Interface Web I

PROGRAMAÇÃO WEB I

Introdução

Figura 1: Internet / Fonte: feebsc

Todos nós temos uma História. Quando entendemos de onde nossa origem, conseguimos então compreender a evolução até chegarmos agora.

Com a Internet não foi diferente. Tudo começa logo após a Segunda Guerra Mundial, com uma treta gigante entre Estados Unidos e União Soviética, que antes eram aliados, mas agora se tornam inimigos. Acompanhe o desenrolar dessa novela e entenda como uma Guerra deu origem à maior rede do mundo. Vamos lá?

TEMA 01 – História da Internet

- ***Guerra Fria***

Depois da Segunda Guerra, EUA e URSS começaram a ter seus desentendimentos, dando origem à Guerra Fria em 1949. Neste contexto, em que os dois blocos ideológicos e politicamente antagônicos exerciam enorme controle e influência no mundo, qualquer mecanismo, qualquer inovação, qualquer ferramenta nova poderia contribuir nessa disputa liderada pela União Soviética e pelos Estados Unidos: as duas superpotências compreendiam a eficácia e a necessidade absoluta dos meios de comunicação.

Figura 2: Guerra Fria – EUA e União Soviética / Fonte: UOL

Nesse meio de tensão da Guerra Fria no dia 4 de outubro de 1957 a União Soviética decide lançar o Sputnik - 1, primeiro satélite artificial lançado em órbita, com o intuito de dar uma volta ao redor da terra e logo cair no oceano, porém os EUA entenderam isso como um ato hostil. Para atender à necessidade militar de proteger os sistemas de defesas temendo um ataque nuclear e cientes do poder da comunicação, os EUA criaram um sistema de descentralização de suas informações no Pentágono para evitar que possíveis ataques causassem a perca irreparável de documentos do governo. Um ataque poderia trazer a público informações sigilosas, tornando os EUA vulneráveis.

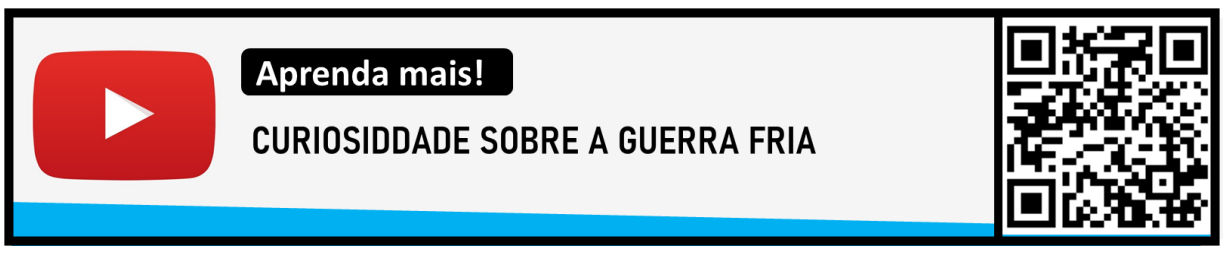


Figura 3: Sputnik – 1º Satélite Artificial / Fonte: Sputnik News

Então o presidente Dwight Eisenhower decidiu fundar a ARPA (Advanced Research Projects Agency Network) mais tarde se transformando na DARPA (Defense Advanced Research Projects Agency) - Agência de Defesa de Projetos e Pesquisa Avançada, onde criou um sistema de compartilhamento de informações entre centros militares distantes geograficamente, a fim de facilitar as estratégias de guerra e manter a segurança dos dados descentralizando as informações apenas em um local físico. Assim foi criada a ARPANET, com a primeira conexão sendo feita 29 de outubro de 1969 entre a Universidade da Califórnia e o Instituto de Pesquisa de Stanford.

Figura 4: ARPANET com 4 pontos / Fonte: theconversation

Momento histórico onde a primeira de mensagem de email foi enviada usando o protocolo NCP (Network Control Protocol - Protocolo de Controle de Rede) entre os computadores SDS Sigma 7 na

Califórnia e o SDS 90 em Stanford protocolo NCP tinha um ponto negativo, quando uma informação era trocada entre eles, outros computadores na rede eram obrigados aguardar o compartilhamento para então fazer sua comunicação via rede, fazendo que a comunicação era feita de apenas 2 computadores por vez, atrasando assim a comunicação entre outros pontos, na época não existiam muitos computadores conectados a ARPANET, eram apenas 4 (SDS Sigma 7 na Califórnia, SDS 90 em Stanford, IBM 370/75 em Los Angeles e o DEC PDP 10 em UTAH).

Figura 4: Computadores SDS Sigma 7 e SDS 90 / Fonte: slideshare

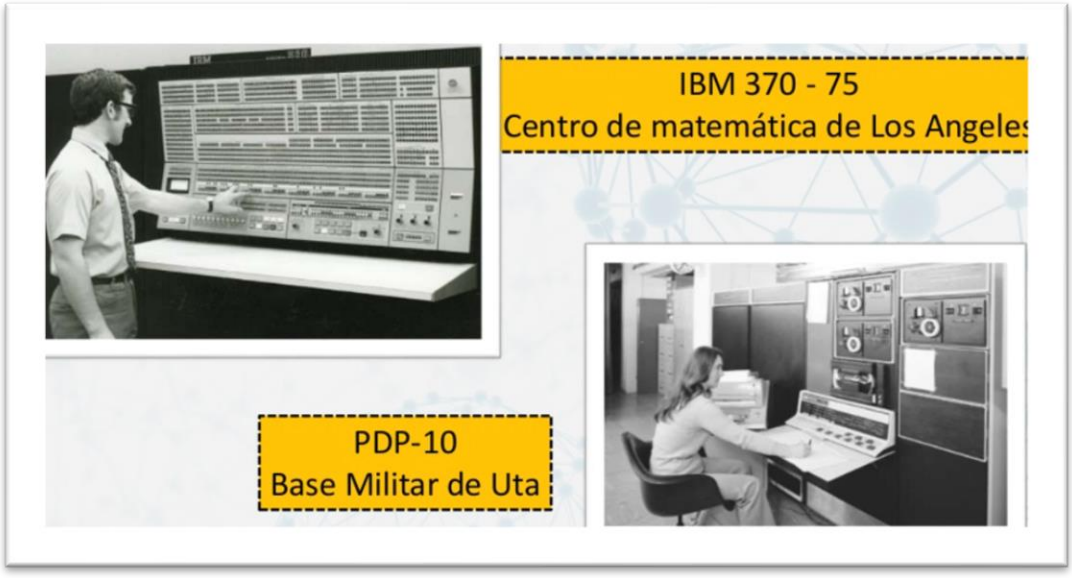


Figura 4: Computadores IBM 370/-75 e PDP-10 Fonte: slideshare

- ***Protocolo TCP/IP***

Em 1972 dois pesquisadores perceberam a necessidade de novos protocolos serem usados na comunicação, Robert Kahn mais conhecido como Bob Kahn criou o protocolo chamado TCP (Transfer Control Protocol - Protocolo de Controle de Transferência), a idéia do TCP era eliminar o problema do NCP, onde teria que parar a rede enquanto estava sendo feita uma transmissão, podendo assim utilizar várias transmissões simultaneamente, porém esse protocolo também tinha um problema, que era identificar os pontos de transferência, ja que na época os pontos tinham aumentado, então o pesquisador Vinton Cerf conhecido como Vint Cerf, criou o protocolo IP (Internet Protocol - Protocolo de Rede) afim de resolver os problema de identificação dos pontos de acesso, fazendo assim uma padronização da comunicação, juntando então os protocolos TCP/IP.

Figura 5: Vincent Cerf e Robert Kahn / Fonte: hollywoodbowles

Com o avanço da Arpanet e o enorme crescimento dos pontos de acesso, os militares decidiram dividir essa rede ficando apenas com a administração da *MILNET*, os centros de pesquisas e universidades responsáveis pela *NFSNET* (National Science Foundation - Fundação Nacional de Ciência) ficando assim as redes comerciais usando a Arpanet.

Com a necessidade de interconexão dessas redes, exceto a *MILNET*, foi criada uma Interconnect Networking, onde foi simplificada e finalmente chegando ao nome que conhecemos hoje, a Internet.

Criação do WWW

Figura 5: Tim Berners-Lee / Fonte: vanityfair

Em meados da década de 50, mais precisamente 8 de junho de 1955, em Londres na Inglaterra, nasce Timothy John Berners-Lee, um dos principais responsáveis pela internet que temos hoje.

Em 1980 Tim Berners-Lee, trabalhou no Instituto CERN (European Organization for Nuclear Research – Conselho Europeu de Pesquisa Nuclear) como consultor em projetos de comunicação. Nesse período em que trabalhou no CERN, Tim elaborou um projeto baseado em hipertextos, que falando em termos resumidos é um conjunto de informações organizadas em blocos de textos.

Esse projeto foi elaborado para facilitar a comunicação entre seus colegas de outros laboratórios, via internet. Em paralelo, Tim Berners-Lee trabalhava em outro projeto, que ele nomeou como *Enquire.* O projeto Enquire foi usado para associar e armazenar as informações em que Tim e seus colegas trabalhavam, muito semelhante ao *Wikipédia* de hoje em dia.

Em 12 de março de 1989, com ajuda de seu colega Robert Cailliau, ele escreveu uma proposta de gerenciamento de informação baseados em elementos do *Enquire*, que descrevia um sistema melhor elaborado para trocar informações.

Essa proposta foi publicada em 1990, sendo reconhecida mais tarde pelo primeiro modelo da World Wide Web, o famoso www. Protocolo usado até os dias de hoje, o www pega as informações da internet, e as organiza em blocos de texto, imagens e multimidia.

Para transportar esses blocos de informações de hipertextos, Tim Berners-Lee criou um protocolo de transferência, o conhecido HTTP (Hypertext Transfer Protocol – Protocolo de Transferência de

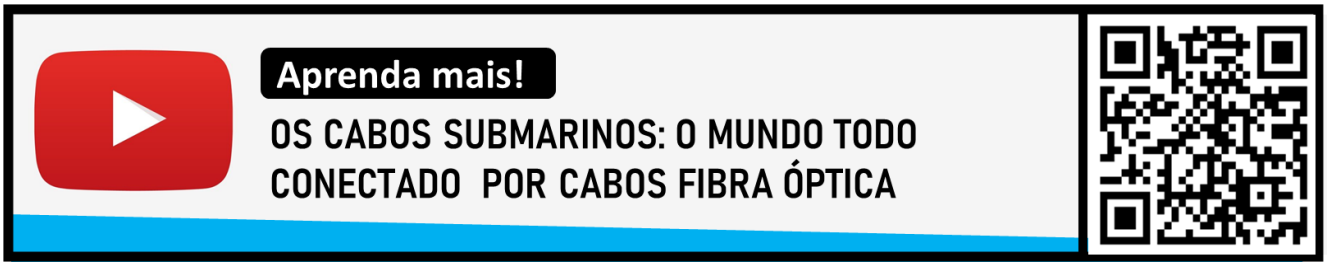
Hipertexto) formando assim a padronização entre a transferência de pacotes de hipertextos. Então Tim Berners-Lee juntamente com Robert Cailliau criam o HTML (Hypertext Markup Language – Linguagem de Marcação de Hipertexto), para a criação de páginas da internet.

Com o crescente uso da linguagem HTML e seus protocolos (HTTP e www), Tim Berners-Lee fundou a W3C (World Wide Web Consortium), o órgão formado por um grupo de empresas dispostas a auxiliar na criação de regras e recomendações para a criação de conteúdos web usando hipertextos.

NAVEGADORES

Falando de forma geral, os navegadores são os softwares mais utilizados e importantes, depois do sistema operacional. Obviamente que o fato de serem usados para interpretar os blocos de hipertextos enviados através do HTTP, ou seja, os documentos HTML, ajudam chegar nesse grau de importância. Vamos imaginar que você quer acessar o site do google, pelo navegador que faz a “requisição” via HTTP para o servidor onde o site do google está hospedado e pedindo para baixar a página (pacote de informações em HTML) e mostrando na tela.

O primeiro navegador também foi criado por Tim Berners-Lee o (WorldWideWeb) em 1990. Em 1992 foi a vez do Mosaic, o navegador mais conhecido na época. O Mosaic foi criado pelos estudantes do NCSA (National Center for Supercomputing Applications - Centro Nacional de Aplicações para Supercomputação. Com a evolução da web, os navegadores também foram evoluindo e oferecendo cada vez mais rapidez de respostas e oferecendo melhor serviços para os usuários.



A Internet no Brasil

A internet surgiu no Brasil em 1989, quando um grupo de universitários buscavam trocar informações com outras universidades internacionais. Essa conexão foi o LNCC (Laboratório Nacional de Pesquisa Cientifica) e a Universidade e Maryland nos Estados Unidos, eram trocadas emails relativamente simples, mas que na época de grande usabilidade.

Os primeiros domínios.br foram registrados em 1989, antes mesmo da primeira conexão de computadores nacionais e internacionais. Nesse período se dá início a uma rede nacional, já que todos os computadores conectados ganham um número de **IP**, essa base de registro foi estruturada pela FAPESP (Fundação de Amparo à Pesquisa do Estado de São Paulo.

O “boom” da internet se deu pelo uso do www, possibilitando um enorme crescimento de usuário criando suas próprias páginas web. Em paralelo surgia o comercio na rede, fortalecendo ainda mais o crescente avanço da internet no Brasil.

Você sabia?

A internet tem o mesmo peso de um morango.
Spams dominam a internet.
A webcam foi criada para vigiar uma cafeteira.
Twitter já teve outro nome.
O primeiro site da história ainda está no ar.

Responda:

1) Em qual período Histórico a Internet surgiu?

2) A agência militar americana DARPA foi a responsável pelo início dos estudos que deram origem à Internet. Logo no início, a rede teve um nome. Você sabe qual foi?

3) A primeira versão da rede que deu origem à Internet interligava quantos computadores?

4) Qual o nome do primeiro protocolo usado na transmissão de dados entre os controladores?

5) Qual o nome do pesquisador que criou o TCP (Transfer Control Protocol)?

6) Qual o nome do criador do IP (Internet Protocol)?

7) O que significa a sigla "www" e qual o nome do seu criador?

8) Como é feita a conexão entre os continentes nos dias de hoje?

9) Em que ano o Brasil fez sua primeira conexão?

10) O cientista inglês Tim Berners-Lee foi o responsável pela criação de três coisas muito importantes para a Internet. Cite quais foram:

DESAFIO

Faça uma linha cronologia dos principais eventos desde a criação a internet até os dias de hoje.

TEMA 02 – CONCEITOS E SERVIÇOS DA WEB (1.0, 2.0 E 3.0)

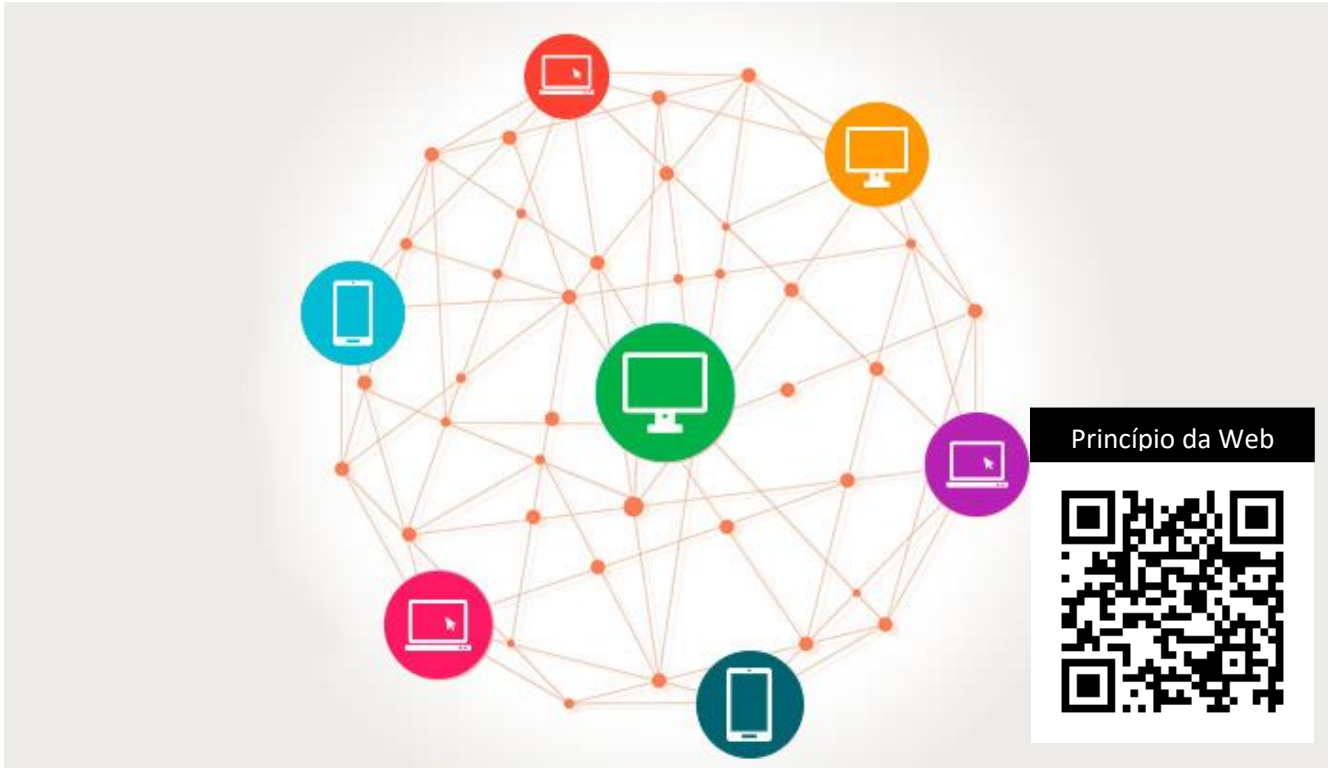


Figura 06 – Serviços WEB - Fonte: opensoft

SERVIÇOS WEB

A internet se tornou a forma mais democrática de comunicação, dentro dela temos os mais diversos tipos de ferramentas para fazermos a busca de todo, e qualquer conteúdo criado ou até mesmo publicado na rede.

Mas o que seria WEB? Falando em termos populares “web” simplesmente é uma das siglas do tão conhecido “www” (World Wide Web), falando em termos técnicos “web” é o termo usado para denominar uma rede com um conjunto de computadores conectados, onde conseguem realizar trocas e recebimento de mensagens.

No capítulo anterior vimos onde começou a internet e sua história, vamos agora entender quais as principais características e diferenças dos avanços da web, possibilitando variadas formas de interações com os dados e usuários, porém a evolução da web está mais relacionada com a forma na qual à utilizamos do que os avanços tecnológicos, as diferentes eras da web até apresentam uma

evolução tecnológica, mas o fator que realmente é representado nas “mudanças” na web, é da forma que o usuário de comporta.

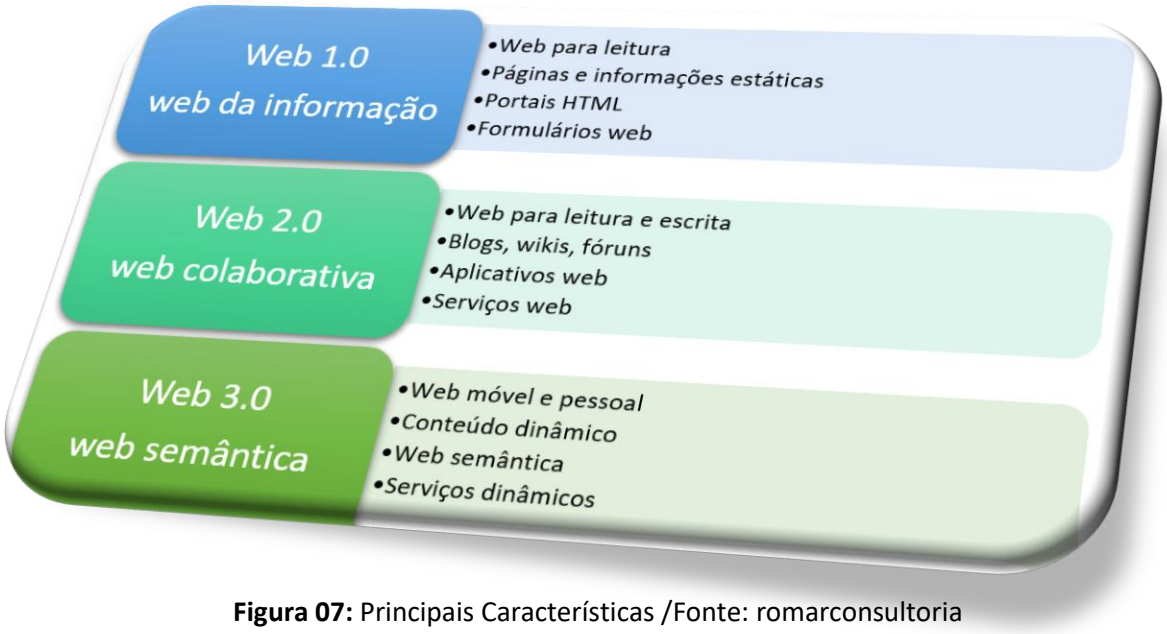
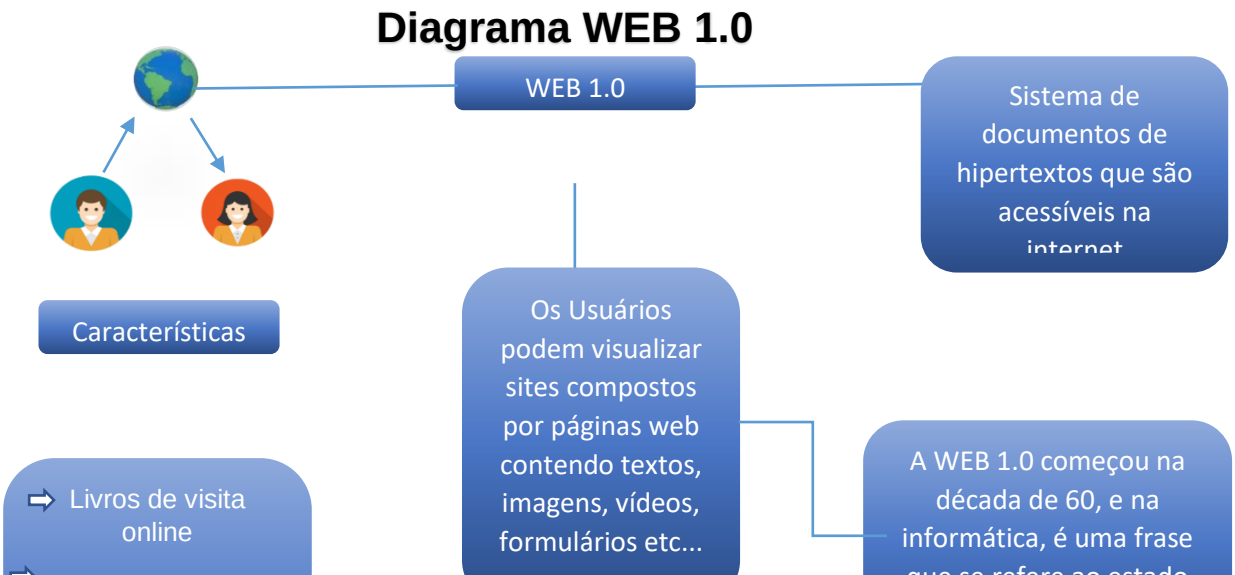


Figura 07: Principais Características /Fonte: romarconsultoria

WEB 1.0

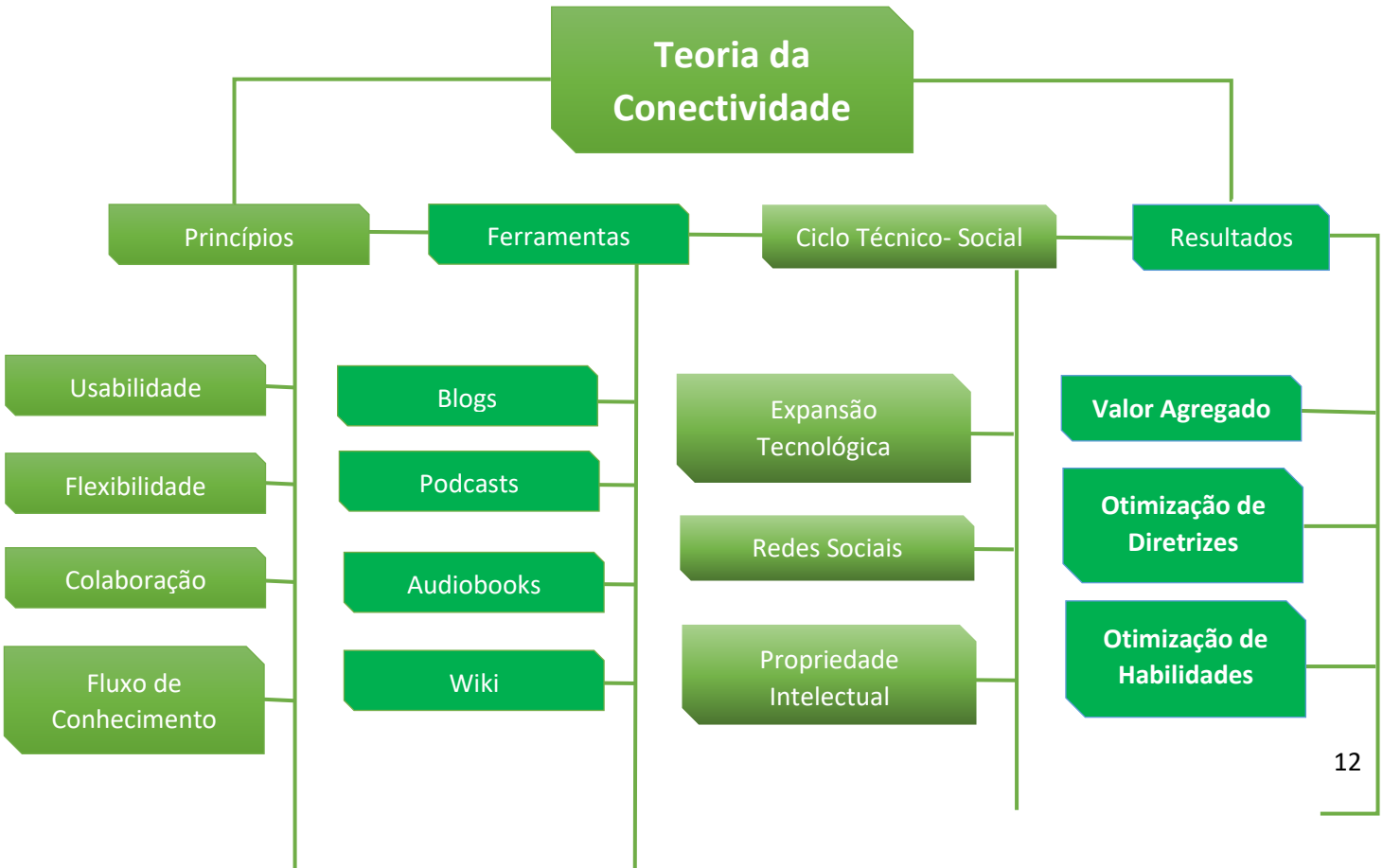
A web 1.0 é o início da era da internet, sua primeira fase de desenvolvimento, começando com poucos usuários e pontos conectados, a web 1.0 é conhecida como a era das informações, com sites estáticos e sem interatividade, sendo feita a comunicação de apenas “uma via”, as páginas não ofereciam aos usuários nenhum dinamismo, sendo usada apenas para leitura, onde a única forma de interatividade era dos comentários deixados por usuários em alguns sites. Evoluindo de suas raízes de uso militar, a web foi tomando um formato diferente devido as necessidades dos diferentes tipos de usuários, e assim iniciando uma mudança e importante na evolução da web.

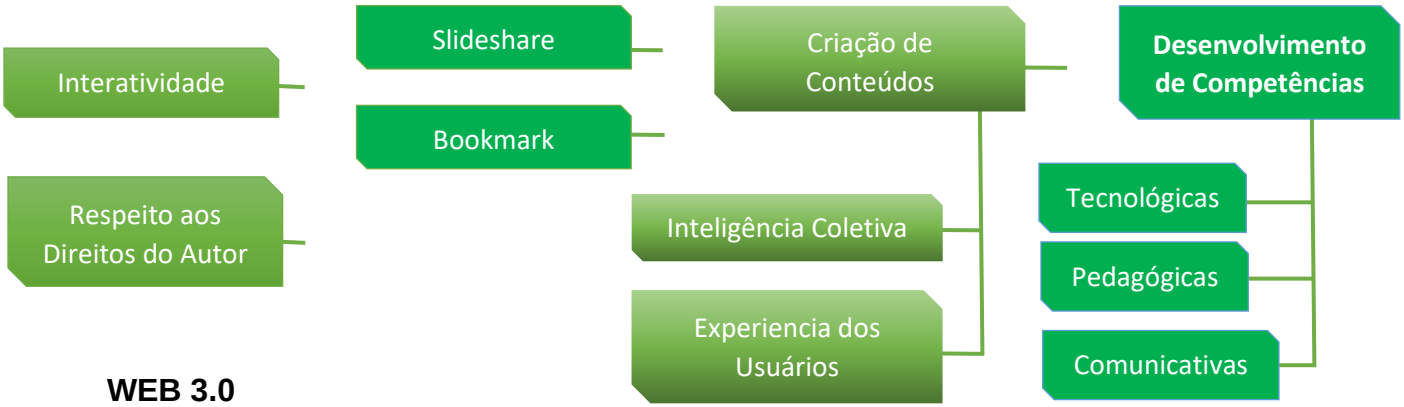


WEB 2.0

A web 2.0 ficou conhecida como a era das redes sociais, trazendo interatividade e dinamismo nas páginas, oferecendo diversas ferramentas para criadores de conteúdo e desenvolvedores, é a web do compartilhamento com novas possibilidades e alvos. Com essa evolução foi possível os ganhos que esses avanços nos proporcionaram, essa versão da web foi criada pelo especialista Tom O' Reilly, que nos mostra nova forma de se usar a internet como uma "plataforma web", as páginas ganharam buscadores otimizados e layouts focados diretamente no usuário. Assim os conceitos da web 2.0 buscou sempre a otimização de sites através de ferramentas altamente acessíveis e essenciais para as páginas da web 2.0.

