



DEFINIÇÃO

O ambiente Web e seu modelo Cliente X Servidor, interfaces com design responsivo (abordagem mobile first) e adaptativo, tecnologias do lado cliente (HTML, CSS, páginas estáticas com Javascript) e do lado servidor (páginas dinâmicas com PHP, acesso a banco de dados).

PROPÓSITO

Compreender a composição do Ambiente Web e seus componentes dos lados cliente e servidor, assim como as tecnologias inerentes a cada componente, é essencial para a formação do profissional de desenvolvimento de sistemas na Web.

OBJETIVOS

MÓDULO 1

Reconhecer o Ambiente Web

MÓDULO 2

Descrever o conceito de interface

MÓDULO 3

Reconhecer as tecnologias do lado cliente

MÓDULO 4

Reconhecer as tecnologias do lado servidor

INTRODUÇÃO

O chamado modelo Cliente x Servidor, com origem na Computação, é uma estrutura de aplicação distribuída, em que temos tarefas partilhadas e executadas entre as duas camadas: de um lado, a origem das requisições e solicitações de recursos ou serviços – o lado cliente – e, do outro, processos sendo executados a fim de prover tais recursos ou serviços – o lado servidor. Atualmente, tal modelo é amplamente utilizado, sendo base do Ambiente Web e de suas aplicações, como veremos ao longo deste tema.



MÓDULO 1

🕒 Reconhecer o Ambiente Web

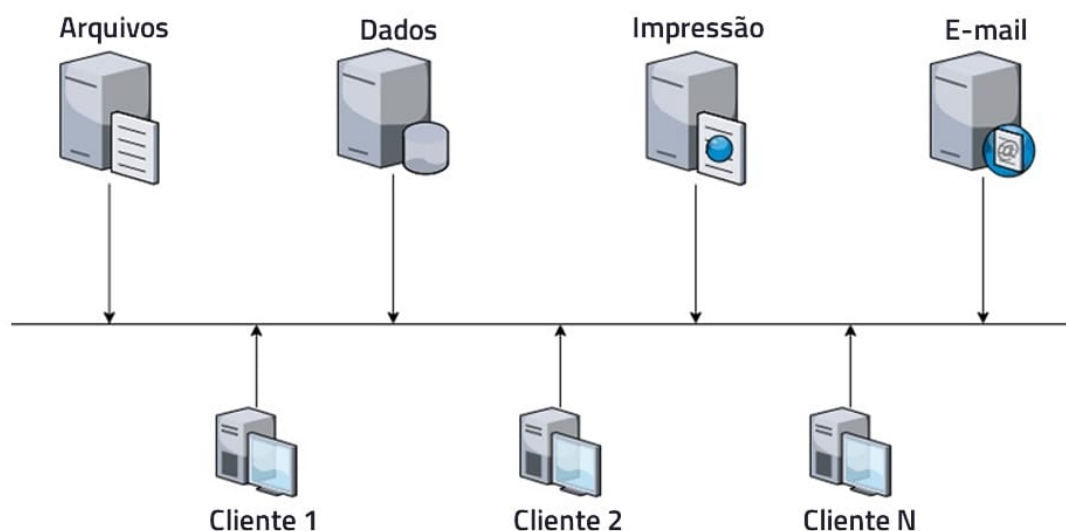
AMBIENTE CLIENTE X SERVIDOR

UM POUCO DE HISTÓRIA

O modelo Cliente x Servidor foi criado pela Xerox PARC nos anos 1970, tendo como principal premissa a separação entre dados e recursos de processamento, ao contrário do modelo predominante à época – conhecido como modelo centralizado, em que tanto o armazenamento dos dados quanto o seu processamento ficavam a cargo dos computadores de grande porte: Mainframe.

COMO É COMPOSTO O MODELO CLIENTE X SERVIDOR

O ponto de partida para entendermos a arquitetura do modelo Cliente x Servidor é tomarmos como exemplo a rede interna de computadores de uma empresa, em que temos máquinas exercendo a função de servidores – provendo serviços como armazenamento de arquivos ou dados, impressão, e-mail etc. – e máquinas exercendo a função de clientes – consumindo os recursos fornecidos pelos servidores. Essa arquitetura pode ser vista na Figura 1.



📷 Figura 1 – Arquitetura Cliente x Servidor em uma Rede Interna. Fonte: Paixão, 2020.

APLICAÇÕES NO MODELO CLIENTE X SERVIDOR

O modelo Cliente x Servidor tornou possível o desenvolvimento de aplicações que fizessem uso de sua arquitetura distribuída. Tais aplicações foram desenvolvidas tendo como base o conceito de desenvolvimento em camadas. Logo, surgiram os modelos de 2, 3 e 4 (ou N) camadas.

MODELO DE 2 CAMADAS

Neste modelo, temos as camadas Cliente e Servidor, sendo função da primeira tratar a lógica do negócio e fazer a interface com o usuário, enquanto a segunda é responsável por tratar os dados – normalmente fazendo uso de Sistemas Gerenciadores de Bancos de Dados (SGDB).

São exemplos deste modelo as aplicações desktop instaladas em cada computador cliente que se comunicam com um servidor na mesma rede.



📷 Modelo de 2 camadas.

MODELO DE 3 CAMADAS

Este modelo foi criado para resolver alguns problemas do modelo anterior, entre eles a necessidade de reinstalação/atualização da aplicação nos clientes a cada mudança de regra ou lógica. Logo, este modelo incluiu uma camada a mais, chamada de aplicação. Com isso, as responsabilidades de cada camada ficaram assim divididas:

CAMADA DE APRESENTAÇÃO

Ainda representada pela aplicação instalada na máquina cliente. Era responsável pela interface com o usuário e passou a acessar o servidor de Aplicação, perdendo o acesso direto ao servidor de dados.

CAMADA DE APLICAÇÃO

Representada por um servidor responsável pela lógica e regras de negócio, assim como pelo controle de acesso ao servidor de dados.

CAMADA DE DADOS

Representada por um servidor responsável pelo armazenamento dos dados.

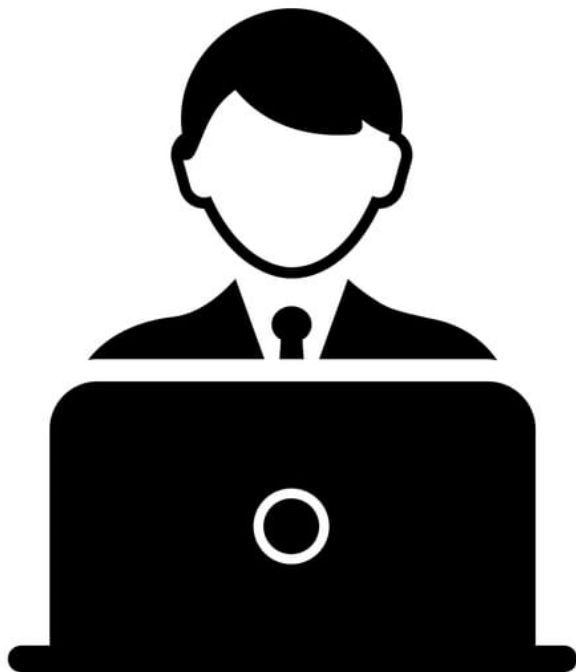
Veja um exemplo do modelo de 3 camadas:



📷 Modelo de 3 camadas.

MODELO DE 4 CAMADAS

O grande avanço obtido neste modelo foi tirar da máquina cliente a responsabilidade pela interface com o usuário, passando a centralizá-la em um único ponto, normalmente em um servidor Web. Com isso, no lugar de aplicações instaladas em cada máquina cliente passamos a ter os clientes acessando aplicações hospedadas em servidores Web a partir de navegadores. Neste modelo, a divisão de responsabilidades ficou dessa forma:



Por AlexeyYakovenko | Fonte: Shutterstock

CLIENTE

Passou a precisar apenas de um navegador para ter acesso à aplicação.



SERVIDOR

Agora composto por 3 servidores – o de Aplicações, o de Dados e o Web, sendo este último o responsável pela apresentação/interface com o usuário cliente.

Veja um exemplo do modelo de 4 camadas:



📷 Modelo de 4 camadas.

O AMBIENTE WEB

Como vimos, inicialmente as aplicações ficavam hospedadas dentro de uma rede interna, onde estavam tanto os clientes quanto os servidores. Posteriormente, elas migraram para a internet, surgindo então o Ambiente Web, cuja base é justamente prover aos clientes, usuários, o acesso a várias aplicações a partir de diversos dispositivos, como navegadores em desktops e smartphones ou a partir de aplicações mobile.

NO CONTEXTO DESTES TEMA É IMPORTANTE DESTACAR UM ASPECTO QUANDO TRATAMOS DO AMBIENTE WEB: A COMUNICAÇÃO.

