de uma coleção é necessário efetuar uma iteração. A mais comum é o **for**:

```
var pessoas = ["João", "José", "Maria", "Sebastião", "Antônio"];
for (var i = 0; i < pessoas.length; i++) {
        alert(pessoas[i]);
}
```

```
alert(pessoas[0]);
alert(pessoas[1]);
alert(pessoas[4]);
```

Interatividade na Web

O JavaScript é a principal linguagem de programação da Web.
Com essa linguagem é possível encontrar um elemento da página e alterar características dele pelo código JavaScript:

```
var titulo = document.querySelector("#titulo");
titulo.textContent = "Agora o texto do elemento mudou!";
```

No exemplo é selecionado um elemento do documento e alterado sua propriedade textContent.Também é possível selecionar elementos de maneira similar no CSS, através de seletores CSS:

```
var painelNovidades = document.querySelector("section#main .painel#novidades");
painelNovidades.style.color = "red"
```

Interatividade na Web

querySelector vs querySelectorAll

A função querySelector sempre retorna um elemento, mesmo que o seletor potencialmente traga mais de um elemento, neste caso, apenas o primeiro elemento da seleção será retornado.
A função querySelectorAll retorna uma lista de elementos compatíveis com o seletor CSS passado como argumento. Sendo assim, para acessarmos cada elemento retornado, precisaremos passar o seu índice conforme o exemplo abaixo:

```
var paragrafos = document.querySelectorAll("div p");
paragrafos[0].textContent = "Primeiro parágrafo da seleção";
paragrafos[1].textContent = "Segundo parágrafo da seleção";
```

Interatividade na Web

As alterações serão feitas quando o usuário executa alguma ação, como por exemplo, clicar em um elemento.
Para fazer isso é necessário utilizar de dois recursos do JavaScript no navegador.
O primeiro é a criação de **funções**:

```
function mostraAlerta() {
    alert("Funciona!");
}
```

Esse código é guardado e só será executado quando chamarmos a função. Para isso será utilizado o segundo recurso, os **eventos:**

```
// obtendo um elemento através de um seletor de ID
var titulo = document.querySelector("#titulo");
titulo.onclick = mostraAlerta;
```

Interatividade na Web

As alterações serão feitas quando o usuário executa alguma ação, como por exemplo, clicar em um elemento.
Para fazer isso é necessário utilizar de dois recursos do JavaScript no navegador.
O primeiro é a criação de **funções**:

```
function mostraAlerta() {
    alert("Funciona!");
}
```

Esse código é guardado e só será executado quando chamarmos a função. Para isso será utilizado o segundo recurso, os **eventos:**

```
// obtendo um elemento através de um seletor de ID
var titulo = document.querySelector("#titulo");
titulo.onclick = mostraAlerta;
```

Interatividade na Web

Chamar a função sem evento:

```
// Chamando a função:
mostraAlerta();
```

A função vai ter algum valor variável que será definido quando executá-la:

```
function mostraAlerta(texto) {
    // Dentro da função  a variável "texto" conterá o valor passado na execução.
    alert(texto);
}
// Ao chamar a função é necessário definir o valor do "texto"
mostraAlerta("Funciona com argumento!");
```

Interatividade na Web

Eventos:
onclick: clica com o mouse; ondblclick: clica duas vezes com o mouse;
onmousemove: mexe o mouse; onmousedown: aperta o botão do mouse;
onmouseup: solta o botão do mouse (útil com os dois acima para gerenciar drag'n'drop);
onkeypress: ao pressionar e soltar uma tecla; onkeydown: ao pressionar uma tecla;
onkeyup: ao soltar uma tecla. Mesmo acima; onblur: quando um elemento perde foco;
onfocus: quando um elemento ganha foco;
onchange: quando um input, select ou textarea tem seu valor alterado;
onload: quando a página é carregada; onunload: quando a página é fechada;
onsubmit: disparado antes de submeter o formulário. Útil para realizar validações.