Diagrama WEB 2.0

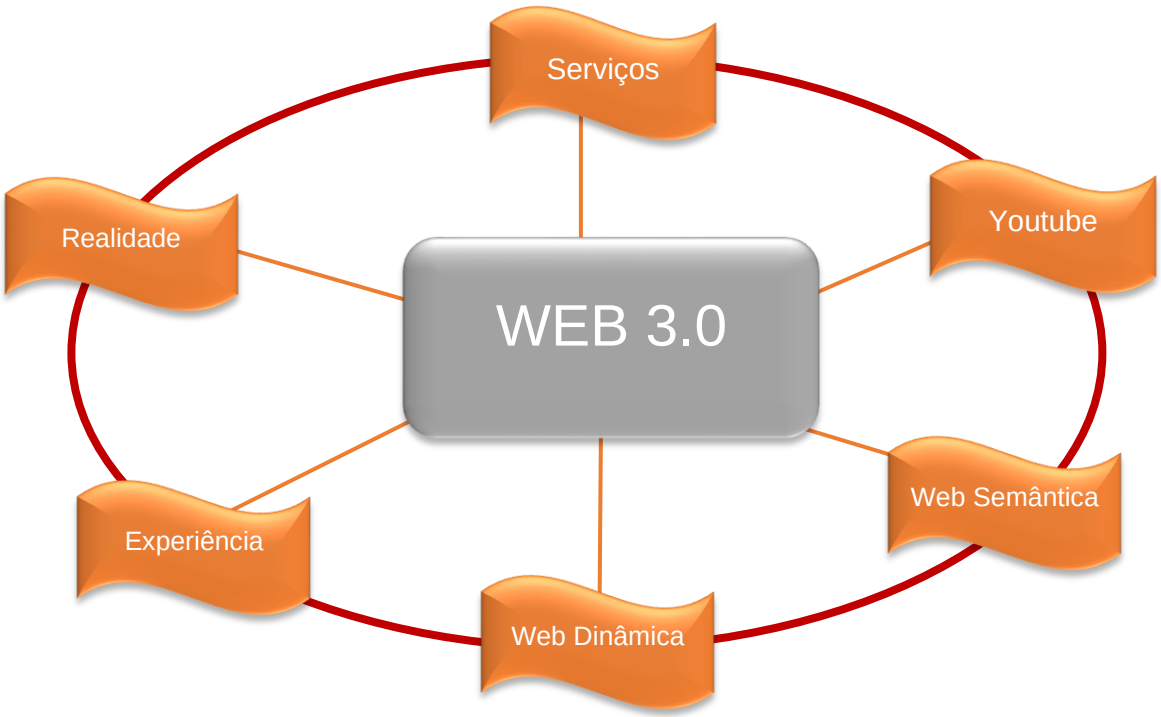




WEB 3.0

A web 3.0 é denominada como a "web inteligente ou web semântica", ela apresenta várias formas de interação trabalhando com machine learning (aprendizado de máquina), blockchain (cadeia de blocos) e inteligência artificial, proporcionando assim uma melhor experiência com o usuário e segurança com os dados compartilhados. A principal característica dessa versão da web, é a inteligência na qual as máquinas aprendem e executam tarefas baseadas nos seus próprios erros extraindo padrões e regras de conjunto de dados, sendo assim se tornando capaz de tomar decisões guiadas pelos resultados inferidos. As grandes facilidades e segurança que temos hoje são possíveis graças a web semântica, a atual geração está cada dia mais evoluindo, e só por curiosidade, já existem estudos e testes em ambiente web 4.0.

Saiba mais das novas tecnologias apresentadas e usadas na web 3.0 Veja esse video que explica muito bem qual as ferramentas e suas Diferenças.



SERVIÇOS WEB

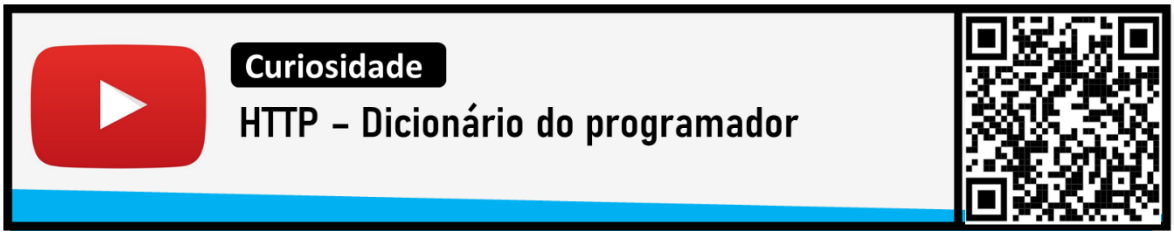
A web service é uma solução sendo usada na interação entre sistemas e aplicações diferentes entre si. Esta tecnologia possibilita que novas aplicações tenham interatividade com aplicações que ja existem, e que sistemas desenvolvidos em diferentes plataformas possam ser compatíveis. As aplicações web fazem uso dos componentes de web services para enviar e receber dados, aplicações que se caracterizam em ter sua própria linguagem, de forma que é feita a tradução para uma linguagem universal em formato intermediário como XML e Json por exemplo.

Quando aconteceu o "boom" da internet, os processos, que antes eram manuais, se tornaram online, pois notava - se a necessidade de uma variedade de ferramentas que automatizasse os processos, sendo tais ferramentas interoperáveis e escaláveis para proporcionar o reuso de sistemas que já estavam sendo utilizados remotamente. Essas tecnologias usadas desde os primórdios da web, continuam e se mantem até hoje, porém o formato XML é o mais usado na lista dos serviços web.

XML (Extensible Markup Language) é uma linguagem de marcação onde a descrição dos documentos utilizados nessa linguagem são de forma autocontida, ou seja, os documentos possuem regras predefinidas. Vamos ver alguns conceitos básicos dos web services:

API (Application Programming Interface) - É uma interface de comunicação de programas. É um intermediário de software que permite que dois aplicativos se comuniquem. Cada vez que você usa um aplicativo como o Facebook, envia uma mensagem instantânea ou verifica a previsão do tempo em seu telefone, você está usando uma API.

HTTP (Hypertext Transfer Protocol) - É o protocolo de comunicação e transferência de dados criado por Tim Bernees- Lee, ele permite a busca de recursos, como documentos HTML. É a base de qualquer troca de dados na Web e é um protocolo cliente-servidor, o que significa que as solicitações são iniciadas pelo destinatário, geralmente o navegador da Web.



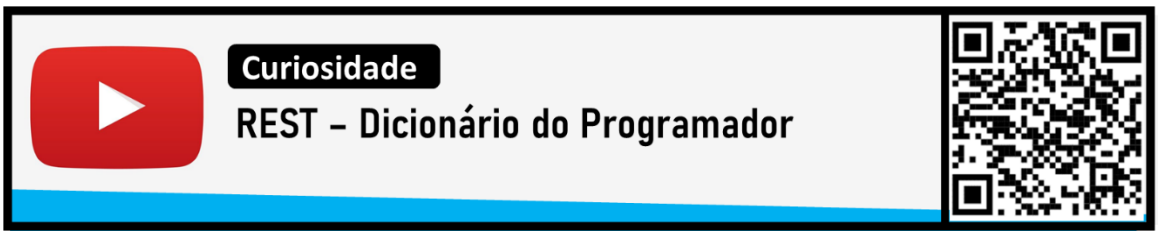
URI (Uniform Resouce Identifier) Um termo técnico de uma cadeia de caracteres compactas que são usadas para identificar ou até mesmo denominar um recurso na internet.

URL (Uniform Resouce Locator) É um termo utilizado para nominar um Localizador Uniforme de Recursos. Um URL nada mais é do que o endereço de um determinado recurso exclusivo na web. Em teoria, cada URL válido aponta para um recurso exclusivo. Esses recursos podem ser uma página HTML, um documento CSS, uma imagem, etc.

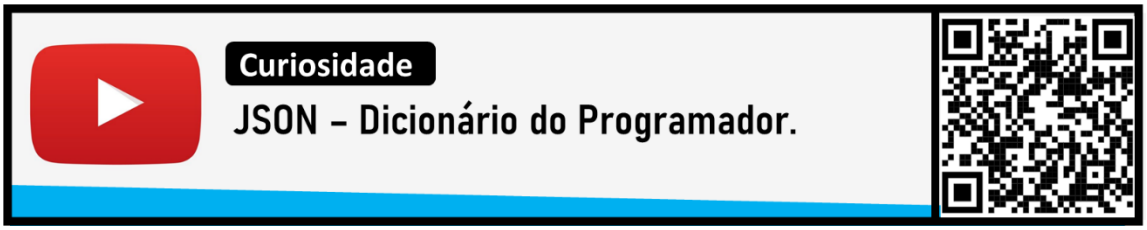
RPC - Remote Procedure Call (Chamada de Procedimento Remoto), é um protocolo que um programa pode usar para solicitar um serviço de um programa localizado em outro computador em uma rede sem ter que entender os detalhes da rede. O RPC é usado para chamar outros processos nos sistemas remotos, como um sistema local. RPC usa o modelo cliente-servidor.

SOAP - Simple Object Access Protocol (Protocolo de Acesso a Objeto Simples) é um protocolo usado para trocar dados entre aplicativos que são construídos em diferentes linguagens de programação. SOAP é baseado na especificação XML e funciona com o protocolo HTTP. Isso o torna perfeito para uso em aplicativos da web.

REST - Representational State Transfer (Transferência de Estado Representacional), é um estilo de arquitetura que fornece padrões entre sistemas de computador na web, tornando mais fácil para os sistemas se comunicarem entre si.



JSON - JavaScript Object Notation é um formato baseado em texto padrão para representar dados estruturados com base o JavaScript. É frequentemente usado para transmitir dados em aplicativos da web (por exemplo, enviar alguns dados do servidor para o cliente, para que possam ser exibidos em uma página da web ou vice-versa).



CLASSIFICAÇÃO DOS WEBSITES

Na internet encontramos diversas possibilidades tanto de uso, como de criação, os usuários da internet possuem propósitos de uso dos mais diversos e particulares entre si, por isso é delicado classificar cada um deles, devido a novas tecnologias aparecerem rapidamente em nosso cotidiano. Mas temos a classificação mais conhecida de cada tipo de websites e os seus propósitos, vejamos alguns exemplos:

Portais web - são sites com informações agrupadas de variadas áreas tais como os portais de notícias, os de tecnologia, e também os portais para ramos mais específicos como os de games, mundo geek, esportes e etc.

Buscadores - são usados como o próprio nome sugere, para realizar buscas na web através de palavras chave, com a evolução da web, automaticamente que os buscadores também evoluíram, facilitando as buscas com tecnologias cada vez mais assertivas e eficientes. O buscador mais conhecido é o Google.

Sites corporativos - são de uso institucional ou corporativo, onde costumam ser a homepage de uma determinada empresa, ou até mesmo de uma instituição, contendo informações gerais da área de prestação de serviço, história da instituição, contatos, produtos entre outras informações. Um exemplo é o site da Coca Cola é um site corporativo da empresa contendo informações que os administradores julgam importantes.

Sites educativos - são usados com o intuito de publicar conteúdos de diversos tipos e finalidades para que cheguem a uma didática eficiente para o processo de aprendizagem de seus usuários. Na internet encontramos diversos tipos de sites educativos, desde conteúdos gratuitos, como também cursos EAD (Ensino à Distância) como instituições de ensino superior.

Sites de rede sociais - são usados com um ambiente onde os usuários possam compartilhar informações, interesses pessoais, festas, amigos, viagens; ou redes sociais de perfis do tipo profissional, estudantil entre outros.

Blogs - são sites criados para publicações de artigos, pesquisas e até mesmo de "fofocas", devido a facilidade de criar um blog, qualquer usuário da internet pode criar seu próprio blog e fazer seus artigos com suas publicações.

Sites da imprensa - são revistas e jornais tradicionais que além da versão impressa, também disponibilizam seu conteúdo de forma online. Esses sites tem o papel importante de manter a informação atualizada.

Sites bancários - são sites de bancos e entidades financeiras que oferecem seus serviços através da internet. Esses sites possuem complementos de segurança de dados, para maior confiabilidade e interação, proporcionando aos usuários segurança nas transações bancárias.

E-Commerce - são sites de lojas online que geralmente oferecem seus produtos e serviços de forma presencial. As lojas online tiveram um crescimento exponencial nos últimos anos devido a tecnologia que usamos atualmente, facilitando a criação e gerenciamento de seus produtos e implementando complementos que facilitam a interação com os usuários.

Sites de aplicativos web - são aplicativos online de softwares, que outrora necessitavam de uma instalação no seu computador para o uso, através desses serviços se tornaram online como por exemplo o Google Docs., que antes da sua criação não era possível usarmos processadores de texto de modo online, sem prévia instalação no computador.

Sites multimidia - são sites dedicados a transmissão de áudio e video, são muitos sites que utilizam esse formato como Youtube, Vimeo entre outros, atualmente o crescimento de podcasts migraram para o uso dessas plataformas para armazenar e disponibilizar seu conteúdo de forma mais efetiva.

RESPONDA

1) O que significa WEB?

2) O que é Web Services?

3) Qual formato é o mais usado na lista de serviços web?

4) Defina U.R.L

5) Qual a utilização de uma API?

6) Quais são as limitações da Web 1.0?

7) A Web 2.0 foi a grande precursora na utilização das redes sociais e interatividade nos sites, qual a característica da Web 2.0 que tornou isso possível?

8) Qual as características da Web 3.0?

9) Entre as evoluções da Web (1.0,2.0 e 3.0), quais as diferenças entre elas?

10) O que é um site E-commerce

DESAFIO

Mostre a importância da evolução da web com suas próprias palavras.

TEMA 03 - INTRODUÇÃO AO HTML

Figura 7: Logo HTML 5 – Fonte: Wikipedia

TERMINOLOGIA

Html - Linguagem de Marcação de Hipertexto é uma linguagem usada em criação de sites, ela é responsável por mostrar no navegador como seu conteúdo será organizado, ou seja, é uma linguagem que transforma o conteúdo "cru" e faz a organização (marcação) desse conteúdo no navegador.

Se o site tem imagem, é o html que determina como e onde ela é mostrada, em caso de tabela também, o html mostra no navegador a tabela de acordo como está no próprio documento html, se o documento possui metadados, é o html que mostra como serão carregados. Através do html são possíveis integrações com outras linguagens, tanto de marcação como o CSS (Cascating Style Sheet) e o JavaScript (programação).

Toda página escrita em HTML deve ser salva *(.html ou .htm)*A linguagem html possui algumas características próprias e distintas na qual possamos identificar, como as “tags”, as tags (etiquetas – português) em html são comandos que se encontram entre os sinais de menor que “<” , e o maior que “>” , e tem o objetivo determinar uma ação de identificação de tópico. O html na sua atual versão que é a 5.0, teve algumas atualizações de grande importância para as páginas web, uma delas é como o html trata as informações nessa versão, não apenas na estrutura e organização, mas também indicando o significado das informações. Vamos conhecer as principais tags encontradas nos documentos html.

TAGS ESTRUTURAIS

<html> ... </html> - Esta tag indica que a página foi escrita em html. Ela aparece na primeira linha da página. É utilizada principalmente para mostrar que a página usa HTML 5 – a versão mais recente da linguagem. Também conhecido como elemento root, esta tag pode ser considerada como uma tag pai para qualquer outra página. Ela é a primeira e a última tag usada nas páginas html.

<head> ... </head> - Esta tag é utilizada para especificar o cabeçalho da página, onde nada entre essa tag será mostrado no documento. É usada também para especificar metadados sobre a página. Inclui o nome da página (tag <title> ... </title>), suas dependências (JavaScript e CSS), é dentro dessa tag que colocamos links de scripts em Javascript e links de estilos em CSS.

<title> ... </title> - Como o nome sugere, essa tag contém o título/nome da página. Você pode ver isso na barra de navegação do seu navegador para qualquer página aberta. Mecanismos de busca utilizam esta tag para extrair o assunto da página, o que é bastante conveniente para o ranqueamento e melhor visualização da página.

<body> ... </body> - Tudo que é mostrado na página, ou seja, tudo o que o usuário vê escrito, está dentro da tag body, nela contém todo o conteúdo da página.

<! -- Comentário em HTML --> - Temos um método onde podemos colocar comentários no código em html, isso facilita na organização do seu documento em html. Os comentários não são mostrados na página dentro do navegador, só é mostrado no código auxiliando o próprio desenvolvedor, ou outros desenvolvedores que irão utilizar aquele documento.

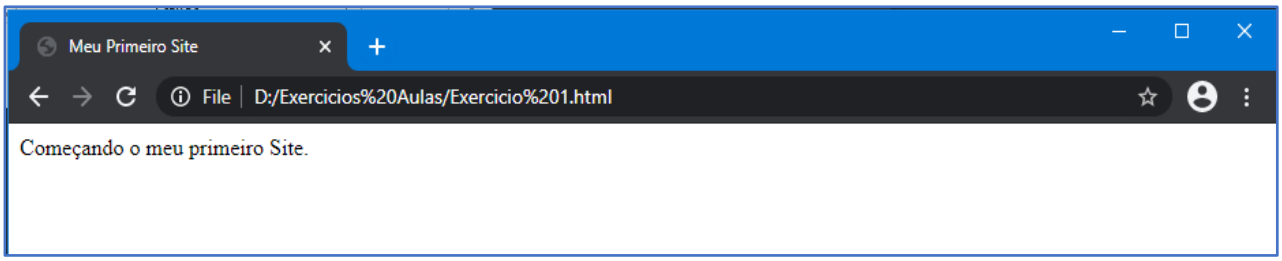
** ... -** Essa tag é muito usada para agrupar elementos de um site, que facilita a identificação nas folhas de estilos (CSS). Pode ser atribuída alguns atributos, são eles:

- ID - Define um identificador
- CLASS - Define uma classe.
- TITLE - Define um título.
- WIDTH – Define a largura.
- HEIGHT – Define a altura.

Veja um exemplo:

```
<html>
    <head>
        <title>Meu Primeiro Site </title>
    </head>
        <body>
            Começando o meu primeiro Site.

        </body>
</html>
```

Esse código em HTML será exibido no navegador da seguinte forma:



Podemos visualizar o título da página na barra de navegação do navegador, o endereço do site (nesse caso o endereço do site onde se encontra salvo no computador) e o corpo da página escrito “Começando o meu primeiro Site”. É assim que funciona uma estrutura básica em HTML, vamos conhecer um pouco mais das novas tags (tags semânticas), que vieram junto com a nova versão do HTML 5.0.

<header> ... </header> - Determina o cabeçalho da página.

<footer) ... </footer> - Especifica o rodapé da página.

<main> ... </main> - Marca o conteúdo principal da página.

<article> ... </article> - Denota um artigo na página. Uma página pode ter vários artigos, desse modo é importante inserirmos o atributo *name* em cada artigo da página.

<aside> ... </aside> - Determina o conteúdo mostrado na lateral da página.

<section> ... </section> - Especifica uma seção da página, semelhante ao *article*, podem ter varias seções na página, sendo necessário o uso do atributo *name* para melhor identificá-las.

<figure> ... </figure> - Essa tag é reservada para diagramas, gráficos em geral.

<figcaption> ... </figcaption> - É usada para colocar uma descrição em uma figura.

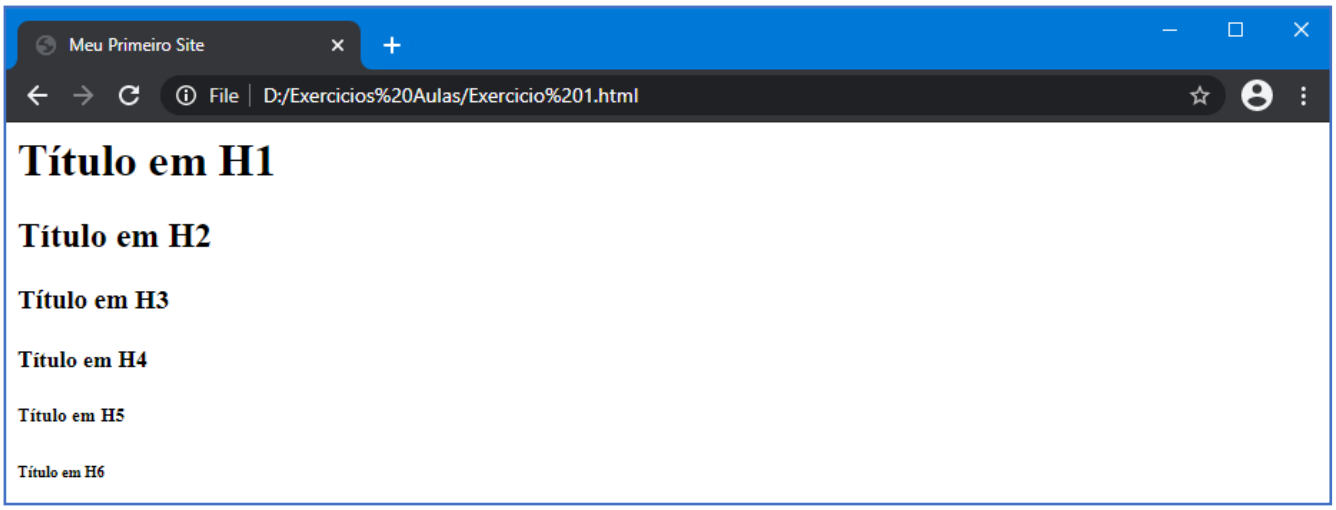
<nav> ... </nav> - É utilizada para separar os links de navegação de uma página, menus de links geralmente ficam separados por essa tag.

<wbr> - É usada para quebra de linha dentro do conteúdo, semelhante ao
, porém a diferença é que essa tag pode ser usada dentro de um texto especifico para melhorar sua visualização no navegador.

- **TAGS DE TEXTO**

<h1 ... h6> ... </h1 ... h6> - São seis variações de escrita de cabeçalho. O <h1> possui o maior tamanho da fonte, enquanto o <h6> o menor.

<hgroup> ... </hgroup> - É uma tag usada para agrupar elementos de títulos (<*h1 ...h6*>) de forma semântica, ajudando os sites de buscas para entender os títulos e facilitar em achar o site.



** ... -** Esta tag indica o começo de um parágrafo, textos sem formatação são colocados dentro dessa tag.

<b> ... <b/> - Deixa o texto em **negrito.**

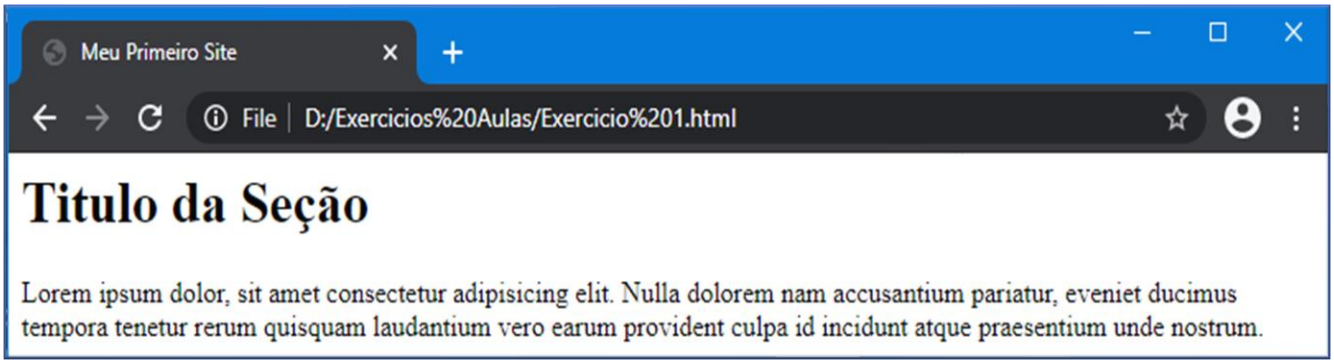
** ... -** Deita o texto em *itálico.*

<s> ... </s> - Deixa o texto sublinhado.

Veja um exemplo:

```
1  <html>
2      <head>
3          <title>Meu Primeiro Site </title>
4      </head>
5              <body>
6                  <h1>Titulo da Seção</h1>
7                  <p>Lorem ipsum dolor, sit amet consectetur adipisicing elit.
8                   Nulla dolorem nam accusantium pariatur, eveniet ducimus tempora
9                   tenetur rerum quisquam laudantium vero earum provident culpa id
10                   incidunt atque praesentium unde nostrum.</p>
11             </body>
12 </html>
```

Esse código html mostrará no navegador da seguinte forma:



**
 -** Esta tag é responsável pela quebra de linha da página. Usada quando você quer escrever em uma nova linha. Essa tag não precisa ser fechada, por isso é usada com um comando único.

- **LINKS E ÂNCORA**

** Nome da âncora -** Esta é uma tag âncora. Primariamente utilizada para inclusão de hyperlinks.

** E-mail -** Tag dedicada para o envio de emails.

Número de telefone - Tag âncora para menção de números de contato. Como os números são clicáveis, é particularmente útil para usuários de celular.

** ... -** Esta tag pode ser utilizada para navegação rápida para outra parte da página.

- **IMAGENS**

A tag <img/> define onde será mostrada uma imagem na página, ela possui propriedades para melhor aplicação das imagens, vejamos quais são:

<img /> - Tag para mostrar imagens em uma página.

src=”URL” - A URL ou caminho onde a imagem está localizada no seu computador ou na web.

alt=" texto" - O texto escrito aqui é exibido quando o usuário passa o mouse em cima da imagem. Pode ser utilizada para dar detalhes adicionais sobre a imagem, facilitando também para os motores de busca.

height=" altura da imagem" - Essa tag especifica a altura da imagem, pode ser em pixels ou porcentagem.

width=" largura da imagem" - Essa tag especifica a largura da imagem, pode ser em pixels ou porcentagem.

border=" ..." - Tag que altera a grossura da borda da imagem, caso não definida ela adota o valor padrão que é 0.

<área/> - Especifica a área de imagem do mapa.

shape=" ..." - Formato da área.

coords=" ..." - Coordenadas da informação vital do formato da área. Exemplo: vértices para triângulos, centro/radianos para círculos.

<map> ... </map> - Indica uma imagem interativa (clicável).

- **LISTAS**

** ... -** Tag usada para lista ordenada ou numerada de itens.

** ... -** Tag para listas não ordenadas.

** ... - Item** individual como parte da lista, pode ser usada nas duas tags de listas.

Podemos fazer alterações nas especificações padrão das listas, alterando valores dos atributos, e assim personalizando as listas. Conheça alguns atributos das listas:

reversed - Este atributo booleano especifica que as partes desta lista serão especificadas em ordem reversa, isto é, a menos importante será listada primeiro.

start - Este atributo inteiro especifica o valor inicial para a numeração dos itens da lista. Embora o tipo de ordenação dos elementos possa ser com algarismos romanos, tal como XXXI, ou letras, o valor inicial sempre é representado como um inteiro. Para iniciar a contagem a partir da letra "C", utilize <ol start="3">.

Type;

Indica o tipo de numeração:

'a' – Indica letras minúsculas.

'A' – Indica letras maiúsculas.

'i' – Indica algarismos romanos minúsculos.

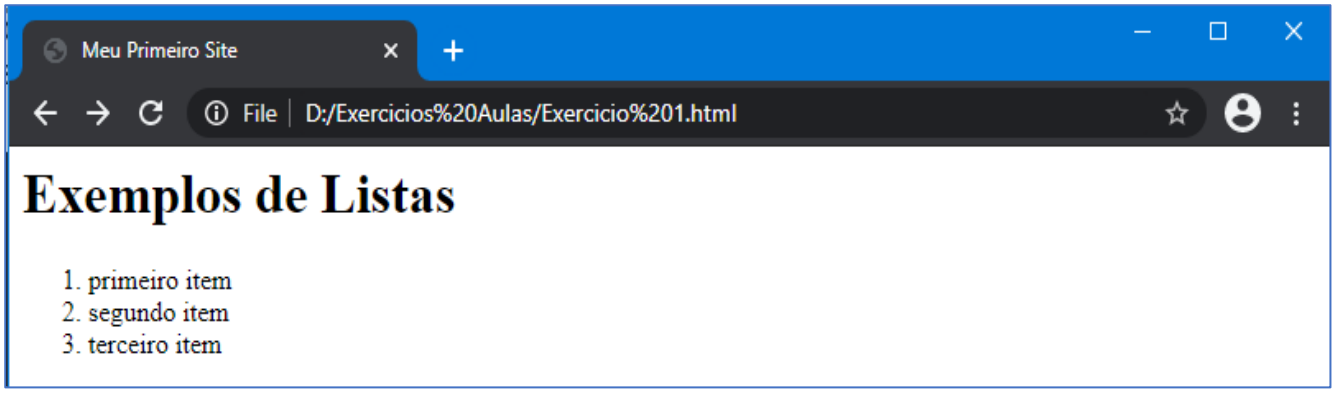
'I' – Indica algarismos romanos maiúsculos.