Até aqui, vimos que este Ambiente é composto pelo:

CLIENTE

Utiliza um navegador ou aplicativo e consome serviços hospedados em um servidor Web.

SERVIDOR WEB

Cuja estrutura pode comportar tanto as camadas citadas anteriormente numa única máquina quanto em diversas máquinas, sendo essa distribuição indistinguível para o cliente.

Quando falamos de comunicação, estamos falando, mais especificamente, de como trafegam os dados entre a requisição enviada por um cliente e a resposta provida por um servidor.

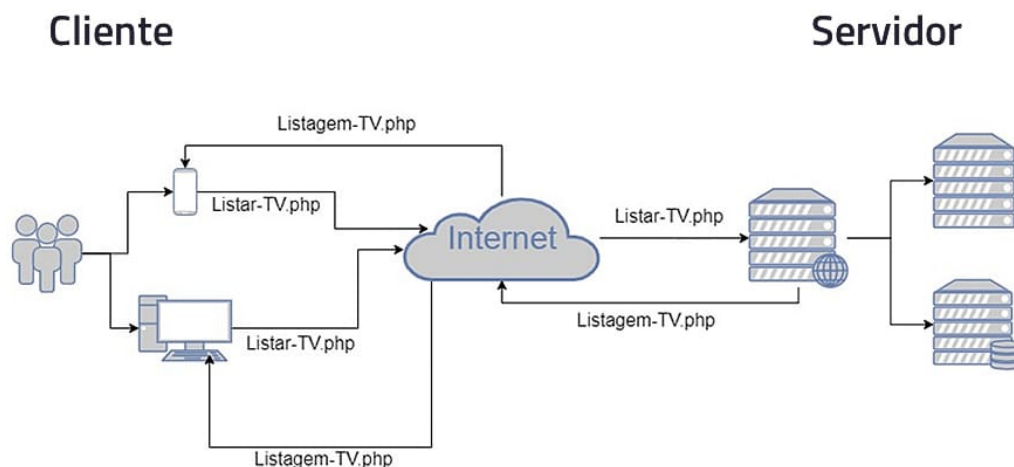


Por Dmitriy Rybin | Fonte: Shutterstock

COMUNICAÇÃO NO AMBIENTE WEB

A comunicação, neste ambiente, é feita sobre a internet, com o uso dos seus protocolos de comunicação, sendo o principal protocolo o HTTP (HyperText Transfer Protocol) – que é um protocolo para transferência de hipertexto. Na Figura 2, podemos ver um exemplo de comunicação no Ambiente Web.

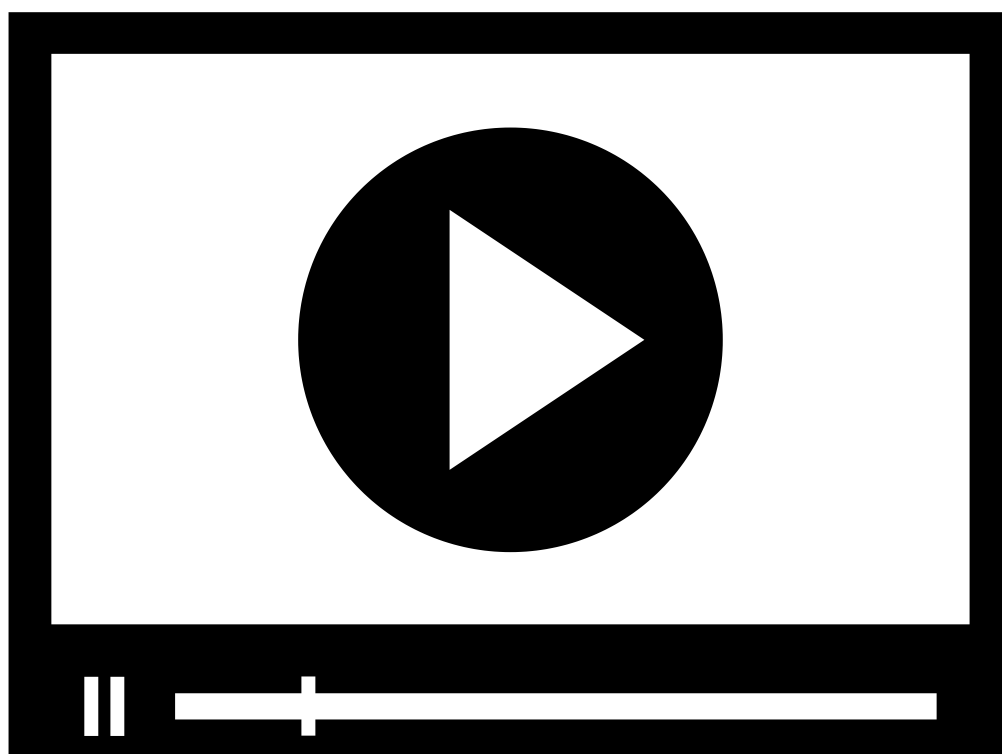
★ EXEMPLO



📷 Figura 2 – Comunicação no Ambiente Web. Fonte: Paixão, 2020.

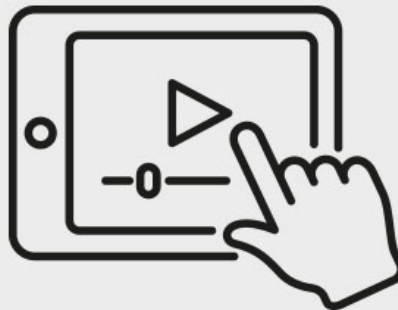
No exemplo mostrado na Figura 2, temos de um lado o cliente que, com um desktop ou smartphone, faz a requisição de um serviço – representada pelo arquivo `Listar-TV.php` – através da internet a um servidor. O servidor Web, após processar a requisição, retorna a informação solicitada, representada pelo arquivo `Listagem-TV.php`.

Com base nesse exemplo, podemos entender como funcionam as aplicações disponíveis no Ambiente Web, como websites de notícias, comércio eletrônico, e-mail, redes sociais etc. Em cada um desses casos, há de um lado uma requisição sendo feita pelo cliente e, do outro, o servidor processando a requisição e respondendo ao cliente com o que foi solicitado.



Para conhecer o **fluxo do usuário requisitante**, assista ao vídeo a seguir.

Para assistir a um vídeo sobre o assunto, acesse a versão online deste conteúdo.



SOLICITAÇÃO E RESPOSTA

O processo de comunicação no Ambiente Web é conhecido como Solicitação (Request) e Resposta (Response). Normalmente, a solicitação é iniciada pelo cliente, mas é possível que também o servidor a inicie, como em serviços PUSH – serviços que disparam notificações/mensagens para os clientes que fizeram opção por recebê-las.



📷 Processo de Solicitação (Request) e Resposta (Response).

CLIENT SIDE X SERVER SIDE

Essas duas expressões são muito comuns quando falamos de aplicações rodando no Ambiente Web. Ambas se referem a tecnologias, códigos, disponibilizados no lado cliente

(neste caso, o dispositivo utilizado por um usuário para fazer uma requisição) e no lado servidor. Nos próximos tópicos trataremos de ambas mais a fundo.



Por Blackboard | Fonte: Shutterstock

VERIFICANDO O APRENDIZADO

1. EM RELAÇÃO À RESPONSABILIDADE DE REALIZAR AS REQUISIÇÕES NO MODELO CLIENTE X SERVIDOR, ASSINALE A ALTERNATIVA CORRETA:

A) Uma das principais vantagens deste modelo é permitir a separação de responsabilidades. Com isso, caberá sempre e unicamente ao cliente realizar as requisições de serviços e/ou recursos, tendo o lado servidor um caráter sempre passivo.

B) Embora inicialmente limitado a redes internas, com o surgimento da internet o modelo Cliente x Servidor evoluiu, tornando-se um modelo híbrido e bastante flexível, separado em N camadas, onde tanto o cliente quanto o servidor podem exercer as mesmas funções, ou seja, ambos podem requisitar e responder a solicitações.

C) Para diminuir o custo de processamento no lado servidor, um cliente poderá solicitar a outros clientes recursos ou serviços já utilizados por eles. Isso é possível graças ao suporte fornecido por este modelo à comunicação Cliente x Cliente.

D) Neste modelo, o lado cliente é, normalmente, o responsável por iniciar a comunicação através da realização de requisições ao lado servidor. Entretanto, o lado servidor também é capaz de iniciar a comunicação, disparando notificações ou enviando mensagens para o lado cliente, por exemplo.