Exercícios

1) Utilizar o Google como mecanismo de busca apenas como ilustração. Para isso, basta fazer o formulário submeter a busca para a página do Google. No <form>, acrescente um atributo action= apontando para a página do Google:

```
<form action="http://www.google.com.br/search" id="form-busca">
<input type="search" name="q" id="q">
```

2) Validar quando o usuário clicar em submeter, verificar se o valor foi preenchido. Se estiver em branco, mostrar uma mensagem de erro em um alert. A validação será escrita em JavaScript. Portanto, crie o arquivo index.js na pasta js/ do projeto. Referencie esse arquivo no index.html usando a tag <script> no final da página, logo antes de fechar o </body>:

```
<body>
    ....
    <script src="js/home.js"></script>
</body>
```

Exercícios

3) Criar a função que verifica se o elemento com id=q (o campo de busca) está vazio. Se estiver, mostramos um alert e abortamos a submissão do formulário com return false:

```
function validaBusca() {
    if (document.querySelector('#q').value == '') {
        alert('Não podia ter deixado em branco a busca!');
        return false;
    }
}
```

4) Indicar que a função anterior deve ser executada quando o usuário disparar o evento de enviar o formulário (onsubmit). Coloque no final do arquivo JavaScript:

```
// fazendo a associação da função com o evento
document.querySelector('#form-busca').onsubmit = validaBusca;
```

Funções temporais

No JavaScript, se pode criar um timer para executar um trecho de código após um certo tempo, ou ainda executar algo de tempos em tempos.

A função **setTimeout** permite que agende alguma função para execução no futuro e recebe o nome da função a ser executada e o número de milissegundos a esperar:

```
// executa a minhaFuncao daqui um segundo
setTimeout(minhaFuncao, 1000);
```

Se for um código recorrente, pode-se usar o **setInterval** que recebe os mesmos argumentos mas executa a função indefinidamente de tempos em tempos:

```
// executa a minhaFuncao de um em um segundo
setInterval(minhaFuncao, 1000);
```

Funções temporais

clearInterval

É utilizado para cancelar a execução no futuro. É especialmente interessante para o caso do interval que pode ser cancelado de sua execução infinita:

```
// agenda uma execução qualquer
var timer = setInterval(minhaFuncao, 1000);

// cancela execução
clearInterval(timer);
```

Exercícios

1) Implementar um banner rotativo na Pagina Principal usando JavaScript. Colocar duas imagens, a destaque-home.png e a destaque-home-2.png para trocar a cada 4 segundos;

```
var banners = ["img/destaque-home.png", "img/destaque-home-2.png"];
var bannerAtual = 0;
function trocaBanner() {
    bannerAtual = (bannerAtual + 1) % 2;
    document.querySelector('.destaque img').src = banners[bannerAtual];
}
setInterval(trocaBanner, 4000);
```

```
<div class="container destaque">
    ...
    <img src="./img/logo.png" alt="Imagem de destaque">
</div
```

Exercícios

2) Faça um botão de pause que pare a troca do banner.
Dica: use o clearInterval para interromper a execução.

```
<a href=”#” class=”pause”></a>
var controle = document.querySelector('.pause');
controle.onclick = function() { if (controle.className == 'pause') { clearInterval(timer);
controle.className = 'play'; } else { timer = setInterval(trocaBanner, 4000);
controle.className = 'pause'; } return false; };
```

3) Faça um botão de play para reativar a troca dos banners.

4) Dentro da página Blog colocar as tags de estrutura e desenvolver uma calculadora.

({(({≫})) ≪ }

CSS Avançado

CSS Avançado

Com CSS, os desenvolvedores front-end utilizam diversas técnicas de estilização.

Mesmo assim algumas coisas eram impossíveis de se conseguir utilizando somente CSS.

O CSS3 agora permite coisas antes impossíveis como elementos com cor ou fundo gradiente, sombras e cantos arredondados. Antes só era possível atingir esses resultados com o uso de imagens e às vezes até com um pouco de JavaScript.

A redução do uso de imagens traz grandes vantagens quanto à performance e quantidade de tráfego de dados necessária para a exibição de uma página.