'1' – Indica números (padrão).

O tipo de marcação é usado na lista inteira, a menos que um atributo type diferente seja utilizado dentro do elemento <li>. Vejamos alguns exemplos do uso das listas em html:

```
<html>
    <head>
        <title>Meu Primeiro Site </title>
    </head>
            <body>
                <h1>Exemplos de Listas</h1>
                <p>
                    <ol>
                        <li>primeiro item</li>
                        <li>segundo item</li>
                        <li>terceiro item</li>
                    </ol>
                </p>

            </body>
</html>
```

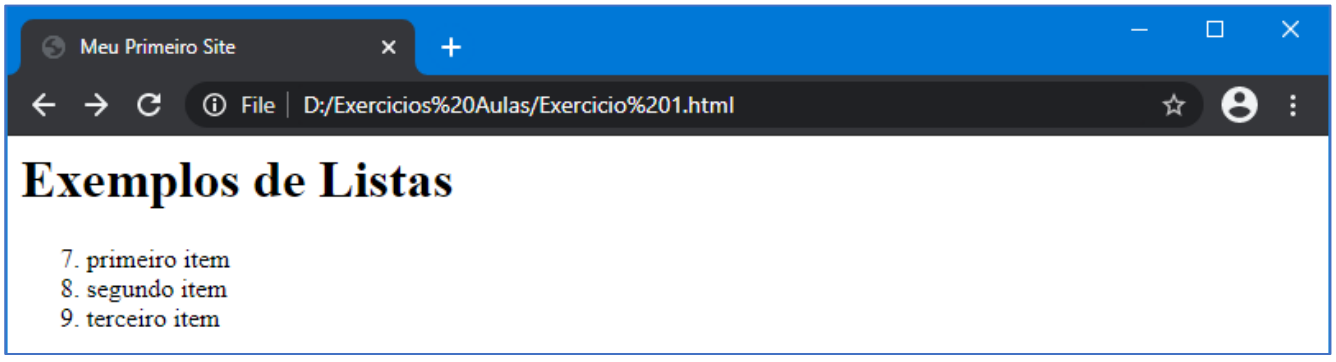
A saída no navegador será assim;



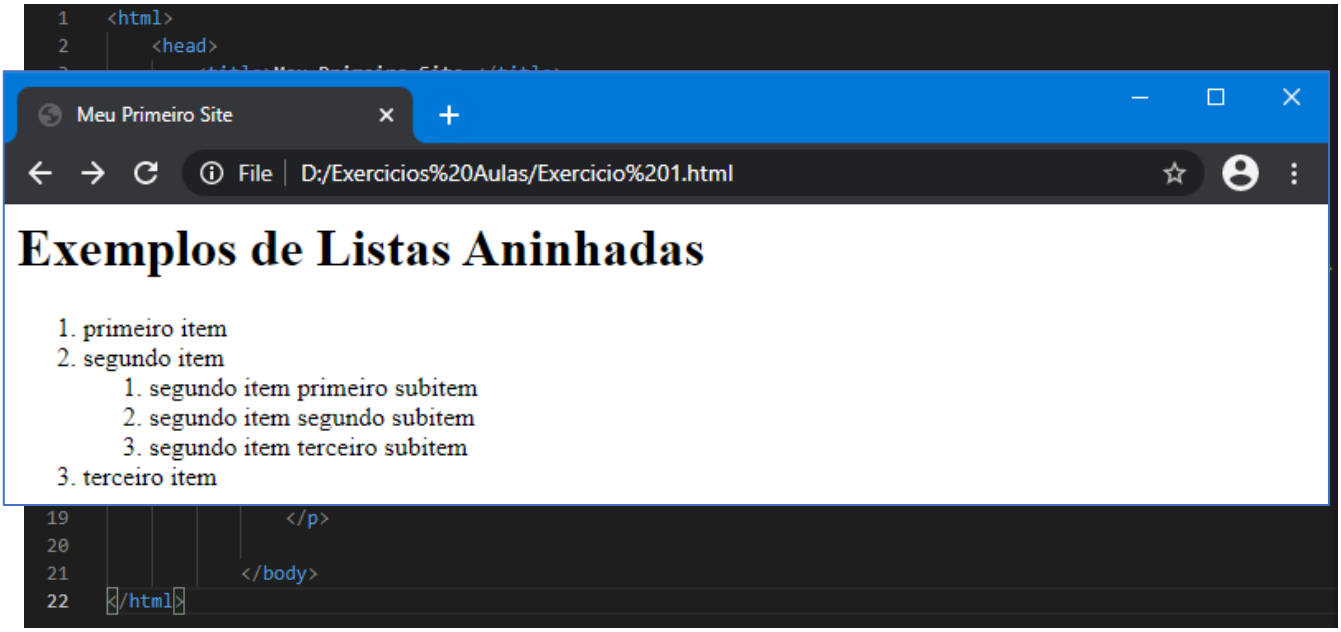
Usando o atributo **Start**;

```
<html>
    <head>
        <title>Meu Primeiro Site </title>
    </head>
            <body>
                <h1>Exemplos de Listas</h1>
                <p>
                    <ol start="7">
                        <li>primeiro item</li>
                        <li>segundo item</li>
                        <li>terceiro item</li>
                      </ol>
                </p>

            </body>
</html>
```

No navegador;



Note que o atributo *start* indica o início do item na lista, nesse caso foi iniciado a partir do número 7. Listas aninhadas também podem ser utilizadas em html. Exemplo:

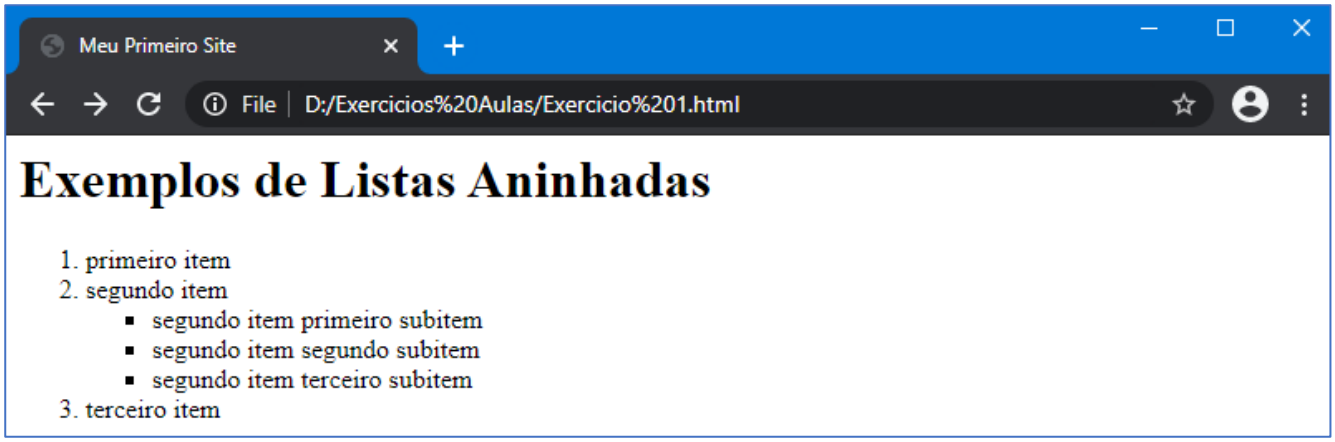


Saída no navegador:

Listas OL e UL aninhadas:

```html
<html>
    <head>
        <title>Meu Primeiro Site </title>
    </head>
            <body>
                <h1>Exemplos de Listas Aninhadas</h1>
                <p>
                    <ol>
                        <li>primeiro item</li>
                        <li>segundo item      <!-- Observe que a tag de fechamento </li> não é colocada aqui! -->
                         <ul type="square">
                            <li>segundo item primeiro subitem</li>
                            <li>segundo item segundo subitem</li>
                            <li>segundo item terceiro subitem</li>
                        </ul>
                        </li>               <!-- Aqui está a tag de fechamento </li> -->
                         <li>terceiro item</li>
                    </ol>
                </p>

            </body>
</html>
```

Saída no navegador:



RESPONDA:

1) O que é HTML?

2) O que são tags?

3) A tag <head> é essencial para uso de arquivos externos no documento HTML, quais tipos de arquivos posso usar para fazer o link através dessa tag?

4) Qual a função da tag <a>?

5) O HTML vem se atualizando no decorrer dos anos, qual versão estamos atualmente do HTML?

6) Qual tag utilizo para inserir uma imagem na página web?

7) Sabemos que as tags <ul> e <ol>, são tags de listas HTML, qual a função de cada uma delas respectivamente?

8) Como escrevo um comentário em um documento HTML?

9) Quais são as tags de estrutura do HTML?

10) Como devo salvar um documento HTML?

DESAFIO

Crie uma página em html como se fosse seu perfil com sua foto, nome e uma seção que diz um pouco sobre você.

Dica: Para criar documento em HTML é necessário um editor de texto, deixei para download o Visual Studio Code, da Microsoft. É um software leve e intuitivo para começar no HTML.

Opção 2: Interpretador pela internet: **https://playcode.io/new/**

TEMA 04 – TABELAS

Figura 9: Estrutura tabelas em HTML – Fonte: homehost.com.br

O QUE SÃO TABELAS

Tabelas são listas de dados compostas por linhas e colunas de forma bidimensional, ou seja, em duas dimensões. São muito utilizadas para apresentar dados de uma forma organizada e profissional. No HTML é possível usarmos tabelas para a organização de dados, mostrando a importância desses dados em nossa página de forma organizada e sistemática. De forma fácil é possível usarmos esse recurso muito bem estruturado e organizado para apresentação de dados em lista.

Vamos conhecer a estrutura básica de uma tabela, para codificarmos uma tabela em HTML.

NOME	IDADE	PROFISSÃO
Eduardo	25	Motorista
Camila	32	Gerente de Vendas
Wesley	18	Ator

Como podemos ver na tabela acima, temos 4 linhas e 3 colunas, com o total de 12 células, sendo a primeira linha sendo usada como título. Agora que sabemos o que é uma tabela podemos então entender como criar uma tabela em html.

- **ESTRUTURA HTML DAS TABELAS**

A tag usada para criar uma tabela é a tag **<table>**, que posteriormente fechada com **</table>**. É entre essas tag que incluímos as tags referente as linhas e colunas da tabela, ou seja, as células da tabela. As tags que formam a estrutura de uma tabela html, são as tags **<tr>** que significa *table row*, usada para definir uma linha, e a tag **<td>** que significa *table data*, usada para criar uma célula. Sendo assim podemos notar que as colunas em uma tabela html são feitas automaticamente através da quantidade de células incluídas dentro de uma linha.

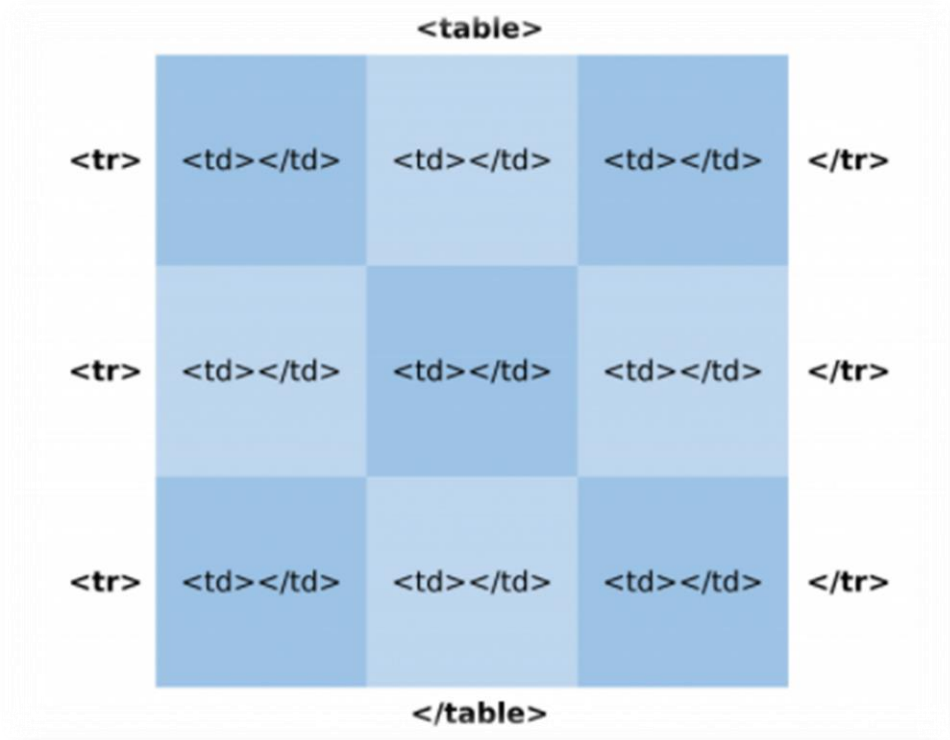
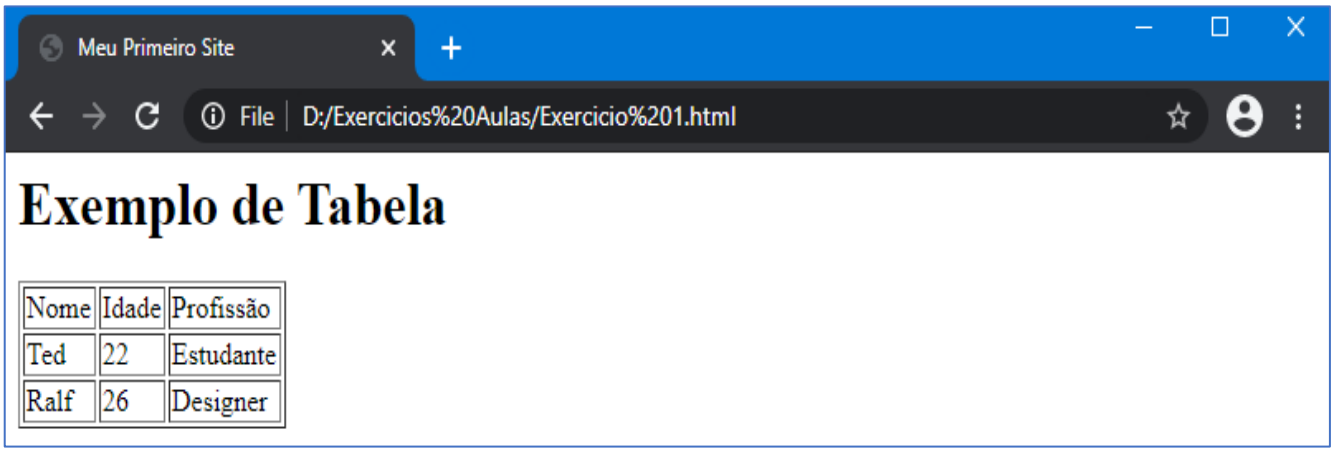


Figura 10: Tag das tabelas / **Fonte:** homehost

Sendo assim vamos criar nossa primeira tabela em html, usando como referência o exemplo acima:

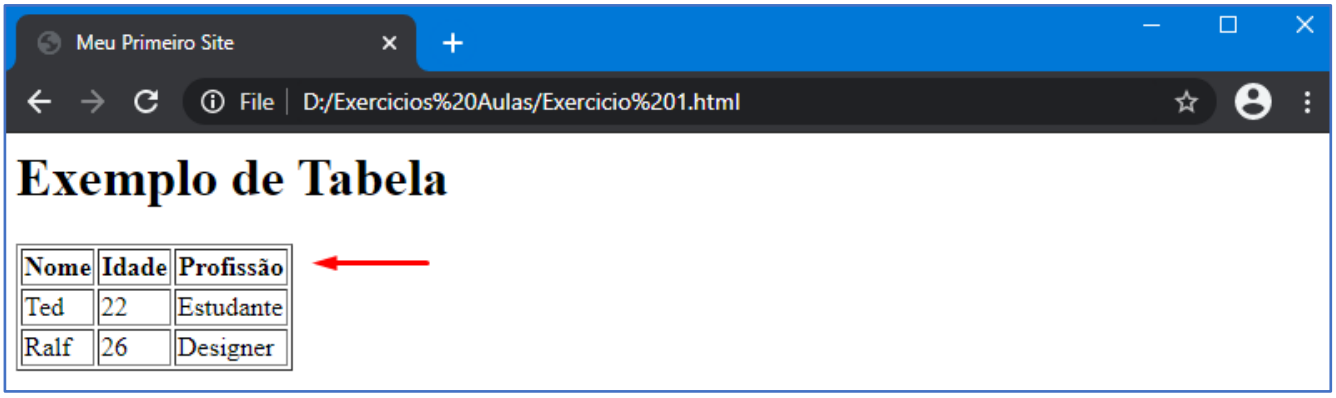
```html
<html>
    <head>
        <title>Meu Primeiro Site </title>
    </head>
            <body>
                <h1>Exemplo de Tabela</h1>
                <p>
                    <table border="1">
                        <tr>
                            <td>Nome</td>
                            <td>Idade</td>
                            <td>Profissão</td>
                        </tr>
                        <tr>
                            <td>Ted</td>
                            <td>22</td>
                            <td>Estudante</td>
                        </tr>
                        <tr>
                            <td>Ralf</td>
                            <td>26</td>
                            <td>Designer</td>
                        </tr>
                    </table>
                </p>

            </body>
</html>
```

Podemos incluir uma ou mais células representando títulos, ganhando certo destaque em relação às demais células, usamos a tag **<th>** que significa *table header.* Vamos aplicar essa tag no mesmo exemplo usado acima e vejamos o resultado:



```html
<html>
    <head>
        <title>Meu Primeiro Site </title>
    </head>
            <body>
                <h1>Exemplo de Tabela</h1>
                <p>
                    <table border="1">
                        <tr>
                            <th>Nome</th>
                            <th>Idade</th>
                            <th>Profissão</th>
                        </tr>
                        <tr>
                            <td>Ted</td>
                            <td>22</td>
                            <td>Estudante</td>
                        </tr>
                        <tr>
                            <td>Ralf</td>
                            <td>26</td>
                            <td>Designer</td>
                        </tr>
                    </table>
                </p>

            </body>
</html>
```

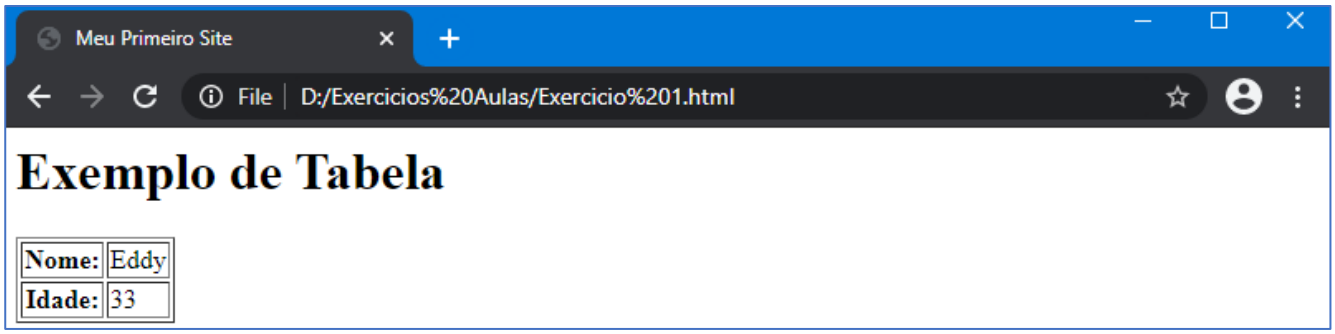
Note que alteramos todas as tags <td> da primeira linha para as tags <th>, dando um destaque no título das células. O resultado no navegador sairá assim:



Podemos aplicar a tag <th> para títulos verticais também, ou seja, na primeira coluna. Dessa forma vamos ver um exemplo olhando a tabela a seguir:

```html
<html>
    <head>
        <title>Meu Primeiro Site </title>
    </head>
            <body>
                <h1>Exemplo de Tabela</h1>
                <p>
                    <table border="1">
                        <tr>
                            <th>Nome:</th>
                            <td>Eddy</td>
                        </tr>
                        <tr>
                            <th>Idade:</th>
                            <td>33</td>
                        </tr>
                    </table>
                </p>

            </body>
</html>
```

No navegador temos o seguinte resultado:

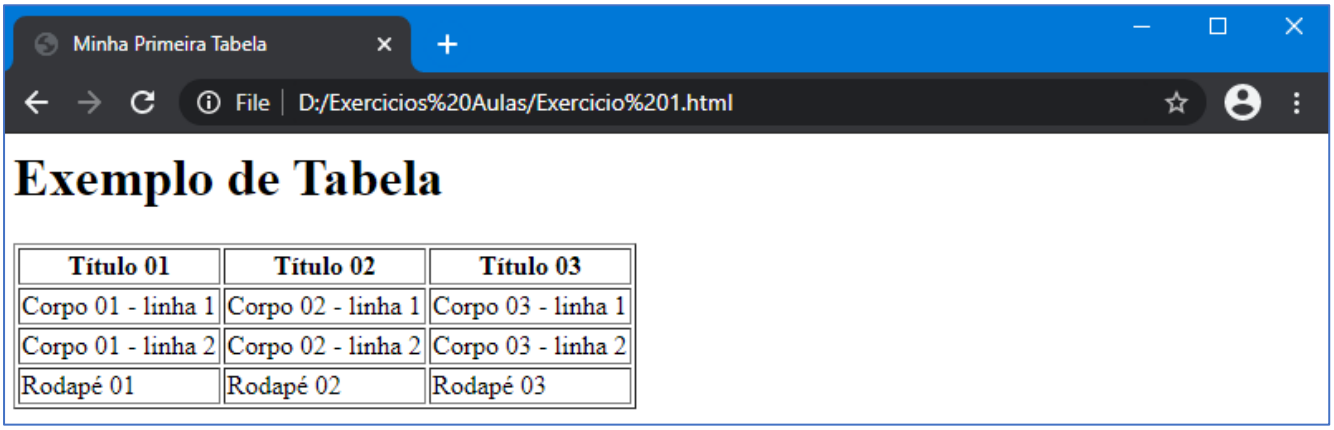


Anteriormente, vimos como funciona a estrutura básica de tabelas em html, com as tags **<tr>** e **<td**>, e as células título **<th>**. Além dessas, também há tags usadas para melhorar a estrutura da tabela, a tag **<thead> ... </thead>** por exemplo, é usada para representar o cabeçalho da tabela, que em geral é composta por títulos.

Temos também a tag **<tbody> ... </tbody>**, usada para representar o corpo da tabela e a tag **<tfoot> ... <tfoot>** que por sua vez é utilizada para representar o rodapé da tabela. Essas tags tem como principal função manter uma semântica adequada para as tabelas, vejamos um exemplo com essas tags aplicada no código html.

```html
<html>
    <head>
        <title>Minha Primeira Tabela  </title>
    </head>
        <body>
            <h1>Exemplo de Tabela</h1>
            <p>
                <table border="1">
                    <thead>
                    <tr>
                        <th>Título 01</th>
                        <th>Título 02</th>
                        <th>Título 03</th>
                    </tr>
                    </thead>
                    <tbody>
                    <tr>
                        <td>Corpo 01 - linha 1</td>
                        <td>Corpo 02 - linha 1</td>
                        <td>Corpo 03 - linha 1</td>
                    </tr>
                    <tr>
                        <td>Corpo 01 - linha 2</td>
                        <td>Corpo 02 - linha 2</td>
                        <td>Corpo 03 - linha 2</td>
                    </tr>
                    </tbody>
                    <tfoot>
                        <td>Rodapé 01</td>
                        <td>Rodapé 02</td>
                        <td>Rodapé 03</td>
                    </tfoot>
                </table>
            </p>

        </body>
</html>
```

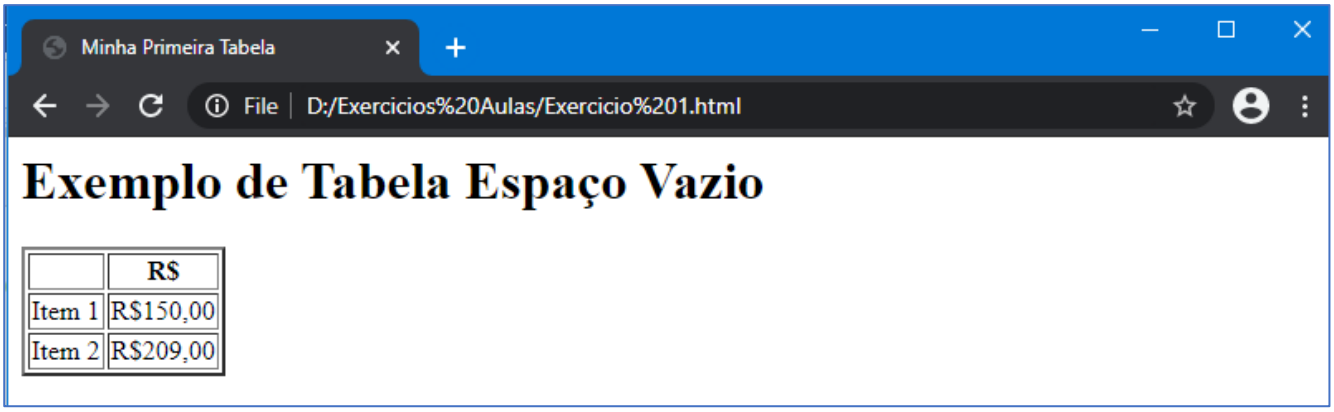
O resultado do código no navegador fica assim:



Em alguns casos, seja necessário o uso de uma célula vazia (em branco), podemos usar as tags **<td>** ou **<th>** sem incluir nenhum conteúdo, porém alguns navegadores não conseguem ler adequadamente nesse formato, então podemos utilizar uma técnica de espaço vazio usando o *** *** (inclui um valor vazio no html). Segue o exemplo abaixo:

```
<html>
    <head>
        <title>Minha Primeira Tabela  </title>
    </head>
            <body>
                <h1>Exemplo de Tabela Espaço Vazio</h1>
                <p>
                    <table border="2">
                        <tr>
                            <th> </th>
                            <th>R$</th>
                        </tr>
                        <tr>
                            <td>Item 1</td>
                            <td>R$150,00</td>
                        </tr>
                        <tr>
                            <td>Item 2</td>
                            <td>R$209,00</td>
                        </tr>
                    </table>
                </p>

            </body>
</html>
```

No navegador o resultado é o seguinte:

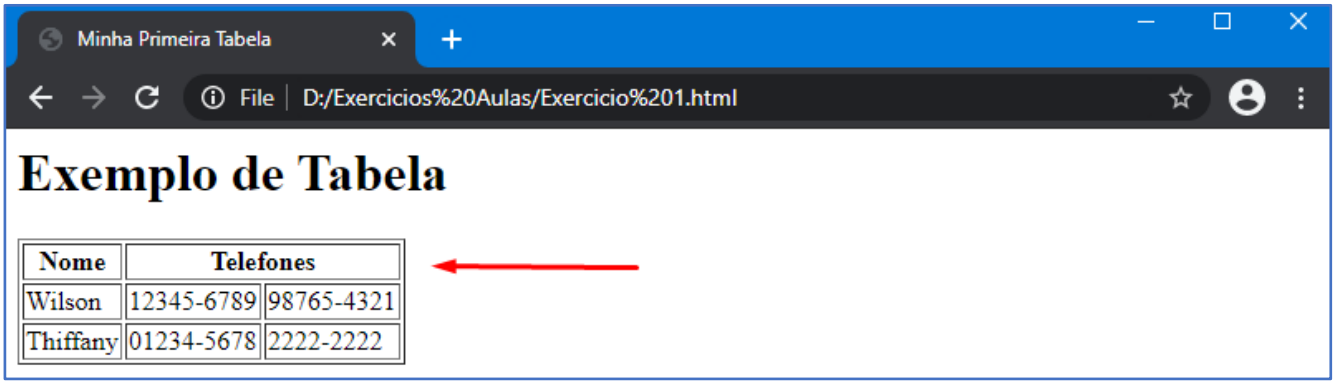


Em alguns casos necessitamos utilizar células mescladas, ou seja, criar uma célula que abrange mais que uma linha ou coluna. Podemos então utilizar os atributos colspan que faz a mesclagem de colunas e rowspan que faz a mesclagem das linhas. O colspan é usado de forma que é necessário indicar o número de colunas a serem mescladas (colspan=”x”, onde o x representa o número de colunas) e o rowspan usado de forma semelhante só que indicando o número de linhas para serem mescladas; (rowspan=”x”, onde o x representa o número de linhas).