2. NÓS VIMOS QUE O MODELO CLIENTE X SERVIDOR É A BASE DO AMBIENTE WEB. ASSINALE A OPÇÃO CORRETA QUE DESCREVE O AMBIENTE WEB:

A) O Ambiente Web é composto por diversos clientes e diversos servidores. Neste cenário, os clientes utilizam a internet e fazem requisições a diferentes servidores, localizados em diferentes partes do mundo. Os servidores então processam a requisição e devolvem a informação requisitada ou executam o serviço solicitado pelo cliente.

B) No Ambiente Web, diferente do que acontecia nos primeiros Modelos de Camadas, há um modelo centralizado. Logo, todas as requisições são feitas a um único servidor, que as distribui para outros servidores e depois envia as respostas para os clientes.

C) O avanço da tecnologia e o suporte oferecido pela internet permitiram uma importante mudança no Ambiente Web em relação aos modelos tradicionais de camadas. Com isso, neste Ambiente, o lado cliente tem as principais responsabilidades, incluindo manter no navegador ou em aplicativos mobile toda a lógica do negócio, facilitando assim o trabalho de processamento pelo lado servidor e agilizando a comunicação.

D) O Ambiente Web é caracterizado, sobretudo, pela transparência, diferentemente do que era visto inicialmente no modelo Cliente x Servidor. Com isso, um cliente sempre terá controle total sobre o processo de comunicação por trás da requisição. Ele terá ciência, por exemplo, de onde se encontra o servidor ou servidores encarregados de receber e processar a sua requisição. Isso permite, por exemplo, que ele cancele a requisição a qualquer momento, caso o servidor encarregado de processá-la fique muito distante de onde ele se encontra.

GABARITO

1. Em relação à responsabilidade de realizar as requisições no modelo Cliente x Servidor, assinale a alternativa correta:

A alternativa **"D "** está correta.

Conforme visto na seção **Solicitação e Resposta**, normalmente é o cliente que inicia a comunicação no modelo Cliente x Servidor. Entretanto, o lado servidor também é capaz de realizar esta tarefa.

2. Nós vimos que o modelo Cliente x Servidor é a base do Ambiente Web. Assinale a opção correta que descreve o Ambiente Web:

A alternativa **"A "** está correta.

Conforme visto na seção **O Ambiente Web**, o ambiente web tem como base o modelo Cliente x Servidor e a evolução de seu Modelo de Camadas. Faz uso, portanto, de um modelo de N camadas, onde a lógica da aplicação e os dados são distribuídos em um ou mais servidores e a interface para acesso a estes servidores fica a cargo do cliente.

MÓDULO 2

⦿ Compreender o conceito de interface

A INTERFACE DO LADO CLIENTE

O conceito de interface está ligado à área de Interação Humano-Computador (IHC), que pode ser resumida como o estudo da interação entre pessoas e computadores. Nesse contexto, a interface, muitas vezes chamada de interface do utilizador, é quem provê a interação entre o ser humano e o computador. No início da utilização dos computadores, tal interação era realizada através de linha de comando e, posteriormente, também através de interfaces gráficas (Graphical User Interface (GUI).). Segundo Moraes (2014), no início, a interação foi, de certa forma, primária, deixando um pouco de lado o ser humano, por não existir um estudo aprofundado desses aspectos.



Por spainter_vfx | Fonte: Shutterstock

Dessa forma, o foco do estudo da interface envolvia principalmente o hardware e o software, e o ser humano simplesmente tinha que se adaptar ao sistema criado.



Por Casezy idea | Fonte: Shutterstock


Posteriormente, com o avanço da tecnologia e do acesso a computadores, e mais recentemente a outros dispositivos, sobretudo os smartphones, a necessidade de melhorar a interação tem crescido continuamente.

Como Silva (2014) explica, a evolução tecnológica levou a uma crescente utilização de dispositivos móveis que possuem os mais **variados tamanhos de tela e funcionalidades**. O site StatCounter Global Stats mantém ativa uma série de dados e estatísticas sobre dispositivos, tamanhos de tela, além de outras informações relacionadas. Sobre o tamanho de telas, e considerando o período de abril de 2019 a abril de 2020, temos os seguintes dados:

VARIADOS TAMANHOS DE TELA E FUNCIONALIDADES

Sobre essa variedade nas características dos dispositivos utilizados como interface para o acesso a aplicações no Ambiente Web, é necessário garantir a usabilidade, ou seja, que sejam desenvolvidos sistemas fáceis de usar e de aprender, além de flexíveis. Em complemento a esse conceito, e partindo do ponto de vista da usabilidade, a mesma deve estar alinhada ao conceito de Design Responsivo, o qual deverá permitir que as páginas Web, e conseqüentemente as aplicações Web, respondam a qualquer dispositivo sem perda de informações por parte do usuário.

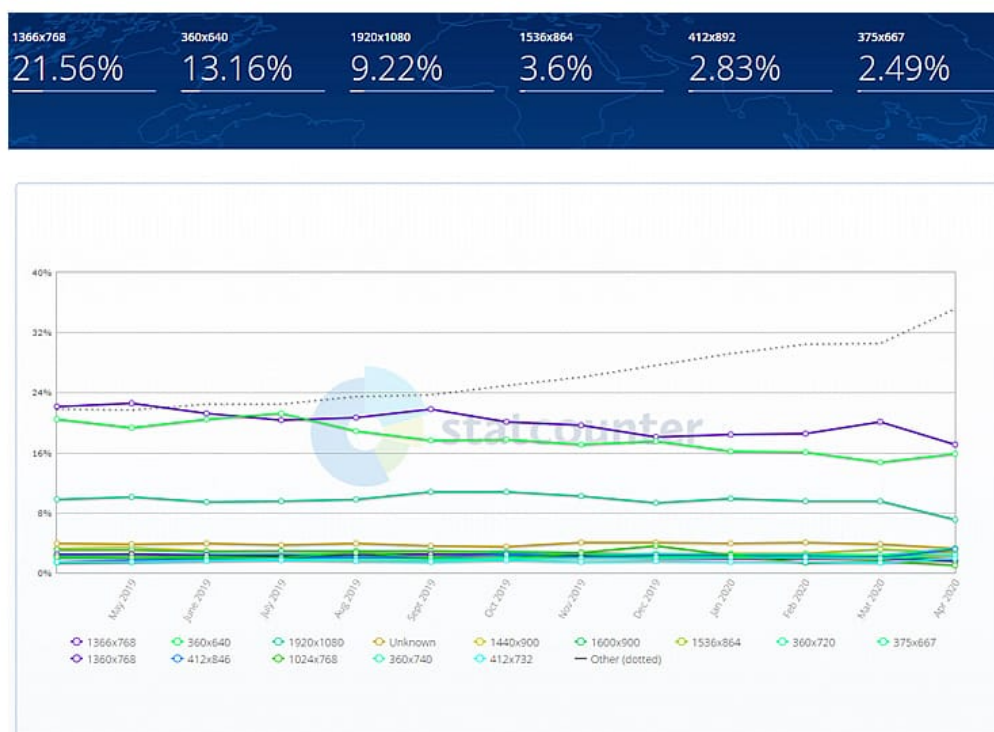
Tamanho da Tela em Pixels (Largura x Altura)	Percentual de Utilização
360 x 640	10,11%
1366 x 768	9,69%
1920 x 1080	8,4%
375 x 667	4,24%
414 x 896	3,62%

 Tabela 1 – Estatísticas mundiais sobre resolução de telas de dispositivos. Fonte: Elaboração do autor, com base em dados de Global Stats, 2020.




Atenção! Para visualização completa da tabela utilize a rolagem horizontal

Quando consideramos essas mesmas estatísticas, mas levando em conta especificamente os dados de navegação do Brasil, temos um cenário diferente, conforme pode ser visto na Figura 3.



Fonte: Global Stats, 2020.

 Figura 3 – Estatísticas sobre resoluções de telas de dispositivos - Brasil.

DESIGN RESPONSIVO

Segundo Knight (2011), o **design responsivo** é a abordagem que sugere que o design e o desenvolvimento devam responder ao comportamento e ao ambiente do usuário com base no tamanho da tela, plataforma e orientação do dispositivo por ele utilizado.

ESSA DEFINIÇÃO, NA PRÁTICA, IMPLICA QUE A PÁGINA WEB/APLICAÇÃO ACESSADA DEVE SER CAPAZ DE,

AUTOMATICAMENTE, RESPONDER ÀS PREFERÊNCIAS DO USUÁRIO E, COM ISSO, EVITAR QUE SEJA NECESSÁRIO CONSTRUIR DIFERENTES VERSÕES DE UMA MESMA PÁGINA/APLICAÇÃO PARA DIFERENTES TIPOS E TAMANHOS DE DISPOSITIVOS.

