

Responsive Web

We create fully responsive website
initial analysis to maintenance
experience guaranteed

DESENVOLVIMENTO WEB

Responsive Web Design.

We can develop responsive website from
scratch or adapt existing website
to different devices and platforms.

[VIEW MORE](#)

Responsive Web Design.

The design is to make website look
good on all devices. It's a good
choice for e-commerce websites.

[VIEW MORE](#)

**UNIBRASIL**
CENTRO UNIVERSITÁRIO

EAD

DIRIGENTES

PRESIDÊNCIA

Prof. Dr. Clémerson Merlin Clève

REITORIA

Prof. Me. Alessandro Kinal

DIRETORIA ACADÊMICA EAD

Prof.ª Me. Daniela Ferreira Correa

DIRETORIA ACADÊMICA PRESENCIAL

Prof.ª Me. Márcia Maria Coelho

DIRETORIA EXECUTIVA

Prof.º Esp. Silmara Marchioretto

COORDENAÇÃO PEDAGÓGICA DE GRADUAÇÃO EAD

Prof. Me. João Marcos Roncari Mari

COORDENAÇÃO PEDAGÓGICA DE PÓS-GRADUAÇÃO EAD

Prof. Me. Marcus Vinícius Roncari Mari

FICHA TÉCNICA

AUTORA

Prof. Esp. Leonel da Rocha

COORDENAÇÃO DA PRODUÇÃO DE MATERIAIS EAD

Esp. Janaína de Sá Lorusso

PROJETO GRÁFICO

Esp. Janaína de Sá Lorusso

Esp. Cinthia Durigan

DIAGRAMAÇÃO

Marcelo Winck

REVISÃO

Esp. Ísis C. D'Angelis

Esp. Idamara Lobo Dias

PRODUÇÃO AUDIO VISUAL

Esp. Rafael de Farias Forte Canonico

Estúdio NEAD (Núcleo de Educação a Distância) - UniBrasil

ORGANIZAÇÃO

NEAD (Núcleo de Educação a Distância) - UniBrasil

IMAGENS

Shutterstock



SUMÁRIO

UNIDADE 01 - HTML

OBJETIVOS DE APRENDIZAGEM	06
INTRODUÇÃO.....	07
1. INICIANDO A CRIAÇÃO DE DOCUMENTOS HTML	08
1.1 Tags HTML.....	08
1.2 Estrutura básica de um documento html.....	08
1.3 Corpo do documento	09
2. FORMATAÇÃO DE TEXTO	14
2.1 Fonte	14
2.2 Parágrafo.....	15
2.3 Cabeçalho.....	15
2.4 Quebra de linha.....	16
2.5 Tabela.....	16
CONSIDERAÇÕES FINAIS	18

UNIDADE 02 - CSS

OBJETIVOS DE APRENDIZAGEM	19
INTRODUÇÃO.....	20
1. CRIAÇÃO DE ESTILOS	21
1.1 Folha de estilo interna	21
1.2 Folha de estilo externa.....	21
2. FORMATAÇÃO DE TEXTO	23
2.1 Body.....	23

SUMÁRIO

2.2	Fonte	24
2.3	Parágrafo	24
2.4	Cabeçalho.....	25
2.5	Agrupamento de seletores.....	26
2.6	Complemento dos seletores	27
2.7	Comentários	27
2.8	Seletor classe.....	27
3.	TAGS PERSONALIZADAS	28
4.	ATALHOS DE CSS.....	29
	CONSIDERAÇÕES FINAIS	30

UNIDADE 03 - JAVASCRIPT - LÓGICA

	OBJETIVOS DE APRENDIZAGEM	32
	INTRODUÇÃO.....	33
1.	CARACTERISTICAS BÁSICAS DO JAVASCRIPT	34
1.1	Eventos.....	34
1.2	Tipos de variáveis	35
1.3	Operadores.....	35
1.4	Atribuições	36
1.5	Comparações.....	36
2.	CRIAÇÃO DE SCRIPTS	37
2.1	Comentários	38
2.2	Alertas.....	39

SUMÁRIO

2.3	Confirmações.....	40
2.4	Avisos.....	41
3.	LÓGICA DE PROGRAMAÇÃO	42
3.1	Tomada de decisão	44
3.2	Laços de repetição	46
3.3	Operadores lógicos: and e or	47
	CONSIDERAÇÕES FINAIS	50

UNIDADE 04 - JAVASCRIPT - PROGRAMAÇÃO

	OBJETIVOS DE APRENDIZAGEM	52
	INTRODUÇÃO.....	53
1.	ESTRUTURAS DE DECISÃO E REPETIÇÃO	54
1.1	Estruturas de decisão.....	55
1.2	Estruturas de repetição.....	55
2.	CRIAÇÃO DE PÁGINAS DINÂMICAS E RESPONSIVAS	57
2.1	Páginas dinâmicas	57
2.2	Design responsivo.....	59
3.	INTERFACE AMIGÁVEL	60
3.1	React	61
4.	FERRAMENTAS DE DEPURAÇÃO	62
4.1	Utilizando o navegador para debug.....	62
	CONSIDERAÇÕES FINAIS	64
	REFERÊNCIAS	66



UNIDADE

01

HTML



OBJETIVOS DE APRENDIZAGEM

- » Conceituar HTML quanto à sua utilização e possibilidades de desenvolvimento de páginas WEB
- » Criar documentos HTML, dominar listas, inserir imagens e tabelas, aplicar cores e formatar fontes e parágrafos



VÍDEOS DA UNIDADE



<https://bit.ly/3q2rWox>



<https://bit.ly/3xt8XWU>



<https://bit.ly/35sGWTd>



<https://bit.ly/3q1B5h0>



<https://bit.ly/2SG9til>



<https://bit.ly/2SG9GVF>



<https://bit.ly/3q1geds>



INTRODUÇÃO

O instrumento utilizado para a construção das páginas da *web*, rede mundial de computadores popularizada a partir dos anos 1990, com a criação de uma interface gráfica que facilitou a interação entre homem e máquina, chama-se *hipertexto*, que são inúmeras páginas conectadas através de *links*. A *web* não se restringe apenas a textos e *links*, podendo oferecer também imagens, vídeos, áudios e demais conteúdos de multimídia.

O desenvolvimento de páginas *web* tem como base a linguagem HTML, CSS e JAVASCRIPT, elementos que serão vistos nesta disciplina de Desenvolvimento WEB. Eles são utilizados para construir e formatar o conteúdo de um *website*, também conhecido como endereço eletrônico, ou simplesmente *site*, hipertextos acessíveis pelo protocolo HTTP ou HTTPS na internet. O conjunto desses *sites* existentes compõe a World Wide Web, rede mundial comumente conhecida como *www* ou *web*.

Quando uma página *web* é visualizada no navegador, programas que disponibilizam conteúdo da internet na tela do computador, conhecidos também como *browser* ou *web browser* e que funcionam em computadores, *notebooks*, dispositivos móveis, televisores e todo tipo de equipamentos que possam ser conectados à internet, ela parece ser uma única entidade. Mas, na verdade, uma página WEB é composta por um número grande de arquivos, imagens, vídeos e um elemento muito importante: o código fonte. Esse código fonte que compõe as páginas da *web* está escrito em uma linguagem chamada HTML, abreviação para a expressão inglesa *HyperText Markup Language*, que significa Linguagem de Marcação de Hipertexto, e é interpretada pelos navegadores.

O HTML não é uma linguagem de programação, é uma linguagem de marcação, usada para definir a estrutura de uma página. Ele é formado por uma série de elementos, que são utilizados para delimitar ou agrupar as partes do conteúdo para que seja exibido de determinada maneira. As TAGs que compõem a linguagem podem transformar uma palavra ou imagem em um hiperlink, podem colocar palavras em itálico, podem aumentar ou diminuir a fonte, e assim por diante.



1. INICIANDO A CRIAÇÃO DE DOCUMENTOS HTML

HTML tem por característica ser uma linguagem de marcação e estruturação de hipertexto. O hipertexto é um conceito associado à tecnologia da informação e se difere de um texto comum na forma de escrita e leitura. Em um texto comum, a escrita e a leitura são lineares, enquanto em um hipertexto elas não são lineares. Esses hipertextos são um conjunto de eventos conectados por *webs* ou *hiperlinks*. O HTML é a base para a construção dos documentos de hipertexto, que aplicam padrões que são representados por TAGs ou etiquetas.

1.1 TAGS HTML

As TAGs, que podem ser compreendidas como etiquetas, delimitam a estrutura de um documento HTML. O *browser* fará a leitura dessas TAGs e montará a página conforme a formatação estipulada pelos comandos HTML.

Uma TAG é iniciada com o sinal de menor (“<”) e finalizada com o sinal de maior (“>”), como mostrado nesse exemplo: <HTML>. O HTML, na maioria dos casos, prevê uma TAG de fechamento para cada TAG aberto – isso delimita a área de influência da TAG. Para a TAG de fechamento, é adicionado o caractere “/” na frente do nome da TAG, que é o mesmo nome da TAG de abertura. Exemplo de TAG de fechamento: </HTML>. As TAGs podem ser escritas em maiúsculas ou minúsculas, porém não podem conter espaços em branco.

1.2 ESTRUTURA BÁSICA DE UM DOCUMENTO HTML

Um documento HTML possui uma estrutura básica dividida em quatro TAGs principais: HTML, HEAD, TITLE e BODY, que definem a parte principal, o cabeçalho e o corpo da página.

<HTML>: delimita os comandos HTML indicando o início e o fim da página.

<HEAD>: utilizado para inserir parâmetros de configuração e para exibir o título da página.

<TITLE>: usado para mostrar o título do documento. Essa TAG deve estar entre as TAGs <HEAD> e </HEAD>.

<BODY>: é todo o conteúdo da página que será mostrado no navegador ou *browser*.

A estrutura básica de um documento HTML, que pode ser digitada em qualquer editor de texto, é a seguinte:



```
<HTML>
<HEAD>
<TITLE> Meu primeiro HTML</TITLE>
</HEAD>
<BODY>
    HELLO WORLD!!!
</BODY>
</HTML>
```

1.3 CORPO DO DOCUMENTO

Parte da página que contém o conteúdo que será visualizado no navegador e é delimitado pelas TAGs `<BODY>` e `</BODY>`. Possui atributos que permitem definir cor de fundo da página, cor das fontes, links e imagem de fundo.

Os principais atributos para a TAG BODY são: `bgcolor`, `font`, `link`, `alink`, `vlink` e `background`.

Para acessar links da internet, a TAG `<a>` e `` é utilizada.