CSS Avançado

Seletores avançados

Os seletores CSS mais comuns e tradicionais são os que já vimos: por ID, por classes e por descendência.

Seletor tag -> header{} **Seletor ID** -> #cabecalho **Seletor classe** -> .divp

Seletor Dependência -> #rodape img {} **Seletor Atributo** -> input[type="text"] {}

Com o CSS3, há muitos outros seletores novos que são bastante úteis.

Já vimos alguns, como os seletores de atributo que usamos anteriormente.

CSS Avançado

Seletor de irmãos (~)

Tem como objetivo selecionar de uma certa maneira todos os elementos após o subtítulo.
Ex.:

```
<article>
    <h1>Título</h1>
    <p>Início</p>

    <h2>Subtítulo</h2>
    <p>Texto</p>
    <p>Mais texto</p>
</article>
```

```
h2 ~ p {
    font-style: italic;
}
```

CSS Avançado

Seletor de irmãos Adjacentes (+)

Tem como objetivo selecionar apenas o parágrafo imediatamente seguinte ao subtítulo.
Ex.:

```
<article>
    <h1>Título</h1>
    <p>Início</p>

    <h2>Subtítulo</h2>
    <p>Texto</p>
    <p>Mais texto</p>
</article>
```

```
h2 + p {
    font-variant: small-caps;
}
```

CSS Avançado

Seletor de Filho Direto (>)

Tem como objetivo selecionar apenas o parágrafo filho seguinte ao subtítulo.
Ex.:

```
<article>
    <h1>Título principal</h1>


    <section>
    <h1>Título da seção</h1>
    </section>
</article>
```

```
/* vai pegar todos os h1 da página */
h1 { color: blue;}

/* vai pegar todos os h1 do article, incluindo de
dentro da section */
article h1 { color: blue; }

/* vai pegar só o h1 principal, filho direto de article e
não os netos */
article > h1 { color: blue; }
```

CSS Avançado

Negação

Tem como objetivo escrever um seletor que pega elementos que não batem naquela regra..
Ex.:

```
<p>Texto</p>
<p>Outro texto</p>
<p class="especial">Texto especial</p>
<p>Mais texto</p>
```

```
p:not(.especial) {
    color: gray;
}
```

A sintaxe do :not() recebe como argumento algum outro seletor simples (como classes, IDs ou tags).

({(({≫})) ≪ }

Pseudo-Classes

Pseudo-Classes

Pseudo-classes são como classes CSS já pré-definidas para nós. É como se o navegador já colocasse certas classes por padrão em certos elementos, cobrindo situações comuns.

Há duas pseudo-classes do CSS3, uma que representa o primeiro elemento filho de outro (first-child) e o último elemento filho (last-child). Essas classes já estão definidas, não precisamos aplicá-las em nosso HTML.

Ex.:

```
<ul>
    <li>Primeiro item</li>
    <li>Segundo item</li>
    <li>Terceiro item</li>
    <li>Quarto item</li>
</ul>
```

```
li:first-child {
    color: red;
}

li:last-child {
    color: blue;
}
```

Pseudo-Classes

nth-child

É um seletor genérico do CSS3 que permite passar o índice do elemento. Por exemplo, podemos pegar o terceiro item com:

```
li:nth-child(3) { color: yellow; }
```

O mais interessante é que o nth-child pode receber uma expressão aritmética para indicar quais índices selecionar.

```
li:nth-child(2n) { color: green; } /* elementos pares */
li:nth-child(2n+1) { color: blue; } /* elementos impares */
```

Pseudo-Classes

Pseudo-Classes de Estado

O CSS possui excelentes pseudo-classes que representam estados dos elementos.

Antes utilizamos javascript para mudar o estado de um elemento.