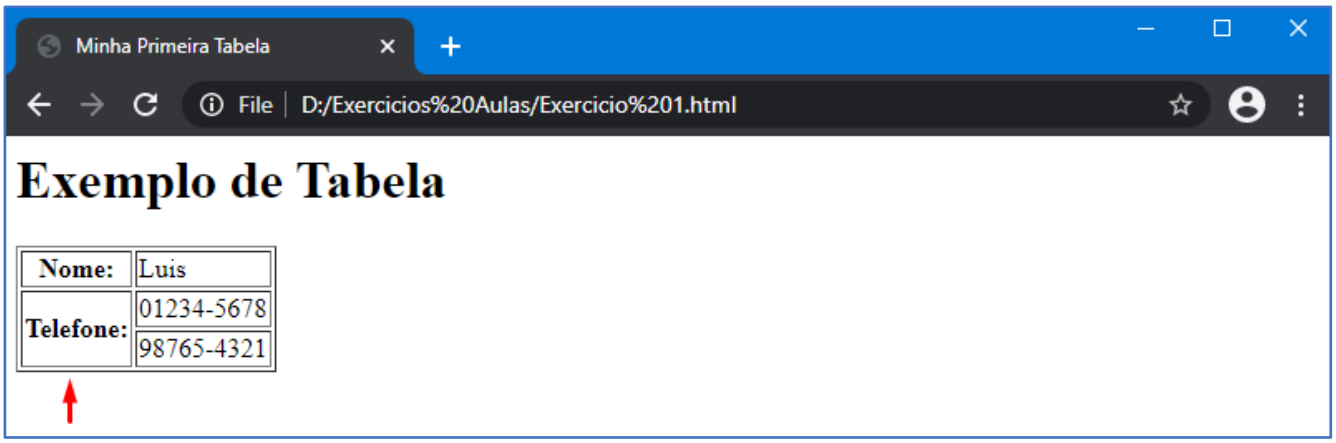
Vamos ver um exemplo usando o colspan:

```html
<html>
    <head>
        <title>Minha Primeira Tabela  </title>
    </head>
            <body>
                <h1>Exemplo de Tabela</h1>
                <p>
                    <table border="1">
                        <tr>
                            <th>Nome</th>
                            <th colspan="2">Telefones</th>
                        </tr>
                        <tr>
                            <td>Wilson</td>
                            <td>12345-6789</td>
                            <td>98765-4321</td>
                        </tr>
                        <tr>
                            <td>Thiffany</td>
                            <td>01234-5678</td>
                            <td>2222-2222</td>
                        </tr>
                    </table>
                </p>

            </body>
</html>
```

No navegador a saída é dessa forma:



Podemos notar que a coluna ‘telefones’ foi mesclada usando o espaço de 2 colunas conforme definida no código html. Veremos um exemplo com o atributo **rowspan**:



O exemplo acima mostra duas linhas mescladas com o atributo **rowspan**. Existem mais algumas propriedades para usarmos nas tabelas, porém muitas delas foram alteradas pelo uso de folhas de estilos para a formatação (CSS), no capitulo de CSS vamos ver algumas formatações adicionais nas tabelas.

RESPONDA:

1) O que são tabelas HTML?

2) As tabelas são formadas pelas _____ e pelas _____.

3) Como é chamado um elemento individual da tabela?

4) Para criamos uma tabela em HTML, devemos começar com a tag _____ .

5) Quais as tag de estrutura de uma tabela HTML?

6) Como aumentar as dimensões de uma tabela em HTML?

7) Para que serve os atributos colspan e rowspan?

8) Qual atributo preciso para definir a bordar de uma tabela HTML?

9) Qual a função das tags <tr> e <td>?

10) Qual a finalidade da tag <th>?

DESAFIO

Criar uma tabela html com 5 colunas (nome, idade, comida favorita, filme favorito e serie favorita) e 8 linhas, com título e rodapé.

TEMA 05 – FORMULÁRIOS

Figura 11: Formulários HTML – Fonte: homehost

O QUE SÃO FORMULÁRIOS?

Os formulários em html são usados como uma das principais ferramentas de interatividade entre os usuários e um site ou aplicativo. Através deles que os usuários enviam dados para o site, sendo que na maior parte do tempo esses dados são enviados para um servidor web, porém a página web pode requisitar para usa-los em qualquer momento. Em geral quando necessitamos que o usuário forneça algum tipo de informação, é pelo formulário que fazemos esse pedido, que por sua vez envia os dados para um banco de dados, para o armazenamento ou utilização em outra página. Existem inúmeras finalidades na qual podemos usar os formulários, vamos conhecer as principais tags e aprender a utilizar o formulário em html.

A principal diferença entre um formulário html e um documento simples html é que na maioria das vezes, o dado coletado é enviado ao servidor. Nesse caso, é necessário ter um servidor web configurado para receber e processar os dados, vamos aprender a configurar um servidor web mais adiante, nesse capitulo vamos nos atentar para o formulário e seus atributos. Os formulários necessitam de validações em seus campos, é até possível fazer essas validações em html, mas no html 5, é indicado fazer as validações no código Javascript incorporado no formulário.

Antes de codificar um formulário, é de grande ajuda ter ele desenhado em algum arquivo, ou até mesmo numa folha de papel, para facilitar na hora de codificar e não esquecer nenhum campo importante.

CRIANDO UM FORMULÁRIO

Todo formulário em html deve começar pela tag **<form>** e terminar fechando-a **</form>**, esse elemento não é visual (quando colocamos somente a tag **<form>** na página, ao executar no navegador não nos mostra nada, é apenas para indicar que naquele local determinado tem um formulário), ela precisa de outros elementos para que o formulário ganhe vida. Vamos conhecer as tags do formulário e seus atributos.

Dentro da tag **<form>** deve ser configurado alguns atributos para indicar qual a finalidade daquele formulário, podemos identificar um formulário através do atributo ***id="nome_do_formulário",*** em casos em que temos mais que um formulário na página , também temos o atributo **method** (método) que define o método *HTTP* para o envio de dados, ele pode ser **method**="*get*" ou **method**="*post*". Vamos ver as diferenças de cada um:

GET = Os dados serão enviados junto com a *URL*, e desta forma, serão visíveis para os usuários, que podem fazer alterações, onde pode gerar alguns problemas e incidentes quanto a segurança dos dados.

POST = Os dados serão enviados no corpo da mensagem *HTTP*, sendo assim serão invisíveis para os usuários, o método mais indicado para envio de dados.

Outro atributo da tag **<form>** é o ***autocomplete***, este atributo permite desabilitar(*off*) o comportamento padrão da maioria dos navegadores modernos, os quais sugerem valores para os campos baseados em preenchimentos anteriores, ou seja, esse atributo desabilita o auto completar dos campos de dados pelos navegadores, mas sempre que se achar necessário ser habilitado pode ser colocado o valor (*on*).

Os formulários em html possui alguns elementos essenciais para seu bom funcionamento, vamos conhecer alguns deles para compreender melhor o uso dos formulários;

INPUT – Com a nova versão do HTML (5.0), esse elemento ganhou novos *types* (tipos), onde possibilita atribuir mais controles, porém ainda temos alguns navegadores que não reconhecem esses novos *types*, adotando assim ao valor padrão **<input type="text">**. O elemento input é um elemento que permite a entrada de vários tipos de dados, seu comportamento é definido pelo atributo *type*, dentre os tipos de entrada de dados usados pela tag **<input>**, podemos destacar alguns:

TEXT – Esse atributo permite a entrada de textos curtos, com limitação de uma linha, não apresenta restrições quanto ao conteúdo utilizado, e não possui validação. Além desse atributo, o ideal é que usamos o atributo *id*, que é usado para estilizar o elemento utilizando o CSS, e o atributo *name*, que serve para determinar o nome do campo, assim podendo ser identificado quando for enviado pro servidor na forma de uma variável, que por sua vez terá o valor igual ao conteúdo do atributo *value.*

FIELDSET – Define um conjunto de campos. Muito usado para agrupar campos no formulário para melhorar a organização.

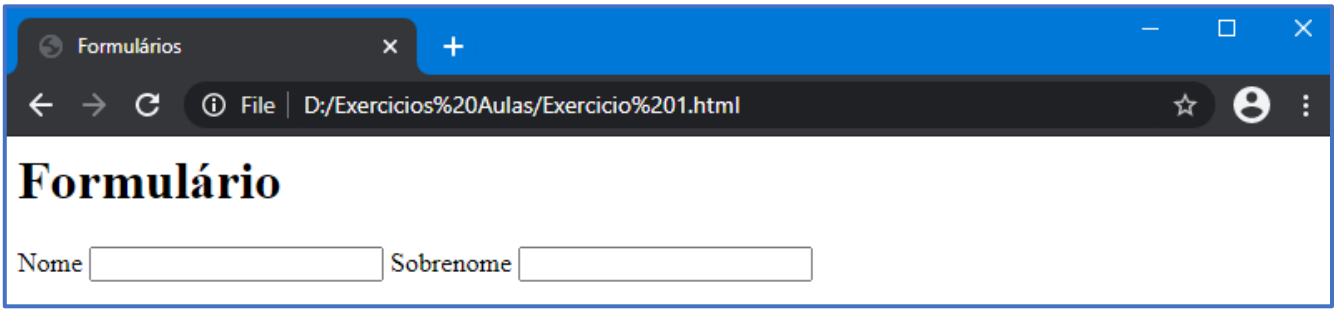
LABEL – Faz a definição de um rótulo para melhorar o controle de campos.

LEGEND – Define um título para um conjunto de campos.

Exemplo do código:

```
<form>
    Nome <input type="text" name="nome" id="txtNone">
    Sobrenome <input type="text" name="sobrenome" id="txtSobrenone">
</form>
```

No navegador:



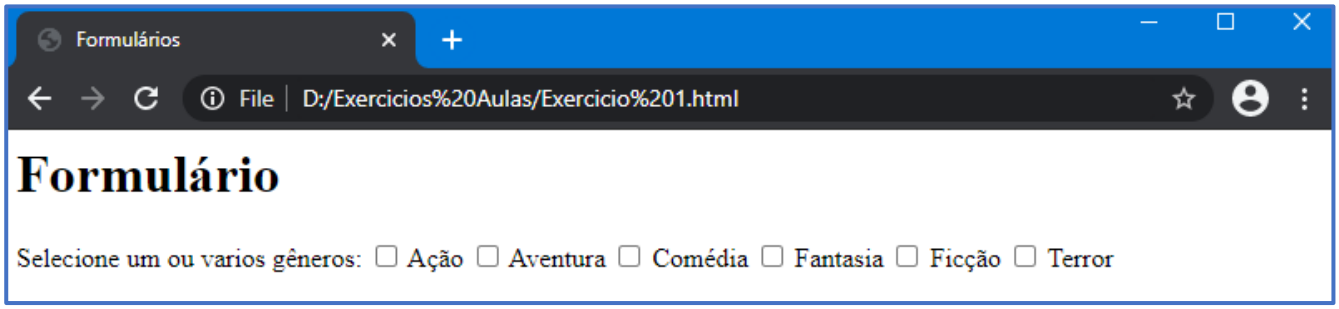
CHECKBOX – Usado juntamente com a tag **input**, esse atributo cria uma variada cadeia de opções selecionáveis, que podem ser escolhidas pelo usuário, permitindo também a seleção de mais de uma opção.

Exemplo do código:

```html
<html>
    <head>
        <title>Formulários  </title>
    </head>
            <body>
                <h1>Formulário</h1>
                <form>
                    Selecione um ou varios gêneros:
                    <input type="checkbox" name="generoFilme" value="1"> Ação
                    <input type="checkbox" name="generoFilme" value="2"> Aventura
                    <input type="checkbox" name="generoFilme" value="3"> Comédia
                    <input type="checkbox" name="generoFilme" value="4"> Fantasia
                    <input type="checkbox" name="generoFilme" value="5"> Ficção
                    <input type="checkbox" name="generoFilme" value="6"> Terror
                </form>

            </body>
</html>
```

Podemos observar que todos os elementos possuem o mesmo valor no atributo “name”, neste caso “*generoFilme*”, mas cada um está associado um valor diferente no atributo *value.*

No navegador:

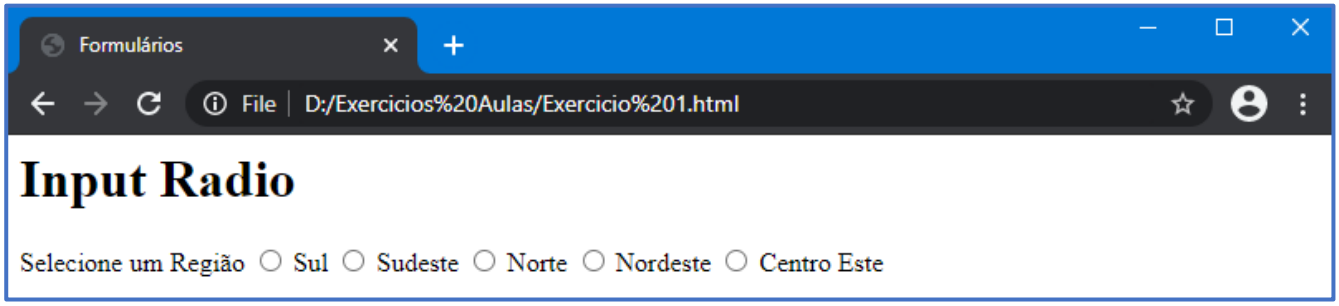


RADIO – Esse atributo é semelhante ao anterior criando uma lista de opções, porém permitindo somente uma opção pra ser selecionada.

Exemplo do código:

```html
<html>
    <head>
        <title>Formulários  </title>
    </head>
            <body>
                <h1> Input Radio </h1>
                <form>
                    Selecione um Região
                    <input type="radio" name="regiao" value="sul"> Sul
                    <input type="radio" name="regiao" value="sudeste"> Sudeste
                    <input type="radio" name="regiao" value="norte"> Norte
                    <input type="radio" name="regiao" value="nordesre"> Nordeste
                    <input type="radio" name="regiao" value="centroeste"> Centro Este
                </form>

            </body>
</html>
```

Neste caso é importante lembrar que todos os elementos *radio*, que estão relacionados devem possuir o mesmo *name*. Confira o resultado no navegador.

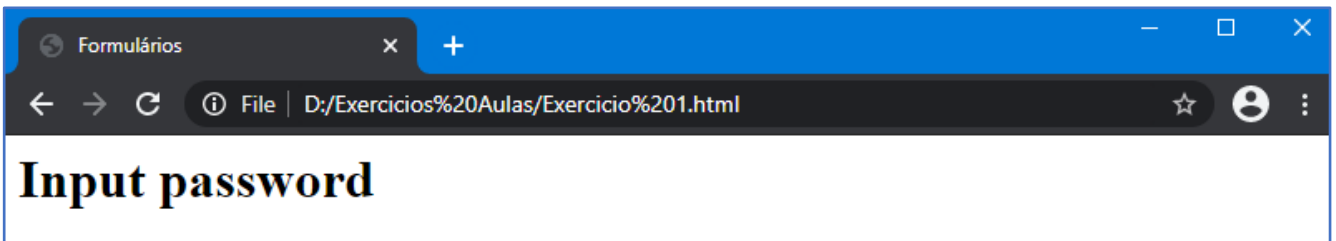


PASSWORD – Esse atributo é muito utilizado para criar campos de entrada de senhas, protegendo os elementos visualmente, substituindo por um carácter padrão o “ * ”. Dificultando os curiosos.

Exemplo de código:

```
<html>
    <head>
        <title>Formulários  </title>
    </head>
            <body>
                <h1> Input password </h1>
                <form>
                    Senha : <input type="password" name="senha"> <br/> <br/>

                    Confirmação : <input type="password" name="confirmação">
                </form>

            </body>
</html>
```

No navegador:

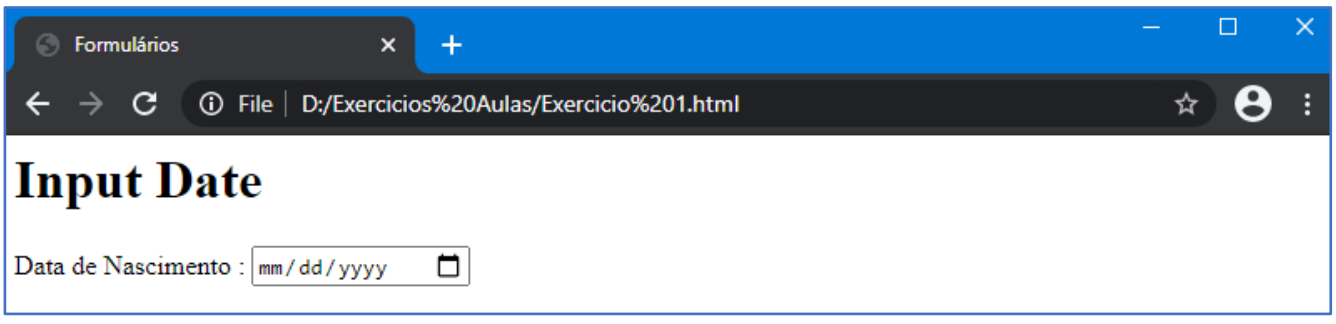


DATE – Esse de tipo de atributo aceita apenas datas. Em alguns navegadores quando este campo é utilizado ele apresenta um calendário no qual podemos escolher uma data especifica, em outros casos ele apresenta apenas uma mascara mostrando um formato que a informação será mostrada.

Exemplo do código:

```
<html>
    <head>
        <title>Formulários  </title>
    </head>
            <body>
                <h1> Input Date </h1>
                <form>
                    Data de Nascimento :
                    <input  type="date"          name="nascimento"
                            min="1950-01-01"     max="2020-12-31">
                </form>

            </body>
</html>
```

Podemos determinar um intervalo de datas válidas, com os atributos “*min*” e “*max*”, sempre usadas no formato de datas norte americano **ano-mês-dia**, onde o ano deve conter 4 dígitos. Veja o resultado no navegador.

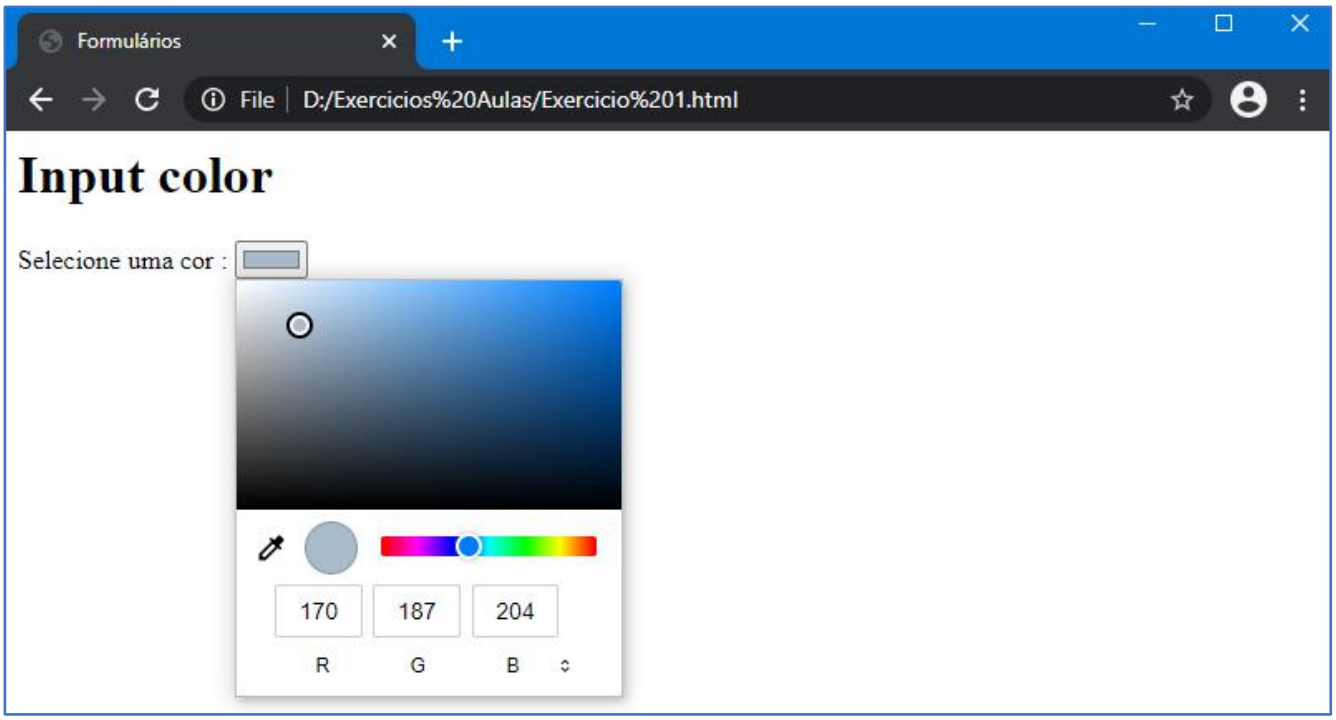


COLOR – Este atributo fornece ao usuário um elemento que permite selecionar uma cor, dependendo do navegador este elemento é apresentado de formas diferentes. Em geral o valor retornado está no formato hexadecimal **(#AABBCC**).

Exemplo do código:

```
<html>
    <head>
        <title>Formulários  </title>
    </head>
            <body>
                <h1> Input color </h1>
                <form>
                    Selecione uma cor :
                <input  type="color" name="cor" value="#AABBCC">
                </form>

            </body>
</html>
```

No navegador:

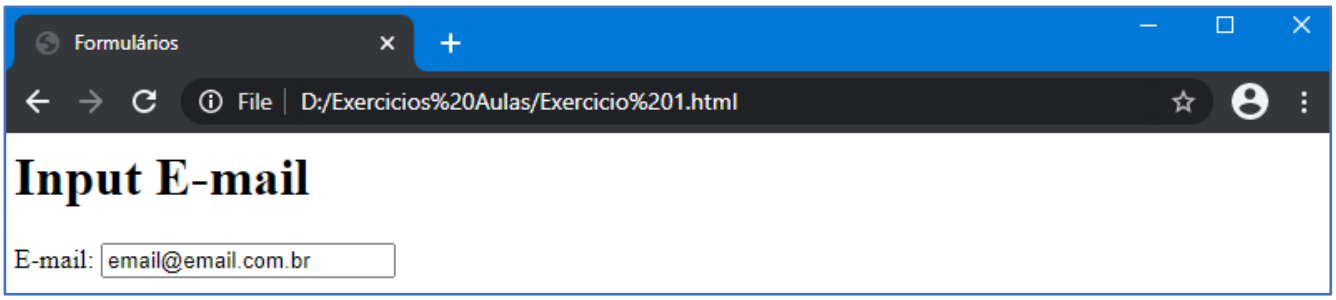


EMAIL – Este atributo permite uma verificação do conteúdo do campo para garantir que o formato digitado seja um email válido, pode ser enviado em branco, salvo com o uso do atributo *required*. Em casos que o atributo *multiple* esteja presente, é possível submeter mais de um endereço de email válido, para isso devemos separar cada email com uma virgula (**,**).

Exemplo do código:

```
<html>
    <head>
        <title>Formulários  </title>
    </head>
            <body>
                <h1> Input E-mail </h1>
                <form>
                    E-mail:
                    <input type="email" required multiple>
                </form>

            </body>
</html>
```

No navegador:

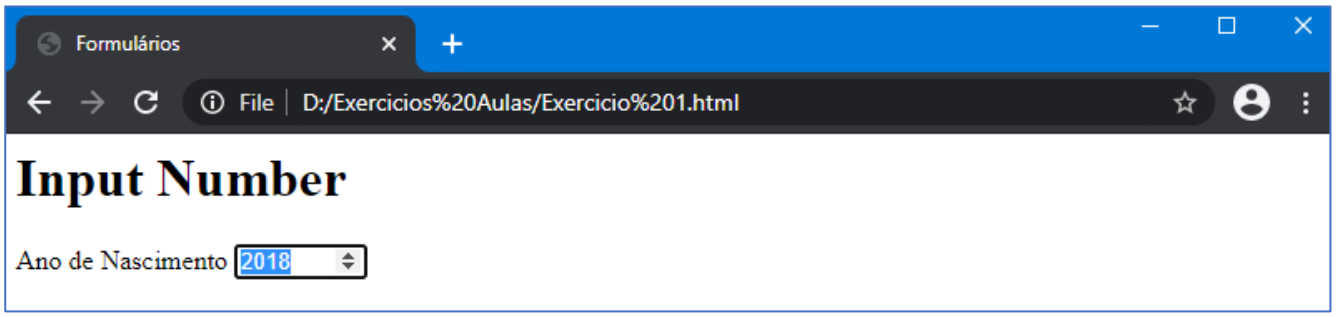


NUMBER – Permite a entrada de dados numéricos permitindo definir através dos atributos "*min*" para o valor mínimo e "*max*" para o valor máximo. Outro atributo que é usado é o "*step*" que define o valor do incremento e decremento do campo.

Exemplo de código:

```
<html>
    <head>
        <title>Formulários  </title>
    </head>
            <body>
                <h1> Input Number </h1>
                <form>

                    Ano de Nascimento
                <input type="number" name="anoNascimento" required
                 min="1910"    max="2018"   step="1">

                </form>
            </body>
</html>
```

No navegador:

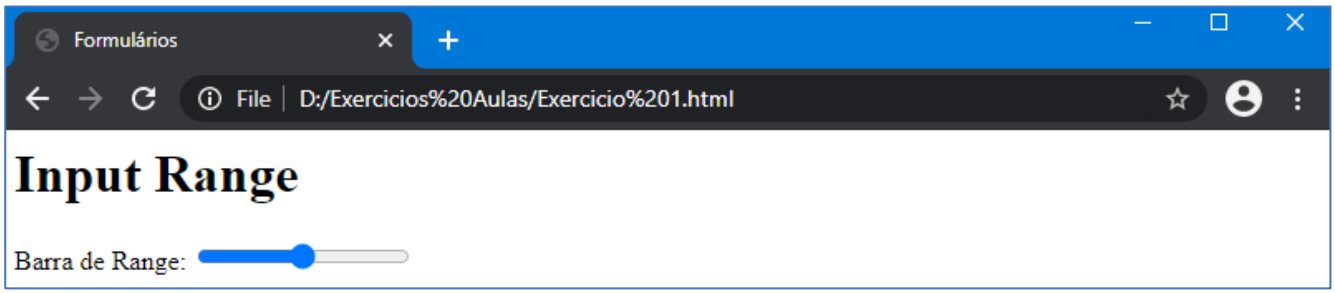


RANGE – Este atributo em geral apresenta uma barra horizontal, cujo os valores são limitados pelos atributos "*min*" e "*max*", outro atributo usado é o "*step*".

Exemplo do código:

```
<html>
    <head>
        <title>Formulários  </title>
    </head>
            <body>
                <h1> Input Range </h1>
                <form>
                    Barra de Range:
                        <input type="range"    name="barra" required
                                min="0"      max="5000"  step="100">
                    </form>
            </body>
</html>
```

No navegador:

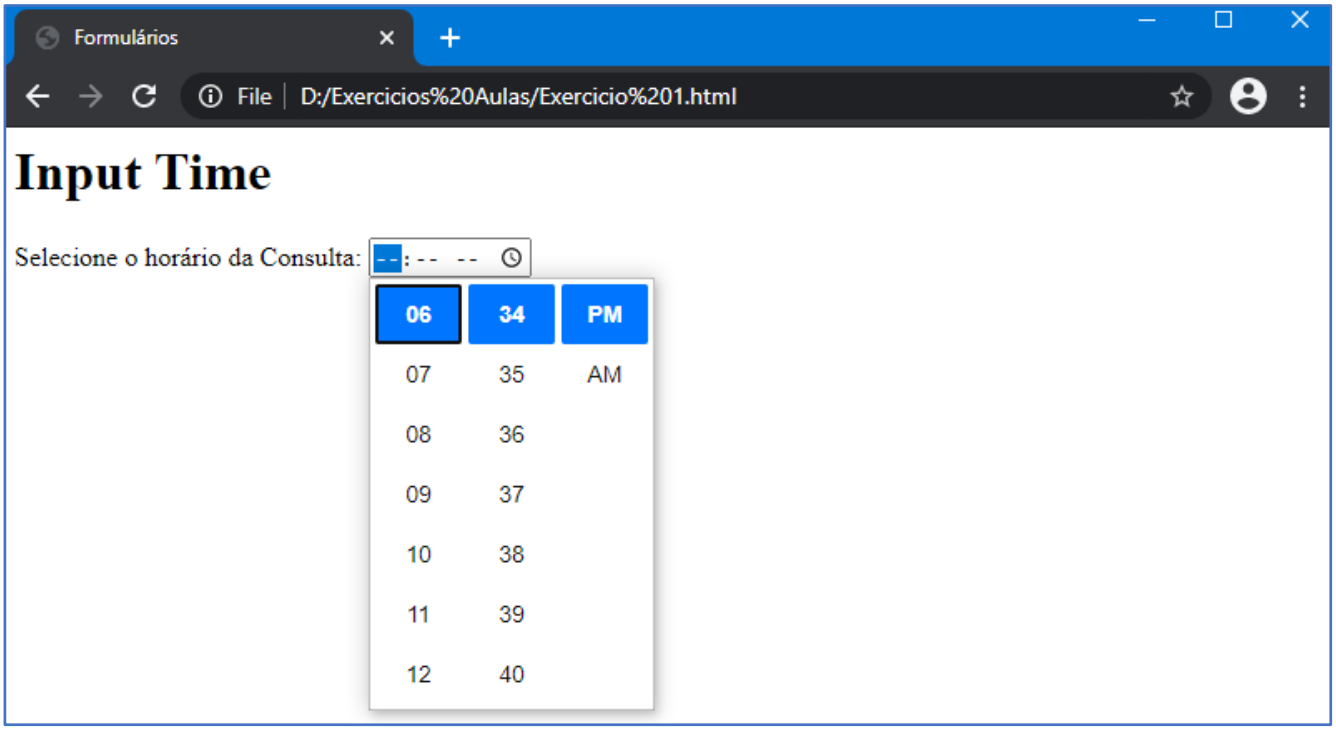


TIME – Este atributo permite ao usuário colocar um horário no formato de 12 ou 24 horas, com o formato hh:mm, em casos de 12 horas com a sigla *AM/PM*, e em casos 24 horas hh:mm:ss, isso dependendo do navegador.