A ORIGEM DO DESIGN RESPONSIVO

O conceito de design responsivo teve sua origem no Projeto Arquitetônico Responsivo. Tal projeto prega que uma sala ou um espaço deve se ajustar automaticamente ao número e fluxo de pessoas dentro dele. Para tanto, é utilizada uma combinação de robótica e tecnologia, como: sensores de movimento; sistemas de controle climático com ajuste de temperatura e iluminação; juntamente com materiais – estruturas que dobram, flexionam e expandem.

Da mesma forma que no Projeto Arquitetônico Responsivo, arquitetos não refazem uma sala ou um espaço de acordo com o número, fluxo e as características de seus ocupantes, no Ambiente Web não devemos ter que precisar construir uma versão de uma mesma página de acordo com as características dos seus visitantes. Isso traria ainda outros custos, como identificar uma enorme combinação de tamanhos de tela e tecnologia, entre outros fatores, para criar uma mesma quantidade de páginas correspondentes.



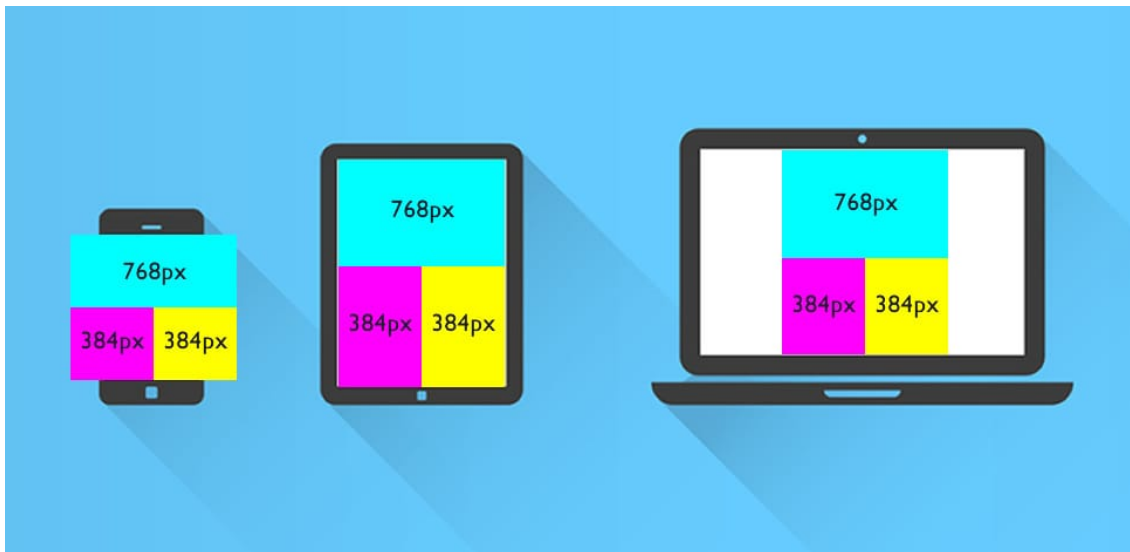
Por REDPIXEL.PL | Fonte: Shutterstock

DESIGN RESPONSIVO NA PRÁTICA

Na prática, ao aplicarmos o conceito de design responsivo, fazemos uso de uma combinação de técnicas, como layouts fluidos, media query e scripts. A seguir veremos cada uma dessas técnicas em detalhes.

LAYOUTS FLUIDOS

Para entender o conceito de layout fluido, é necessário entender primeiro o que seria o seu oposto, ou seja, o layout fixo.

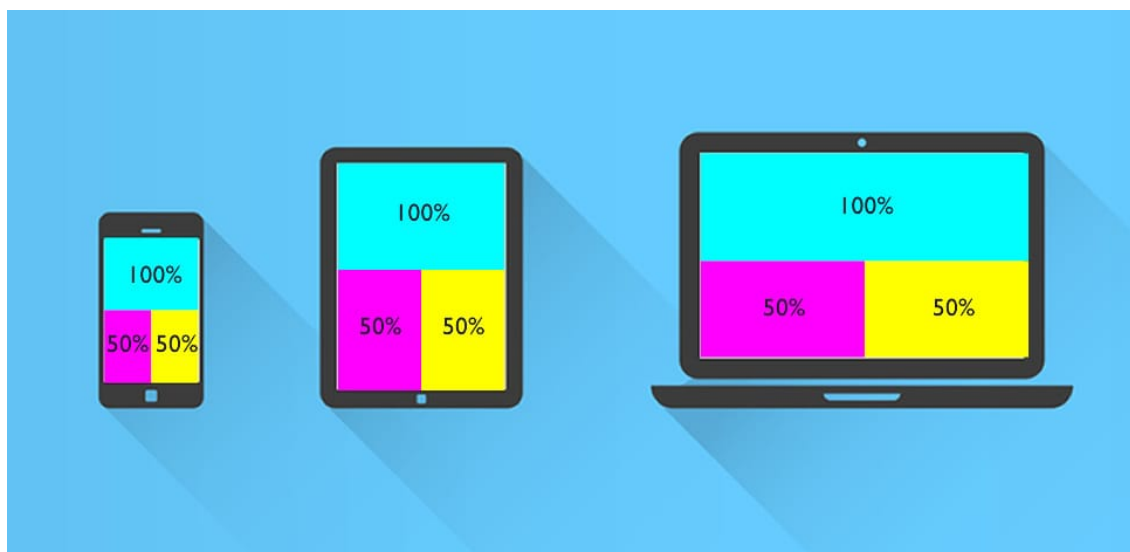


autor/shutterstock

LAYOUT FIXO

As dimensões (largura e altura) dos elementos de uma página Web são definidos com a utilização de unidades de medidas fixas, como os '**pixels**'. Com isso, tais elementos não se adaptam às alterações no tamanho do campo de visão dos dispositivos que os visualiza.





LAYOUT FLUIDO

Já os layouts fluidos fazem uso de unidades flexíveis – no lugar de definir as dimensões com o uso de quantidades fixas são utilizados valores flexíveis. Isso permite, por exemplo, que no lugar de definir que o cabeçalho de uma página tenha 1366 pixels de largura, possamos definir que ele ocupe 90% do tamanho da tela do dispositivo que o visualiza. Daí o conceito de fluido, ou seja, de adaptabilidade ao campo de visão conforme dimensões do dispositivo que visualiza a página.

PIXEL

Um pixel é o menor ponto que forma uma imagem digital, sendo que um conjunto de pixels com várias cores formam a imagem inteira.

Além dos valores percentuais, há outras unidades de medidas flexíveis, como, por exemplo:

EM

Unidade de medida tipográfica, estando relacionada à letra “M”. O tamanho base dessa unidade equivale à largura da letra “M” em maiúscula.

REM

Enquanto o EM está relacionado ao tamanho do elemento de contexto (ou seja, definimos o valor EM de um elemento tomando como base o seu elemento pai), no REM definimos que o

elemento de contexto, o elemento pai, será sempre a tag HTML <body>. Daí a letra “R” nessa unidade, que faz referência à raiz (root).

✚ SAIBA MAIS

Além das unidades, fixas e flexíveis já mencionadas, há ainda outras disponíveis. A listagem completa pode ser acessada através do site do W3C – CSS Units.

MEDIA QUERY

A função de apresentação, de estruturar o layout de uma página, no Ambiente Web, cabe às Folhas de Estilo (CSS). Trataremos mais a fundo do CSS no próximo tópico. Por ora, para definir o que é Media Query, falaremos um pouco também sobre CSS.

Com base na afirmação de que cabe ao CSS estruturar o layout de uma página Web, temos normalmente associada a uma página Web uma ou mais Folhas de Estilo – que são códigos que definem aspectos de toda a página, como as dimensões dos elementos, cores de fundo, as cores e os tipos de fonte etc.




Por NicoElNino | Fonte: Shutterstock

NESSE CONTEXTO, MEDIA QUERY É A UTILIZAÇÃO DE MEDIA TYPES (TIPOS DE MÍDIA) A PARTIR DE UMA OU MAIS EXPRESSÕES PARA DEFINIR FORMATAÇÕES PARA

DISPOSITIVOS DIVERSOS. COM O SEU USO PODEMOS, POR EXEMPLO, DEFINIR QUE DETERMINADO ESTILO DE UM OU MAIS ELEMENTOS SEJA APLICADO APENAS A DISPOSITIVOS CUJA LARGURA MÁXIMA DE TELA SEJA IGUAL OU MENOR QUE 600PX.

A **Figura 4** mostra um fragmento de código onde uma Media Query é utilizada para impedir que um menu lateral (o elemento HTML cuja classe equivale a “menu_lateral”) seja exibido caso a largura da tela do dispositivo seja menor que 360 pixels.

```
1 <style type="text/css">
2 @media (max-width: 360px)
3 {
4   .menu_lateral
5   {
6     display: none;
7   }
8 }
9 </style>
```

 **Figura 4** – Exemplo de Declaração de Media Query. Fonte: Paixão, 2020.

O resultado das expressões utilizadas na Media Query pode ser verdadeiro ou falso. No caso acima, será verdadeiro sempre que a largura da tela do dispositivo que visualiza a página seja inferior a 360 pixels. Do contrário, será falso. Ou seja, para todos os dispositivos cuja largura de tela seja superior a 360px, o código CSS em questão será ignorado.