Agora temos pseudos-classes de estados que fazem isso.
Ex: Uma que representa o momento que o usuário está com o mouse em cima do elemento, a :hover.

```
/* seleciona o link no exato momento em que passamos o mouse por cima dele */
 a:hover { background-color:#FF00FF; }
```

Pseudo-Classes

Pseudo-Classes de Estado

```
/* seleciona todas as âncoras que têm o atributo "href", ou seja, links */
 a:link { background-color:#FF0000; }

/* seleciona todos os links cujo valor de "href" é um endereço já visitado */
a:visited { background-color:#00FF00; }

/* seleciona o link no exato momento em que clicamos nele */
a:active { background-color:#0000FF; }
```

Pseudo Elementos

Pseudo Elementos

São elementos que não existem no documento mas podem ser selecionados pelo CSS. É como se houvesse um elemento lá!
Ex.:

```
<p>Pseudo Elementos</p>
```

```
p:first-letter {
    font-size: 200%;
}

p:first-line {
    font-style: italic;
}
```

Pseudo Elementos

Novos Conteúdos

Um outro tipo de pseudo-elemento mais poderoso que nos permite gerar conteúdo novo via CSS é o **after** e **before**.

[Link 1] [Link 2] [Link 3]

HTML

```
<a href="...">Link1</a>
<a href="...">Link2</a>
<a href="...">Link3</a>
```

CSS

```
a:before {
        content: '[ ';
}
a:after {
        content: ' ]';
}
```

Exercícios

Exercícios

1) Alterar a página de sobre voces, a contato.html modificando as primeiras letras dos parágrafos fiquem em negrito. Altere o arquivo contato.css e use a pseudo-classe :first-letter pra isso. p:first-letter { font-weight: bold; }

2) Remover apenas a identação do primeiro parágrafo da página. Usando seletor de irmãos adjacentes. Acrescente ao contato.css: h1 + p { text-indent: 0;}

3) Alterar o visual da primeira linha do texto com :first-line. Por exemplo, transformando-a em small-caps usando a propriedade font-variant: p:first-line {font-variant: small-caps;}

4) Agora no index.css. Temos (.menu-opcoes) que é uma lista de links. Altere seus estados quando o usuário passar o mouse (:hover) e quando clicar no item (:active). Adicione ao arquivo index.css: .menu-opcoes a:hover { color: #007dc6;} .menu-opcoes a:active { color: #867dc6; }

Exercícios

5) Fazer um menu que abre e fecha em puro CSS. Temos o .menu-conteudo na esquerda da página com várias categorias de aulas. Adicionar sucategorias que aparecem apenas quando o usuário passar o mouse.

HTML

```
<li>
        <a href="#">Aula 1</a>
        <ul>
                <li><a href="#">Exercício 1</a></li>
                <li><a href="#">Exercício 2</a></li>
                <li><a href="#">Exercício 3</a></li>
        </ul>
</li>
```

CSS

```
.menu-conteudo li ul {
        display: none; /*por padrão escondida*/
}
.menu-conteudo li:hover ul {
        display: block; /*no mouse aparece*/
}
.menu-conteudo ul ul li {
        background-color: #dcdcdc;
 }
```

Exercícios

6) Colocar um traço na frente para diferenciar os links dos menus. Podemos alterar o HTML colocando os traços - algo visual e não semântico -, ou podemos gerar esse traço via CSS com pseudo elementos. Utilise o :before para injetar um conteúdo novo via propriedade content no CSS: .menu-conteudo li li a:before {content: '- ';}

7) Na página Contato, abra-a no navegador. No exercício 3, as primeiras linhas foram colocadas em small-caps usando o seletor p:first-line. Mas todos os parágrafos foram afetados. Para resolver utilize a negação. Com o seletor :not() do CSS3:
:not(p) + p:first-line { font-variant: small-caps; }

CSS3 Border Radius

CSS3 Border Radius

No CSS3 foi adicionado as bordas arredondadas via CSS. Até então, a única forma de obter esse efeito era usando imagens, o que deixava a página mais carregada e dificultava a manutenção.