1.3.1 COR DE FUNDOS

A cor de fundo da página pode ser alterada adicionando-se o comando `bgcolor` na tag BODY. As cores devem ser escritas em inglês, como: preto = `black`; branco = `white`; vermelho = `red`; amarelo = `yellow`; azul = `blue`; rosa = `pink`; etc. A escolha das cores deve ser feita com critério, pois o estilo de uma boa página pode ficar prejudicado se elas não forem bem utilizadas.

```
<HTML>
<HEAD>
<TITLE> Meu primeiro HTML</TITLE>
</HEAD>
<BODY bgcolor = "blue">
    HELLO WORLD!!!
</BODY>
</HTML>
```

1.3.2 COR DAS FONTES

Para alterar a cor das fontes, é necessário adicionar a TAG `` junto ao texto e especificar a cor para as fontes. Lembrando que nessa formatação de cor das fontes também é necessário escrever o nome das cores em inglês.



```
<HTML>
<HEAD>
<TITLE> Meu primeiro HTML </TITLE>
</HEAD>
<BODY bgcolor="blue">
<FONT color=white> HELLO WORLD!!! </FONT>
</BODY>
</HTML>
```

1.3.3 ABRINDO PÁGINAS DA INTERNET

Para abrir uma página da internet, a TAG `<a>` é utilizada. O atributo `target` com o valor `_blank` é utilizado para abrir a página em uma nova aba.

```
<HTML>
<HEAD>
<TITLE> Meu primeiro HTML </TITLE>
</HEAD>
<BODY bgcolor="blue">

<a href="https://www.unibrasil.com.br/" target="_blank"> Unibrasil </a>

</BODY>
</HTML>
```

1.3.4 COR DOS LINKS

A TAG `<LINK>` altera a cor dos links de internet disponibilizados na página.

```
<HTML>
<HEAD>
<TITLE> Meu primeiro HTML </TITLE>
</HEAD>
<BODY bgcolor="blue">

<LINK color=yellow> <a href="https://www.unibrasil.com.br/"> Unibrasil </a> </LINK>

</BODY>
</HTML>
```



1.3.5 COR DOS LINKS ACIONADOS

A TAG <ALINK> modifica a cor dos links de internet acionados na página.

```
<HTML>
<HEAD>
<TITLE> Meu primeiro HTML </TITLE>
</HEAD>
<BODY bgcolor="blue">

<ALINK color=blue> <a href="https://www.unibrasil.com.br/">Unibrasil</a> </ALINK>

</BODY>
</HTML>
```

1.3.6 COR DOS LINKS DEPOIS DE VISITADOS

A TAG <VLINK> altera a cor dos links de internet que já foram visitados pelo usuário.

```
<HTML>
<HEAD>
<TITLE> Meu primeiro HTML </TITLE>
</HEAD>
<BODY bgcolor="blue">

<VLINK color=blue> <a href="https://www.unibrasil.com.br/">Unibrasil</a> </VLINK>

</BODY>
</HTML>
```

Todas as TAGs que modificam a cor dos links podem ser utilizadas em conjunto. A ordem de abertura das TAGs deve ser inversa na hora de fechá-las.

```
<HTML>
<HEAD>
<TITLE> Meu primeiro HTML </TITLE>
</HEAD>
<BODY bgcolor="blue">

<LINK color=yellow>
<ALINK color=blue>
<VLINK color=red>
```



```
<a href="https://www.unibrasil.com.br/" target=_blank> Unibrasil </a>

</VLINK>
</ALINK>
</LINK>

</BODY>
</HTML>
```

Links também podem ser utilizados na TAG <BODY>. Dessa maneira, todos os links existentes na página terão a mesma configuração.

```
<HTML>
<HEAD>
<TITLE> Meu primeiro HTML </TITLE>
</HEAD>
<BODY bgcolor="blue" LINK color=yellow ALINK color=blue VLINK color=red>

<a href="https://www.unibrasil.com.br/" target=_blank> Unibrasil </a>

</BODY>
</HTML>
```

1.3.7 IMAGEM DE FUNDO

O atributo BACKGROUND é utilizado para adicionar uma imagem de fundo na página. A imagem deve estar disponibilizada no mesmo diretório do documento que está sendo utilizado como página principal, senão será necessário informar o caminho onde o arquivo está armazenado. No exemplo a seguir, o arquivo que está sendo utilizado é o “fundo.jpg”.

```
<HTML>
<HEAD>
<TITLE> HTML com imagem de fundo </TITLE>
</HEAD>
<BODY background="fundo.jpg" bgcolor="blue">

<a href="https://www.unibrasil.com.br/" target=_blank> Unibrasil </a>

</BODY>
</HTML>
```

1.3.8 Imagem

Em uma página da internet, é normal a utilização de imagens para ilustrar o conteúdo disponibilizado. A HTML trata de uma maneira bem simples a disponibilização de imagens no documento. Para isso, é utilizada a TAG .

```
<HTML>
<HEAD>
<TITLE> Meu primeiro HTML </TITLE>
</HEAD>
<BODY bgcolor="blue" LINK color=yellow ALINK color=blue VLINK color=red>



</BODY>
</HTML>
```

1.3.9 Comentários

Comentários são utilizados para documentar o desenvolvimento de uma página ou de qualquer sistema de informática. Servem para identificar o responsável pelo desenvolvimento, pela programação, por alterações que por acaso sejam feitas no sistema. Em HTML, comentários podem ser inseridos em qualquer parte da página por meio da TAG <!-- comentário-->

```
<HTML>
<HEAD>
<TITLE> HTML com imagem de fundo </TITLE>
</HEAD>

<!-- COMENTÁRIO: Página desenvolvida por José da Silva em 29/12/2020-->

<BODY background="fundo.jpg" bgcolor="blue">

<a href="https://www.unibrasil.com.br/" target=_blank> Unibrasil </a>

</BODY>
</HTML>
```



REFLITA

Muitas vezes os desenvolvedores se deparam com erros de funcionamento em seus códigos. Para os iniciantes, isso pode ser uma dificuldade a mais. De que maneira esse tipo de problema pode ser facilmente resolvido?

Uma maneira simples de resolver problemas em código é a validação. A W3C disponibiliza uma ferramenta para validar códigos HTML simplesmente copiando e colando o código no seguinte endereço: https://validator.w3.org/#validate_by_input.

2. FORMATAÇÃO DE TEXTO

Para uma perfeita harmonização do conteúdo da página, faz-se necessário a formatação dos textos nela inseridos. Tipo e tamanho das fontes, alinhamento e espaçamento dos parágrafos são itens importantes a serem formatados em uma página. E essa formatação é realizada de forma bem simples na HTML, utilizando TAGs específicas para cada uma das opções citadas.

2.1 FONTE

Os caracteres que descrevem o conteúdo do site são comumente chamados de fontes. As fontes podem ser formatadas quanto ao seu tamanho (size), estilo (face), cor (color), negrito , itálico <I> e sublinhado <U>, entre outros elementos. Para a utilização de caracteres especiais, basta acrescentar ao <head> a tag <meta> com o atributo charset e seu valor conforme o padrão a ser utilizado.

```
<HTML>
<HEAD>

<meta charset="utf-8"/>

<TITLE> Meu primeiro HTML </TITLE>

</HEAD>

<BODY bgcolor="blue">

<!-- Formatação de Fontes:
SIZE: Indica o Tamanho da fonte que pode variar entre 1 e 7
FACE: Indica o TIPO da fonte
COLOR: Indica a Cor da fonte
B: Texto em Negrito
U: Texto Sublinhado
I: Texto em Itálico -->

<FONT size="3" face="arial" color="white"> <B><U><I> UNIBRASIL </I></U></B> </FONT>

</BODY>
</HTML>
```



2.2 PARÁGRAFO

Parágrafo é um conjunto de caracteres que descreve um determinado assunto. Para a HTML, ele pode ser alinhado à esquerda (left), à direita (right) e pode ser centralizado. Para textos com mais do que uma linha, poderá ser utilizado o alinhamento justificado (justify), que alinha tanto a margem esquerda quanto a margem direita.

```
<HTML>
<HEAD>
<TITLE> Meu primeiro HTML </TITLE>
</HEAD>
<BODY bgcolor="blue">

<!-- Formatação de Parágrafos:
RIGHT: Texto alinhado à direita da página
CENTER: Texto centralizado
LEFT: Texto alinhado à esquerda-->

<p align="right"> <B><U><I> UNIBRASIL </I></U></B> </p>
<p align="center"> <B><U><I> UNIBRASIL </I></U></B> </p>
<p align="left"> <B><U><I> UNIBRASIL </I></U></B> </p>

</BODY>
</HTML>
```

2.3 CABEÇALHO

Os cabeçalhos são utilizados para identificar os títulos e subtítulos de uma página.

```
<HTML>
<HEAD>
<meta charset="utf-8"/>
<TITLE> Meu primeiro HTML </TITLE>
</HEAD>
<BODY bgcolor="blue">

<!-- Cabeçalhos: Podem ter nível de 1 a 6 -->

<h1> Cabeçalho Nível 1 </h1>
<h2> Cabeçalho Nível 2 </h2>
<h3> Cabeçalho Nível 3 </h3>
<h4> Cabeçalho Nível 4 </h4>
<h5> Cabeçalho Nível 5 </h5>
<h6> Cabeçalho Nível 6 </h6>

</BODY>
</HTML>
```

2.4 QUEBRA DE LINHA

Para textos que precisam estar em linhas diferentes, existe a necessidade de inserir uma quebra de linha. Para isso, é só adicionar a TAG
 entre os dois textos.

```
<HTML>
<HEAD>
<meta charset="utf-8"/>
<TITLE> Meu primeiro HTML </TITLE>
</HEAD>
<BODY bgcolor="blue">

<FONT size="3" face="arial" color="white"> UNIBRASIL </FONT>
<BR>
<FONT size="2" face="arial" color="white"> CURITIBA- PR </FONT>

</BODY>
</HTML>
```

2.5 TABELA

As tabelas são elementos importantes para a criação e organização do *layout* de uma página. Por meio das tabelas é possível distribuir de uma maneira mais fácil os elementos da página.

A utilização da tabela se dá com qualquer elemento da página, não necessariamente apenas com valores tabulados, como a relação de funcionários e seus dados. Pode-se, por exemplo, inserir o logotipo da empresa na primeira linha e coluna de uma tabela, o endereço da empresa em outra coluna, e assim por diante. Nas células de uma tabela, que é a intersecção de uma linha e uma coluna, é possível inserir qualquer elemento suportado pela HTML.

A seguir estão os elementos utilizados para a criação e formatação de tabelas:

<TABLE>: Identifica o início da tabela;

<TR>: Indica uma linha na tabela;

<TD>: Indica uma coluna na tabela;

WIDTH: Tamanho da tabela;

BORDER: Tipo da borda da tabela;

ALIGN: Alinhamento da tabela;

BORDERCOLOR: Cor das bordas da tabela.



```
<HTML>
<HEAD>
<meta charset="utf-8"/>
<TITLE> TABELAS </TITLE>
</HEAD>
<BODY bgcolor="blue">

<BR>

<TABLE WIDTH="500" BORDER="1" ALIGN="CENTER" BORDERCOLOR="WHITE">
<TR>
<TD><FONT size="3" face="arial" color="white"> UNIBRASIL (L1C1) </FONT></TD>
<TD><FONT size="2" face="arial" color="white"> CURITIBA- PR (L1C2) </FONT></TD>
</TR>

<TR>
<TD><FONT size="3" face="arial" color="white"> UNIBRASIL (L2C1) </FONT></TD>
<TD><FONT size="2" face="arial" color="white"> CURITIBA- PR (L2C2) </FONT></TD>
</TR>

</TABLE>
</BODY>
</HTML>
```



LEITURA

Para complementar o conhecimento sobre HTML, é interessante ler a respeito do HTML5, que incorporou novas soluções à linguagem, facilitando e aprimorando o desenvolvimento de páginas WEB. Conteúdos sobre HTML5 são facilmente encontrados na internet, porém no link a seguir pode-se acessar um artigo resumido e bem explicativo sobre a versão 5 da HTML:

TECMUNDO. O que é HTML5? 16 jun. 2009. Disponível em: <https://bit.ly/31y9cSB>. Acesso em: fev. 2021.

CONSIDERAÇÕES FINAIS

Esta Unidade teve como objetivo mostrar as características básicas da linguagem de formatação HTML e sua relevância no desenvolvimento de conteúdo para a internet.

Foram mostrados a estrutura básica de um documento HTML e o funcionamento das TAGs. Na sequência, trabalhou-se a formatação do corpo do documento e sua estrutura de formatação de cor de fundo, cor das fontes, *links*, imagens e imagens de fundo, além dos comentários durante o desenvolvimento de uma página ou sistema, que são muito importantes para a manutenção futura da estrutura. Em formatação de texto, foi visto como alterar as fontes, parágrafos, cabeçalho, quebra de linha e tabelas. Esses itens são básicos e compõem o conteúdo de uma página ou sistema WEB.