ATENÇÃO

Outras expressões podem ser utilizadas, além da demonstrada acima, como a definição do tipo de mídia (Media Type – ou seja, um estilo que se aplica apenas a um ou mais tipos de documento, como a versão para impressão de uma página web, por exemplo) e a combinação entre escalas de valores.

SCRIPTS

Quando falamos em Scripts no lado cliente, no Ambiente Web, estamos falando de linguagens de programação que rodam no navegador e cujo exemplo mais comum é o Javascript.

Esta linguagem adiciona interação a uma página Web, permitindo, por exemplo, a atualização dinâmica de conteúdos, o controle de multimídia, a animação de imagens e muito mais. No contexto do design responsivo, sua faceta mais importante é a de atualização dinâmica de conteúdo – e não só do conteúdo, mas também da apresentação do mesmo.



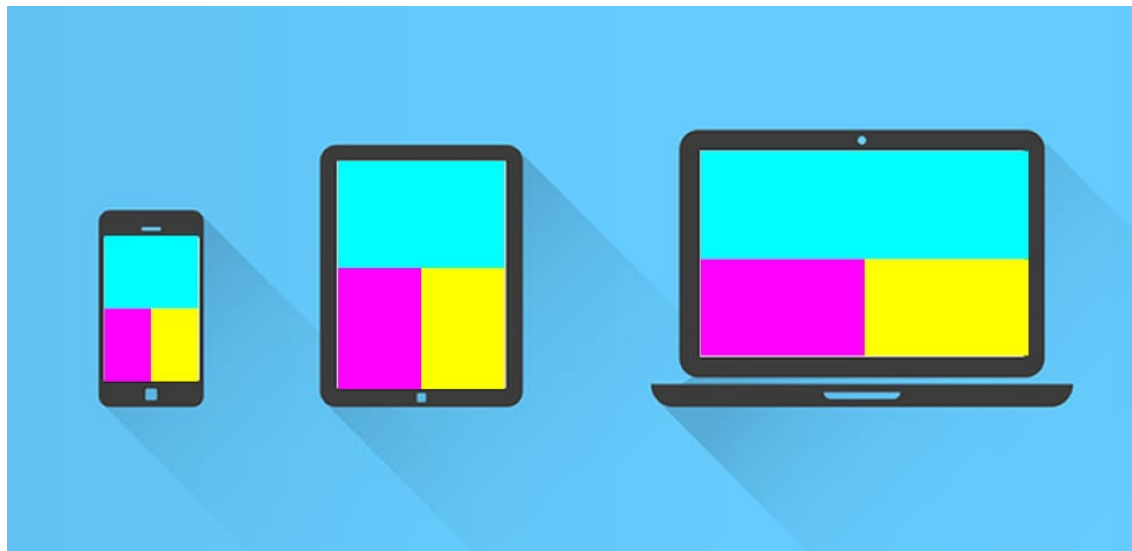
Por fatmawati achmad zaenuri | Fonte: Shutterstock

DESIGN RESPONSIVO X DESIGN ADAPTATIVO

O conceito de design adaptativo, muitas vezes, confunde-se com o de design responsivo. Enquanto o segundo, como já visto anteriormente, consiste na utilização de uma combinação de técnicas para ajustar um site automaticamente em função do tamanho da tela dos dispositivos utilizados pelos usuários, no design adaptativo são usados layouts estáticos baseados em pontos de quebra (ou de interrupção), onde, após o tamanho de tela ser detectado, é carregado um layout apropriado para ele. Em linhas gerais, no design adaptativo, são criados layouts com base em seis tamanhos de tela mais comuns. A aplicação desses dois conceitos na prática acontece da seguinte forma:

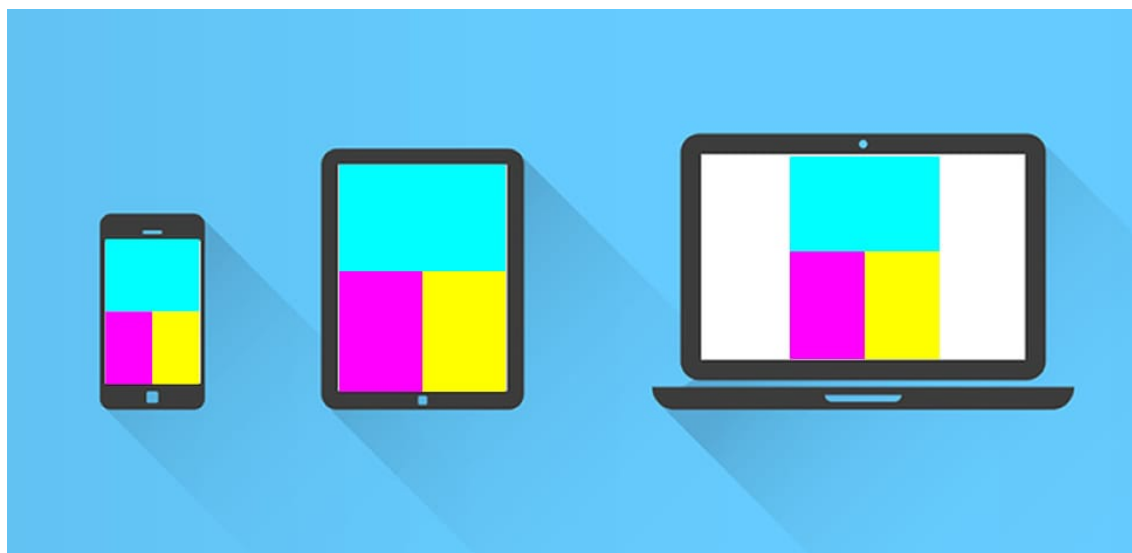
DESIGN RESPONSIVO

Medias Queries são utilizadas, em conjunto com scripts, para criar um layout fluido que se adapte – através, sobretudo, da adequação das dimensões de seus elementos – ao tamanho da tela do dispositivo utilizado pelo visitante.



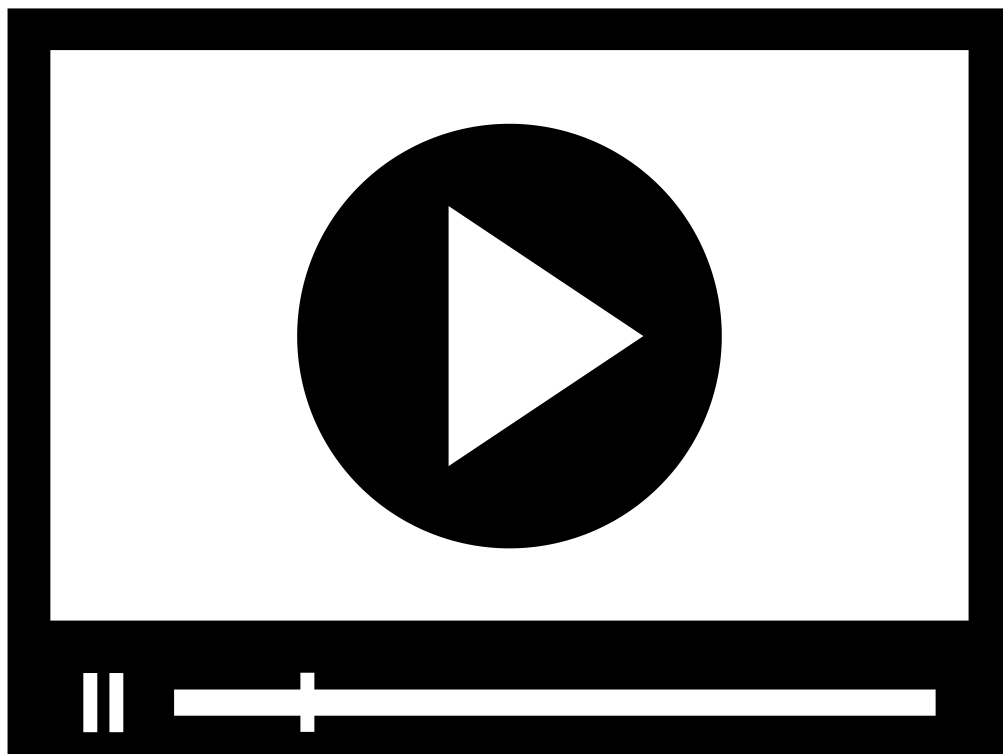
DESIGN ADAPTATIVO

Um site é planejado e construído com a definição de seis layouts predefinidos, onde são previstos pontos de quebra para que a página se adapte às seis diferentes dimensões utilizadas.