Exemplo do código:

```html
<html>
    <head>
        <title>Formulários  </title>
    </head>
            <body>
                <h1> Input Time </h1>
                <form>
                    Selecione o horário da Consulta:
                    <input type="time"    name="horaConsulta" required>
                </form>
            </body>
</html>
```

No navegador:

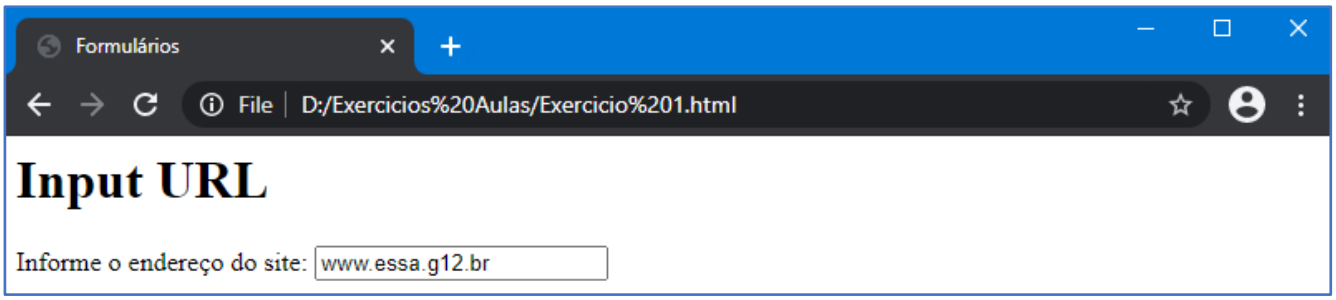


URL – Este atributo é usado quando precisamos de uma validação do campo para verificar uma *url* válida digitada pelo usuário.

Exemplo do código:

```html
<html>
    <head>
        <title>Formulários  </title>
    </head>
            <body>
                <h1> Input URL </h1>
                <form>
                    Informe o endereço do site:
                    <input type="url" required>
                </form>
            </body>
</html>
```

No navegador:



HTML 5 NOVAS TAGS

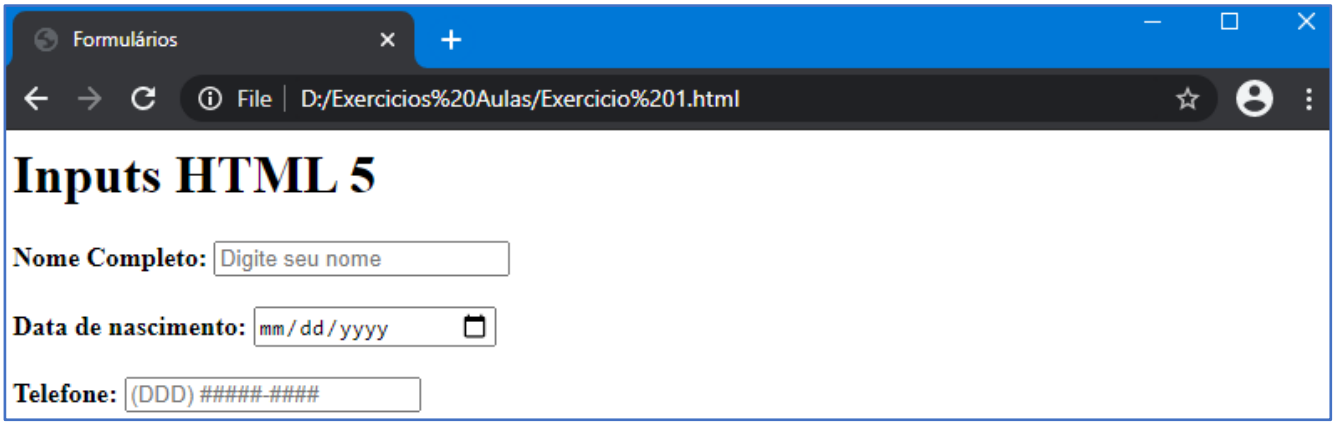
Esses atributos do **<input>** apresentados até aqui são atributos que fazem parte do html já algum tempo, com a chegada da versão 5.0 do html, veio também alguns novos atributos que ajudam a interatividade com o usuário e a praticidade na implementação, vamos conhecer os principais.

PLACEHOLDER – Este atributo mostra uma “dica” de como o campo dever ser preenchido, na versão anterior só era possível com auxílio de scripts (JavaScript ou jQuery). Essa “dica” aparece no campo até que o usuário começa a digitar algo. As informações do *placeholder* ficam visíveis somente enquanto o campo está vazio, e não influencia no valor do campo em si.

Exemplo de código:

```html
<html>
    <head>
        <title>Formulários  </title>
    </head>
            <body>
                <h1> Inputs HTML 5 </h1>
                <form>
                    <b>Nome Completo:</b>
                    <input type="text" name="nome" placeholder="Digite seu nome"/> <br/><br/>
                    <b>Data de nascimento:</b>
                    <input type="date" name="nasc" placeholder="dd/mm/aaaa"/><br/><br/>
                    <b>Telefone:</b>
                    <input type="tel" name="tel" placeholder="(DDD) #####-####"/>
                </form>
            </body>
</html>
```

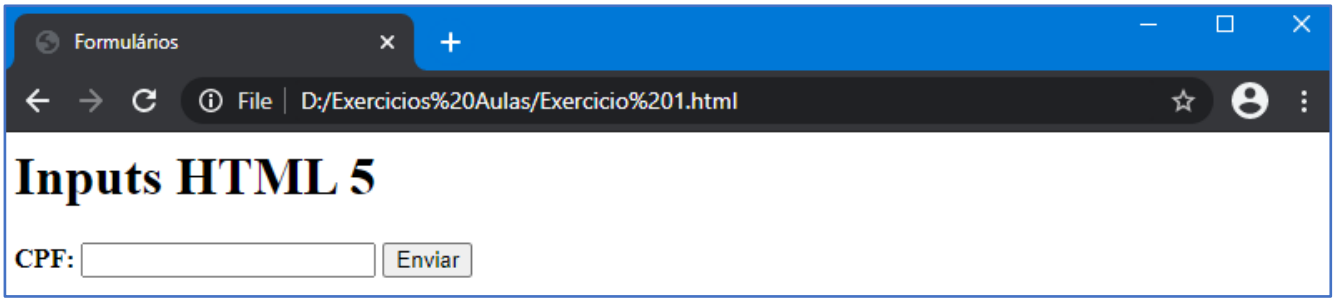
No navegador:



REQUIRED – Este atributo quando presente auxilia na validação do campo, o formulário não será "submetido/enviado" caso o campo esteja vazio. Veja o exemplo do código:

```html
<html>
    <head>
        <title>Formulários  </title>
    </head>
            <body>
                <h1> Inputs HTML 5 </h1>
                <form action="http://www.google.com">

                    <b>CPF:</b>
                    <input type="number" required/>
                    <input type="submit" value="Enviar"/>

            </body>
</html>
```

No navegador:



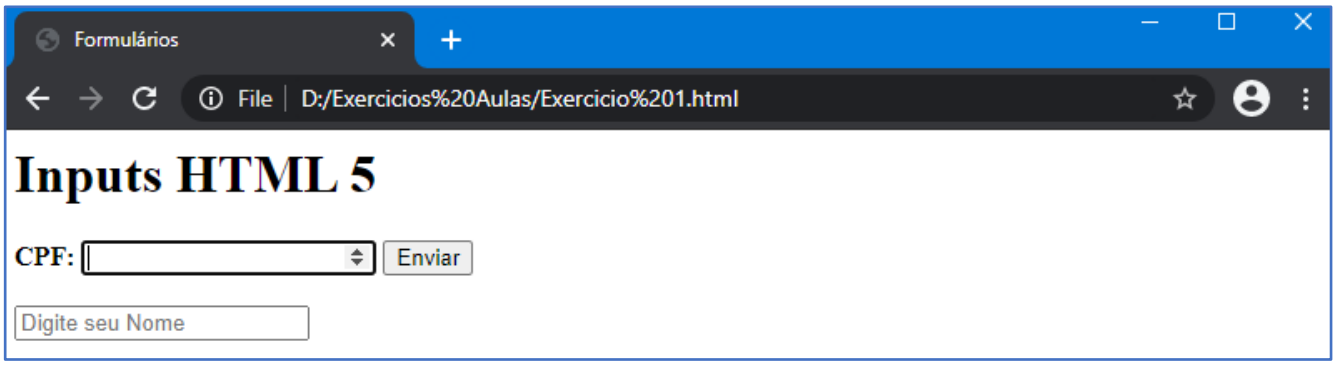
AUTOFOCUS – Esse atributo é possível implementa-lo somente em um campo do formulário, pois sempre que o navegador carregar a página onde está o formulário o campo terá o foco, sendo assim o cursor aparecerá nesse campo predefinidamente. Este atributo garante que ao carregar a página do formulário, sempre seremos direcionados para o mesmo campo.

Exemplo do código:

```
<html>
    <head>
        <title>Formulários  </title>
    </head>
            <body>
                <h1> Inputs HTML 5 </h1>
                <form action="http://www.google.com">

                        <b>CPF:</b>
                        <input type="number" required autofocus/>
                        <input type="submit" value="Enviar"/> <br/> <br/>
                        <input type="text" placeholder="Digite seu Nome" name="nome">


            </body>
</html>
```

Devemos colocar o atributo no campo que terá o foco toda vez que a página será carregada, pode – se colocar em qualquer campo em específico, só devemos lembrar que esse atributo pode ser usado somente em um campo do formulário.

Veja no navegador o resultado do código:



MAXLENGTH – Esse atributo permite a limitação da quantidade máxima de caracteres que podem ser inseridos na tag **<input>,** ideal para limitar campos com o *type="text"*. Vamos ver o código como exemplo.

```
<html>
    <head>
        <title>Formulários  </title>
    </head>
            <body>
                <h1> Inputs HTML 5 </h1>
                <form action="">
                    <h2>Digite seu Primeiro Nome</h2>
                    <input type="text" maxlength="20" required/>
                    <input type="submit" value="Enviar"/>
                </form>


            </body>
</html>
```

Esse atributo, em particular, não é possível visualizar em imagem, pois não mostra nada referente ao limite determinado no campo, é possível a visualização comente implementando na página e visualizando que o campo não permite mais caracteres que o estipulado no código.

AUTOCOMPLETE - Em geral por padrão, os navegadores oferecem a função de autocomplete em formulários, porém em alguns momentos seja necessário desabilitar esta função, por exemplo, quando precisamos confirmar o endereço de um e-mail, login ou senha.

Exemplo do código:

```html
<html>
    <head>
        <title>Formulários  </title>
    </head>
            <body>
                <h1> Inputs HTML 5 </h1>
                <form action="">
                    <h3>Digite seu email:</h3>
                    <input type="email" required />
                    <h3>Confirme seu email:</h3>
                    <input type="email" required autocomplete="off"/>
                    <input type="submit" value="Enviar"/>
                </form>


            </body>
</html>
```

Esses são os principais atributos utilizados na tag **<input>**, claro que temos muitos mais, recomendo que baixe da internet a documentação do HTML 5.0 para visualizar todas as tags na nova versão, baixe do site:

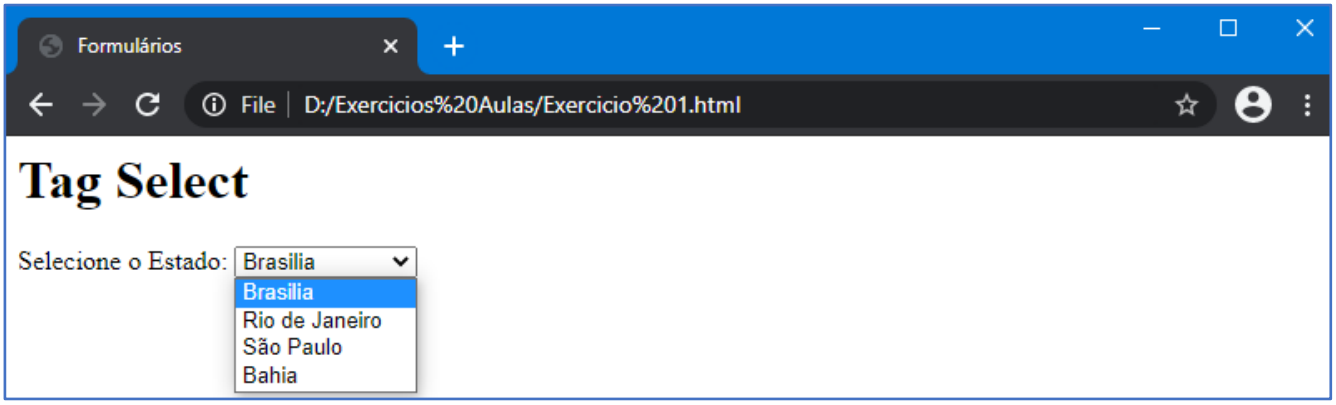
Vamos aprender sobre outras tags também bastante utilizadas nos formulários:

SELECT – Essa tag utilizamos quando desejamos criar uma estrutura no modelo "*drop-down*", que mostrar pro usuário uma série de opções selecionáveis. Dependendo dos parâmetros implantados no código, os usuários poderão selecionar uma ou mais opções.

Exemplo do código:

```html
<html>
    <head>
        <title>Formulários  </title>
    </head>
            <body>
                <h1> Tag Select </h1>
                <form>
                    Selecione o Estado:
                    <select name="estado">
                        <option value="DF">Brasilia</option>
                        <option value="RJ">Rio de Janeiro</option>
                        <option value="SP">São Paulo</option>
                        <option value="BH">Bahia</option>
                    </select>
                </form>

            </body>
</html>
```

Podemos notar que o modo no qual escrevemos essa tag é diferente, sendo aberto a tag com **<select>** e após seu término fechando – a com **</select>**, onde todas as opções são usadas com ao atributo **<option>** ... **</option>** Vamos ver no navegador como será o resultado:

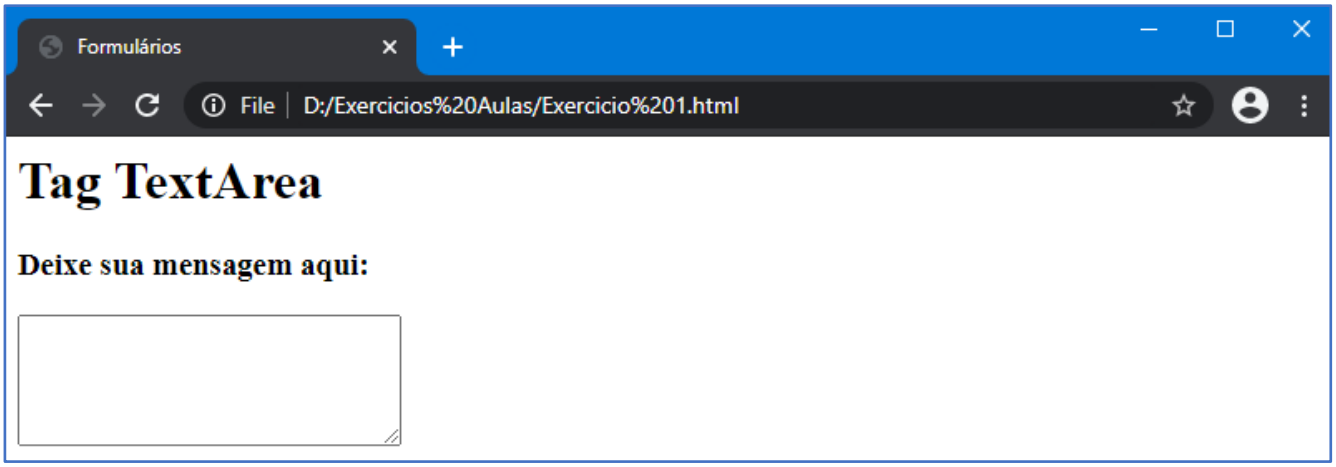


TEXTAREA – Diferente da tag **<input>**, que permite o usuário fornecer apenas uma linha de dados, esta tag permite que seja fornecido uma quantidade de informações muito superior. Permitindo a inserção de texto de múltiplas linhas, sendo muito útil para armazenar mensagens mais longas ou trechos de textos. Pode ser inserido 2 atributos para determinar o número máximo de linhas e colunas dentro da área de texto, o atributo **rows** é usado para limitar o número de linhas, e o atributo **cols** a quantidade de colunas.

Exemplo do código:

```
1  <html>
2      <head>
3          <title>Formulários  </title>
4      </head>
5              <body>
6                  <h1> Tag TextArea  </h1>
7                  <form>
8                      <h3>Deixe sua mensagem aqui:</h3>
9                      <textarea rows="5" cols="30"></textarea>
10                 </form>
11
12             </body>
13 </html>
```

No navegador podemos visualizar da seguinte forma:



BUTTON – É possível adicionar botões aos formulários de duas maneiras, primeiro é usando a tag **<input>** e segundo é usando a tag **<button>**. Os botões no formulário são de extrema importância, pois são através deles que a maioria das interações são acionadas. Uma das principais funções é invocar o método que realiza o envio/submissão dos dados do formulário para um servidor ou página em *PHP*. Essa tag tem o modo de utilização sendo aberta com **<button>** e fechada com **</button>**. Vamos ver um exemplo do código:

```
1   <html>
2       <head>
3           <title>Formulários  </title>
4       </head>
5               <body>
6                   <h1> Tag Button  </h1>
7                   <form action="form.php" method="POST">
8                   <button type="submit">Enviar</button>
9                   </form>
10
11              </body>
12  </html>
```

Assim como a tag **<input>**, a tag **<button>** possui uma série de atributos importantes para diversas utilizações. É importante conhecermos os mais usados para deixar nossos formulários mais funcionais e práticos. Vamos conhecer os principais atributos da tag **<button>**:

TYPE – Determina o tipo do botão, ele aceita alguns valores como; (*button*), para botão no formato padrão, (*reset*), para um botão com a finalidade de resetar os campos preenchidos para um valor padrão, e (*submit*), para criar um botão com a função de enviar os dados do formulário.

NAME – É usado para nomear o botão.

VALUE – Usamos o value para definir o nome/valor que aparece no botão.

FORMACTION - Define a URL onde os dados submetidos serão processados.

AUTOFOCUS – Determina que o botão deve receber o foco sempre quando a página for carregada.

DISABLED – Faz com que o botão seja desabilitado e não responda aos eventos, como por exemplo ao click ou passar do mouse.

FORMMETHOD – Especifica como será realizada a submissão do formulário, seja no método *GET* ou *POST*.

FORMNOVALIDATE – Determina que o formulário seja submetido/enviado sem ser validado, esse atributo só deve ser usado nos botões tipo *submit*.

OBJETOS E IFRAMES

Novas tags adicionadas como a **<object>** e a **<iframe>** vieram para ajudar na implementação e uso do html, e deixar mais leves a prática de adicionar objetos multimidia na página, vamos conhecer essas tags com mais detalhes:

<object> ... </object> - Esta tag é utilizada para incorporar tipos de mídias adicionais à página. Podendo ser áudio, video, documento em pdf, entre outros. Com alguns atributos importantes para a utilização como *width* (largura) e *height* (altura) para ajustar as dimensões onde o objeto será adicionado e o *type* que determina o tipo do conteúdo exibido.

<iframe> ... </iframe> - Essa tag é uma espécie de bloco de conteúdo em linha, que é utilizado como um contêiner flexível para arquivos de mídia. Ele tem a posição flutuante dentro da página, significando que é colocado relativo à outros itens da página. Conheça alguns atributos da tag *iframe.*

NAME – Determina o nome do frame.

SRC – A fonte URL/caminho do objeto multimidia para ser inserido dento do iframe.

SRCDOC – Insere um conteúdo HTML para ser mostrado dentro do iframe.

WIDTH – Determina a largura do iframe.

HEIGHT – Determina a altura do iframe.

<PARAM/> - Serve para customizar o iframe. Incluindo parâmetros adicionais para ir junto com o conteúdo.

<embed> ... </embed> - É usado para incorporar na página objetos externos, como plugins (flash player entre outros).

RESPONDA:

1) O que é um formulário HTML?

2) Qual a principal diferença entre um formulário em HTML e um documento simples HTML?

3) Para a tag <form> podem ser atribuídos o atributo _____ (atributo onde define o local através da url) e o _____ (atributo que define o método HTTP).

4) Qual a diferença entre os métodos de um formulário HTML?

5) Qual a tag usada para usada para adicionar uma área de texto no formulário?

6) Para inserir um botão no formulário eu uso a tag _____ ou a tag _____.

7) A tag <input type='date'> me retorna qual valor?

8) Qual a função da tag <map> no código em HTML.

9) Qual tag utilizo para inserir um campo de texto?

10) Qual o uso da tag <label>?

DESAFIO

Faça um formulário de contato, com os campos nome, e-mail comentários e botão enviar.

TEMA 06 – CSS (Cascating Styles Sheets – Folhas de Estilo em Cascata)

Figura: 11 - CSS – folhas de estilo em cascatas – **Fonte:** creativebloq

O QUE SÃO CSS

A CSS (Folhas de Estilo em Cascata) é uma linguagem usada para estilizar documentos de forma leve e elegante, ela possibilita determinar o comportamento de um determinado documento em telas de diferentes dispositivos. Ela é altamente compatível com o HTML, fazendo parte de praticamente toas as páginas da web, fazemos o uso das css para estilizar os documentos HTML, visando sempre a rapidez e eficácia na visualização dos elementos. As CSS permitem aplicar estilos de modo seletivo em elementos de documentos HTML.

Por exemplo, para selecionar todas as tags (elementos) título de uma página e fazer qualquer tipo de alteração e estilização, com as CSS não há necessidade de modificar elemento individualmente (como era feito antigamente), posso simplesmente determinar no código CSS que os títulos terão um determinado comportamento, e a alteração é feita em todos. Para que a página em HTML possa fazer uso dos estilos em CSS, é necessário inserir na tag **<head>** um link para a página em CSS onde estão configurados os estilos. Veja no exemplo:

Podemos notar na linha 3 da imagem acima, indicando a inserção da página **estilo.css** no atributo *href*, fazendo assim o link para que essa página seja usada esses estilos. O link é uma tag utilizada para indicar ao html que uma página externa será adicionada, nesse caso usamos o essa tag **<link rel=" stylesheet">** indicando que esse link é relativo às folhas de estilos do tipo (type) *text/css*. E por último fazemos o uso do *href=" nome_da_pagina.css",* para indicar o local da página do estilo CSS.

Vamos ver um exemplo simples do uso do html na tag <p> do html:

Podemos notar nesse exemplo onde o elemento <p> foi configurado para que tenha a cor blue(azul) em todos os elementos que a página está linkada. Toda estrutura é chamada de conjunto de regras (em geral usamos o termo “regra”, por ser mais curto). Vamos conhecer a anatomia do conjunto de regras do CSS.

SELETOR (SELECTOR) – O nome do elemento (tag) em HTML no começo do conjunto de regras. Ele é responsável por selecionar o(s) elemento(s) a serem aplicados os estilos (nesse caso, os elementos **<p>**). Para dar estilo a um outro elemento, é só mudar o seletor.

DECLARAÇÃO (DECLARATION) – Uma regra simples como **color: red**; especifica quais das **propriedades** do elemento deseja estilizar. Nesse caso irá mudar a cor do elemento para vermelho.

PROPRIEDADES (PROPERTY) – Forma pela qual usamos para estilizar um elemento em HTML. Nesse caso, **color** é uma propriedade dos elementos **<p>**. Em CSS, podemos escolher quais propriedades queremos afetar com sua regra.

VALOR DA PROPRIEDADE (PROPERTY VALUE) – Com o valor da propriedade podemos escolher uma, dentre muitas aparências possíveis para uma determinada propriedade (cada propriedade em particular, possui inúmeros valores de distintos).

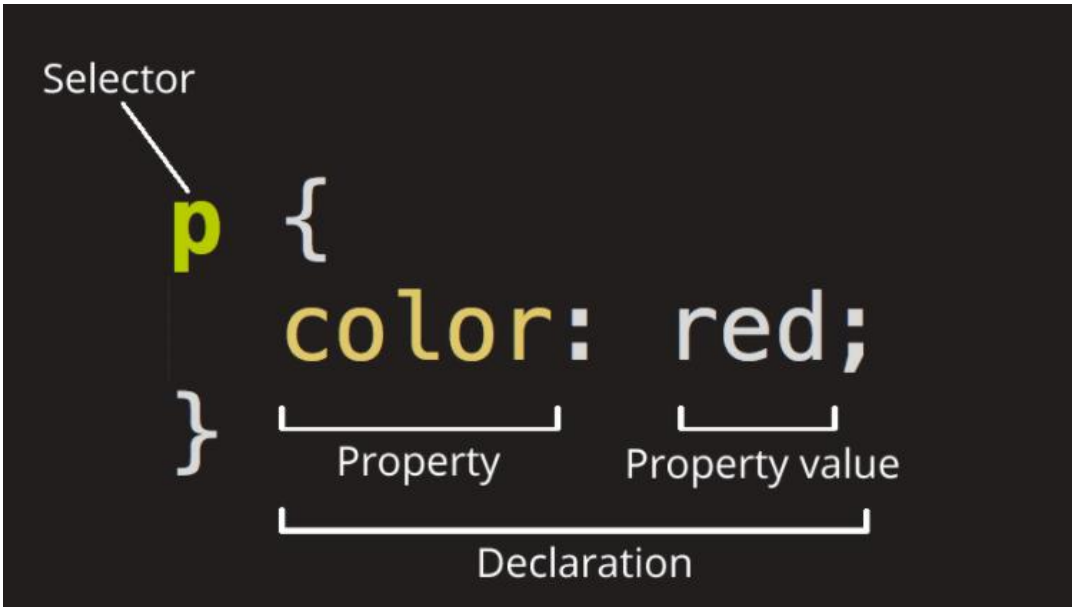


Figura 12 – Sintaxe do CSS – Fonte: guilhermemuller

Na imagem acima nota-se a estrutura do conjunto de regras que se faz necessário para a aplicação do CSS, temos primeiro que indicar o *seletor* (nesse caso o elemento **<p>**), para definir as propriedades desse seletor sempre devemos definir a propriedade entre **chaves ({ })** , depois a declaração da propriedade seguida de **dois pontos (:)**, e atribuindo o valor da propriedade, finalizando a linha de cada declaração com o **ponto e vírgula (;)**. Devemos lembrar que essa regra deve ser usada em todos os elementos que serão estilizados com o CSS, mudando algumas definições pontualmente. Em casos que é necessário a estilização de mais de uma propriedade, é só definir na linha de baixo uma nova propriedade, declarando o valor da mesma. Veja o exemplo abaixo:

```
p {
    color: red;
    width: 500px;
    border: 1px solid black;
    }
```

Podemos perceber que definimos o estilo do elemento **<p>** com o uso de três propriedades diferentes.