Na próxima Unidade, será visto o CSS, que é uma lista de formatação para melhorar a apresentação de um documento HTML, responsável por muitos dos *layouts* presentes nas páginas.



ANOTAÇÕES



UNIDADE

02

CSS



OBJETIVOS DE APRENDIZAGEM

- » Compreender e aplicar Folhas de Estilo em Cascata (CSS)
- » Criar folha de estilo do tipo interna ou externa
- » Formatar tipos de folha de estilo
- » Personalizar TAGs e atalhos



VÍDEOS DA UNIDADE



<https://bit.ly/3insixW>



<https://bit.ly/3xq130l>



<https://bit.ly/3guGYQU>



<https://bit.ly/35wTAAJ>



<https://bit.ly/2SwOUIj>



<https://bit.ly/3cP2yR4>



<https://bit.ly/3gsLRtD>

INTRODUÇÃO

Cascading Style Sheets (CSS), ou Folha de Estilo em Cascata, é um estilo de desenvolvimento de *sites* muito interessante, pois, a partir de uma lista de configurações, é possível melhorar o visual e facilitar a criação e a manutenção das páginas. Além disso, é possível diminuir o tamanho do código da página e aumentar a velocidade de carregamento. Com a diminuição do código, há mais controle sobre o *layout* do documento.

Os estilos criados são definidos como uma regra CSS. Cada regra criada utiliza a seguinte sintaxe:

Elemento {atributo1: valor; atributo2: valor; ...; atributoN: valor}

Onde:

- **Elemento** - é nome pelo qual será chamado o estilo a ser aplicado no conteúdo da página. A partir da definição do elemento, ele será utilizado no corpo (*body*) da página;
- **Atributo** – são as configurações utilizadas pelo elemento, como cor, tamanho, alinhamento, tipo de fonte e outros itens que podem ser aplicados à página;
- **Valor** – é a configuração aplicada ao atributo, como blue, 12, center, arial. Se o valor for uma palavra composta, deverá estar entre aspas duplas ou simples.

A criação dos elementos é feita nas TAGs `<style>` e `</style>`, e a utilização deles será nas TAGs `<body>` e `</body>`, como pode ser visto no exemplo a seguir:

```
<html>
  <head>
    <meta charset="utf-8"/>
    <TITLE> CSS </TITLE>
    <style>
      h1 {color: blue; text-align: center; font-family: arial; font-size: 20}
    </style>
  </head>
  <body>

    <h1> Formatação utilizando CSS </h1>

  </body>
</html>
```

A linguagem HTML possui controles nativos para exibir botões em páginas WEB, porém os botões têm estilos originais que dependem do browser e do sistema operacional. Para controlar o estilo desses botões, utiliza-se o CSS e define-se cores de plano de fundo, dimensões, fonte, entre outras características. Mas, mesmo com CSS, essa customização pode se tornar repetitiva e trabalhosa, principalmente para desenvolvedores com pouca experiência. Para facilitar a criação de botões personalizados com CSS, as ferramentas conhecidas como CSS Button Generators permitem escolher várias características dos botões em uma interface gráfica, necessitando-se apenas copiar o código CSS para a aplicação. E o melhor de tudo: são gratuitas e estão disponíveis on-line.

1. CRIAÇÃO DE ESTILOS

Os estilos são as formatações aplicadas aos objetos de uma página. Esses estilos podem ser criados em uma folha de estilo. Essas folhas, por sua vez, podem ser inseridas em uma página de três maneiras: localmente (Unidade I), em uma folha de estilo interna e em uma folha de estilo externa.

1.1 FOLHA DE ESTILO INTERNA

Uma folha de estilo interna é indicada quando uma página tem um estilo único. A criação de estilos internos é feita na seção head, com a tag <style>.

```
<head>
<style type="text/css">
body {background-color: blue;}
</style>
</head>
```

1.2 FOLHA DE ESTILO EXTERNA

A folha de estilo externa é a declaração de CSS mais utilizada. Com uma folha de estilo externa é possível concentrar todas as configurações de aparência de todo o *site* em um local separado. Basicamente ela consiste em colocar todo o código CSS em um arquivo externo (.css) e, no HTML, é feita a chamada desse arquivo para que o código de estilo seja utilizado.

A página deve ser vinculada à folha de estilo usando a tag <link>. A tag <link> vai dentro da seção head, conforme exemplo a seguir:



```
<head>
<link rel="stylesheet" type="text/css" href="Folha_de_Estilos.css">
</head>
```

Para esse exemplo, é necessário criar um arquivo e chamá-lo de Folha_de_Estilos.css, de preferência no mesmo diretório onde está o arquivo HTML. Senão, será necessário colocar o caminho onde o arquivo está armazenado.

```
body {
    background-color: blue;
}

h1 {
    color: white;
    text-align: center;
}

p {
    font-family: arial;
    font-size: 20;
}
```

No código HTML, deve-se referenciar o arquivo de folha de estilo criado:

```
<html>
<head>
<meta charset="utf-8"/>
<TITLE> CSS </TITLE>

<link rel="stylesheet" type="text/css" href="Folha_de_Estilos.css">

</head>

<body>

<h1> Formatação utilizando CSS- Folha de Estilo Externa </h1>
<br>
<p> Conteúdo da página </p>

</body>
</html>
```



REFLITA

O que realmente é CSS?

Assim como o HTML, o CSS não é uma linguagem de programação. Também não é uma linguagem de marcação, como o HTML. CSS é uma linguagem de folhas de estilos. Isso significa que o CSS permite aplicar estilos seletivamente a elementos nos documentos HTML.

Portanto, CSS é utilizado para definir o estilo de um documento HTML e descreve como os elementos HTML devem ser exibidos na página WEB.

2. FORMATAÇÃO DE TEXTO

Para uma boa harmonização de uma página, é necessária a formatação dos textos, quanto ao tipo e tamanho das fontes, alinhamento e espaçamento dos parágrafos, cores de fundo e dos cabeçalhos. Com a utilização do CSS, esse trabalho é facilitado, pois com a definição das formatações é necessário *setar* somente esses valores uma única vez, a partir dos estilos definidos na TAG <style>.

2.1 BODY

Todas as formatações da TAG <body> podem ser criadas em um único estilo. O objetivo é simplificar a criação e, principalmente, facilitar a manutenção e modificação das formatações conforme surjam necessidades durante o tempo de vida da página.

- background-color: cor de fundo;
- color: cor das fontes.

```
<html>
<head>
<meta charset="utf-8"/>
<TITLE> CSS </TITLE>
<style>
body { background-color: blue; color: red}
</style>
</head>
<body>
Formatação utilizando CSS
</body>
</html>
```



2.2 FONTE

Os caracteres que descrevem o conteúdo do *site* são comumente chamados de fontes. As fontes podem ser formatadas quanto ao seu tamanho (size), estilo (face), cor (color), entre outros elementos. O CSS possui oito propriedades de formatação das fontes:

- color: cor;
- font-family: tipo;
- font-size: tamanho;
- font-style: estilo;
- font-variant: fontes maiúsculas mais baixas;
- font-weight: espessura da fonte;
- font-stretch: fonte mais estreita ou larga;
- font: maneira abreviada para declarar todas as propriedades anteriores (exceto cor).

```
<html>
<head>
<meta charset="utf-8"/>
<TITLE> CSS </TITLE>

<style>

body { background-color: green; color: red}
font { color: white; font-family: arial; font-size: 20; }

</style>

</head>

<body>

Formatação utilizando CSS
<br>
<font> Formatação fonte utilizando CSS </font>

</body>
</html>
```

2.3 PARÁGRAFO

Um parágrafo pode ser alinhado à esquerda (left), à direita (right) e pode ser centralizado. Para textos com mais do que uma linha, poderá ser utilizado o alinhamento justificado (justify), que alinha tanto a margem esquerda quanto a direita. Os parágrafos também podem ser identados, que é o espaço na primeira linha que facilita a identificação do início do parágrafo.



- text-align: alinhamento do texto (left, right, center ou justify);
- margin-top: margem superior;
- margin-bottom: margem inferior;
- text-indent: identação do parágrafo.

```
<html>
<head>
<meta charset="utf-8"/>
<TITLE> CSS </TITLE>

<style>

body { background-color: green; color: red}
font { color: white; font-family: arial; font-size: 20; }
p { text-align: center }

</style>

</head>

<body>

Formatação utilizando CSS
<br>
<font> Formatação fonte CSS </font>
<br>
<p> Formatação parágrafo CSS </p>

</body>
</html>
```

2.4 CABEÇALHO

Os cabeçalhos são utilizados para identificar os títulos e subtítulos de uma página. Com CSS é possível formatar as seguintes características de um cabeçalho:

- display: modo de exibição dos elementos, block ou inline;
- font-size: tamanho da fonte;
- margin-top: margem superior;
- margin-bottom: margem inferior;



- margin-left: margem esquerda;
- margin-right: margem direita;
- font-weight: espessura da fonte;

```
<html>
<head>
<meta charset="utf-8"/>
<TITLE> CSS </TITLE>

<style>
body { background-color: green; color: red}
font { color: white; font-family: arial; font-size: 20; }
h1 { color: black; text-align: center; font-weight: 10}
p { text-align: center }
</style>

</head>

<body>

<h1> Formatação cabeçalho CSS </h1>
<br>
<font> Formatação fonte CSS </font>
<br>
<p> Formatação parágrafo CSS </p>

</body>
</html>
```

2.5 AGRUPAMENTO DE SELETORES

Existem elementos que têm o mesmo estilo na folha de estilos. Para evitar a repetição desses estilos, é possível usar o seletor de pacotes, separando os elementos por vírgulas.

Exemplo de elementos com as mesmas características na folha de estilos:

```
<style>
font {color: green;}
h1 {color: green;}
p {color: green;}
</style>
```



Para economizar tempo e repetição dos mesmos valores, pode-se utilizar o agrupamento de seletores:

```
<style>
font, h1, p {color: green;}
</style>
```

2.6 COMPLEMENTO DOS SELETORES

Quando se utiliza o agrupamento de seletores para os valores iguais, muitas vezes há a necessidade de atribuir novos itens aos seletores. Para isso, é possível complementar os seletores adicionando novos itens na folha de estilo.

```
<style>
font, h1, p {color: green;}
p {text-align: center}
</style>
```

2.7 COMENTÁRIOS

Comentários são utilizados para explicar o código e auxiliar na manutenção do código fonte. Os comentários não são interpretados pelos *browsers*.

Um comentário CSS é escrito dentro do elemento `<style>`, sendo iniciado com `/*` e finalizado com `*/`.

```
<style>

/* COMENTÁRIOS NO CSS - AGRUPAMENTO DE ELEMENTOS */

font, p {color: red; font-family: arial; font-size: 20; }

p {text-align: center} /*COMENTÁRIOS NO CSS- COMPLEMENTO DOS SELETORES*/

</style>
```

2.8 SELETOR CLASSE

Não há restrição quanto à criação de regras de estilo para aplicar a qualquer elemento HTML, complementando as regras existentes. O objetivo é criar uma regra que tenha, por exemplo, uma



cor padrão utilizada na página. Toda vez que for necessário usar essa cor, é só referenciar a classe criada. O seletor classe pode ser utilizado mais de uma vez dentro do documento.

```
<html>
<head>

<meta charset="utf-8"/>
<TITLE> CSS </TITLE>

<style>
.cor { color: red;}
font, p { font-family: arial; font-size: 20; }
p { text-align: center }
</style>

</head>

<body>

<font class="cor"> Formatação fonte CSS </font>
<br>
<p class="cor"> Formatação parágrafo CSS </p>

</body>

</html>
```

3. TAGS PERSONALIZADAS

Com as classes de estilo se consegue criar variações de uma TAG. Pode-se, por exemplo, criar um estilo de parágrafo com o texto alinhado à esquerda, um estilo com texto alinhado à direita e outros alinhamentos, tendo, assim, vários temas da TAG <p>.