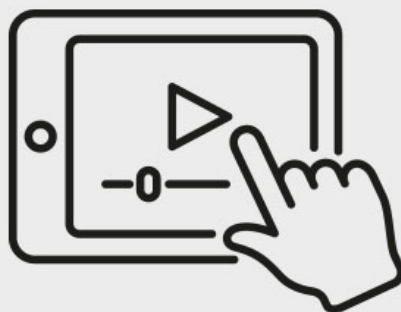
RESUMINDO

Poderíamos ainda dizer que o Design Responsivo é mais complexo, porém mais flexível. Já o Adaptativo, mais trabalhoso, embora menos flexível.



Assista ao vídeo e entenda mais sobre a diferença entre design responsivo e design adaptativo.

Para assistir a um vídeo sobre o assunto, acesse a versão online deste conteúdo.



Como dito, no design responsivo é preciso criar uma série de combinações de Media Query para que o layout se adapte aos mais variados tamanhos de tela. Já no adaptativo, imaginemos a situação onde foram definidos os seguintes layouts e quebras: 360px, 720px, 900px, 1080px, 1440px e 1800px. Caso a largura da tela do dispositivo seja superior a 360px e inferior a 720px – por exemplo, 700px –, será carregado o layout de 360px, que equivale, praticamente, à sua metade. É possível imaginar que, neste caso, o resultado não seja visualmente muito agradável ou otimizado.

MOBILE FIRST



Por DisobeyArt | Fonte: Shutterstock

Uma das abordagens de design responsivo mais utilizadas atualmente é a *mobile first*. Tal abordagem está centrada no crescente uso de dispositivos móveis na navegação no Ambiente Web e defende que em primeiro lugar seja pensado o design para telas menores e, posteriormente, para telas maiores. Trata-se de um enfoque progressivo (*progressive enhancement*), no qual parte-se dos recursos e tamanhos de tela disponíveis nos dispositivos menores, progredindo com a adição de recursos e conteúdo tendo em vista as telas e os dispositivos maiores.

A partir da definição de *mobile first* podemos identificar o seu contraponto com o desenvolvimento Web tradicional, em que temos o conceito de degradação graciosa (*graceful degradation*):

AS PÁGINAS WEB SÃO PROJETADAS TENDO EM VISTA DISPOSITIVOS DESKTOP E TELAS MAIORES E, POSTERIORMENTE, ADAPTADAS PARA DISPOSITIVOS MÓVEIS E TELAS MENORES.

A aplicação prática do *mobile first* consiste em planejar o desenvolvimento de um site priorizando os recursos e as características presentes nos dispositivos móveis, como o tamanho de tela, a largura de banda disponível e até mesmo recursos específicos, como os de localização, por exemplo.

VERIFICANDO O APRENDIZADO

1. ASSINALE A ALTERNATIVA QUE NÃO CORRESPONDE AO CONCEITO DE INTERFACE:

- A) A interface tem como objetivo proporcionar uma comunicação mais natural entre usuário e sistema computacional.
- B) Interface é o meio pelo qual interagimos com um software, com uma aplicação, permitindo o acesso às opções e informações disponíveis.
- C) É o nome dado à parte de um sistema com a qual o usuário mantém contato ao usá-lo.
- D) A interface é a disciplina responsável pelo layout no desenvolvimento de software. Um dos seus princípios é garantir a criação de telas mais bonitas, que chamem a atenção de quem utiliza um software ou aplicativo.

2. EM RELAÇÃO AO DESIGN RESPONSIVO, ASSINALE QUAL OPÇÃO CORRESPONDE À MELHOR AÇÃO A SER TOMADA PARA SUA APLICAÇÃO:

- A) Estudar os dados provenientes das visitas ou, na ausência destes, os relacionados às pesquisas de comportamento de acesso a websites para planejar a construção ou remodelação de um site a fim de garantir que ele se adapte às características dos dispositivos que o acessa.
- B) Construir um site a partir de seis ou mais layouts fixos predefinidos.
- C) Escolher uma das três técnicas possíveis, preferencialmente o Javascript, uma vez que sua implementação é mais simples, além de ser mais completo que as demais técnicas.
- D) Aplicar simultaneamente as técnicas de Design Responsivo e Adaptativo.

GABARITO

1. Assinale a alternativa que não corresponde ao conceito de interface:

A alternativa "D " está correta.

O objetivo da interface vai além de aspectos como definir o que é 'mais ou menos bonito'. Seu cerne está em garantir que, sobretudo, haja uma comunicação mais natural e intuitiva entre usuário e sistema computacional.

2. Em relação ao design responsivo, assinale qual opção corresponde à melhor ação a ser tomada para sua aplicação:

A alternativa **"A "** está correta.

Como visto anteriormente, para aplicar o Design Responsivo devemos fazer uso de uma combinação de técnicas a fim de garantir que uma página corresponda às preferências e características dos seus usuários com base no tamanho da tela, plataforma e orientação dos dispositivos por eles utilizados.

MÓDULO 3

🕒 Reconhecer as tecnologias do lado cliente

AS TECNOLOGIAS DO LADO CLIENTE: HTML5, CSS E JAVASCRIPT

HTML

A HTML (Hypertext Markup Language – Linguagem de Marcação de Hipertexto.) é considerada a tecnologia fundamental da Web, pois sua função é definir a estrutura de uma página Web. Essa linguagem de marcação, criada por Tim Berners-Lee na década de 1990, inicialmente objetivava permitir a disseminação de pesquisas entre Lee e seus colegas pesquisadores, mas foi rapidamente difundida até formar a rede que, posteriormente, veio a se tornar a World Wide Web como a conhecemos atualmente.

```

1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4 <title>My perfect website</title>
5 <meta charset="utf-8" />
6
7 <link rel="preconnect" href="//s3.mysite.com" />
8 <link rel="preconnect" href="//www.mysite.com" />
9
10 <meta name="viewport" content="width=640, initial-scale=1">
11
12 <script>
13   var mytag = mytag || {};
14   mytag.cmd = mytag.cmd || [];
15   (function() {
16     var gads = document.createElement('script');
17     gads.async = true;
18     gads.type = 'text/script';
19     var useSSL = 'https:' == document.location.protocol;
20     gads.src = (useSSL ? 'https:' : 'http:') + '//www.mytagservices.com/tag/js/gpt.js';
21     var node = document.getElementsByTagName('script')[0];
22     node.parentNode.insertBefore(gads, node);
23   })();
24   mytag.cmd.push(function() {
25     var homepageSquareSizeMapping = mytag.sizeMapping();
26     addSize([945, 250], [200, 200]);
27     addSize([0, 0], [300, 250]);
28     build();
29     mytag.defineSlot('/1023782/homepageDynamicSquare', [[300, 250], [200, 200]], 'reserved-div-1');

```

Por Lars Poyansky | Fonte: Shutterstock

Em linhas gerais, a HTML é uma linguagem de marcação simples, composta por elementos, chamados **tags**, que são relacionados a textos e outros conteúdos a fim de lhes dar significado. Por exemplo: Podemos marcar um texto como sendo um parágrafo, uma lista ou uma tabela. É possível, ainda, inserir vídeos e imagens. Além disso, também utilizamos essa marcação para definir a estrutura de um documento de forma lógica: menu de navegação, cabeçalho, rodapé etc. As tags podem ser agrupadas em tipos:

AS TAGS HTML

Como já mencionado, na HTML a tag é usada para definir um elemento. O exemplo a seguir mostra a tag utilizada para a marcação de um parágrafo de texto:

<p>Este texto está contigo em uma tag de parágrafo</p>

A anatomia de uma tag é composta pela tag de abertura, pela tag de fechamento e pelo conteúdo – no exemplo anterior, o texto. Com essa combinação temos, ainda, a definição de elemento na HTML.

ESTRUTURAIS

Juntamente com o elemento de definição do DocType, como pode ser visto na Figura 5, compõem a estrutura obrigatória de uma página web.

```

1  <!DOCTYPE html>
2
3  <html>
4
5  <head>
6
7  </head>
8
9  <body>
10
11 </body>
12
13 </html>

```

📷 Figura 5: Estrutura obrigatória de uma página Web. Fonte: Paixão, 2020.

DE CONTEÚDO

Como nome sugere, têm o papel de marcar o conteúdo pelo seu tipo.