Com o CSS3 há o suporte a propriedade border-radius que recebe o tamanho do raio de arredondamento das bordas. Por exemplo:

```
div {
    border-radius: 5px;
}
```

CSS3 Border Radius

Também pode passar valores diferentes para cantos diferentes do elemento:

```
.a{ /* todas as bordas arredondadas com um raio de 15px */ border-radius: 15px; }
.b{ /*borda superior esquerda e inferior direita com 5px borda superior direita e inferior esquerda com 20px*/ border-radius: 5px 20px; }
.c{ /*borda superior esquerda com 5px borda superior direita e inferior esquerda com 20px borda inferior direita com 50px */ border-radius: 5px 20px 50px; }
.d{ /*borda superior esquerda com 5px borda superior direita com 20px borda inferior direita com 50px bordar inferior esquerda com 100px */ border-radius: 5px 20px 50px 100px; }
```

CSS3 Text Shadow

CSS3 Text Shadow

Outro efeito do CSS3 são as sombras em textos com text-shadow. Sua sintaxe recebe o deslocamento da sombra e sua cor:

```
p {
    text-shadow: 10px 10px red;
}
```

Ou ainda receber um grau de espalhamento (blur):

```
p {
    text-shadow: 10px 10px 5px red;
}
```

É possível até passar mais de uma sombra ao mesmo tempo para o mesmo elemento:

```
text-shadow: 10px 10px 5px red, -5px -5px 4px red;
```

CSS3 box-shadow

CSS3 box-shadow

Agora também é possível colocar sombras em qualquer elemento com box-shadow. A sintaxe é parecida com a do text-shadow:

```
box-shadow: 20px 20px black;
```

Podemos ainda passar um terceiro valor com o blur:

```
box-shadow: 20px 20px 20px black;
```

CSS3 box-shadow

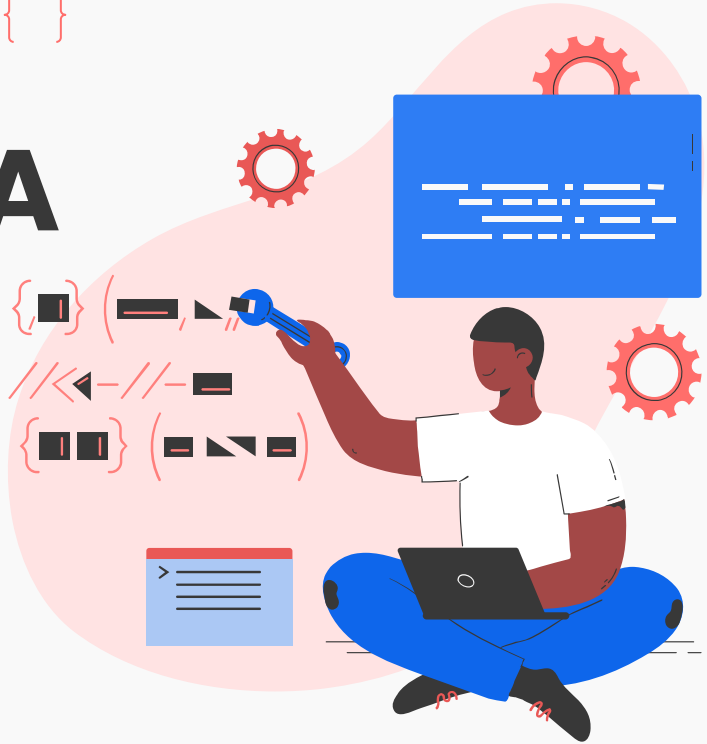
Diferentemente do text-shadow, o box-shadow suporta ainda mais um valor que faz a sombra aumentar ou diminuir:

```
box-shadow: 20px 20px 20px 30px black;
```

Por fim, é possível usar a keyword inset para uma borda interna ao elemento:

```
box-shadow: inset 0 0 40px black;
```

Opacidade RGBA



Opacidade RGBA

No CSS2 é possível mudar a opacidade de um elemento para que ele seja mais transparente com o uso da propriedade opacity.

```
p { opacity: 0.3; }
```