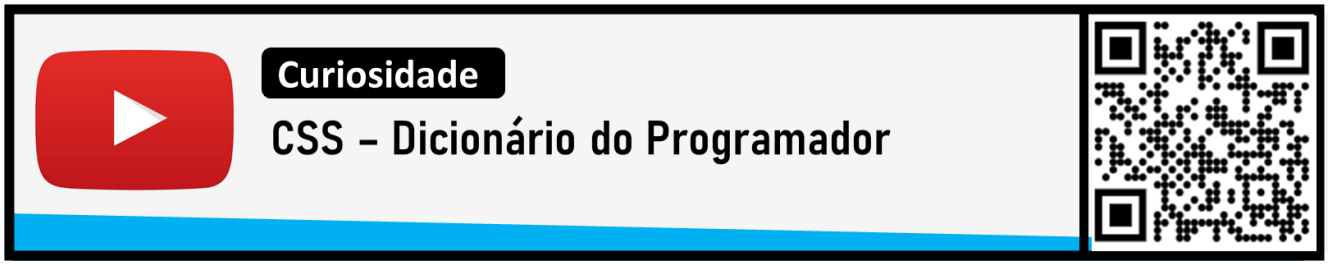
```
html{
    background-color: skyblue;

    }
p {
    color:black;
    font-family: 'Trebuchet MS', 'Lucida Sans Unicode', 'Lucida Grande', 'Lucida Sans', Arial, sans-serif;

    }

h1{
    color: teal;
    text-shadow: slategrey;
    }
```

Nesse exemplo acima, podemos notar que foram inseridos no CSS três elementos, o elemento HTML, onde é aplicado em todo o documento ***HTML***, o elemento **p** e o elemento **h1**. Podemos fazer diversas alterações deixando nossas páginas muito mais atrativas visualmente com o CSS.É possível também definir um mesmo conjunto de regras para múltiplos seletores, vamos ver um exemplo:

```
html{

    background-color: skyblue;
    }

p, li, h1 {

    color: red;

    }
```

Na imagem acima definimos três seletores com o mesmo conjunto de regras, ou seja, mesma propriedade e valor para os três, apenas separando-os com uma vírgula **(,)**. Nesse exemplo determinamos que os elementos <p>, <li> e <h1>, serão apresentados na página em vermelho.

O CSS vem evoluindo em paralelo com o HTML, por isso é importante saber suas versões e como contribuem para nosso uso hoje em dia, o canal código fonte explica essa história com detalhes, veja esse video e entenda um pouco mais dessa linguagem "mágica" de estilos.



DIFERENTES TIPOS DE SELETORES

Existem diversos tipos de *seletores* diferentes, vamos conhecer os *seletores* de elementos. Que selecionam todos os elementos de um determinado tipo nos documentos HTML. Porém é preciso fazer seleções mais especificas, veja alguns dos tipos mais comuns de seletores.

Nome do seletor	O que seleciona	Exemplo
Seletor de Elemento (pode ser chamado tag ou seletor de tipo)	Todos os elementos HTML de determinado tipo.	**P** Seleciona <p>
Seletor de ID	O elemento na página com o ID especificado. Em uma determinada página HTML, é uma boa prática usar um elemento por ID (e claro, um ID por elemento) mesmo que seja permitido usar o mesmo ID para vários elementos.	**#id** Seleciona <p id="meu id"> ou <a id="meu-id">.
Seletor de Classe	O(s) elemento(s) na página com a classe especificada (varias instancias de classe podem aparecer em uma página)	**.class** Seleciona <p class="minha-classe"> e <a class="minha-classe">
Seletor de Atributo	O(s) elemento(s) na página com o atributo especificado.	**Img[src]** Seleciona <img src="minha_imagem.png">. Lembrando seleciona o atributo *src* e não o img

Seletor de Pseudo-classe	O(s) elemento(s) especificado(s), mas somente quando estiver no estado especificado, como por exemplo o mouse sobre ele.	**a:hover** Seleciona <a>, mas somente quando o mouse está em cima do link.

FONTES E TEXTOS

No CSS podemos vincular fontes em diferentes formas, uma delas é um vínculo online de fonte, nesse caso usando o *google fonts* podemos inserir uma ou mais famílias de fontes com esse método, vamos ver no código:

```
<link href="http://fonts.googleapis.com/css?family=Open+Sans" rel="stylesheet">
```

Nota – se no código acima que vinculamos uma folha de estilo que baixa a família de fontes *Open Sans* junto com a página web, permitindo que seja definido seus elementos HTML usando a própria folha de estilos. No CSS é possível estilizar várias propriedades de um mesmo elemento de texto, vamos conhecer agora esses elementos:

PROPRIEDADE FONTE	VALOR DA PROPRIEDADE
Font	font-style | font-variant | font-weight | font-size | font-family | caption icon | menu | message-box | smallcaption | status-bar
Font-size	xx-small | x-small | small | medium | large | x-large | xx-large | smaller larger | inherit | length (%)
Font-size-adjust	None | inherit | number (%)
Font-stretch	normal | wider | narrower | ultra-condensed | extracondensed | condensed | semi-condensed | semiexpanded | expanded | extra-expanded | ultraexpanded | inherit
Font-style	Normal | italic | oblique | inherit
Font-variant	Normal | small-caps | inherit

Font-weight	Normal | bold | bolder | lighter | 100 | 200 | 300 | 400 | 500 | 600 | 700 | 800 | 900 | inherit
Font-family	Family-name | generic-family | inherit

Na tabela acima estão as possibilidades que podemos definir os estilos das fontes, importante que na hora da prática você faça testes para descobri qual melhor estilo para seu documento. A seguir vamos ver a tabela de propriedades de texto:

PROPRIEDADE DE TEXTO	VALOR DA PROPRIEDADE
direction	ltr | rtl | inherit
hanging-punctuaction	None | [start | end | adjancent]
letter-spacing	normal | length (%)
punctuation-trim	None | [start | end | adjancent]
text-align	Start | end | left | right | center | justify
text-align-last	Start | end | left | right | center | justify
text-decoration	none | underline | overline | line-through | blink
text-emphasis	none | [accent | dot | circle | disc] [before | after]
text-indent	Length (%)
text-justify	Auto | inter-word | interideograph | inter-cluster | distribute | kashida | Tibetan
text-outline	None | color | length
text-shadow	None | color | length
text-transform	None | capitalize | uppercase | lowercase
text-wrap	Normal | unrestricted | none | suppres
unicode-bidi	Normal | embed | bidioverride
white-space	Normal | pre | nowrap | prewap | pre-line
white-space-collapse	Preserve | collapse | preserve-breaks | discard
word-break	Normal | keep-all | loose | break-strict | breal-all
word-spacing	Normal | length (%)
word-wrap	Normal | break-word

Na tabela acima é mostrado a propriedade de texto com seus respectivos valores, dessa forma é possível entender que cada propriedade tem um valor especifico com uma ação especifica. Não é necessário que saiba todas, o importante é usa-las de acordo achar melhor em sua folha de estilo, por isso é recomendado que faça testes com cada um desses valores de cada propriedade, assim é possível visualizar o efeito de cada uma delas.

DESENVOLVIMENTO E APLICAÇÃO

Vamos começar a utilizar o CSS na prática para fixar melhor esse conjunto de regras e seu modo de aplicação;

HTML:

```html
<html>
    <head>
        <link rel="stylesheet" href="D:\Exercicios Aulas\CSS\estilo.css">

        <title> - Estilos em CSS - </title>
    </head>
            <body>

                <h1>Aplicando Folhas de Estilo No HTML</h1>

                <img src="D:\Exercicios Aulas\Imagens\logo.png" alt="Imagem logo">

                <p>Assim como o HTML, o CSS não é realmente uma linguagem de programação.
                  Também não é uma linguagem de marcação  – é uma linguagem de folhas de estilos.
                  Isso significa que o CSS permite aplicar estilos seletivamente a elementos em documentos HTML.</p>

            </body>
</html>
```

CSS:

```css
/**ALTERANDO A COR DE FUNDO**/
html {
    background-color: #424142;
  }
```

Alterando o background da página;

```css
body {
  width: 600px;
  margin: 0 auto;
  background-color: #0e4d58;
  padding: 0 20px 20px 20px;
  border: 3px solid black;
```

Agora para o elemento **<body>**. Há algumas declarações aqui, então vamos passar por elas uma a uma:

width: 600px; — Isso força o corpo a ter 600 pixels de largura.

margin: 0 auto; — Quando você define dois valores em uma propriedade como margin ou padding, o primeiro valor afeta a parte superior do elemento e a parte inferior (tornando-os 0 nesse caso), e no segundo valor os lados esquerdo e direito (aqui, auto é um valor especial que divide o espaço horizontal uniformemente entre esquerda e direita). Você também pode utilizar um, três ou quatro valores, como documentado aqui.

background-color; — Como antes, isso define a cor de fundo do elemento.

padding: 0 20px 20px 20px; — Temos quatro valores definidos no padding, para criar um pouco de espaço em torno do nosso conteúdo. Dessa vez, estamos definindo sem padding na parte superior do corpo, e 20 pixels no lado esquerdo, na parte inferior e no lado direito. Os valores definem a parte superior, o lado direito, a parte inferior e o lado esquerdo, nessa ordem. Como com a margin, você também pode usar um, dois, ou três valores.

border: 3px solid black; — Isso simplesmente define uma borda preta sólida de 3 pixels de largura em todos os lados do corpo.

```css
h1 {
  margin: 0;
  padding: 20px 0;
  color: #f08902;
  text-shadow: 2px 2px 2px black;
  text-align: center;
  font-family: Cambria, Cochin, Georgia, Times, 'Times New Roman', serif;
}
```

Margin: – Definimos o margin do h1 em 0, para tirar os espaços da margem do texto.

Padding: – O padding no texto é importante para espaçar dos outros elementos da página.

Color: - Definimos a cor do texto.

Text-shadow – Uma propriedade bastante interessante que usaremos aqui é o text-shadow, que aplica uma sombra ao conteúdo de texto do elemento. Seus quatro valores são os seguintes:

- O primeiro valor em pixel define o deslocamento horizontal da sombra do texto — até onde ele se move: um valor negativo deve movê-la para a esquerda.
- O segundo valor em pixel define o deslocamento vertical da sombra do texto — o quanto ela se move para baixo, neste exemplo; um valor negativo deve movê-la para cima.
- O terceiro valor em pixel define o raio de desfoque da sombra — um valor maior significará uma sombra mais borrada.
- O quarto valor define a cor base da sombra.

Text-align: - Essa propriedade centraliza o texto.

Font-family: - Definimos uma família de fontes, ele funciona em ordem, nesse caso se a fonte *Cambria* não estiver instalada no computador ou no servidor (caso o site já esteja hospedado), ele define a fonte *Georgia* para uso, e assim sucessivamente.

```
img {
  display: block;
  margin: 0 auto;
}
```

Com essa propriedade centralizamos a imagem para melhorar a aparência. Nós poderíamos usar novamente o truque margin: 0 auto que aprendemos anteriormente para o corpo, mas também precisamos fazer outra coisa. O elemento **<body> é em nível de bloco**, o que significa que ocupa espaço na página e pode ter margens e outros valores de espaçamento aplicados a ele. Imagens, por outro lado, são elementos **em linha**, o que significa que não podem ter margens. Então, para aplicar margens a uma imagem, temos que dar o comportamento de nível de bloco a imagem usando display: block;

```
p{
  margin: 25px 25px;
  padding: 25px;
  text-align: justify;
  font-size: 25px;
  color: white;
  text-shadow: 2px 2px 1px black;
  font-family: Verdana, Geneva, Tahoma, sans-serif;
  }
```

Nos parágrafos aqui definimos propriedades para melhor alocar o texto na página, nota-se que as propriedades são bem parecidas com as do elemento **<h1>**;

Margin: - Definimos o margin com 25 pixels dos lados, para melhorar e espaçar o texto.

Padding: - Com o padding deixamos um espaço interno entre o bloco do texto.

Text-align – Aqui nessa propriedade definimos que o parágrafo seja justificado.

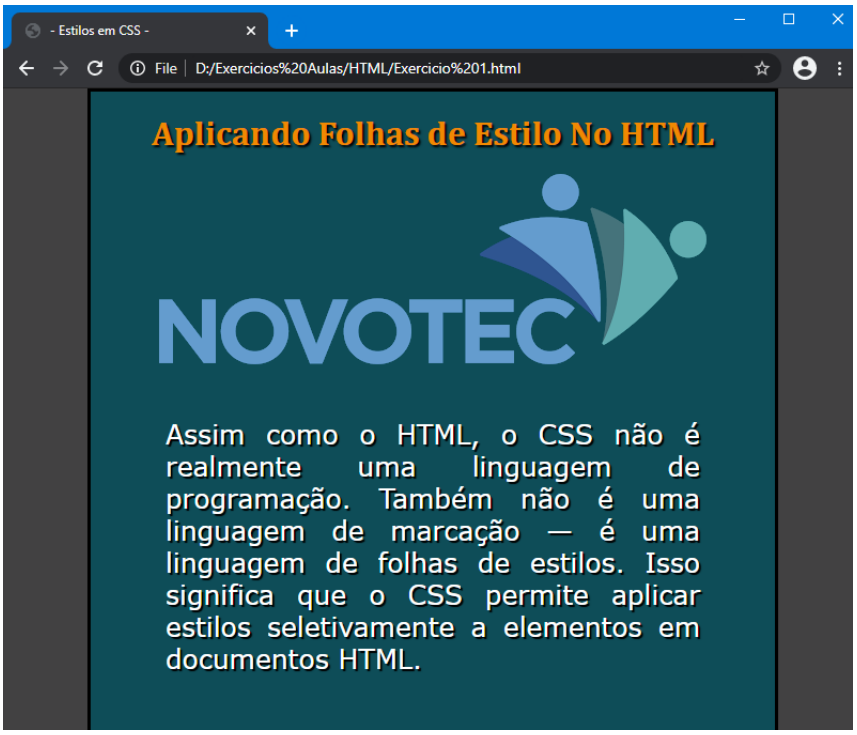
Font-size – Com a font-size deixamos a fonte do parágrafo com tamanho de 25 pixels.

Color: - Mudamos a cor do texto para branco, dando destaque no texto e contrastando com o fundo.

Text-shadow: - Colocamos uma sombra em preto no texto para dar um contraste com a cor branca.

Font-family – Definimos a família de fontes para nosso parágrafo, assumindo a ordem de importância em caso a fonte não esteja instalada, automaticamente assume a próxima fonte da sequência.

Resultado no navegador:



Podemos notar que as definições no código CSS são de muita importância para uma página HTML, sem o uso do CSS é praticamente inviável fazer o mesmo estilo em HTML puro, por isso o uso do CSS se fez muito importante desde a sua criação.

RESPONDA:

1) O que é CSS?

2) O que faz um CSS dentro de um documento em HTML?

3) Qual a sintaxe para incorporar uma folha de estilo dentro de uma página em HTML?

4) Quais os principais tipos de seletores em CSS?

5) Qual a função da propriedade *text-align*?

6) Qual propriedade CSS preciso usar para mudar a cor do texto de um elemento <p> do HTML?

7) Para mudar a cor do background do HTML, qual a sintaxe?

8) Como funciona a propriedade text-shadow?

9) As propriedades _____ e _____ são essenciais para definir as dimensões de um elemento, como largura e altura.

10) Qual a diferença entre *margin* e *padding*?

DESAFIO - Faça uma folha de estilo mudando o layout do desafio 3

TEMA 07 – BOX MODEL

BOX MODEL USADO PARA LAYOUTS

No CSS sempre quando estilizamos algum elemento, é comum, que alguma alteração que iremos fazer, possa impactar em outros elementos, e precisamos estar cientes disso. Deste modo é importante saber compreender o conceito de Box Model, que é utilizado em layout de páginas web.

Basicamente, a ideia do box model é composta por quatro partes:

- ✓ Conteúdo
- ✓ Espaçamento
- ✓ Bordas
- ✓ Margens

Em resumo, podemos dizer que o box model trata-se de como as 4 propriedades acima se relacionam para compor a dimensão do elemento.

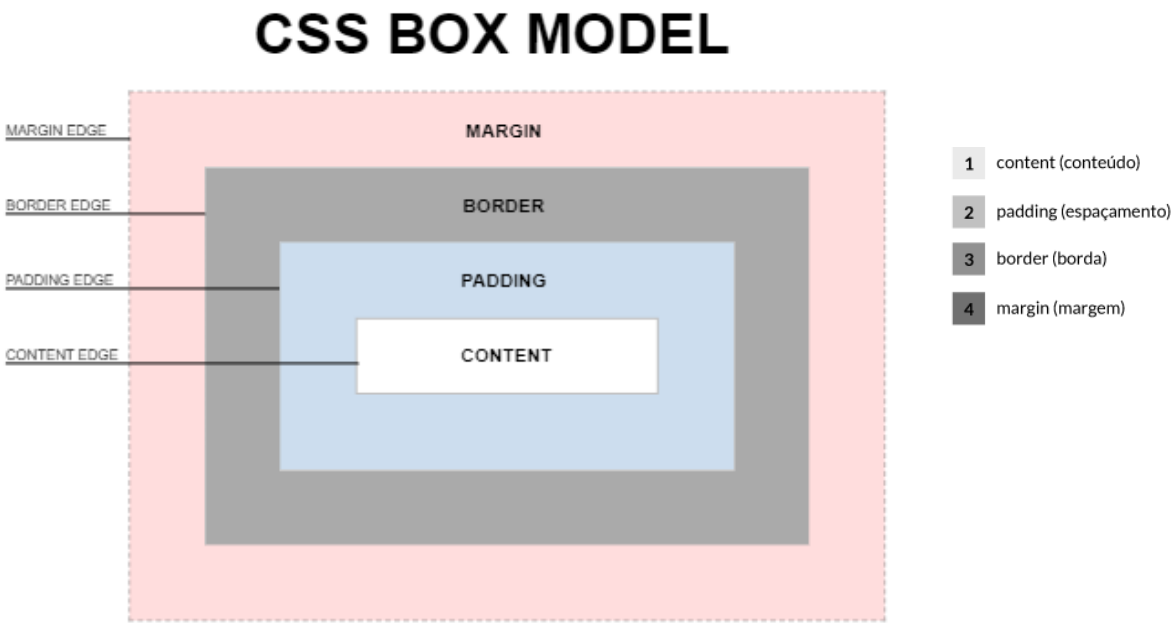


Figura 13: Estrutura do box model com seus atributos – Fonte: dev

Analisando a área de conteúdo (content) do box model é a área ocupada pelo conteúdo real do elemento. Muitas das vezes possuem um fundo, cor de fonte ou até mesmo uma imagem. É localizada dentro do conteúdo as dimensões, elas são; largura do conteúdo ou largura do box, e altura do conteúdo ou a altura do box conteúdo.

A área de preenchimento *(padding)* estende-se para a borda em torno do preenchimento. Em casos em que a área de conteúdo tem um fundo, imagem ou cor, será estendido para a área do preenchimento, desse modo, podemos entender que o preenchimento*(padding)* é a extensão do conteúdo*(content)*. O preenchimento se localiza dentro do padding edge, e suas dimensões são a largura do *padding-box* e a altura do *padding-box*. O espaço entre o padding e o content podem ser controlados pelas seguintes propriedades CSS;

- *Padding*
- *Padding-top*
- *Padding-right*
- *Padding-bottom*
- *Padding-left*

Temos também a área da borda (*border*) se estende a área de preenchimento para a área que contém as bordas. Esta é a área do *border,* e as dimensões dela são a largura e a altura do *border-box*. A área do border depende do tamanho da borda que definimos pela propriedade ***border*** ou pela ***border-width***.

E por fim a área de margem (*margin*) se estende a área da borda com um espaço vazio usado para fazer a separação dos elementos vizinhos. Esta área contém as dimensões de largura e altura do *margin-box.* O tamanho da sua área é definido usando as seguintes propriedades:

- *Margin*
- *Margin-top*
- *Margin-right*
- *Margin-bottom*
- *Margin-left*

Vamos aplicar uma classe(***.class***) em um elemento para entendermos melhor a forma que o box model funciona, definindo as propriedades de estilo. Assim podemos ver que o box model é de suma importância para a estilização das páginas, deixando os elementos melhor organizados e visualmente mais bonito e elegante os elementos da página em HTML.

```
1   .classe {
2     width: 50px;
3     height: 50px;
4     border: 1px solid gray;
5     padding: 10px 20px;
6   }
```

Nesse caso, quando aplicamos a classe acima em um elemento, ele vai ter as dimensões que definimos (50 pixels de altura e largura), correto? Não, errado. O elemento será mostrado com 72 pixels de altura e 92 pixels de largura. Isso acontece pelo fato das propriedades padding e border serem somadas à largura e alturas definidas anteriormente, consequentemente, aumentando as dimensões do elemento. Com esse acontecimento podemos entender que as propriedades ***width*** e ***height*** definem as dimensões do seu conteúdo e não do elemento em sua totalidade. Vamos ver como acontece esse tipo de “erro”.

Largura: 50 (largura definida) + 20(padding left) + 20(padding right) + 1(border left) + 1(border right) totalizando 92 pixels de largura.

Altura: 50(altura definida) + 10(padding top) + 10(padding bottom) + 1(border top) + 1(border top) totalizando 72 pixels de altura.

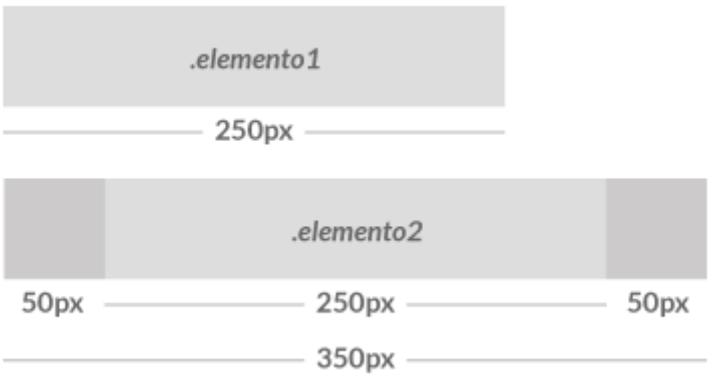
Para a resolução desse tipo de problema, como tantos outros que podem acontecer, a atualização do CSS 2 para o CSS 3, trouxe novas propriedades para auxiliar na correção desses problemas.

BOX-SIZING

Uma das propriedades que o CSS3 trouxe para facilitar a vida dos desenvolvedores e o ***box-sizing***. Essa propriedade nos permite alterar o comportamento do box model. Por padrão, os elementos possuem o valor *content-box* para essa propriedade.

Um exemplo para descrever melhor o box model com o uso do *content-box*. Considere os elementos a seguir:

```css
.elemento1 {
  background: #ddd;
  width: 250px;
  height: 50px;
}

.elemento2 {
  background: #ddd;
  width: 250px;
  height: 50px;
  padding: 0 50px;
}
```



Os elementos são estilizados com o mesmo valor na largura, porem como podemos notar na imagem acima, o *elemento2* é mostrado maior pelo fato do padding ser adicionado a ele.

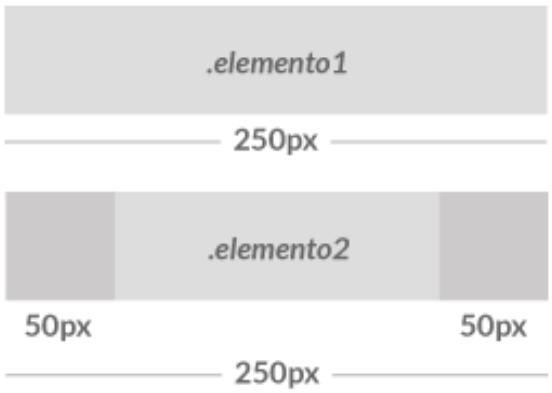
Uma propriedade que ajuda a resolver esse empasse é o box-sizing, permitindo um novo valor resolvendo os problemas dos exemplos anteriores:

Sintaxe do Código:	**box-sizing: border-box;**

Qual a função dela? Simples. Essa propriedade altera o comportamento do box model, fazendo que o browser (navegador) faça o cálculo a largura e altura do elemento, contando não somente o seu conteúdo (como no content-box), mas considerando também, o *padding* e o *border* do elemento. Vamos melhorar o exemplo anterior com a utilização do *border-box;*

```
.elemento1,
.elemento2 { box-sizing: border-box; }

.elemento1 {
    background: #ddd;
    width: 250px;
    height: 50px;
}

.elemento2 {
    background: #ddd;
    width: 250px;
    height: 50px;
    padding: 0 50px;
}
```

Duas linhas de código do CSS foram adicionadas, aplicando a propriedade *border-box*, deste modo melhorando a interação com o navegador e corrigindo o erro de espaçamento definido no CSS. Vamos ver o resultado:



Entretanto essa propriedade não é aceita total ou parcialmente em alguns navegadores (em média seria 90%, com 80% suporte parcial e com 18% suporte total), sendo assim se faz necessário utilizar

os prefixos para que essa propriedade aconteça da forma que for definida no código CSS. Segue a sintaxe dos prefixos para ser mostrada nos navegadores;

```
1  * {
2      -webkit-box-sizing: border-box;
3         -moz-box-sizing: border-box;
4          -ms-box-sizing: border-box;
5              box-sizing: border-box;
6  }
```

RESPONDA:

1) Qual a estrutura do box model?

2) Quais as propriedades usadas para posicionamento de elementos no box model?

3) A area do preenchimento estende-se para a _____ em torno do _____.

4) O espaço entre p padding e o content podem ser controlados por quais propriedades?

5) Qual a Sintaxe da propriedade “box-sizing”?

6) Sabemos que nem todos os navegadores entendem a box-sizing nativamente, qual a linha de comando devo utilizar para que não aconteça um erro na interpretação dessa propriedade?

7) Em qual momento é recomendado o uso da propriedade do border-box?

DESAFIO – Crie Um Formulário De Contato Usando As Propriedades Do Box Model

TEMA 08 – LAYOUT RESPONSIVO

Figura 13: Layout de sites – Fonte: igluonline

O QUE É LAYOUT

Layout é um termo técnico para aparência, significa arranjo, esquema ou design. Na área de design gráfico, layout é um rascunho da estrutura de uma página de revista, jornal ou página web. O layout engloba muitos elementos como: gráficos, textos, imagem, e a forma que eles se comportam dentro do espaço determinado pelo desenvolvedor.

É feito geralmente pelo design gráfico, em nossa área como desenvolvedores web, geralmente nos é enviado o layout predefinido, pronto para a implementação em código. Mas em alguns momentos, temos a necessidade de criarmos um layout, consequentemente devemos saber como funciona e o modo que é criado um layout, não só isso, mas um layout responsivo (adaptável em todos os dispositivos apresentados).

O layout de uma página depende da criatividade e do conteúdo que ela irá conter, deste modo, precisamos entender a necessidade do cliente, a função do site, quais informações que ele precisa ter destaque, e outras como alocar corretamente os menus e informações contidas nele.

OTIMIZAÇÃO DE LAYOUTS WEB

A otimização do layout é de suma importância para as páginas web, pois através dele, exploramos de maneira eficaz e eficiente todo o espaço determinado para as informações contidas na página web. Devemos analisar alguns itens para a otimização/criação de um layout responsivo, vamos analisar alguns deles;

S.E.O (Search Engine Optimization - Otimização de Mecanismos de Busca) – O SEO faz uso de ferramentas disponíveis na internet como o planejador de palavras chave do google, onde é feito a filtragem através das palavras chaves relacionadas ao tema do site, otimizando nas buscas, quando usamos os mecanismos de busca.

Foco e Clareza – Vamos supor que você tenha uma página da sua empresa na internet, e um cliente faz a busca com o mesmo assunto relacionado a sua empresa procurando um produto ou serviço. Ao navegar pelo site esse cliente deve achar a informação que precisa imediatamente para que não saia do seu site e vai buscar em outro. Pra isso o site deve conter clareza e foco em seu design, isso é a importância de um layout responsivo, que realmente que funciona como ponte e se transforme em venda. Clareza e objetividade no site, é o que faz toda a diferença, seu produto pode ser até melhor, ou seu serviço ser referencia no mercado, mas se seu site não tem clareza nas informações, com menus intuitivos e facilidade no acesso, o cliente consequentemente irá procurar outro site.

Carregamento Rápido – É comum pensarmos que o carregamento de um site não faz tanta diferença assim, mas ao pensar assim, estamos totalmente errados, pois a velocidade do carregamento de um site é determinante no quesito numero de acessos. Por exemplo você está navegando pela internet com os dados móveis no seu celular, ao entrar em um determinado site, se ele demora muito pra carregar, consequentemente você irá procurar outro site. Por esse, entre outros motivos, devemos nos atentar pelo rápido carregamento do site. O layout responsivo entra nessa parte, definindo regras para que o carregamento do site seja o mais rápido e eficiente possível.

Sem Excessos – Conhecemos o tão conhecido termo “*Menos é Mais*”, qualquer excesso em sites é prejudicial, tanto em sites que possuem cores em excesso que podem causar cansaço nos usuários, ou em sites monocromáticos que tornam o site sem vida e chatos e fazem que os usuários não fiquem muito tempo. Imagens em tamanhos desproporcionais (muito pequena ou muito grande), acaba tirando

o foco das informações que realmente importa ou passando desapercebida. Por isso é importante ter responsividade no layout para auxiliar a navegação do usuário no site. Devemos nos atentar em criar um layout simples e objetivo, onde as informações mais importantes com destaque, e links intuitivos pois devemos lembrar nem todos nossos usuários sabem navegar na internet.

Devemos lembrar que cada caso é um caso, não existem uma regra absoluta sobre layouts, a regra usada é que o layout deve proporcionar uma experiência boa para o usuário, ajudando a encontrar informações, sendo elegante, fácil acesso as informações e rápido carregamento. Esses são os pontos importantes para que possamos criar um layout responsivo que realmente funcione.

CSS GRID E FLEX BOX

O CSS Grid é um sistema usado para estruturar layouts permitindo configurar em duas dimensões com linhas e colunas. A grade é formada por essas linhas e colunas, por isso o nome *Grid.* Esse sistema é muito completo e poderoso, pois ele permite a definição de certos elementos de maneiras diferentes, o que no início pode até confundir, mas com o uso na prática acaba facilitando o entendimento.