Sintaxe de criação de TAGs personalizadas:

```
<style>

elemento.nomedaClasse {atributo:valor; ... }

</style>
```



Para criar uma TAG personalizada, é preciso escolher o elemento ao qual ela será aplicada e o nome da classe, o nome do atributo e o seu valor. Para utilizar a classe, é só digitar classe="Nome da Classe" juntamente com o elemento para o qual ela foi criada.

Exemplo de criação e utilização de uma TAG personalizada:

```
<html>
<head>
<meta charset="utf-8"/>
<TITLE> CSS </TITLE>
<style>

p.c1 {color: green; text-align: left; }
p.c2 {color: red; text-align: center; }
p.c3 {color: black; text-align: right; }

</style>
</head>

<body>

<br>
<p class="c1"> Formatação parágrafo CSS </p>
<br>
<p class="c2"> Formatação parágrafo CSS </p>
<br>
<p class="c3"> Formatação parágrafo CSS </p>

</body>
</html>
```

4. ATALHOS DE CSS

Alguns atributos de CSS podem ter diversos valores em uma declaração. Pode-se, por exemplo, definir várias formatações para as fontes de um parágrafo, como pode ser visto a seguir:

```
P {
font-family: "arial";
font-size: 5;
font-style: italic;
font-variant: small-caps ;
font-weight: bold;
font-stretch: condensed }
```

Nesse caso, é possível especificar todas essas formatações utilizando o atributo Font, definindo todas de uma só vez:

```
P { Font : "arial" 5 italic small-caps bold condensed }
```

Pode-se verificar a simplicidade do código utilizando os atalhos para a declaração dos valores, onde eles são separados apenas por espaço em branco, poupando espaço e tempo de desenvolvimento. O atalho Font só não pode ser utilizado com cores da fonte.



VÍDEO

Apresentação sobre CSS no workshop “Tecnologias Web e Publicações Digitais”, no Google Campus, em São Paulo. Esse workshop foi promovido por ceweb.br, nic.br e W3C do Brasil, e pode ser assistido em: <https://bit.ly/3fwWewj>. Acesso em: fev. 2021.

CONSIDERAÇÕES FINAIS

A Unidade II apresentou a CSS ou Folha de Estilo em Cascata, que é um estilo de desenvolvimento de *sites* muito interessante, pois a partir de uma lista de configurações, é possível melhorar o visual, facilitar a criação e a manutenção das páginas e sistemas WEB. Sua principal característica é manter os comandos de formatação separados dos comandos HTML. Foram mostradas a criação de folhas de estilos, tanto internas como externas, além das características de formatação de fontes, parágrafos e cabeçalhos. Foi possível ver como funciona o agrupamento e o complemento dos seletores e a inclusão de comentários nas folhas de estilo.

A próxima Unidade tratará da linguagem de programação JavaScript, que permite incluir regras de negócio em uma estrutura HTML. Com JavaScript também é possível controlar o fluxo de execução dos algoritmos das regras de negócio por meio das estruturas de decisão e repetição.



ANOTAÇÕES

UNIDADE

03

JAVASCRIPT - LÓGICA



OBJETIVOS DE APRENDIZAGEM

- » Conceituar JavaScript e adicionar interatividade à HTML
- » Criar scripts em JavaScript
- » Entender lógica de programação, tomada de decisões e laços de repetição
- » Implementar scripts com lógica de programação e operadores lógicos E e OU



VÍDEOS DA UNIDADE



<https://bit.ly/35tyH9w>



<https://bit.ly/3vxLWkf>



<https://bit.ly/3q3D0SA>



<https://bit.ly/3wxuspr>



<https://bit.ly/3vBKsoW>



<https://bit.ly/35wuubQ>



<https://bit.ly/3wAho2I>



INTRODUÇÃO

JavaScript é uma linguagem de programação utilizada para adicionar interatividade a uma página HTML. Essa interatividade pode ser vista nas respostas quando botões são pressionados, na aplicação de animações, na inserção de dados em formulários e na criação de um estilo dinâmico e responsivo da página.

Os *scripts* em JavaScript serão incluídos na página HTML dentro da TAG <script>. A TAG poderá ser inserida dentro da seção <head> ou da seção <body>, como pode ser visto no exemplo a seguir:

```
<html>
  <head>
    <meta charset="utf-8"/>
    <TITLE> JavaScript </TITLE>
    <style>
      body {background-color: blue; }
      h1 {color: white; }
      p {font-family: arial; font-size: 20; text-align: right; }
    </style>
  </head>

  <body>
    <h1><p>
      <script language="javascript" type="text/javascript">

        document.write('Hello World')

      </script>
    </p> </h1>
  </body>
</html>
```

JavaScript pode interagir com quase todos os elementos de uma página HTML. Trabalha com variáveis, gera relatórios, modifica elementos sem a necessidade de recarregar constantemente a página. Em conjunto com HTML e CSS, o JavaScript, forma uma das principais tecnologias da internet. Com ele é possível a criação de páginas da *web* interativas, sendo utilizado pela maioria dos *sites*, e os principais navegadores têm um mecanismo JavaScript dedicado para executá-lo.

JavaScript®, que pode ser abreviado como JS, é uma linguagem leve, interpretada e baseada em objetos com funções de primeira classe. Conhecida por ser uma linguagem de scripts para páginas web, pode ser utilizada em ambientes sem browser, tais como node.js, Apache CouchDB e Adobe Acrobat. O JavaScript é uma linguagem baseada em protótipos, multiparadigma e dinâmica, suportando estilos de orientação a objetos, imperativos e declarativos.

1. CARACTERÍSTICAS BÁSICAS DO JAVASCRIPT

Algumas características do Javascript são interessantes na utilização da linguagem:

- Roda em qualquer plataforma;
- Tem prioridade sobre o HTML e o CSS;
- Possibilita criar funcionalidades que não são possíveis com HTML e CSS;
- Possui sintaxe similar ao Java, C e C++;
- Tem modelo de objetos baseado em protótipos e não em classes, como no Java, por exemplo;
- Seu código é inserido no arquivo HTML utilizando a tag <script>;
- Suporta a utilização de variáveis sem necessidade de declaração;
- É dirigido a eventos. Um clique em um botão, por exemplo, é um evento utilizado pelo JavaScript.

1.1 EVENTOS

Eventos são ações que o usuário executa quando visita uma página. Movimentar o *mouse* sobre uma imagem ou clicar em um botão são exemplos de eventos. O JavaScript controla os eventos por meio de comandos conhecidos como *handlers de eventos*.

Existem 14 *handlers* de eventos no JavaScript:

GESTO DE EVENTO	AÇÃO
onAbort	Encerra o carregamento da página.
onBlur	Foco retirado do objeto.
onChange	Troca de valor do elemento de um formulário.
onClick	Clique sobre um elemento de um formulário.
onDbclick	Duplo clique sobre um elemento de um formulário.
onError	Erro ao carregar um objeto.
onFocus	Foco atribuído a um elemento.
onLoad	Página carregada.



onMouseover	Cursor do <i>mouse</i> passando sobre um <i>link</i> .
onMouseout	<i>Mouse</i> afastado do objeto.
onmousedown	Botão do <i>mouse</i> pressionado.
onSelect	Seleção de um objeto.
onSubmit	Formulário enviado.
onUnload	Usuário sai da página

1.2 TIPOS DE VARIÁVEIS

Variáveis são elementos básicos na programação, nas quais são atribuídos valores para descrever um objeto tratado no programa. Por exemplo, um nome de produto: NomeProduto. E essa variável receberá um valor a ela atribuído: NomeProduto = ‘computador’. Existem tipos diferentes de valores que as variáveis suportam em JavaScript, como pode ser visto a seguir:

TIPO DE VALORES	DESCRIÇÃO
String	Caracteres.
Number	Números.
Boolean	Verdadeiro ou Falso.
Null	Vazio.
Function	Valor de retorno de uma função.
Object	Valor associado ao objeto.

O nome das variáveis no JavaScript não pode conter espaços em branco. Se for necessário, poderá ser usado o *underline*. Exemplo: Nome_Produto. Outra característica do JavaScript em relação às variáveis é que ele é *case sensitive*, ou seja, faz distinção entre maiúsculas e minúsculas. Exemplo: Nome_Produto é diferente de nome_producto.

1.3 OPERADORES

Operadores são símbolos utilizados para operações aritméticas com variáveis numéricas. Os operadores podem ser utilizados também com variáveis do tipo *string*, porém somente é utilizado o operador + que fará a concatenação de dois conjuntos de *strings*. A seguir, é apresentado o conjunto de operadores suportados pelo JavaScript:

OPERADOR	FUNÇÃO
x + y (string)	Concatena x e y.
x + y (number)	Soma x e y.
x - y	Subtrai x e y.
x * y	Multiplica x e y.
x / y	Divide x e y.
x % y	Módulo de x por y.
x ++	Soma 1 a x.
++ x	Soma 1 a x.

x--	Subtrai 1 de x.
-- x	Subtrai 1 de x.
- x	Multiplica x por -1.

1.4 ATRIBUIÇÕES

Atribuições são valores setados para as variáveis. Com exceção do sinal de igualdade, as outras atribuições funcionam como atalhos para a atribuição de valores às variáveis. Deve ser tomado um certo cuidado em relação a isso, pois em algumas situações pode haver confusão. Este material adota a notação longa, para ficar mais claro qual atribuição está sendo realizada.

ATRIBUIÇÃO	FUNÇÃO
x = y	x recebe y.
x += y	x recebe x + y.
x -= y	x recebe x - y.
x *= y	x recebe x * y.
x /= y	x recebe x / y.
x %= y	x recebe x % y.

1.5 COMPARAÇÕES

Em programação, é comum comparar o valor de variáveis com o valor de outras variáveis e de variáveis com valores fixos. Para isso, o JavaScript disponibiliza várias maneiras de comparar valores. O retorno de uma comparação sempre será falso ou verdadeiro. As comparações são utilizadas com estruturas de tomada de decisão e laços de repetição, que são situações utilizadas com frequência em *scripts* de programação, como pode ser visto nos exemplos a seguir:

- **Estrutura de Tomada de Decisão:**

Se **Valor > 10** então Valor = Valor – Valor * 10% FimSe

- **Laço de Repetição:**

Enquanto **Valor < 100** faça Valor = Valor + 1 FimEnquanto

COMPARAÇÃO	FUNÇÃO
x == y	Retorna Verdadeiro se x for igual a y.
x != y	Retorna Verdadeiro se x for diferente de y.
x > y	Retorna Verdadeiro se x for maior que y.
x >= y	Retorna Verdadeiro se x for maior igual a y.
x < y	Retorna Verdadeiro se x for menor que y.
x <= y	Retorna Verdadeiro se x for menor igual a y.
x && y	Retorna Verdadeiro se x E y forem verdadeiros.
x y	Retorna Verdadeiro se x OU y forem verdadeiros.
! x	Retorna Verdadeiro se x for falso.



REFLITA

Javascript é a mesma coisa que **Java**?