SEMÂNTICO

Relacionado ao tipo de conteúdo e à criação de seções para agrupá-lo de acordo com sua função no documento. Para melhor entender esse conceito, veja a Figura 6.

```

1  <!DOCTYPE html>
2
3  <html>
4
5  <head>
6    <title>Titulo da Página</title>
7  </head>
8
9  <body>
10
11    <header>
12      Cabeçalho da Página
13    </header>
14
15    <nav>
16      Barra de Navegação
17    </nav>
18
19    <main>
20      Conteúdo da Página
21
22      <aside>
23        Barra Lateral
24      </aside>
25    </main>
26
27    <footer>
28      Rodapé
29    </footer>
30
31  </body>
32
33 </html>

```

📷 Figura 6: Tags estruturais básicas de uma página Web. Fonte: Paixão, 2020.

ESTRUTURAIS

Juntamente com o elemento de definição do DocType, como pode ser visto na Figura 5, compõem a estrutura obrigatória de uma página web.

```

1  <!DOCTYPE html>
2
3  <html>
4
5  <head>
6
7  </head>
8
9  <body>
10
11 </body>
12
13 </html>

```

📷 Figura 5: Estrutura obrigatória de uma página Web. Fonte: Paixão, 2020.

DE CONTEÚDO

Como nome sugere, têm o papel de marcar o conteúdo pelo seu tipo.

SEMÂNTICO

Relacionado ao tipo de conteúdo e à criação de seções para agrupá-lo de acordo com sua função no documento. Para melhor entender esse conceito, veja a Figura 6.

```

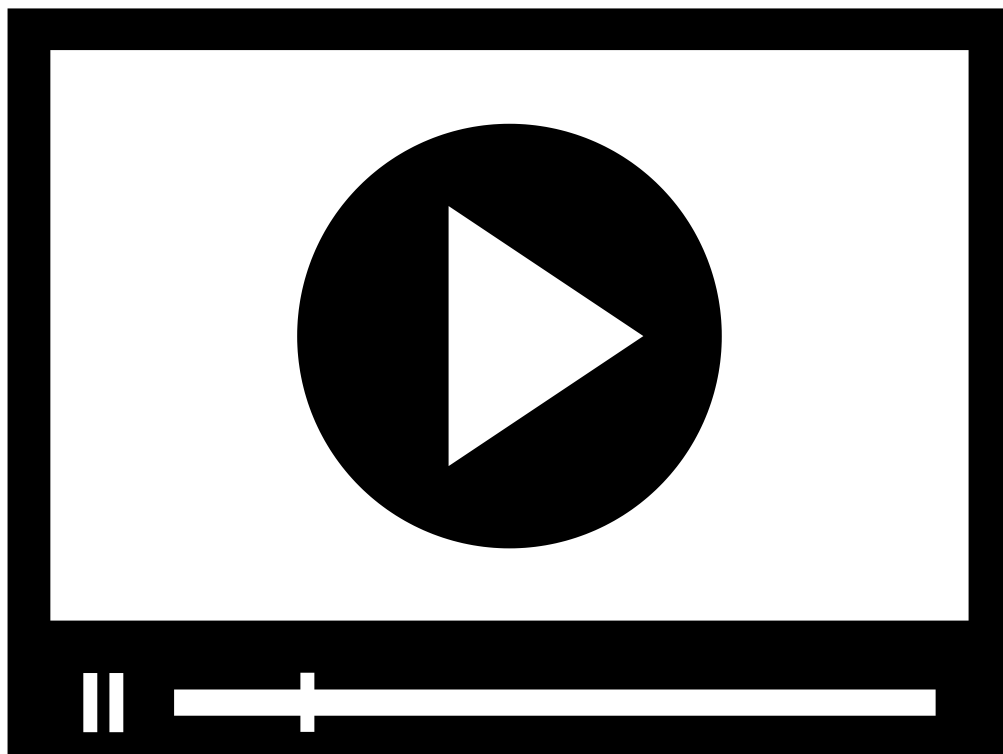
1  <!DOCTYPE html>
2
3  <html>
4
5  <head>
6    <title>Titulo da Página</title>
7  </head>
8
9  <body>
10
11    <header>
12      Cabeçalho da Página
13    </header>
14
15    <nav>
16      Barra de Navegação
17    </nav>
18
19    <main>
20      Conteúdo da Página
21
22      <aside>
23        Barra Lateral
24      </aside>
25    </main>
26
27    <footer>
28      Rodapé
29    </footer>
30
31  </body>
32
33 </html>

```

📷 Figura 6: Tags estruturais básicas de uma página Web. Fonte: Paixão, 2020.

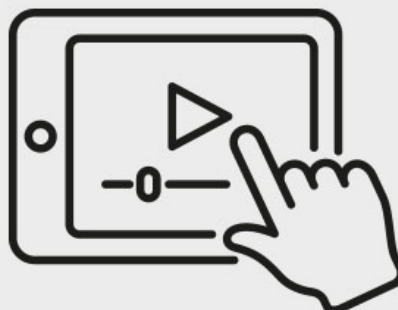
+ SAIBA MAIS

Uma listagem completa de tags e atributos (usados para adicionar características a uma tag) pode ser encontrada no site do W3C.



Conheça, agora, as diferenças entre uma estrutura em HTML e em folhas de estilo neste vídeo.

Para assistir a um vídeo sobre o assunto, acesse a versão online deste conteúdo.



Como visto na Figura 6, as tags `<header>`, `<nav>`, `<main>` e `<footer>` desempenham papel semântico, uma vez que estruturam a página em seções. Como seus nomes indicam, elas separam o conteúdo em partes lógicas que formam o esqueleto da maioria das páginas HTML, ou seja: Cabeçalho, menu de navegação, conteúdo principal e rodapé. Logo, tags de parágrafo, imagem, entre outras, são inseridas dentro de cada uma dessas seções, formando assim um documento HTML completo.

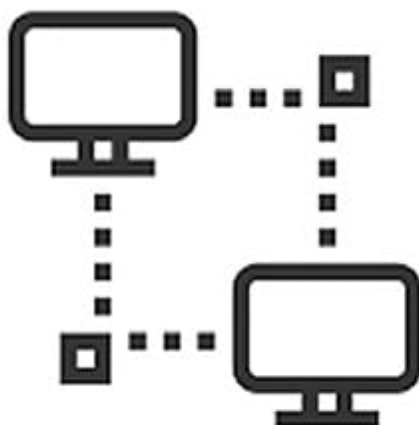
HTML 5

A versão mais recente da HTML é a 5, que trouxe algumas importantes evoluções em relação às anteriores. Entre tais novidades destacam-se:



Por ODSERG | Fonte: Shutterstock

Novos atributos e elementos, com foco sobretudo na semântica.



Por howcolour | Fonte: Shutterstock

Melhorias de conectividade.



Por 32 pixels | Fonte: Shutterstock

Possibilidade de armazenamento de dados no lado cliente.



Por Digital Bazaar | Fonte: Shutterstock

Otimização nas operações off-line.



Por Palau | Fonte: Shutterstock

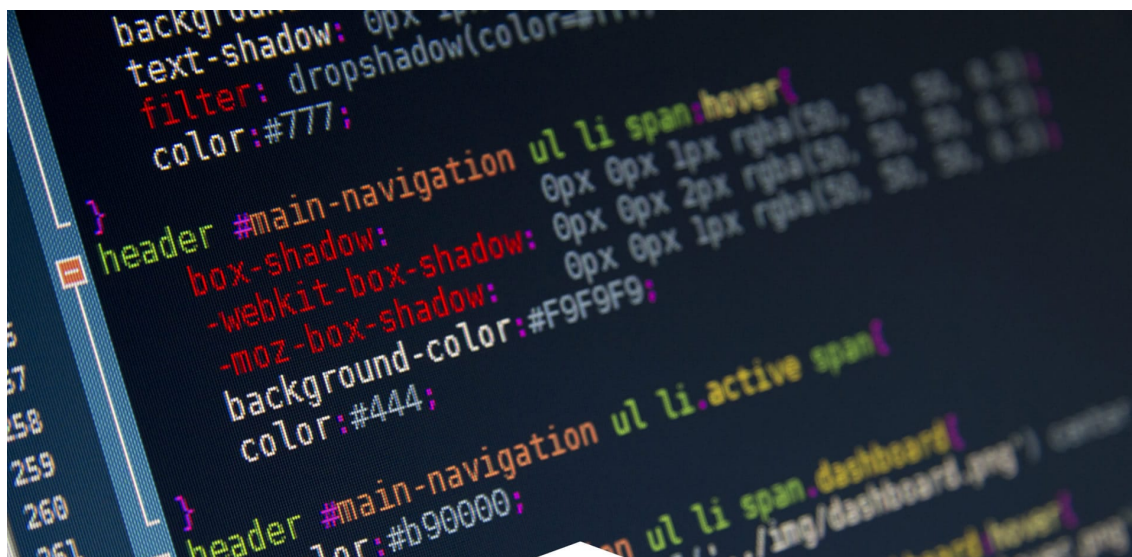
Suporte estendido a multimídia – áudio e vídeo.