Nesse capítulo vamos conhecer um pouco mais desse sistema que é bastante usado pelos desenvolvedores em CSS. Antes de mais nada temos que entender que novas propriedades CSS são essenciais para o uso do grid, onde temos elementos como grid container ou elemento pai e propriedades para os itens da grid ou *elementos filhos.*

Alguns detalhes são importantes para o aprendizado da CSS GRID, ele requer um desenvolvimento prático das habilidades não só de um layout, mas pensando como um sistema inovador de uma nova forma de estruturação de layouts. Vejamos algumas dicas importantes para o correto uso do CSS GRID, facilitando no entendimento das propriedades, e consequentemente usando de forma correto e simples.

Sempre use nomes ao invés de números, em CSS GRID sabemos que são compostos por linhas e colunas em sua essência. Por exemplo, esse CSS configura uma grade com 2 colunas e coloca o conteúdo principal da página na 2ª coluna:

```
.container {
  display: grid;
  grid-template-columns: 1fr 2fr;
}

.content {
  grid-column: 2;
}
```

O código acima até funciona, porém não utiliza um recurso importante do CSS GRID, onde é possível dar nomes específicos as linhas e colunas. Recomendo que use esse recurso sempre que puder, para facilitar no uso e em futurar alterações do layout. Veja esse mesmo exemplo de forma melhorada;

Embora esse exemplo seja simples, mas serve para entendermos de como esse recurso funciona. Em casos em que o layout tenha um código muito maior, usando esse recurso podemos identificar os elementos mais facilmente, acelerando o processo tanto de criação como o de alteração.

A legibilidade é essencial, já que estamos falando de GRID. A linha 3 desse modo está descrevendo tudo que está acontecendo dentro do grid container. Aqui não estamos listando apenas listando as colunas, mas também delineando a intenção de cada coluna. É importante esse uso, pois aqui estamos falando apenas de 2 colunas, mas quando o site tem mais colunas na página, a negligencia no uso desse recurso do CSS GRID, acaba complicando a vida do desenvolvedor web.

Nesse video temos com detalhes o uso do CSS GRID na prática, nele percebemos a importância de uma boa organização do código em CSS, quanto na responsividade das informações do layout.

```css
.container {
  display: grid;
  grid-template-columns: [sidebar] 1fr [content] 2fr;
}

.content {
  grid-column: content;
}
```

Como estamos trabalhando com uma grade (grid), precisamos fazer a indicação de quantas linhas e colunas essa grade terá. Para as colunas usamos o *grid-template-columns* e para as linhas usamos o g*rid-template-rows*.

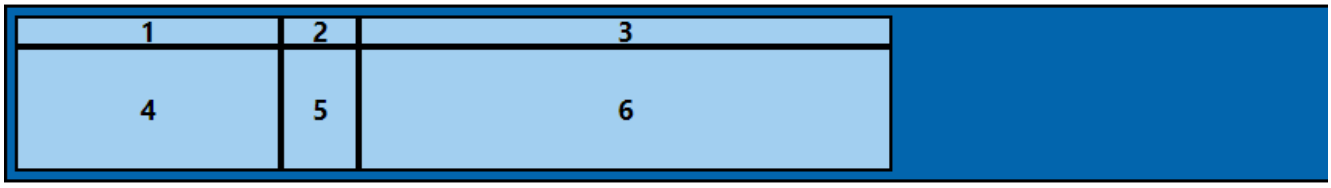
```css
.container {
  display: grid;
  grid-template-columns: 120px

}
```

Temos esse resultado no navegador;

Do mesmo modo podemos declarar as linhas utilizando a propriedade *grid-template-rows*, veja o código abaixo;

```css
.container {
  display: grid;
  grid-template-columns: 20% 50px 40%;
  grid-template-rows: 20px 80px
  }
```

Temos o resultado no navegador;



Para melhorar o uso dessas propriedades podemos adicionar a *grid-template-areas*, pois o CSS GRID nos permite dar nomes para cada área da nossa grade(grid), e indicar onde cada elemento deve ir. até agora só declaramos o tamanho das nossas colunas e linhas, notamos que os nossos itens vão se organizando na grid de acordo com a ordem que estão no documento HTML.Com essa propriedade fica simples entender a estrutura da grade. Vamos definir o nosso HTML colocando as classes nos itens da grade(grid) para indicarmos a posição de cada um deles no CSS.

```html
<!DOCTYPE html>
<html lang="pt">
<head>
    <meta charset="UTF-8">
        <title>Grid - Container</title>
</head>
<body>
    <div class="container" >
        <div class="item header" ></div>
        <div class="item sidenav" ></div>
        <div class="item content" ></div>
        <div class="item footer" ></div>
    </div>
</body>
</html>
```

Nota – se que nomeamos cada item das classes, facilitando a identificação de cada um deles no CSS. Para cada item, declaramos a propriedade *grid-area*, que nos permite indicar o nome da área que cada

elemento deverá ocupar na *grid.* Aqui está nomeado como exemplo, mas sinta-se a vontade para nomeá-las de acordo com a necessidade.

```
.header{
  grid-area: header;
}
.sidenav{
  grid-area: sidenav;
}
.content{
  grid-area: content;
}
.footer{
  grid-area: footer;
}
```

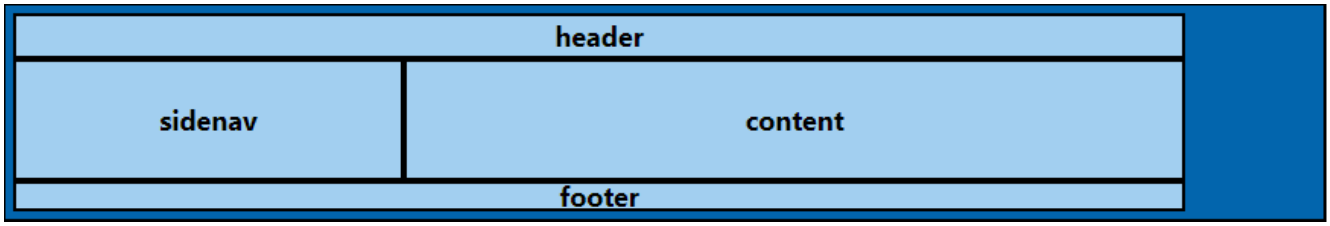
Para nomearmos às areas, escrevemos dentro das aspas (“ ”), sendo assim , ao abrir novas aspas estamos indicando que estamos indo para a linha seguinte. Note o exemplo abaixo;

```
.container{
  grid-template-areas:
   "header header"
   "sidenav content"
   "footer footer";
}

```

Nesse caso, repetimos *header* e *footer* duas vezes porque declaramos 2 colunas, e queremos que eles ocupem as duas.

```
.container {
  display:grid;
  grid-template-columns: 30% 60%;
  grid-template-rows:30px 80px 20px;
  grid-template-areas:"header header" "sidenav content" "footer footer"
  }
```

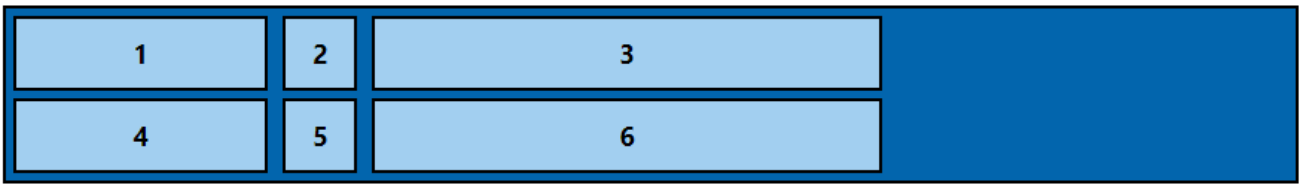
Veja o resultado no navegador;



Percebemos que nossos itens estão colados, temos outra propriedade que podemos espaçar as linhas e colunas, a propriedade *grid-column-gap* e *grid-column-row*. Vamos ver o código em CSS;

```
14  .container {
15    display: grid;
16    grid-template-columns: 20% 50px 40%;
17    grid-template-rows: 50px 50px;
18    grid-column-gap: 10px;
19    grid-row-gap: 5px;
20
```

Veja o resultado no navegador:



O *grid-gap* é um atalho para declararmos as suas propriedades em um só lugar. Podemos usar de duas formas, passando um único valor, onde os dois serão aplicados para o espaço de linhas e colunas, ou passando dois valores o primeiro será para linhas e o segundo para colunas.

```
14    .container{
15      grid-gap: 20px;
16      grid-gap: 20px 50px;
17  }
```

O uso do prefixo *grid-* destas propriedades mostradas aqui, estão sendo removidos, e já são suportados nos navegadores modernos. Deste modo também pode-se utilizar as propriedades *column-gap, row-gap* e *gap*.

Temos outras propriedades onde podemos definir, de maneira efetiva e simples, vejamos elas a seguir:

JUSTIFY-ITEMS – Faz o alinhamento de todos os itens da *grid* horizontalmente dentro da sua própria célula(área). Segue os valores dessa propriedade;

- **stretch (padrão):** estica os itens horizontalmente para eles preencherem sua célula.
- **start:** alinha os itens no início de suas células.
- **end:** alinha os itens no final de suas células.
- **center:** alinha os itens no centro de suas células.

ALIGN-ITEMS – Determina o alinhamento de todos os itens da grid verticalmente dentro da sua própria célula (área). Segue os valores dessa propriedade;

- **stretch (padrão):** estica os itens verticalmente para eles preencherem sua célula.
- **start:** alinha os itens na parte de cima de suas células.
- **end:** alinha os itens na parte de baixo de suas células.
- **center:** alinha os itens no centro de suas células.

JUSTIFY-CONTENT – O *justify-item* alinha cada item dentro da sua própria área. Já o *justify-content* alinha horizontalmente as áreas em relação ao container.

stretch: redimensiona os itens da grid para preencherem a largura do container. Apenas funciona se você não tiver declarado uma largura para as colunas.

- **start:** alinha as áreas no início da grid.
- **end:** alinha as áreas no final da grid.
- **center:** alinha as áreas no centro da grid.
- **space-around:** distribui as áreas com o espaço a sua volta.
- **space-between:** distribui as áreas com o espaço entre elas.
- **space-evenly:** distribui o espaço igualmente entre as áreas.

ALIGN-CONTENT – O *align-item* faz o alinhamento de cada item dentro da sua própria área. Já o *align-content* alinha verticalmente as áreas em relação ao container. Veja os valores dessa propriedade;

- **stretch:** redimensiona os itens da grid para preencherem a altura do container. Apenas funciona se você não tiver declarado uma altura para as linhas.
- **start:** alinha as áreas no topo da grid.

- **end:** alinha as áreas na base da grid.
- **center:** alinha as áreas no centro da grid.
- **space-around:** distribui as áreas com o espaço a sua volta.
- **space-between:** distribui as áreas com o espaço entre elas.
- **space-evenly:** distribui o espaço igualmente entre as áreas.

Essas propriedades são as principais de uso quando falamos de CSS GRID, fazem parte de um layout responsivo, em sua maioria, recomendo que treine essas propriedades modificando os valores para compreender melhor seu uso.

FLEXBOX

O CSS Flexible Box Layout Model ou simplesmente **Flexbox** é uma das especificações do CSS3 que promete fazer a organização dos elementos na página web previsivelmente quando o layout responsivo se faz necessário, sendo visualizado em diversos tamanhos de tela em diferentes dispositivos. Ele nos ajuda na organização desses elementos sem usar o *float* e também nos auxilia resolver problemas de ***BOX MODEL*** que normalmente acontecem quando acrescentamos o *border*, *margin* e *padding* além da largura do elemento.

A utilização do Flexbox é bem simples; os filhos deum elemento em Flexbox pode se posicionar em qualquer direção e pode ser definido em dimensões flexíveis para se adaptar. Podemos posicionar diversos elementos independente da sua posição no HTML, e isso é muito bom, pois deste modo está resolvendo um dos problemas do float e da sua dependência com a estrutura dos elementos em HTML.

É necessário entender que o Flexbox é uma maneira nova de posicionar elementos no seu layout, com isso precisamos de novos nomes para a identificação dos elementos da estrutura. Vamos conhecer cada um deles;

FLEX CONTAINER – É o elemento que envolve a estrutura como um todo. Podemos definir que um elemento é um *Flex Container* com a propriedade *display* com os valores *flex ou inline-flex*.

FLEX ITEM – São os elementos filho (child) do *flex container*.

AXES OU EIXOS – São as direções básicas que existem em um *Flex Container*: *main axis*, que seria o eixo horizontal ou principal, e o *cross axis* que seria o eixo vertical.

DIRECTIONS – Determina o inicio e o fim do fluxo dos itens. Seguem o vetor estabelecido pelo modo tradicional de escrita; da esquerda pra direita, direita pra esquerda, etc.

As propriedades *order, flex-wrap* e *flex-flow* também se fazem importantes para a definição de um layout feito com Flexbox. A propriedade *order* determina o lugar que os elementos aparecerão, a propriedade *flex-flow* determina a ordem do fluxo em que os elementos aparecerão e a propriedade *flex-wrap* define se os elementos serão forçados a ficar a mesma linha ou se eles irão quebrar em várias linhas.

DEFININDO UMA ESTRUTURA FLEXBOX

Para darmos inicio vamos começar com o código em HTML;

```html
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <title>Exemplo Flexbox</title>
    <meta charset="utf-8">
</head>
<body>

<div class="main">
    <div>Primeiro div</div>
    <div>Segundo div</div>
    <div>Terceiro Div</div>
</div>

</body>
</html>
```

Agora no CSS vamos definir que a *div main* irá virar um Flex Container, para isso definimos a propriedade display com os valores *flex* ou *inline-flex.* O valor do Flex faz o elemento ser um BLOCO, com o valor *inline-flex*, o elemento vira um inline-block com as propriedades do Flex Container. Vamos ver como fica no código em CSS;

```css
.main {
  display: -webkit-flex;
  display: flex;
}

/** Para deixar uma cor de fundo nos divs filhos **/
.main div {
  background: #CCC;
  margin-right:10px;
}
```

Nesse código automaticamente os divs da nossa classe. main ficarão um ao lado do outro. Para definirmos que os *flex itens* comecem a ter uma distribuição na largura, basta definirmos para eles a propriedade *flex*, segue modelo abaixo:

```
.main {
    width:500px;
    background: #EFEFEF;
    height:500px;
    display: -webkit-flex;
    display: flex;
}

.main div {
    background: #CCC;
    margin-right:10px;

    -webkit-flex:1 1 auto;
    flex:1 1 auto;
}
```

A propriedade *flex* é um atalho para as propriedades *flex-grow, flex-shrink* e *flex-basis*. Vamos supor que queremos modificar a ordem que os elementos aparecem. É bem simples, é so usarmos a propriedade *order* e modificar as colunas:

```
.main div.primeiroDiv {-webkit-order:2;}
.main div.segundoDiv {-webkit-order:3;}
.main div.terceiroDiv {-webkit-order:1;}
```

FLEX-FLOW

Vamos supor que ao invés de colunas, nós optamos por linhas. Caso seja necessário criar uma versão para mobile, em mobile ter colunas é ruim, então devemos definir apenas a propriedade *flex-flow* com o valor *column*, vamos ver no código a seguir:

```
.main {
    width:500px;
    background: gray;
    height:500px;
    display: -webkit-flex;
    display: flex;

    -webkit-flex-flow: column;
    flex-flow: column;
```

Essa propriedade possui outros valores, como *row, row-reverse* e *column-reverse*. O *row-reverse* e o *column-reverse* invertem a ordem dos elementos automaticamente sem a obrigação de modificar novamente a propriedade *order* definido de cada elemento. O Flexbox funciona perfeitamente no Google Chrome, alguns testes foram feitos no Safari e no Mozilla e não funcionaram, no Edge funciona com o (-ms) aplicado no CSS. O importante saber pra qual uso será o site assim definindo assim suas configurações dentro do código para não ter erros no navegador.

Esse video mostra de forma mais detalhada o uso do Flexbox, essa poderosa ferramenta usada em layouts de páginas. Confira no video:

RESPONDA:

1) Porque devo fazer um layout de um site?

2) Porque é importante definir um layout responsivo?

3) Quais os principais aspectos devo observar na hora de criar um layout?

4) Quais as ferramentas que o CSS3 disponibiliza para um layout responsivo e eficaz?

5) O que é CSS GRID?

6) O que é Flexbox?

7) Qual a função da propriedade *align-item*?

8) Qual a diferença entre o *align-item* e o *align-content?*

9) Como chama as estruturas de Flexbox?

10) Qual a vantagem do Flexbox e o box model?

DESAFIO – Crie um site de perfil pessoal com uso do CSS Grid ou Flexbox.

TEMA – 09 JAVASCRIPT

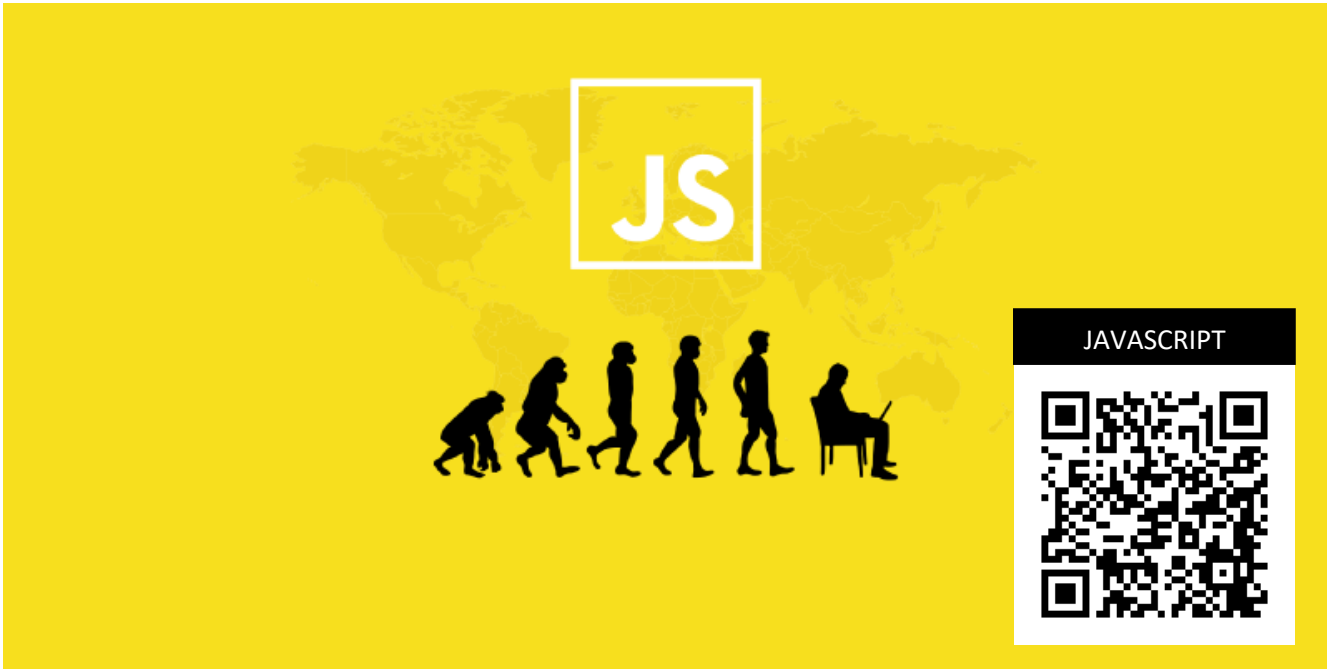


Figura 15: Javascript – Fonte: dankicode

O QUE É JAVASCRIPT

Em 1996 o programador Brendan Eich cria a linguagem Javascript, uma linguagem de programação de alto nível muito usado em páginas web devido ser uma linguagem que permite interações diferentes com o usuário e de fácil aplicação, o javascript tem compatibilidade com todos os navegadores, facilitando a aplicação em páginas web. O javascript também é conhecida como uma linguagem de programação comportamental, criando conteúdos dinâmicos, controles de mídias e animações para deixar as páginas web com mais interatividade e mais interessantes. Mas porque usar o Javascript nas páginas web e não outras linguagens? Vamos conhecer algumas características importantes do javascript que se sobressaem das outras linguagens.

- O javascript é uma linguagem que não precisa de um compilador, pois a interpretação é feita nos navegadores internet.
- É uma linguagem de fácil aprendizado.
- É totalmente compatível com várias plataformas e navegadores.
- É mais leve que as outras linguagens de programação.
- Os erros no javascript são mais fáceis de serem localizados, logo são mais fáceis de serem corrigidos.
- Com o javascript é possível fazer sites mais interativos e que segura a atenção do usuário.

SINTAXE BÁSICA

O javascript é uma linguagem *case-sensitive (*linguagem que faz diferenciação entre letras maiúsculas e minúsculas), que usa o conjunto de caracteres **Unicode**. Por exemplo se uma variável se chama **Loja**, quando eu for usa-la se eu escrever **loja**, certamente vai dar erro, pois ela é *case-sensitive*. As instruções no javascript são chamadas de ***declaração*** e são separadas por ponto e vírgula **(;)**. O código fonte dos scripts em Javascript são lidos da esquerda pra direita e são convertidos em uma sequencia de elementos de entrada. Vamos conhecer as sintaxes do javascript e entendermos seu uso:

COMENTÁRIOS – Os comentários em javascript são definidos de duas formas, a primeira é comentário de uma única linha *(// comentário de uma linha)*, e a segunda é o comentário de múltiplas linhas *(/* comentário de múltiplas linhas*/)*.

DECLARAÇÕES – Existem 3 tipos de declarações em javascript, são elas:

- Declaração de variáveis, onde opcionalmente, inicializando-a com um valor.
- Declaração let, onde permite que declare várias variáveis limitando o escopo no bloco.
- Declaração de constantes, onde possuem escopo de bloco. O valor de uma constante não pode ser alterado por uma atribuição ou ser redeclarada.

VARIAVEIS - As variáveis são como o sistema trata uma parte da memória para alocar/guardar informações, onde podem ser usadas posteriormente. Exemplo:

Preciso fazer uma conta aritmética simples e mostrar o resultado através de uma variável. Então vamos lá – Tenho a variável chamada soma, nela será armazenada a soma entre dois números, *numero1* +

numero2 = *soma,* logo se preciso mostrar a soma dos números não preciso de um *numero3*, apenas eu chamo a variável *soma*.

Em javascript o nome das variáveis é chamado de *identificadores* e obedecem a algumas regras – Um identificador javascript deve começar com uma **letra, underline (_), ou cifrão ($)**; os caracteres subsequentes podem também ser números (0-9). Pelo fato do javascript ser *case-sensitive*, letras incluem caracteres de “A” a “Z” (maiúsculos) e de “a” a “z” (minúsculos).

Vamos ver algumas formas usadas para a declaração de variáveis:

- ✓ Com a palavra-chave **var**. Exemplo – *var x = 75.* Esta sintaxe pode ser usada para declarar tanto variáveis locais como globais.
- ✓ Com a palavra-chave **let**. Exemplo – *let y = 50.* Esta sintaxe pode ser usada para declarar uma variável local de escopo de bloco.

Uma variável sendo declarada usando *var* ou *let* sem especificar o valor inicial tem o valor **undefined**.

As variáveis se dividem nos seguintes tipos:

String – São variáveis de texto, quase sempre chamada de “cadeia de caracteres”. Os valores atribuídos a esse tipo normalmente usam aspas duplas (“ “) ou aspas simples (‘ ’).

Float – Variáveis com ponto flutuante, ou seja, variáveis com números em casas decimais.

Boolean – Tipos de variáveis com valores true(verdadeiro) e false(falso).

Int – São variáveis com números inteiros.

Arrays – Uma variável array faz a referência a vários espaços na mémoria, é um conjunto de valores organizados por um índice.

JAVASCRIPT NA PRÁTICA

O Javascript pode ser inserido na página HTML de duas formas:

INTERNO – Inserido no próprio documento em HTML utilizando as tags **<script> e </script>**, conforme na imagem abaixo:

```
<!DOCTYPE html>
<html>
  <head>
    <script type="text/javascript">
      //código JavaScript
    </script>
  </head>
  <body>
  </body>
</html>
```

EXTERNO – Implantado em um documento externo **(.js)**, em um documento separado do HTML. Deste modo devemos fazer a referencia desse documento javascript na página HTML, conforme imagem abaixo:

```
<!DOCTYPE html>
<html>
  <head>
    <script type="text/javascript" src="meuArquivo.js"></script>
  </head>
  <body>
  </body>
</html>
```

Usando as variáveis no javascript como vimos acima, vamos ver na prática o funcionamento da linguagem, sempre atento nas regras de declaração de variáveis. Vamos ver um exemplo:

```
var nome;
nome = 'Fulano';
var idade = 30;
idade = 30 + 20 - 10*5;
```

O javascript por ser uma linguagem de programação, nos oferece métodos mais avançados do que as linguagens vistas até aqui, uma delas são as funções. As funções podem receber parâmetros e retornar valores, porém o tipo de retorno e o tipo dos parâmetros não precisa ser definido previamente. Vamos ao exemplo abaixo:

```
function exibirMensagem()
{
  alert('Olá, seja bem vindo(a)!');
}
```

Esse é um exemplo de uma função em javascript sem parâmetros e sem retorno.

```
function somar(A, B)
{
  return A + B;
}
```

Esse é um exemplo de uma função em javascript com parâmetro e com retorno. Nota – se que para definirmos um retorno em uma função, devemos utilizar a palavra *return* seguida do valor ou expressão do resultado.

Controle de Fluxo e Estruturas Condicionais

Estruturas de controle de fluxo, conhecidas também como estruturas condicionais, são estruturas onde se define condições em forma de looping, encadeando processos onde o resultado depende de ações predefinidas em uma das condições. Vamos ver o código abaixo como exemplo:

```
if(condição1)
{
  //ação se condição 1 verdadeira
}
else if (condição2)
{
  //ação se condição 2 verdadeira
}
else
{
  //ação se nenhuma das condições for verdadeira
}
```

Nesse exemplo vimos a sintaxe da estrutura *if-else*, o bloco de instruções *else* pode ser omitido, em casos que apenas uma condição precise ser avaliada. No próximo exemplo temos uma variável chamada idade sendo avaliada, e dependendo do seu valor, uma mensagem é exibida na tela.

```
if(idade > 18)
{
alert('É maior de idade')
}
else
{
alert('É menor de idade');
}
```

Em outros casos onde necessitamos de mais condições, usamos a estrutura *switch* para verificar o valor da variável e retornar o valor.

```
switch(variável)
{
case valor1:
//ações caso valor1
break;
case valor2:
//ações caso valor2
break;
case valor3:
//ações caso valor3
break;
default:
//ações caso nenhum dos valores
break
}
```

O comando *switch* faz a verificação do valor de uma variável, sendo que pra cada opção executa um conjunto de ações. Caso nenhum dos valores for verificado, os comandos do bloco *default* são executados. O bloco *default*, porém, pode ser omitido caso não exista uma ação padrão a ser executada se nenhuma das opções for observada.

```
switch(dia)
{
case 1:
alert('Hoje é domingo');
break;
case 2:
alert('Hoje é segunda');
break;
case 3:
alert('Hoje é terça');
break;
default:
alert('Hoje não é nem domingo, nem segunda, nem terça');
break
}
```

O comando *switch* usado em outro exemplo, nesse caso a verificação de qual dia da semana.

LAÇOS DE REPETIÇÃO

Existem estruturas de repetições, onde comando executa o comando dentro do loop até que o retorno seja diferente do determinado para entrar em loop. No javascript uma desses laços se chama *while* (enquanto – português). Vamos ver um exemplo no código:

```
1   while(condição)
2   {
3     //ações
4   }
```

Essa estrutura de repetição *while* é muito usada para executar um conjunto de ações enquanto uma condição for verdadeira. Quando a condição retorna o valor falso, o bloco de comando é finalizado. Vamos ver um exemplo bastante usado em javascript:

```
1   var contador = 0;
2   while(contador < 5)
3   {
4     alert('Olá');
5     contador = contador + 1;
6   }
```

Outra estrutura semelhante é a *do – while*, que executa um bloco de ações até que uma condição seja falsa. Mas, essa condição é validada após a execução dos comandos, fazendo com que estes sejam executados pelo menos uma vez.

```
1   do
2   {
3     //ações
4   }
5   while(condição)
6
```

Um exemplo semelhante ao comando *while* pode ser usado para representar a sintaxe do comando *do-while*. Vamos ver o código abaixo:

```
var contador = 0;
do
{
alert('Olá');
contador = contador + 1;
}
while(contador < 5)
```

E a última estrutura de repetição usada no javascript é o *for,* ele usa um contador para executar um bloco de ações uma determinada quantidade de vezes. Vejamos um exemplo da sintaxe:

```
for(inicialização; condição; complemento)
{
  //ações
}
```

Fica mais fácil entender essa estrutura quando observamos um exemplo prático. No exemplo abaixo, temos uma variável *contador* é inicializado com o valor zero, e enquanto for menor que 10, o laço deve ser executado.