Definitivamente não. Java é uma linguagem de programação e JavaScript se apresenta como uma linguagem de “scripting”, executada e não compilada como é o caso do Java. As Linguagens executadas funcionam sob uma plataforma, neste caso os browsers, sua ação é realizada pelo navegador e depende dele para rodar, diferente de uma linguagem compilada que gera um código executável.



2. CRIAÇÃO DE SCRIPTS

Scripts são a essência do JavaScript e podem ser inseridos nas TAGs <head> e <body> em uma página HTML. E, para delimitar os *scripts*, eles devem estar inseridos entre as TAGs <script>, como pode ser visto no exemplo a seguir, que mostra na tela a mensagem “Hello World”:

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

<TITLE> JavaScript </TITLE>

<style>
body {background-color: blue; }

h1      {color: white;}

p       {font-family: arial; font-size: 20; text-align: right; }

</style>

</head>

<body>
<p>

<script language="javascript" type="text/javascript">

    document.write('Hello World')

</script>

</p>
</body>
</html>
```

2.1 COMENTÁRIOS

Comentários são importantes para documentar pontos importantes em programação. Devem ser inseridos durante o desenvolvimento e sempre que qualquer item for inserido ou alterado após a implementação de qualquer *software* ou página de internet. Durante o desenvolvimento, muitas vezes, os comentários podem parecer redundantes, mas eles se mostram muito importantes principalmente na manutenção do *software* ou da página. Outra situação é auxiliar outros desenvolvedores quando eles forem fazer alterações em programas de terceiros. O JavaScript tem uma notação diferente do HTML para os comentários. Para comentários de uma linha só, é utilizando “//Comentário” e, para comentários de várias linhas, utiliza-se “/* Comentário */”, como pode ser visto no exemplo a seguir:

```
<html>
<head>
<meta charset="utf-8"/>
<TITLE> JavaScript </TITLE>
<style>
body {background-color: blue; }
h1      {color: white; }
p       {font-family: arial; font-size: 20; text-align: right; }
</style>

</head>

<body>
<h1><p>

<script language="javascript" type="text/javascript">

/* Este é um comentário de múltiplas linhas */
document.write('Hello World')

// Este é um comentário de apenas uma linha

</script>
</p></h1>

</body>
</html>
```



2.2 ALERTAS

As caixas de diálogo de alerta auxiliam na comunicação com o usuário. Isso pode ajudar quem navega na página a utilizar os recursos de uma forma mais efetiva ou, ainda, quem tem problemas para encontrar meios de resolvê-los. No exemplo a seguir, foi inserido um alerta para a mensagem “Hello World”.

```
<html>
  <head>
    <meta charset="utf-8"/>
    <TITLE> JavaScript </TITLE>

    <style>
      body {background-color: blue; }
      h1 {color: white; }
      p {font-family: arial; font-size: 20; text-align: right; }
    </style>

  </head>

  <body>
    <script language="javascript" type="text/javascript">

      /* Este é um comentário de múltiplas linhas */

      alert('Hello World')

      // Este é um comentário de apenas uma linha

    </script>

    <noscript>
      <h2>Esta página requer JavaScript</h2>
    </noscript>

  </body>
</html>
```

O Elemento HTML `<noscript>`, inserido no final do exemplo, define uma seção de HTML a ser inserida se um tipo de `script` não é suportado pela página ou se o `script` está desativado no navegador.



2.3 CONFIRMAÇÕES

Confirmações servem para receber informações dos usuários quanto a situações que necessitam de escolhas que ele deve fazer. O exemplo a seguir mostra uma pergunta feita ao usuário: se ele quer sair do sistema ou de uma página de compra. Nesse *script* é apresentado o conceito de condicional, que será vista de maneira mais aprofundada na sequência do material. Condicional permite direcionar o sistema para ações distintas. Se o resultado dela for verdadeiro, será realizada a ação da primeira parte, ao passo que, se o resultado for falso, a ação da segunda parte é que será realizada.

```
<html>
<head>
<meta charset="utf-8"/>
<TITLE> JavaScript </TITLE>
<style>
body {background-color: blue; }
h1      {color: white; }
p        {font-family: arial; font-size: 20; text-align: right; }
</style>
</head>

<body>
<h1><p>
<script language="javascript" type="text/javascript">

if (confirm("Tem certeza que deseja sair?"))
{
    // Ação se o resultado da condicional for verdadeiro
    alert("Você clicou em OK")
}
else
{
    // Ação se o resultado da condicional for falso

    alert("Você clicou em Cancelar")
}

</script>
</p></h1>
</body>
</html>
```



2.4 AVISOS

Avisos servem para o sistema interagir com o usuário. Pode ser feita uma pergunta padrão, a qual será respondida ou não pelo usuário. O método prompt possui dois parâmetros: a pergunta que será feita ao usuário e a resposta padrão. No exemplo a seguir, está sendo feita uma pesquisa com os usuários para verificar se eles preferem praia ou montanha. E como resposta padrão está sendo passado "", que significa uma resposta em branco. O método retorna a resposta do usuário ou nulo, quando o usuário clica em CANCELAR ou pressiona OK sem preencher a resposta.

```
<html>
<head>
<meta charset="utf-8"/>
<TITLE> JavaScript </TITLE>
<style>
body {background-color: blue; }
h1      {color: white; }
p       {font-family: arial; font-size: 20; text-align: right; }
</style>
</head>

<body>
<h1><p>

<script language="javascript" type="text/javascript">

pergunta = prompt ("Praia ou Montanha", "")

if (pergunta)
{
  alert("Sua escolha foi " + pergunta)
}
else
{
  alert("Sem Resposta")
}

</script>
</p></h1>
</body>
</html>
```

3. LÓGICA DE PROGRAMAÇÃO

A principal característica para a resolução de problemas é a capacidade de pensar de maneira lógica. E a lógica de programação está ligada diretamente a essa capacidade. Outra situação importante relacionada à lógica de programação é a divisão da resolução desses problemas em partes menores.

É nesse ponto que entra o conceito de algoritmo, descrito como uma sequência lógica de ações capaz de resolver um problema. Em desenvolvimento de sistemas computacionais, a soma de dois números é o exemplo mais clássico de um algoritmo. Para a resolução de um sistema de processamento de dados, é necessária a realização de três fases: entrada de dados, processamento e saída de informações. Para a entrada de dados, é preciso informar quais são os dois números que serão somados. No processamento de dados, é feita a operação aritmética de soma e, por fim, há a necessidade de mostrar o resultado da operação realizada com a saída das informações, que, nesse caso, é a soma dos dois números.

No exemplo a seguir, podem ser observadas as etapas para a soma de dois números:

- **Entrada de Dados:**

Informar primeiro número.

Informar segundo número.

- **Processamento de Dados:**

Somar os dois números.

- **Saída de Informações:**

Mostrar o resultado da soma.

Um algoritmo pode ser entendido como um *script* de resolução de problemas. A seguir, pode ser visto o *script* para a soma dos dois números utilizando uma notação de uma pseudolínguagem de programação:

```
Início Algoritmo  
  
    //Entrada de Dados  
    Ler (Num1);  
    Ler (Num2);  
  
    //Processamento de Dados  
    Soma := Num1 + Num2;  
  
    //Saída de Informações  
    Escrever (Soma)  
  
Fim Algoritmo
```



Não há a necessidade de fazer todos os *scripts* utilizando uma notação de pseudocódigo, porém, no início profissional de um desenvolvedor, muitas vezes isso pode ajudar. A seguir o pseudocódigo é transformado em JavaScript, onde é possível notar uma complexidade bem maior:

```
<html>
<head>
<meta charset="utf-8"/>
<TITLE> JavaScript </TITLE>

<style>
body {background-color: blue; }
h1      {color: white;text-align: center;}
p       {font-family: arial; font-size: 20; text-align: left; }
</style>

</head>

<body>
<script language="javascript" type="text/javascript">

function somar() {
    var num1 = document.querySelector(".num1").value;
    var num2 = document.querySelector(".num2").value;

    var resultado = parseInt(num1) + parseInt(num2);
    document.querySelector(".resultado").innerHTML = resultado;
}

</script>

<section>
<div>
    <h1>Soma de Valores</h1>
    Número 1:
    <input class="num1" type="number"><br>
    Número 2:
    <input class="num2" type="number"><br>
    <br>
    <button onclick="somar()">Total</button>
    <p class="resultado"></p>
</div>
</section>
</body>
</html>
```

3.1 TOMADA DE DECISÃO

Em qualquer linguagem de programação, o código precisa tomar decisões e realizar ações à medida que situações específicas aconteçam durante a sua execução. Por exemplo, em um sistema de vendas, se o valor de vendas é maior que 1000, então deve ser concedido um desconto de 5%, senão deve ser concedido um desconto de apenas 1%. É possível ver que, na explicação do sistema de vendas, foi utilizada uma linguagem natural, porém é possível perceber algumas palavras que se caracterizam na tomada de decisão e são conhecidas como condicionais. Essas palavras são: **se**, **então** e **senão**. Dentro dessa declaração condicional, tem-se a seguinte situação: SE a comparação for verdadeira, ENTÃO faça a primeira parte dos comandos, SENÃO faça a segunda parte dos comandos. Em uma pseudolinguagem, ficaria com a seguinte notação:

SE condição

Comandos /* Quando a condição for verdadeira */

SENÃO

Comandos /* Quando a condição for falsa */

No JavaScript, o pseudocódigo terá a seguinte sintaxe:

```
IF (condicao)
{
    Comandos
}
ELSE
{
    Comandos
}
```

No JavaScript o código pode ser representado como exposto a seguir, levando em consideração o sistema de vendas e o desconto que deve ser aplicado quando o valor de vendas for maior que 1000, que é de 5%. E, para as vendas menores, deve ser aplicado um desconto de 1%:

```
var Vendas === 5000;

if (Vendas > 1000) {
    Vendas = Vendas * 0,95;
} else {
    Vendas = Vendas * 0,99;
}
```



Agora, segue o código completo no JavaScript:

```
<html>
<head>
<meta charset="utf-8"/>
<TITLE> JavaScript </TITLE>
<style>
body {background-color: blue; }
h1      {color: white;}
p       {font-family: arial; font-size: 20; text-align: center; }
</style>

</head>

<body>
<h1>
<p>
<script language="javascript" type="text/javascript">

let Vendas = 500;

if (Vendas > 1000)
{
    Vendas = Vendas * 0.95; //Cálculo do desconto de 5%
}
else
{
    Vendas = Vendas * 0.99; //Cálculo do desconto de 1%
}

document.write(Vendas)

</script>
</p>
</h1>
</body>
</html>
```

Os operadores de comparação são usados para testar as condições dentro das declarações condicionais:

OPERADOR	FUNÇÃO
====	Igual
!==	Diferente
>	Maior que
<	Menor que
>=	Maior igual
<=	Menor igual



3.2 LAÇOS DE REPETIÇÃO

Para executar tarefas que se repetem, as linguagens de programação possuem estruturas que podem ser utilizadas em trechos do programa por um determinado número de vezes. Elas são as estruturas de repetição.

Essas estruturas de repetição são apresentadas em três tipos: Enquanto (WHILE), Repita (DO WHILE) e Para (FOR). Também são conhecidas como *loop*, pois esse nome lembra uma execução finita em círculos para resolver um algoritmo. Elas consistem em uma estrutura de controle do fluxo de execução que permite repetir diversas vezes em um mesmo trecho do algoritmo, com seu teste no início da estrutura no final. Para este material, será utilizado o laço chamado ENQUANTO.

Sintaxe – Estrutura de Repetição ENQUANTO

```
Enquanto <condição>
{ comandos;
  contador;
}
```

A estrutura de repetição ENQUANTO permite que um bloco seja executado enquanto a condição de repetição for verdadeira; quando o resultado for falso, o comando de repetição será finalizado. Uma característica importante da estrutura de repetição ENQUANTO é que, caso já na primeira vez o resultado da condição seja falso, nenhum comando será executado, o que representa a característica principal desse modelo de repetição. Os outros dois modos executam uma vez os comandos e depois fazem a verificação da condição.

No exemplo a seguir, é possível verificar a implantação do laço de repetição WHILE (Enquanto) – estrutura de repetição, condição, comando e contador.