No CSS3, além das cores hex normais (#FFFFFF pro branco), podemos criar cores a partir de seu valor RGB, passando o valor de cada canal (Red, Green, Blue) separadamente (valor entre 0 e 255):

```
/* todos equivalentes */
color: #FFFFFF;  color: white;  color: rgb(255, 255, 255);
```

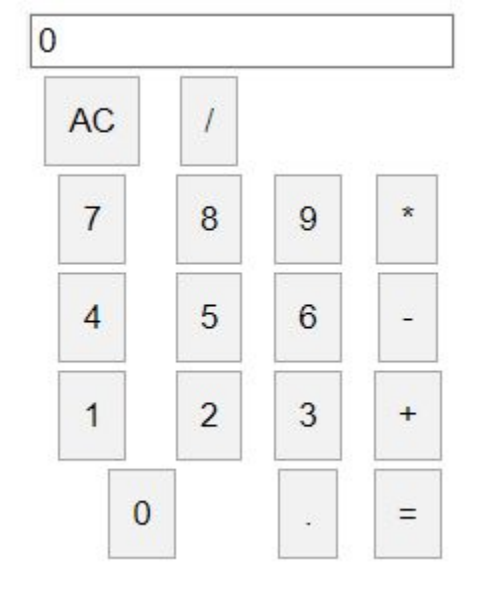
Porém, existe uma função chamada RGBA que recebe um quarto argumento, o chamado canal Alpha. Na prática, seria como o opacity daquela cor (um valor entre 0 e 1):

```
/* branco com 80% de opacidade */
color: rgba(255,255,255, 0.8);
Ex,: p { background: rgba(0,0,0,0.3); color: white; }
```

Exercícios

Opacidade RGBA

1) Criar uma calculadora com todas as funções básicas, utilizando table para estruturar os botões. Além de resolver as questões aritméticas com funções em javascript.



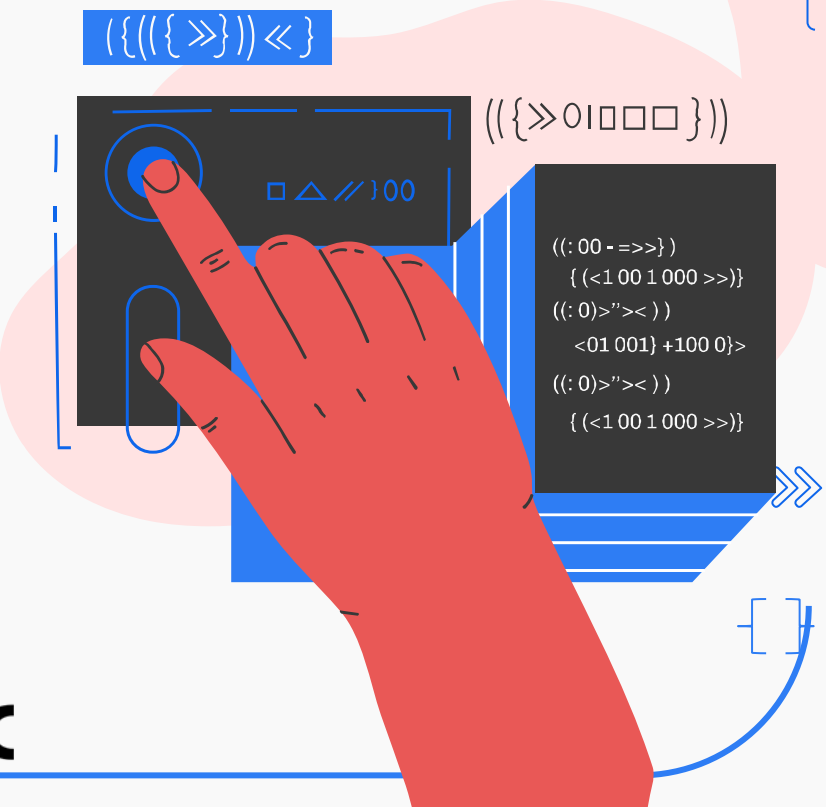
Professor

Lucas G. F. Alves

Senac

Obrigado!

E-mail :lucas.g.f.alves@gmail.com



Senac