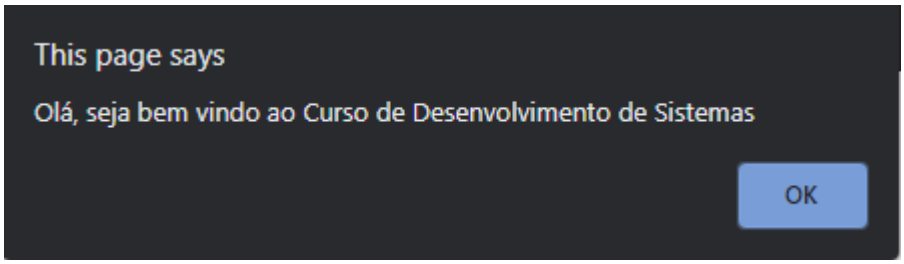
```
var contador;
for(contador = 0; contador < 10; contador++)
{
  alert(contador);
}
```

COMANDO ALERT

A linguagem javascript possui diversas funções nativas, vamos conhecer as principais. Sendo uma delas a função ***alert.*** Essa função serve para exibir uma mensagem em uma janela pop-up. A função *alert* recebe apenas um parâmetro, que é texto a ser exibido. Vejamos o exemplo abaixo:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8"/>
    <script type="text/javascript">
      alert("Olá, seja bem vindo ao Curso de Desenvolvimento de Sistemas")
    </script>
  </head>
  <body>
  </body>
</html>
```

Veja o resultado no navegador:

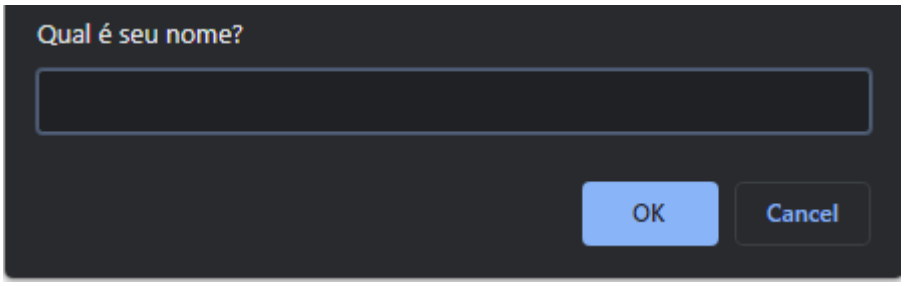


COMANDO PROMPT

A função *prompt* é semelhante a função *alert*, a diferença, que na janela pop-up que essa função exibe é possível digitar texto, assim coletando dados digitados pelo usuário. Vamos ao exemplo da sintaxe:

```
1   <!DOCTYPE html>
2   <html>
3     <head>
4       <meta charset="UTF-8"/>
5       <script type="text/javascript">
6         var nome;
7         nome = prompt("Qual é seu nome?");
8         alert("Olá, " + nome);
9       </script>
10    </head>
11    <body>
12    </body>
13  </html>
```

Veja o resultado no navegador:



Nesse caso o valor coletado pelo *prompt*, foi atribuído a uma variável, e a função *alert* exibiu o valor dessa variável, sendo o valor digitado “Aluno NovoTec”.

OBJET WINDOW

O objeto window é usado para fazer a manipulação das janelas do navegador. Toda vez que abrirmos o navegador simultaneamente esse objeto é gerado, isso acontece porque esse objeto representa exatamente essa janela que foi aberta. Para criarmos esse objeto, não precisamos usar nenhum tipo de linguagem, como HTML, porque isso é feito automaticamente pelo navegador. Com esse objeto podemos criar e abrir novas janelas de maneiras diferentes. Esses processos são possíveis através das *propriedades* e *métodos* que o objeto window possui. Vamos ver a sintaxe desse objeto:

- *window.propriedade*
- *window.metodo*

PROPRIEDADES

As propriedades do objeto window tem como objetivo modificar os aspectos em relação à janela do navegador.

- ***Closed –*** Esta propriedade retorna um valor booleano indicando se a janela foi fechada.
- ***DefaltuStatus –*** Esta propriedade nos permite definir uma mensagem padrão que será exibida no status do navegador.
- ***Document –*** Esta propriedade possui todas as características da página HTML, onde podemos integrar no meio de um comando javascript tag HTML.
- ***Frames:*** Array de frames em uma janela.
- ***History:*** Esta propriedade contém uma lista representando as URLs que o usuário já visitou. Podemos acessar todas as URLs visitadas pelos usuários atribuindo os seguintes métodos a essa propriedade: current, next e previous.
- ***InnerHeigth:*** Com esta propriedade podemos definir ou obter a altura da área onde o conteúdo é apresentado, não a altura do navegador em si.
- ***InnerWidth***: Com esta propriedade podemos definir ou obter a largura da área onde o conteúdo é apresentado, não a largura do navegador em si.
- ***Length:*** Informa a quantidade de frames existentes em uma janela.
- ***Location****:* Esta propriedade tem informações referentes ao endereço corrente. Um exemplo é a propriedade hostname, que retorna o nome do servidor que está hospedando a página carregada. Nessa propriedade temos mais dois métodos:

    - **Reload:** Que força o navegador a carregar a página.
    - **Replace:** Que carrega a URL informada.

- ***Name**:* Podemos utilizar para definir ou obter o nome da janela.
- ***Navigator**:* Esta propriedade contém informações sobre o navegador, nome, versão, e outras informações. O objeto navigator possui também um método interessante:

    - javaEnabled(): Que informa se o navegador está com o Java habilitado.

Vamos utilizar o objeto *document* como exemplo para aprendermos como funciona a sintaxe do *objeto window.*

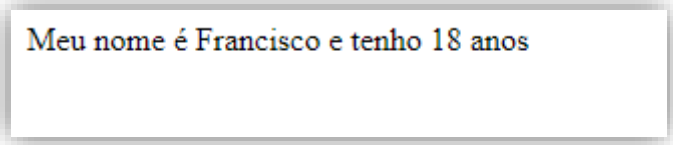
DOCUMENT.WRITE

O método *document.write* serve para escrever informações no documento HTML. Ele é muito simples de usar, basta colocar dentro de parênteses **( )** o que deseja que apareça no documento. Vamos definir duas

```
<!DOCTYPE html>
<html>
 <head>
   <title>Curso Desenvolvimento de Sistemas</title>

   <script type="text/javascript">
    var nome="Francisco";
    var idade=18;

    document.write("Meu nome é ", nome," e tenho ", idade, " anos");
   </script>

 </head>
</html>
```

variáveis, uma com um nome e outra com idade, então vamos escrever no documento a string (texto): “Meu nome é [nome] e tenho [idade] anos” No código HTML fica assim:

Note que quando usamos texto para aparecer no documento, ele deve estar entre **aspas “ ”** , e quando usamos uma variável para mostrar o valor é somente o nome da variável, nesse caso *nome e idade.* Vamos conferir o resultado no navegador:

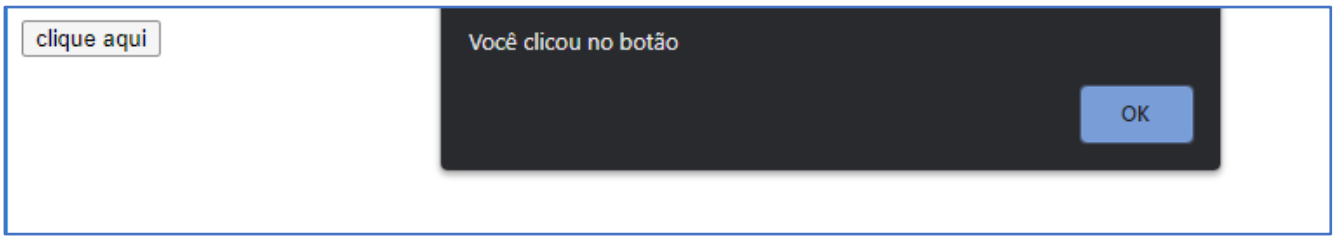


MANIPULANDO EVENTOS

Os eventos são disparados quando alguma ação é executada, como clique num botão, ou a digitação de valores em um input. No código podemos atribuir valores aos eventos como se fossem propriedades, desde que o valor atribuído seja um código a ser executado. Por exemplo na imagem abaixo, mostra o código de uma página com um botão, que, ao ser clicado, exibe uma mensagem (alert):

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8"/>
  </head>
  <body>
<button onclick="alert('Você clicou no botão');">clique aqui</button>
  </body>
</html>
```

Vamos ver o resultado no navegador:



Nesse exemplo as tags **<script> ... </script>** não foram necessárias, pois o código encontra-se dentro do próprio *button*. Caso o código a ser executado no evento seja muito extenso, é possível criar uma função para isso e associa-la ao evento em questão. Vamos ver um exemplo:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8"/>
    <script type="text/javascript">
      function clique_botao()
      {
        alert("Você clicou no botão");
      }
    </script>
  </head>
  <body>
    <button onClick="clique_botao()">clique aqui</button>
  </body>
  </html>
```

Esse exemplo tem como resultado o mesmo do anterior no navegador, a diferença dele, é que o evento foi chamado no próprio botão.

OnMouseover – O evento onmouseover executa um código em Javascript quando o ponteiro do mouse é movido para um elemento ou para um de seus filhos.

OnMouseout – O evento executa um Javascript quando o ponteiro o mouse é movido para fora de um elemento ou de seus filhos.

OnLoad – Esse evento carrega o elemento Javascript.

OnBlur – Esse evento remove o foco do elemento.

OnChange – Esse evento altera o valor do elemento.

OnClick – Esse evento executa um comando quando é clicado pelo usuário.

OnKeyPress – Esse evento é executado quando o usuário pressiona uma tecla sobre o elemento.

OnSubmit - Esse evento é usado em formulários, onde é definida uma ação em Javascript.

OnFocus – Esse evento é aplicado um foco no elemento.

Os eventos são muito importantes no Javascript, pois facilita na interação com o usuário e é carregado de modo instantâneo.

Usamos os eventos de duas maneiras:

INLINE – Dessa maneira o evento é definido diretamente na tag do elemento, vamos ao exemplo:

```
<!DOCTYPE html>
<html>
<head>
<title>Usando o evento onClick no Javascript</title>
</head>
<body>

<h1 onclick="this.innerHTML='Isso acontece quando usamos o evento onClick!'"
>Clique nesse link para testar o evento onClick!</h1>

</body>
</html>
```

EXTERNO – Para usarmos um evento externo, é preciso de um manipulador de evento, nesse exemplo vamos usar o *event listener* que adiciona ou remove um evento sobre qualquer elemento. Esse evento nos disponibiliza duas funções principais, são elas:

***addEvent* –** Adiciona uma função que será disparada quando ocorrer determinado evento no objeto.

***removeEvent* –** Remove um *listener* previamente adicionado em um objeto e retorna o valor **true**, em caso de sucesso.

Vamos ver um exemplo do uso desse evento:

```
<script type="text/javascript" src="/path/to/event-listener.js"></script>

<form>
  <input type="text" name="a" />
  <input type="submit" />
</form>
```

Vamos ver mais alguns exemplos de eventos mais usados no Javascript:

EVENTO ONLOAD

```html
<!DOCTYPE html>
<html>
<head>
<title>Usando eventos no Javascript</title>
</head>
<body onload="checkCookies()">

<script>
function checkCookies()
{
if (navigator.cookieEnabled==true)
  {
  alert("Cookies são permitidos")
  }
else
  {
  alert("Cookies não são permitidos")
  }
}
</script>

<p>Irá aparecer um alert dizendo se os cookies estão ou não liberados em seu navegador</p>
</body>
</html>
```

EVENTO ONCHANGE

```html
<!DOCTYPE html>
<html>
<head>
<title>Usando eventos no Javascript</title>
<script>
function myFunction()
{
var x=document.getElementById("fname");
x.value=x.value.toUpperCase();
}
</script>
</head>
<body>

Insira seu Nome: <input type="text" id="fname" onchange="myFunction()">
<p>
Ao clicarmos fora do input text o texto escrito nele ficará todo em caixa alta.</p>

</body>
</html>
```

EVENTOS ONMOUSEOVER E ONMOUSEOUT

```
<!DOCTYPE html>
<html>
<head>
<title>Usando eventos no Javascript</title>
</head>
<body>

<div onmouseover="mOver(this)" onmouseout="mOut(this)"
style="background-color:#D94A38;width:120px;height:20px;padding:40px;">Passe o mouse em mim</div>

<script>
function mOver(obj)
{
obj.innerHTML="Obrigado"
}

function mOut(obj)
{
obj.innerHTML="Passe o mouse em mim"
}
</script>
</body>
</html>
```

EVENTOS ONMOUSEDOWN, ONMOUSEUP E ONCLICK

```
<!DOCTYPE html>
<html>
<head>
<title>Usando eventos no Javascript</title>
</head>
<body>

<div onmousedown="mDown(this)" onmouseup="mUp(this)"
style="background-color:#D94A38;width:90px;height:20px;padding:40px;">Clique aqui</div>

<script>
function mDown(obj)
{
obj.style.backgroundColor="#1ec5e5";
obj.innerHTML="Solte o clique"
}

function mUp(obj)
{
obj.style.backgroundColor="#D94A38";
obj.innerHTML="Obrigado"
}
</script>

</body>
</html>
```

Por essas e outras que o javascript vem se tornando cada vez mais usado e ganhando adeptos todos os dias, uma linguagem leve, interativa e de fácil aplicação.

RESPONDA:

1) O que é o javascript?

2) Qual o nome do criador do javascript e em que ano foi criado?

3) Quando usamos o javascript para páginas web, qual as vantagens dele quanto as outras linguagens?

4) Como posso inserir um arquivo javascript numa página em HTML?

5) Quais as estruturas de repetição do Javascript?

6) O object window é usado para _____ no navegador.

7) O que faz a função do alert?

8) Qual a função do prompt?

9) Qual a diferença de inserir um link para um arquivo javascript interno e um externo?

10) Quais os principais tipos de variáveis no javascript?

DESAFIO

Crie uma rotina em Javascript em que o usuário digite um nome e a idade, e na janela do navegador apareça o nome e a idade digitados pelo usuário.

TEMA 10 – COOKIES, BIBLIOTECAS E FRAMEWORKS

Figura 16: Bibliotecas e Frameworks Javascript – Fonte: elo7

O QUE SÃO COOKIES

Os cookies são pequenos arquivos criados por sites em que o usuário visitou, e que são salvos no computador do usuário, por meio do Browser. Eles contem informações que são usadas para identificar o visitante, seja para personalizar a página de acordo com o perfil, ou até mesmo para facilitar o transporte de dados entre as páginas de um mesmo site. Cookies também são comumente relacionados a casos de violação de privacidade em sites web.

Essa tecnologia existe desde seu início da internet doméstica, nos anos 1990, foi criada para atender às demandas do E-commerce, mas foi ganhando diferentes propósitos ao decorrer dos anos. A lei GDPR (General Data Protection Regulation – Regulamento Geral de Proteção de Dados). Ela regulamenta a transparência das informações fornecidas ao site pelo usuário e obriga a exibição de um aviso sobre a politica de cookies adotada pelo site. Os cookies podem ser divididos em duas categorias principais, com muitos subconjuntos:

COOKIES DE SESSÃO – São cookies que permanecem no navegador mantêm suas informações até que as sessões sejam fechadas. Quando uma nova janela do navegador é aberta, o mesmo usuário é tratado como um novo visitante e deve inserir suas credenciais de login.

COOKIE PERSISTENTE – Os cookies persistentes têm uma vida útil determinada e permanecem no navegador até o período decorrido, ou até mesmo, seja excluído manualmente. Os sites que usam cookies persistentes lembram os usuários mesmo depois de fecharem o navegador. Esse tipo de cookie ativa recursos como carrinhos de comprar persistentes, que retém produtos adicionados ao carrinho entre sessões.

O cookie em si é apenas uma pequena linha de texto, nele não contém informações sobre o usuário ou a máquina do usuário. Nos cookies normalmente contém a URL do site que colocou o cookie, um número único gerado exclusivamente para aquele acesso e a data de validade do cookie.

Vamos ver os tipos de sites que usam essa tecnologia e como funciona em cada um deles:

E-COMMERCE (COMÉRCIO ELETRÔNICO)

No comércio eletrônico essa estratégia serve para manter itens adicionados a um carrinho de compras mesmo que o usuário mude de página, ao navegar entre produtos diferentes. Sem o uso dos cookies, os sites não seriam capazes de manter informações salvas no meio do caminho. Além de permitir a preservação de dados durante o acesso, os cookies funcionam como uma espécie de identificador do computador. Usando esse recurso, os endereços online podem personalizar anúncios e destaques a serem exibidos para os usuários, com base no histórico de visitas anteriores.

Os sites de E-commerce usam uma combinação de cookie de sessão e cookie persistentes para criar uma experiencia perfeita no carrinho de compras. Quando o usuário adiciona itens ao carrinho, os cookies da sessão acompanham os itens. Caso o usuário abandonar o carrinho, os cookies persistentes fazem a recuperação de suas seleções do banco de dados na próxima vez que ele visitar, ou até mesmo permite que o desenvolvedor crie campanhas personalizadas de redirecionamento que incentivem o usuário a revisitar o carrinho. Esse recurso é de grande ajuda para incentivar as conversões para vendas.

LOGIN

A recuperação de algo que foi digitado pelo usuário pode ser feita pelo uso de cookies no seu site. Usando esse recurso, nomes, endereços, dados de cadastro em formulários ou até mesmo termos de busca usados em campos de pesquisa podem ser restaurados. Os cookies funcionam para lembrar que um usuário está logado em um determinado site. Ao entrar em um serviço que exija senha, a autenticação fica ativa mesmo após atualizar a página ao navegar nos endereços internos. Sem o uso

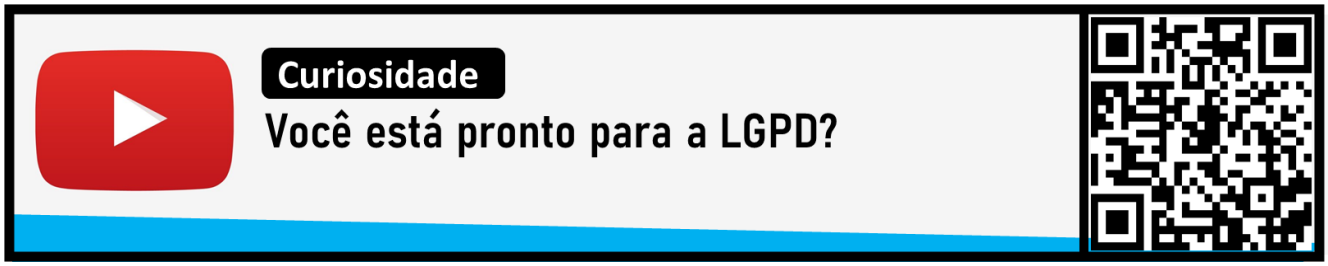
dos cookies, sites como Youtube, Netflix, Amazon Prime ou qualquer outro com conta de usuário vinculadas precisaria de um novo login toda vez que um video for carregado.

PRIVACIDADE E SEGURANÇA

Ao fazer usos carrinhos de compras, logins com validade duradoura e outros elementos que melhoram a experiência de navegação, os cookies pedem em troca a privacidade do usuário. Esses sites podem usar as informações disponibilizadas pelos usuários nesses arquivos para compor um padrão de identificação na web, que por sua vez não depende de visitas diretas ao site. Com o uso dessa prática, uma loja pode conhecer o padrão de consumo do usuário mesmo que ele nunca tenha visitado o site.

A segurança é muito importante ao usarmos a politica de cookies, os hackers mal-intencionados podem aproveitar as informações presentes em cookies para aplicar golpes na web. Um dos ataques mais conhecidos é aquele que fazem uso dos dados de validação de login e senha para navegar em uma determinada loja virtual, assim comprando produtos no nome da vítima.

Um video que explica um pouco mais sobre a LGPD e suas práticas, aprenda, mais sobre esse recurso muito bom para ajudar os desenvolvedores de website.



BIBLIOTECAS

Biblioteca é um arquivo que organiza código pré-definido para uso em aplicações. São uma coleção de recursos usados por programas de computador e podem incluir:

- Dados de configuração
- Documentação
- Procedimentos
- Classes
- Funções
- Templates
- Especificações de tipos
- Entre outros recursos

Mas o que deve estar se perguntando, porque usar bibliotecas? Vou te apresentar algumas vantagens para você entender melhor:

- Funções comuns são pré-definidas para programação modular.
- Rapidez no desenvolvimento da aplicação, pois podem ser usadas por vários programas distintos, mas que necessitem de funcionalidades similares. Assim implementação *reuso de código* em aplicações.
- Com as bibliotecas podemos escrever códigos menores e mais organizados, poupando tempo precioso no desenvolvimento e diminuindo a ocorrência de erros.
- Facilitam a atualização do programa.
- Ao usar uma biblioteca, um programa passa a ser capaz de executar suas funções sem que seja necessário implementa-las no programa em si.

Centenas de bibliotecas estão disponíveis para uso de aplicações. Algumas das mais conhecidas incluem:

- **NumPy** – Utilizada na linguagem Python
- **jQuery** – Utilizada na linguagem Javascript
- **Matplotlib** – Utilizada na linguagem Python
- **D3.js** – Utilizada em Javascript
- **Processing.js** – Utilizada no Javascript
- **STL – C++ Standard Library** – Utilizada no C++

BIBLIOTECAS PADRÃO

Disponibilizadas por Padrão nas implementações de uma linguagem de programação, geralmente descritas na especificação da linguagem à qual pertencem.

Tratada como parte da linguagem em si, mesmo sendo uma entidade separada. Traz funções gerais, como acesso a disco, a manipulação de strings, entrada e saída, etc.

Exemplos de Bibliotecas Padrão:

- Biblioteca Padrão C
- JCL – Java Class Library
- Python Standard Library
- FCL – Framework Class Library (.NET Framework)
- C++ Standard Library

VINCULAÇÃO DE BIBLIOTECAS

As Bibliotecas podem ser vinculadas e acessadas em dois momentos distintos, dependendo da linguagem e bibliotecas empregadas:

- Ligadas ao programa durante o processo de compilação: Bibliotecas Estáticas
- Acessadas somente durante a execução do programa (em runtime): Bibliotecas Dinâmicas ou ainda Bibliotecas Compartilhadas.

Exemplos de arquivos de bibliotecas incluem arquivos com as extensões tipo: .**dll, .so, .h e .lib.**

Quando começamos a falar de bibliotecas, geralmente acaba confundindo com arquivo de cabeçalho, mas existe algumas diferenças. Um arquivo de cabeçalho é um arquivo que contém referencias a bibliotecas em uma linguagem. Traz, por exemplo, listas de nomes e funções (protótipos) e como chamar essas funções, além de tipos de dados e constantes usadas pelas bibliotecas. Sendo assim, trata-se de uma *interface* para uma implementação – a biblioteca em si.

BIBLIOTECA/FRAMEWORK - Sabemos que o Javascript é uma das linguagens mais usadas na web, juntamente com a HTML e a CSS, dão vida e interatividade a boa parte dos sites. Assim como as outras linguagens, ao longo dos anos o Javascript foi sendo aperfeiçoado e ganhando cada vez mais notoriedade no meio web. Dessa forma foi sendo criados vários plugins/frameworks/bibliotecas para se usar o Javascript, visando a facilidade do uso da linguagem em tarefas comuns do nosso cotidiano de desenvolvimento web.

A grande maioria desses frameworks são compartilhados de forma gratuita na internet, e ainda com open Source. Alguns desses frameworks, devido o grande poder que possuem, acabam se destacando e tornando-se fundamental para o desenvolvimento web. Vale lembrar que bibliotecas são um conjunto

de códigos prontos para uso, esses códigos podem conter classes, instruções, funções e rotinas que podem ser implementadas em diversas aplicações web.

Framework é um conjunto de bibliotecas em um formato padrão para o uso, podemos inserir o nosso próprio código, a um código já existente no framework, mas o seu código tem que respeitar as regras do framework, caso eu optar para usar um framework, preciso entender como ele trabalha e fazer meu código pensando nele.

A desvantagem de se trabalhar com framework, é que seu código fica “amarrado” a ele, então se futuramente esse framework se tornar obsoleto, o seu código continua amarrado a ele, tornando inviável seu uso. Por isso é mais funcional o uso de bibliotecas devido a facilidade de utilização e implementação nos códigos, não criando uma dependência de utilização. Vamos conhecer algumas bibliotecas e frameworks mais conhecidos e usados atualmente.

PRINCIPAIS BIBLIOTECAS

jQuery é a principal biblioteca Javascript atualmente, ela foi criada em 2006 pelo engenheiro de software americano John Resig, e tem sido usado em sites ao redor do mundo. Algumas estimativas indicam que o jQuery é usado em mais de 50% dos sites ativos hoje. Uma das vantagens do jQuery é que ele “entende” que nem todos os navegadores podem possuir os mesmos recursos iguais, mas com nomes diferentes permitindo uma boa compatibilidade entre eles. Uma das desvantagens é que podem ocorrer conflitos entre scripts dificultando a depuração em alguns casos.

D3.JS é uma biblioteca Javascript ideal para se trabalhar com dados, essa biblioteca é uma ótima ferramenta para visualizações personalizadas, serve tanto para gráficos simples em formato de barras por exemplo, como para os mais complexos em 3D. Uma das vantagens do D3.js é que tem um rico conjunto de ferramentas para visualização de dados. Uma das desvantagens é a falta de privacidade caso os dados sejam sigilosos.

REACT é uma das principais bibliotecas usadas hoje em dia, sendo uma biblioteca de código aberto apoiada pelo Facebook, o *react* é usado para ajudar no desenvolvimento de aplicativos web em pequena ou larga escala. Ele usa componentes, que ajudam a encapsular código e estado. O uso de componentes facilita a construção de interfaces de usuário mais complexos. Uma das vantagens do react é que ele é muito fácil para aprender e utilizar, por isso o crescimento exponencial no uso dessa

biblioteca. Uma das desvantagens é que ela abrange apenas uma camada da interface do usuário do aplicativo, ou seja, pode exigir em algum momento, outra tecnologia como complemento para a parte de visualização.

GLIMMER.JS é uma biblioteca Javascript com componentes de UI (User Interface – Interface do Usuário) e foco em rapidez e leveza para web, o objetivo dos fundadores, foi desenvolver algo menor e mais leve que o Ember.js (outra biblioteca Javascript). Uma das vantagens dessa biblioteca é que ela faz a diferenciação entre elementos estáticos e dinâmicos. Uma das desvantagens é que ela é destinada a ser usada prioritariamente no Ember, causando a dependência dessa tecnologia;

PRINCIPAIS FRAMEWORKS

BOOTSTRAP é um dos principais frameworks Javascript web do mercado, ele é capaz de oferecer padrões para o desenvolvimento HTML, CSS e Javascript. Com ele é possível criar designs responsivos com uma aparência razoável sem precisar sem um expert em design ou front-end. Uma das vantagens do bootstrap é que ele oferece uma estrutura consistente que suporta grande parte dos navegadores. Uma das desvantagens dele é que todos os sites com esse framework terão uma aparência semelhante se você não fizer uma personalização adequada.

ANGULARJS é um framework mantido pelo Google e apoiado por uma enorme comunidade de desenvolvedores em todo mundo, é um dos frameworks Javascript mais importantes quando falamos de desenvolvimento web. Trata-se de um framework front-end open Source para o desenvolvimento dinâmico de aplicações web. Uma das vantagens é que ele suporta o cache e muitos outros processos, o angular reduz a carga da cpu do servidor. Uma das desvantagens é que dependendo da qualidade dos códigos, aplicações dinâmicas mais complexas tendem a apresentar alguns atrasos e mostrar falhas durante a execução.

EMBER.JS o ember é um framework open source que combina muito dos melhores recursos de seus concorrentes, tornando-a uma ferramenta perfeita para criar aplicações complexas. O objetivo dele é aumentar a produtividade de desenvolvimento de aplicações web, com a mentalidade de que é melhor perder um pouco de tempo no desenvolvimento com o fim de facilitar a manutenção. Uma das vantagens do ember.js é o foco em produtividade. Uma das desvantagens é não ter a renderização do lado do servidor, tornando esse processo mais lento.

VUE.JS é outro framework open source, que permite facilmente misturar e combinar o Vue entre diferentes projetos, ele é muito utilizado para criar aplicações single page (página única) e também para desenvolver vários tipos de interfaces, que possuem necessidades de uma maior interação e uma melhor experiencia para o usuário. Uma das vantagens do Vue são os recursos e a flexibilidade de integração, pois depende somente do Javascript e não requer outras ferramentas para funcionar. Uma das desvantagens é a falta de suporte para projetos de larga escala.

Nesse video o canal código fonte tv faz uma análise dos frameworks mais utilizados no javascript, vale a pena conferir e entender mais dessa poderosa ferramenta.

RESPONDA:

1) O que são cookies?
2) Para que servem os cookies?
3) O que é biblioteca Javascript
4) O que são frameworks?
5) Por que o uso de bibliotecas é importante no desenvolvimento de aplicativos web?
6) Qual a diferença entre bibliotecas e frameworks?
7) Qual a biblioteca javascript mais usada atualmente?
8) Qual biblioteca Javascript é recomendada para uso de dados gráficos?
9) Para qual propósito inicialmente os cookies foi criado?
10) Qual o framework mais utilizado no mercado hoje em dia?

DESAFIO

Com suas palavras explique as vantagens e desvantagens de usar framework.