```
<html>
<head>
<meta charset="utf-8"/>
<TITLE> JavaScript </TITLE>
<style>
body {background-color: blue; }
h1      {color: white; }
p       {font-family: arial; font-size: 20; text-align: center; }
</style>

</head>

<body>
<h1>
<p>
```

```
<script language="javascript" type="text/javascript">  
  
let Numero = 500;  
  
while (Numero < 1000)  
{  
    Numero = Numero + 100; //Incremento do valor  
}  
  
document.write(Numero)  
  
</script>  
</p>  
</h1>  
</body>  
</html>
```

3.3 OPERADORES LÓGICOS: AND E OR

São palavras que servem para ligar duas ou mais proposições simples, tornando-as compostas. São utilizadas nas tomadas de decisões e nas estruturas de repetição.

3.3.1 OPERADOR LÓGICO AND (E)

O operador lógico AND retorna verdadeiro quando as duas comparações forem verdadeiras; caso contrário, ele retorna falso.

SE condição1 E condição2

Comandos /* Quando AMBAS as condições forem verdadeiras */

SENÃO

Comandos /* Quando uma condição ou as duas forem falsas */

No JavaScript, a notação utilizada para o operador lógico AND é o “&”:

```
IF (condição1 & condição2)  
{  
    Comandos /* Quando AMBAS as condições forem verdadeiras */  
}  
ELSE  
{  
    Comandos /* Quando uma condição ou as duas forem falsas */  
}
```



Exemplo do operador lógico AND no JavaScript:

```
<html>
<head>
<meta charset="utf-8"/>
<TITLE> JavaScript </TITLE>
<style>
body {background-color: blue; }
h1      {color: white; }
p       {font-family: arial; font-size: 20; text-align: center; }
</style>

</head>

<body>
<h1>
<p>
<script language="javascript" type="text/javascript">

let Valor = 500;
let Qtde = 10;

if (Valor > 100 & Qtde > 5)
{
    document.write("As duas comparações são verdadeiras")
}
else
{
    document.write("Ao menos uma das comparações é falsa ou as duas")
}

</script>
</p>
</h1>
</body>
</html>
```

3.3.2 OPERADOR LÓGICO OR (OU)

O operador lógico OR retorna verdadeiro quando uma das comparações forem verdadeiras; caso as duas forem falsas, ele retorna falso.

SE condição1 OU condição2

Comandos /* Quando UMA ou AMBAS as condições forem verdadeiras */

SENÃO

Comandos /* Quando as DUAS condições forem falsas */

No JavaScript, a notação utilizada para o operador lógico OR é o “|” (*pipe*):

```
IF (condição1 | condição2)
{
    Comandos /* Quando UMA ou AMBAS as condições forem verdadeiras */
}
ELSE
{
    Comandos /* Quando as DUAS condições forem falsas */
}
```

Exemplo do operador lógico OR no JavaScript:

```
<html>
<head>
<meta charset="utf-8"/>
<TITLE> JavaScript </TITLE>
<style>
body {background-color: blue; }
h1      {color: white;}
p       {font-family: arial; font-size: 20; text-align: center; }
</style>

</head>

<body>
<h1>
<p>
<script language="javascript" type="text/javascript">

let Valor = 500;
let Qtde = 10;

if (Valor > 100 | Qtde > 5)
{
```

```
        document.write("Ao menos uma das comparações é verdadeira")
    }
else
{
    document.write("As duas comparações são falsas")
}

</script>
</p>
</h1>
</body>
</html>
```



LEITURA

O *Guia JavaScript* mostra como usar JavaScript e dá uma visão geral dessa linguagem. Ele é disponibilizado pela Mozilla.org (www.mozilla.org/pt-BR/), a qual é uma comunidade de software livre que usa, desenvolve, divulga e suporta os produtos Mozilla® – entre eles, o JavaScript. Disponível em: <https://mzl.la/2QUrds>. Acesso em: fev. 2021.

CONSIDERAÇÕES FINAIS

Esta Unidade abordou a linguagem JavaScript, muito conhecida no mercado, a qual viabiliza a implantação de regras de negócio em uma página HTML. Foi possível implementar variáveis, operadores aritméticos, estruturas de repetição e decisão. Com a aplicação dessas estruturas, consegue-se controlar o fluxo do algoritmo com repetições controladas e decisões a partir de comparações inseridas na estrutura da página. Foi trabalhada, também, a introdução de lógica de programação, na qual foram vistos os operadores lógicos AND e OR, que possibilitam a aplicação de testes lógicos nas estruturas de repetição e decisão.

A lógica de programação, os operadores lógicos e a aplicação deles são muito importantes e imprescindíveis no desenvolvimento de um sistema. Para comprovar isso, serão abordadas na Unidade IV mais situações de aplicações de lógica de programação, além de páginas dinâmicas e responsivas, interfaces amigáveis e ferramentas de depuração.



ANOTAÇÕES

UNIDADE 03

UNIBRASIL EAD | DESENVOLVIMENTO WEB

UNIDADE

04

JAVASCRIPT - PROGRAMAÇÃO



OBJETIVOS DE APRENDIZAGEM

- » Implementar estruturas de decisão e de repetição
- » Criar páginas dinâmicas e responsivas
- » Incorporar interface amigável a uma página HTML e utilizar ferramentas de depuração



VÍDEOS DA UNIDADE



<https://bit.ly/3cPczyH>



<https://bit.ly/3xtwtCS>



<https://bit.ly/3gwJvd8>



<https://bit.ly/3wxuYUp>



<https://bit.ly/3wzK77p>



<https://bit.ly/3cQ02Xo>



<https://bit.ly/3xr8sfN>

INTRODUÇÃO

Conforme já abordado anteriormente, JavaScript é uma linguagem de programação utilizada para adicionar interatividade a uma página HTML e para a aplicação de regras de negócio que implementam cálculos, decisões e repetições. Para a aplicação dessas regras de negócio, a linguagem JavaScript apresenta estruturas de decisão que podem desviar o fluxo do algoritmo aplicado à página. Existem, também, estruturas de repetição, que permitem repetir comandos até que uma ou mais condições sejam atingidas.

Outra situação importante no desenvolvimento de sistemas é a possibilidade de depuração do algoritmo de programação aplicado à estrutura da página WEB. Essa depuração auxilia na verificação, validação e busca de erros.

No desenvolvimento de um sistema, também é importante a criação de uma interface amigável que permita interação, facilidade de navegação e uma fluidez na utilização do sistema. Para o desenvolvimento de sistemas ou *sites* amigáveis e que utilizem a HTML e o JavaScript, existe o React, que, conforme seus criadores (React.org), é uma biblioteca JavaScript declarativa, eficiente e flexível para a criação de interface do usuário (UI).

Há muitas maneiras de escrever um script pensando em sua forma e concatenação. Mesmo assim, ele funcionará corretamente. Nas estruturas de decisão e de repetição, as chaves não são exigidas quando, e somente quando, houver apenas um comando após a chave condicional.

Exemplo:

```
If (num < 10) {num = 20} else {num = 30}
```

é equivalente a:

```
If (num < 10) num = 20 else num = 30
```

1. ESTRUTURAS DE DECISÃO E REPETIÇÃO

As estruturas de decisão e repetição auxiliam a implementação de regras de negócio ao sistema proposto, tanto para tomada de decisão dentro da estrutura de programação quanto para a repetição de ações em determinados pontos do sistema.

1.1 ESTRUTURAS DE DECISÃO

As declarações condicionais permitem implementar as tomadas de decisão no JavaScript. Já foi vista anteriormente a condicional IF; nesta Unidade, serão apresentadas outras condicionais, que podem ajudar em algumas situações de programação.

1.1.1 CONDICIONAL IF E ELSE

Estrutura básica de condicional no JavaScript, na qual é feita uma comparação e, a partir do resultado verdadeiro, é realizado um conjunto de comandos, senão outro conjunto é realizado.

```
if (condicao)
    {codigo para executar caso a condição seja verdadeira}
Else
    {senão, executar este código}
```

1.1.2 CONDICIONAL ELSE IF

Quando se tem mais do que um valor a ser verificado, é possível usar a condicional ELSE IF, que possibilita a utilização de várias condições.

```
if (condicao)
    {codigo para executar caso a condição seja verdadeira}
else if (condição) {se condição verdadeira, executar este código}
else if (condição) {se condição verdadeira, executar este código}
else {senão, executar este código}
```

1.1.3 ANINHANDO IF E ELSE

É comum colocar uma condicional IF dentro da outra quando existe mais do que um tipo de valor a ser testado. Por exemplo, ao verificar a filial de uma empresa e, ainda, valores de produtos para a concessão de desconto.

```
if (filial = F1)
{if valor > 1000 {valor = desconto 10%} else {valor = desconto 5%}}
else
{if valor > 1000 {valor = desconto 5%} else {valor = desconto 2%}}
```

1.1.4 INSTRUÇÕES SWITCH

As instruções **switch** tomam uma única variável como uma entrada e, em seguida, examinam várias opções até encontrarem uma que corresponda ao valor da variável, executando o código correspondente.

```
var N1 = select.value;

switch (N1)
{
    case '1': comando;
    break;
    case '2': comando;
    break;
    case '3': comando;
    break;
    case '4': comando;
    break;
    default: comando;
}
```

1.1.5 CONDICIONAL BOOLEAN

Existe o teste de valores boolean (true/false). Qualquer valor que não seja false, undefined, null, 0, NaN, ou uma string vazia ("") retorna o valor true quando é testado como uma instrução condicional, sendo possível usar um nome de variável para testar se é verdadeiro.

```
var N1 = '1';

if (N1)
{codigo para executar}
else
{senão, executa esse codigo}
```



SAIBA MAIS

A Estrutura Condicional pode ser simples ou composta. A estrutura condicional simples executa um comando ou vários comandos se a condição for verdadeira e, se a condição for falsa, a estrutura é finalizada sem executar os comandos. Por sua vez, a estrutura condicional composta executa um comando ou vários se a condição for verdadeira e, se a condição for falsa, executa um ou mais comandos também. O comando que define a estrutura é representado pela palavra "IF".

1.2 ESTRUTURAS DE REPETIÇÃO

As estruturas de repetição são utilizadas para repetir um determinado conjunto de comandos que necessitam ser realizados até uma condição ser atingida.



1.2.1 WHILE

A estrutura de repetição while, conforme já visto na Unidade III, executa seus comandos enquanto uma condição especificada seja avaliada como verdadeira. A característica do while é que os comandos podem não ser executados caso a condição já seja falsa na sua primeira execução.

```
while (condicao)
{comandos}
Exemplo:
var contador = 0;
while (contador < 10)
{// Executa 10 vezes, com os valores do contador de 0 a 9
document.write(contador);
contador = contador + 1;}
```

1.2.2 FOR

Um laço for é repetido até que a condição especificada seja falsa.

```
for ([expressaoInicial]; [condicao]; [incremento])
{comandos}
Exemplo:
var passo;
for (passo = 0; passo < 10; passo++)
{// Executa 10 vezes, com os valores de passos de 0 a 9
document.write(passo);}
```

1.2.3 DO...WHILE

A instrução do...while repetirá até que a condição especificada seja falsa. Diferentemente da estrutura de repetição while, a estrutura do...while executa ao menos uma vez os comandos, pois a condição está no final da sua estrutura.

```
do
{comandos}
while (condicao);
Exemplo:
var contador = 0;

do
{// Executa 10 vezes, com os valores do contador de 0 a 9
document.write(contador);
contador = contador + 1;
}
while (contador < 10)
```



VÍDEO

Para rever as estruturas de controle e repetição, existem muitos materiais escritos e em vídeo que poderão ser úteis para aprender como aplicá-los. Entre os vários vídeos que é possível encontrar na internet, fica aqui como indicação o “JavaScript Essencial – Estrutura de controle e repetição”, do RBtech, que traz uma revisão das estruturas de controle e repetição. Esse tipo de material é útil para visualizar e poder utilizar uma solução em projetos futuros. Disponível em: <https://bit.ly/3fA2bj1>. Acesso em: fev. 2021.

2. CRIAÇÃO DE PÁGINAS DINÂMICAS E RESPONSIVAS

2.1 PÁGINAS DINÂMICAS

Com o JavaScript é possível fazer atualização de elementos da página no navegador do usuário, em vez de baixar essas atualizações do servidor WEB. Isso proporciona uma agilidade maior para a página, e esse uso de *scripts* para atualização de elementos é chamado de *páginas dinâmicas*.

Uma situação bastante comum na atualização de elementos de uma página é a disponibilização de data e hora. Por isso, o JavaScript traz várias opções para manipulação de data e hora:

FUNÇÃO	ACÃO
getDay()	Dias da Semana de 0 a 6 (Domingo = ZERO).
getDate()	Dias do Mês de 1 a 31.
getMonth()	Meses de 0 a 11 (Janeiro = ZERO).
getYear()	Ano com 2 dígitos.
getFullYear()	Ano com 4 dígitos.
getHours()	Hora de 0 a 23.
getMinutes()	Minuto de 0 a 59.
getSeconds()	Segundo de 0 a 59.
getMilliseconds()	Milissegundo de 0 a 999.

O mês é de 0 a 11; será necessário somar 1 quando for disponibilizado na página.

Outra situação que ocorre na disponibilização de datas é a necessidade de apresentar por extenso os meses e os dias da semana. Assim é preciso utilizar uma funcionalidade relativamente simples para resolver essa característica, que são os vetores, como pode ser visto no exemplo a seguir:



```
var data = new Date();
var diasemana = new Array('Domingo','Segunda-feira','Terça-feira','Quarta-feira','Quinta-feira','Sexta-feira','Sábado');
document.write(diasemana[data.getDay()]);
```

A seguir, um exemplo para a disponibilização da data e hora, lembrando que elas são do computador do usuário:

```
<html>
<head>
<meta charset="utf-8"/>
<TITLE> JavaScript </TITLE>
<style>
body {background-color: blue; }
h1      {color: white; }
p       {font-family: arial; font-size: 20; text-align: center; }
</style>
</head>
<body>
<h1>
<p>
<script language="javascript" type="text/javascript">
// Seta a variável data com a data e hora atuais
var data = new Date();
var diasemana = new Array('Domingo','Segunda-feira','Terça-feira','Quarta-feira','Quinta-feira','Sexta-feira','Sábado');

// Seta as variáveis com as partes da data e hora
var semana = data.getDay();      // Dias da Semana de 0 a 6 (domingo = ZERO)
var dia   = data.getDate();      // Dias do Mês de 1 a 31
var mes   = data.getMonth();     // Meses de 0 a 11 (Janeiro = ZERO)
var ano_2 = data.getYear();      // Ano com 2 dígitos
var ano_4 = data.getFullYear();  // Ano com 4 dígitos
var hora  = data.getHours();    // Hora de 0 a 23
var min   = data.getMinutes();   // Minuto de 0 a 59
var seg   = data.getSeconds();  // Segundo de 0 a 59
var mseg  = data.getMilliseconds(); // Milisegundo de 0 a 999

// Seta a data e a hora nas variáveis (obs.: mês + 1)
var datav = dia + '/' + (mes+1) + '/' + ano_4;
var horav = hora + ':' + min + ':' + seg;

// Mostra a data
document.write('Data: ' + datav + ' Hora: ' + horav);

// Mostra o dia da semana
document.write(' Hoje é ' + diasemana[data.getDay()]);</script>
</p>
</h1>
</body>
</html>
```

2.2 DESIGN RESPONSIVO

No início do desenvolvimento das páginas *web*, o *design* não se preocupava com o tamanho das telas, pois a maioria delas tinha o mesmo tamanho. Se o tamanho fosse maior ou menor que o padrão, apareciam indesejadas barras de rolagem e o espaço da página era inadequado. Conforme as medidas das telas utilizadas foram ficando cada vez mais diferentes, desde *smartphones* até TVs de muitas polegadas, surgiu o conceito de *web design responsivo* (RWD), que é um conjunto de práticas onde páginas *web* alteram seu *layout* e aparência adequando-se a diferentes larguras, alturas e resoluções de telas. Isso alterou o modo de projeto e desenvolvimento das páginas *web*, apresentando múltiplos dispositivos de acesso a essas páginas e com usuários cada vez mais atentos e exigentes quanto à formatação e disposição dos elementos de *design* das páginas, independentemente do dispositivo que eles estejam utilizando.

A seguir, serão mostradas as principais técnicas que podem ser aplicadas no desenvolvimento de páginas *web* para elas apresentarem um *design* responsivo. Lembrando que são técnicas combinadas, que utilizam vários conceitos ao mesmo tempo.

2.2.1 MEDIA QUERIES

Media Queries possibilitam testar o tamanho e a resolução da tela e utilizar o CSS para adequar a página de acordo com as necessidades. No exemplo a seguir, *media queries* testa se exibição da página é uma tela, e não um documento impresso, e se o monitor tem ao menos 800 pixels de largura. O seletor *.container* será aplicado apenas se essas duas condições forem verdadeiras:

```
@media screen and (min-width: 800px)
{
    .container {margin: 1em 2em; /* relativo ao tamanho do texto */}
}
```

Podem ser adicionadas diversas *medias queries* dentro de uma folha de estilo, possibilitando adequar o *layout* a vários tamanhos de tela. Esses pontos onde um *media query* é inserido são conhecidos como *breakpoints*. Outra abordagem comum é conhecida como *mobile first*, que consiste em criar um *layout* com somente uma coluna para dispositivos de tela pequena, como um *smartphone*, depois verificar se a tela é maior, e, então, implementar um *layout* de várias colunas para monitores maiores.

2.2.2 GRIDS FLEXÍVEIS

Os *sites* responsivos, além de mudarem o *layout* entre *breakpoints*, também podem ser construídos em *grids* flexíveis. Quando um *grid* flexível é utilizado, não há necessidade de marcar todos os tamanhos de telas, mas sim criar um *layout* baseado em *pixels* que irá se adaptar automaticamente à tela. Quando se usa *grids* flexíveis, não é preciso adicionar *breakpoint* e modificar o *layout* da página para telas diferentes.



CÓDIGO NO CSS

```
.container {  
    display: grid;  
    grid-template-columns: repeat(3, 1fr);  
    gap: 10px;  
    grid-auto-rows: minmax(100px, auto);}  
.lin1 {grid-column: 1/3; grid-row: 1;}  
.lin2 {grid-column: 2/4; grid-row: 1/3;}  
.lin3 {grid-column: 1; grid-row: 2/5;}  
.lin4 {grid-column: 3; grid-row: 3;}  
.lin5 {grid-column: 2; grid-row: 4;}  
.lin6 {grid-column: 3; grid-row: 4;}
```

CÓDIGO NO HTML

```
<div class="container">  
<div class="lin1">Um</div>  
<div class="lin2">Dois</div>  
<div class="lin3">Três</div>  
<div class="lin4">Quatro</div>  
<div class="lin5">Cinco</div>  
<div class="lin6">Seis</div>  
</div>
```

3. INTERFACE AMIGÁVEL

Interface amigável é uma interação na qual o usuário encontra facilidades de navegação e que permite uma fluidez na utilização de um *software* ou página da internet. Para o desenvolvimento de *sites* amigáveis e que utilizem a HTML e o JavaScript, será visto a seguir o React, o qual, de acordo com a definição dos seus criadores (React.org), é uma biblioteca JavaScript declarativa, eficiente e flexível para a criação de interface do usuário (UI).

3.1 REACT

O React apresenta algumas características que o destacam entre as linguagens modernas de desenvolvimento de aplicativos WEB:

- É uma biblioteca, ou seja, uma coleção de funcionalidades;
- Utiliza a mesma linguagem de programação do JavaScript;
- É uma linguagem de programação declarativa;
- No React tudo é componente;
- O React é eficiente propondo o seu próprio DOM (Document Object Model);
- É utilizado para desenvolver aplicativos móveis;
- É muito usado e admirado pelos desenvolvedores.

Essas características tornam a utilização do React muito popular e fácil de ser feita pelos desenvolvedores JavaScript, sendo que três aspectos são fundamentais:

- a) Para a descrição de interfaces no React são utilizados componentes, que são como funções, onde é informado um valor, processado, e a função retorna um resultado. E os componentes funcionam da mesma maneira: são chamadas as entradas de “propriedades” e “estado” e a saída de um componente é a descrição de uma interface.
- b) Quando há mudança de estado de um componente, sua interface também muda. Essa mudança deve refletir no dispositivo que está sendo usado.
- c) O React representa visualmente os *views* de dados na memória.

3.1.1 VANTAGENS DA UTILIZAÇÃO DO REACT

Uma das vantagens do uso do React é a possibilidade de organizar o código, pois o React permite separar o código da aplicação e criar pequenos componentes. Um exemplo são os botões constantemente usados em uma aplicação: pode-se criar um componente para esses botões, e então eles poderão ser utilizados na aplicação. O código fica centralizado e qualquer alteração que se faça necessária somente será feita no componente, e todos os locais que utilizam esses componentes serão corrigidos.

Também há vantagem na divisão de tarefas. Quando se utiliza o React, o *cliente* fica somente com a tarefa de renderizar a interface corretamente, enquanto todas as regras de negócio ficam no *servidor*, que será utilizado pelo React.

E a última vantagem é a característica de aceitar múltiplos clientes. O React é uma biblioteca-base, utilizada pelo ReactJS para projetos *web* e usada com o React Native para projetos *mobile*. Uma vez desenvolvido um projeto *servidor* para suportar uma aplicação ReactJS ou React Native, com facilidade poderá ser desenvolvido um projeto *cliente* React e utilizar o mesmo projeto *servidor*. Também é comum o compartilhamento de componentes e bibliotecas JS entre projetos *web* e *mobile*.

O React com certeza traz muitos benefícios ao desenvolvimento de aplicativos com JavaScript e deve ser estudado mais profundamente por quem tem interesse em implementar projetos de sistemas WEB. O objetivo de mencioná-lo neste material é mostrar que existe essa tecnologia que facilita e incrementa os projetos quando ela é incorporada ao desenvolvimento. Então fica aqui a dica para um estudo complementar do React para futuros projetos.



4. FERRAMENTAS DE DEPURAÇÃO

Depurar ou *debugar* o código de um sistema é analisar passo a passo a execução, geralmente de uma parte que está apresentando problema. O JavaScript é uma linguagem *Client-Side*, ou seja, é uma linguagem executada no “lado do cliente”, especificamente no equipamento do cliente, e não no servidor WEB. E essa depuração pode ser realizada através do navegador/*browser*.

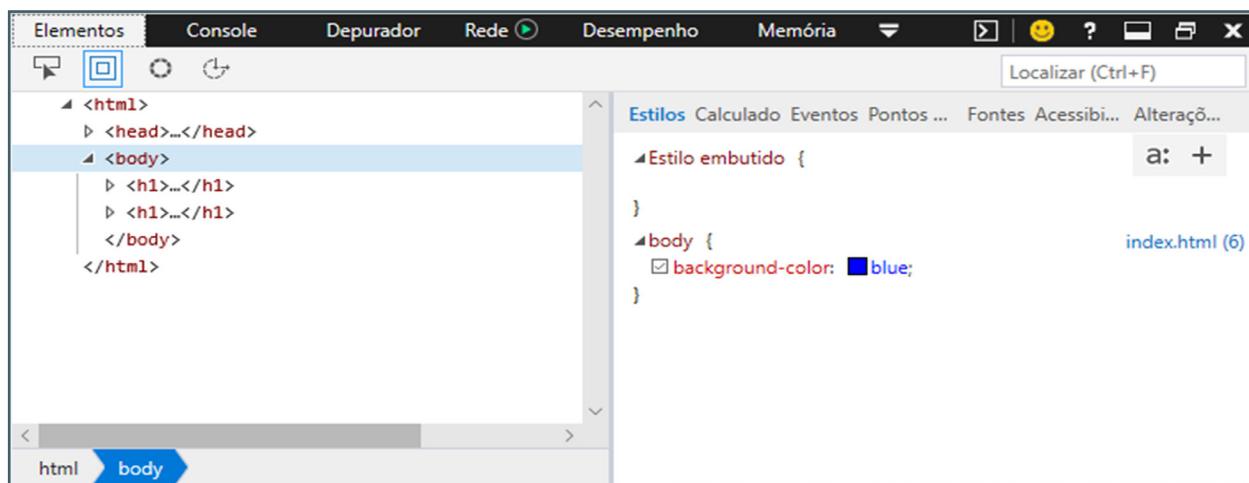
A desvantagem dessa possibilidade do JavaScript é que, se o desenvolvedor pode fazer essa depuração pelo navegador, o usuário também poderá fazer. E isso pode trazer alguns riscos, dependendo de como o código foi implementado, já que o JavaScript é uma linguagem auxiliar e parâmetros de conexão com banco de dados e segurança de usuários estão expostos, deixando o sistema vulnerável a ataques. Para evitar esse tipo de problema, esses parâmetros devem ser implementados na linguagem principal do sistema, utilizando o JavaScript para efetivar interfaces amigáveis e procedimentos que exigem respostas em tempo real.

4.1 UTILIZANDO O NAVEGADOR PARA DEBUG

Para efeitos práticos, será utilizado aqui o Microsoft Edge® para a depuração dos arquivos do JavaScript, mas qualquer outro navegador que tenha esse recurso pode ser utilizado. Para a depuração, o importante é entender o funcionamento, pois, dessa maneira, poderá ser aplicado em qualquer navegador, console ou versão. Visando a uma melhor depuração, é aconselhável que os arquivos do HTML, CSS e JavaScript estejam separados.

PASSO 1 – Abrindo o Console no navegador

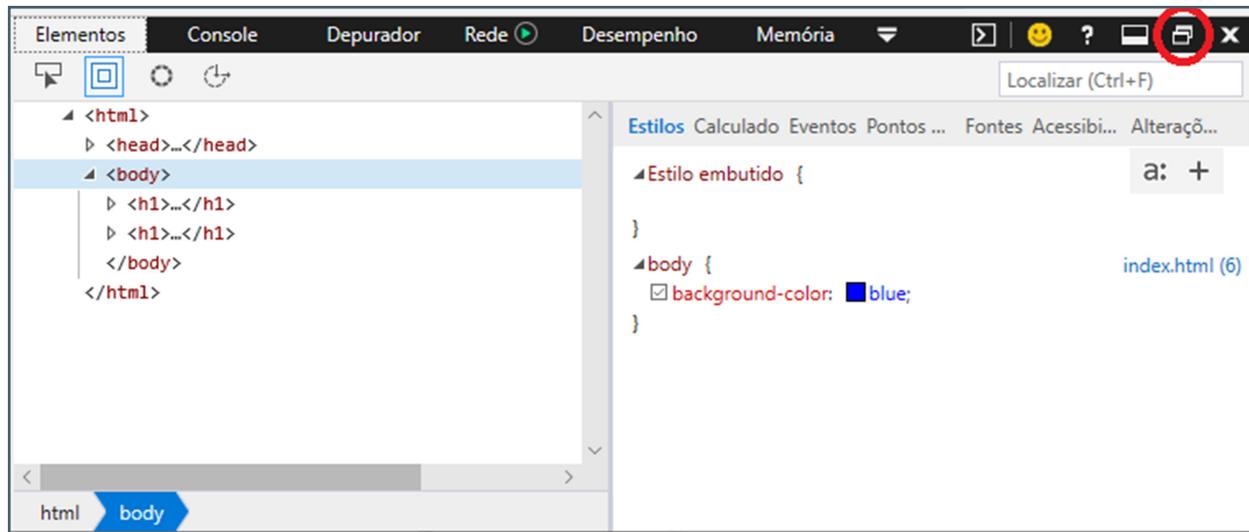
Com a página no navegador, pressionar a tecla F12 para acessar o console:



Fonte: O autor (2021).

PASSO 2 – Desencaixando o console da página

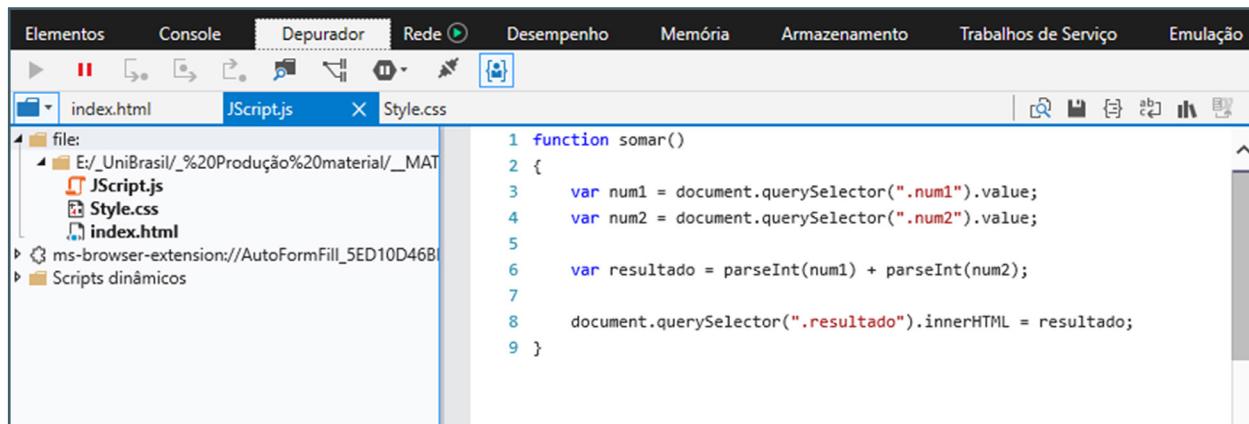
O console poderá ser separado da página para melhor manipulação, clicando-se no botão “desencaixar”, mostrado na tela a seguir:



Fonte: O autor (2021).

PASSO 3 – Depurando os arquivos do sistema

Existem várias abas, e cada uma delas tem determinada função, mas, para a depuração, deve-se selecionar a aba Depurador – é nela que poderão ser vistos os arquivos que compõem a página. Por isso é importante eles estarem separados, onde é fácil ver os arquivos HTML, CSS e o do JavaScript, que contém o algoritmo que será depurado. Na imagem a seguir, pode-se ver que foi selecionado o arquivo JScript.js:



Fonte: O autor (2021).

PASSO 4 – Incluindo um *Breakpoint*

Para verificar a execução do sistema, deve ser inserido um *breakpoint* clicando na linha de processamento; quando o sistema for executado, o navegador vai parar a execução e o fluxo será direcionado para o depurador. Isso ajudará a encontrar algum tipo de problema que possa estar acontecendo no sistema.

```

1 function somar()
2 {
3     var num1 = document.querySelector(".num1").value;
4     var num2 = document.querySelector(".num2").value;
5
6     var resultado = parseInt(num1) + parseInt(num2);
7
8     document.querySelector(".resultado").innerHTML = resultado;
9 }

```

Fonte: O autor (2021).

PASSO 5 – Verificando a execução do sistema

Depois de inserir a marcação da linha, deve-se executar o sistema até o *breakpoint* ser ativado. Na imagem a seguir, pode ser visualizada a função que foi chamada e a execução do procedimento interrompido para *debug*. No lado direito do painel, é possível acompanhar o valor das variáveis e outros detalhes que podem ajudar na resolução de alguma anomalia no algoritmo, a qual esteja comprometendo a execução do sistema. Pressionando F8, o depurador irá para o próximo *breakpoint*.

[Locals]
this [object Window]
arguments [object Arguments]
num1 3
num2 3
resultado undefined

Fonte: O autor (2021).

A depuração em linguagens *Client-Side* ainda é um recurso pouco explorado, mas de uma utilidade enorme. Ela pode agilizar o processo de desenvolvimento e testes.

CONSIDERAÇÕES FINAIS

Esta Unidade trouxe o complemento da lógica de programação com mais formas de estruturas de repetição e decisão. Foi vista na Unidade III a relevância dessas estruturas na aplicação das regras de negócio, mas também é essencial conhecer as várias formas em que essas estruturas podem ser aplicadas, pois podem ser úteis em diversos momentos do desenvolvimento de uma página.

Outro ponto importante visto nesta Unidade foi a criação de páginas dinâmicas e responsivas que se adaptam aos diversos equipamentos utilizados para a navegação nas páginas e nos sistemas WEB. Também foi abordada a implantação de interfaces amigáveis que possibilitam fluidez na navegação de uma página.

Para finalizar, as ferramentas de depuração de uma página foram mostradas. Com elas é possível verificar o fluxo com que uma estrutura de lógica de programação foi implantada, e essa possibilidade facilita a resolução de problemas de programação.

Os conteúdos das quatro Unidades trabalhadas nesta disciplina de Desenvolvimento WEB possibilitem a criação de páginas e sistemas WEB de baixa complexidade. Portanto, há muitas possibilidades de crescimento e aprendizado com base nos assuntos tratados na disciplina. O ambiente de desenvolvimento de sistemas é muito dinâmico e está em constante evolução, cabendo aos desenvolvedores ficarem atentos a novas tecnologias e tendências do mercado.



ANOTAÇÕES



REFERÊNCIAS

DEITEL, Harvey M; DEITEL, Paul J; NIETO, T. R. **Internet & World Wide Web:** como programar. 2. ed Porto Alegre: Bookman, 2003. 1274 p.

DEITEL, Paul J; DEITEL, Harvey M. **Ajax, rich internet applications e desenvolvimento web para programadores.** São Paulo: Pearson, 2009. 747 p.

MORRISON, Michael. **Use a cabeça:** JavaScript. Rio de Janeiro: Alta Books, c2008. 606 p.

NIEDERAUER, Juliano. **Web interativa com Ajax e PHP.** São Paulo: Novatec, 2007. 288 p.

NIEDERST, Jennifer. **Aprenda Web design.** Rio de Janeiro: Ciência Moderna, 2002. 464 p.

SILVA, Mauricio Samy. **Construindo sites com CSS e (X) HTML:** sites controlados por folhas de estilo em cascata. São Paulo: Novatec, 2007. 446p.

Responsive Web

We create fully responsive websites from initial analysis to maintenance. Best User experience guaranteed.



Responsive Web Design.

We create fully responsive websites from initial analysis to maintenance. Best User experience guaranteed.

[VIEW MORE](#)



[VIEW MORE](#)



EAD