CSS

O CSS (Sigla de Cascading Style Sheets. Em português, folhas de estilos em cascata.)

corresponde à segunda camada no tripé de tecnologias que formam o lado cliente, no Ambiente Web. Trata-se de uma linguagem declarativa cuja função é controlar a apresentação visual de páginas Web. Com isso, têm-se a separação de funções em relação à HTML:

UMA CUIDA DO CONTEÚDO, DA ESTRUTURAÇÃO; A OUTRA CUIDA DA APRESENTAÇÃO, DO LAYOUT.



Por Melody Smart | Fonte: Shutterstock

SINTAXE

A sintaxe da CSS consiste em uma declaração onde são definidos o(s) elemento(s) e o(s) estilo(s) que desejamos aplicar a ele(s) ou, em outras palavras:

O SELETOR

Um elemento HTML (body, div, p etc.) ou o seu identificador (atributo id) ou classe (atributo class).

A PROPRIEDADE

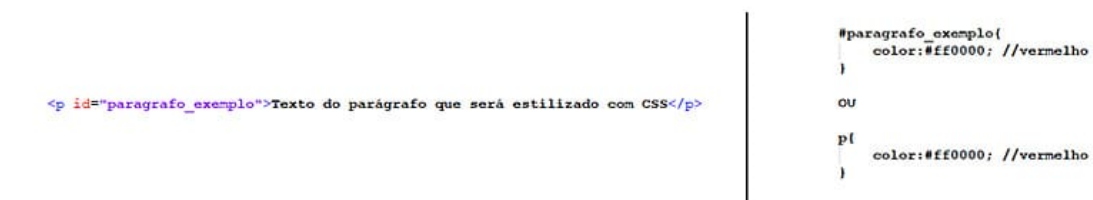
Característica do elemento (cor, fonte, posição etc.).

O VALOR

Novo parâmetro a ser aplicado à característica do elemento.

★ EXEMPLO

Para alterar a cor da fonte de um texto inserido em um parágrafo, poderíamos utilizar uma das variações apresentadas na Figura 7:



📷 Figura 7 – Exemplo de aplicação de CSS. Fonte: Paixão, 2020.

No exemplo anterior, vimos duas formas para definir o estilo de uma tag de parágrafo. No primeiro, o elemento ao qual o estilo foi aplicado foi definido com a utilização de seu atributo ID. A respeito dos seletores, propriedades existentes e mais detalhes sobre a CSS é recomendado ler o Guia de Referência do próprio W3C.

COMO INSERIR O CSS NA PÁGINA WEB

Há quatro formas de inserir o CSS em um documento: Inline, Interno, Externo e Escopo.

INLINE

Os estilos, neste caso, são aplicados com a utilização do atributo “style” seguido de uma ou mais propriedades/valores.

INTERNO

Os estilos são definidos com a utilização da tag <style>, dentro da tag <head> no documento.

EXTERNO

Essa é a forma preferencial de inserir estilos. Nela, é utilizado um arquivo externo, com extensão “.css”, contendo apenas estilos. Para vincular esse arquivo externo ao documento é utilizada a tag <link> dentro da tag <head>.

ESCOPO

Esta forma foi introduzida pela HTML5. Com ela, um estilo pode ser definido em nível de escopo, ou seja, declarado em seções específicas do documento. Sua declaração é feita da mesma forma que na Inline. Entretanto, no lugar de ser declarada no <head>, é declarada dentro da tag à qual se quer aplicar os estilos.

SELETORES CSS

A CSS permite uma série de combinações para a aplicação de estilos. Pode-se usar aplicações simples, como as vistas até aqui, nas quais apenas um elemento foi selecionado, ou combinações mais complexas, em que vários elementos podem ser agrupados a fim de receberem um mesmo estilo.

BOAS PRÁTICAS RELACIONADAS À CSS

É boa prática e fortemente recomendado utilizar a forma Externa para incluir CSS em uma página Web. Entre outras vantagens, como uma melhor organização do código, separando o HTML do Estilo, devemos ter em mente que um mesmo arquivo CSS pode ser usado em várias páginas de um site.



Por ZinetronN | Fonte: Shutterstock

★ EXEMPLO

Vamos imaginar a situação oposta: Temos um site onde o layout (topo, rodapé e menu, por exemplo) é comum em todas as suas páginas – arquivos .html independentes. Ao usarmos as formas Inline e Interna, precisaríamos replicar um mesmo código em todas as páginas. Imagine agora ter que fazer alguma alteração ou inclusão, tal operação precisaria ser repetida inúmeras vezes. Em contrapartida, se usarmos um arquivo externo e o linkarmos em todas as páginas, precisaremos trabalhar apenas em um único código, tornando-o muito mais fácil de ser mantido e ainda diminuindo consideravelmente nosso trabalho.

Outra **boa prática**, tendo em vista o **desempenho do carregamento da página Web** é **compactar o arquivo** – normalmente chamamos este processo de **minificação**. Existem softwares e até mesmo sites que fazem esse trabalho, que consiste em, resumidamente, diminuir os espaços e as linhas no arquivo .css, reduzindo assim o seu tamanho final.

OUTRAS CONSIDERAÇÕES SOBRE A CSS

Uma nova funcionalidade tem ganhado bastante espaço ultimamente no que diz respeito à CSS: os **pré-processadores**, como Sass, Less, Stylus etc.

PRÉ-PROCESSADORES

Em linhas gerais, um pré-processador é um programa que permite gerar CSS a partir de uma sintaxe – própria de cada pré-processador –, que inclui inúmeras facilidades não suportadas naturalmente pelo CSS, como variáveis, funções, regras aninhadas, entre outras.

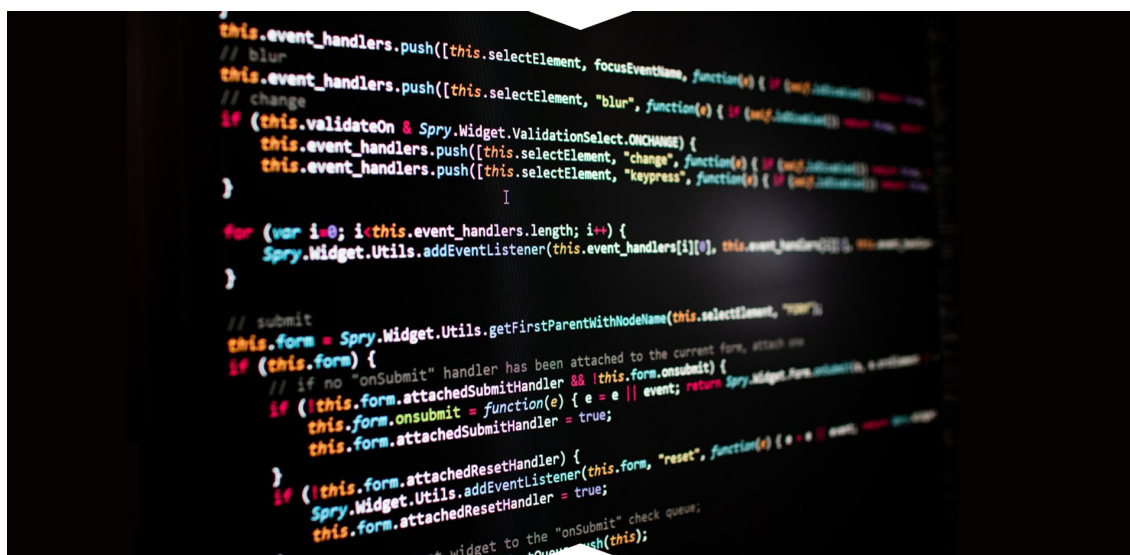
O fluxo de gerar CSS através de um pré-processador consiste na escrita do código contendo as regras a serem aplicadas, fazendo uso da sintaxe de cada pré-processador. Ao final, esse código será compilado, gerando então o código CSS normal.

JAVASCRIPT

O **Javascript** completa o tripé de tecnologias Web que rodam no lado cliente. Trata-se de uma linguagem de programação que, assim como o CSS, é interpretada pelo Navegador. Entre suas principais características, destaca-se o fato de ser multiparadigma (orientação a objetos, protótipos, funcional etc.).

JAVASCRIPT

Costuma-se abreviar o seu nome como “JS”, que é também a extensão de seus arquivos – quando vinculados externamente ao documento HTML.



Por Paolo De Gasperis | Fonte: Shutterstock

Sua função é, sobretudo, fornecer interatividade a páginas Web, e foi criada com o intuito de diminuir a necessidade de requisições ao lado Servidor, permitindo a comunicação assíncrona e a alteração de conteúdo sem que seja necessário recarregar uma página inteira.

SINTAXE

O Javascript é, ao mesmo, amigável, mas também completo e poderoso. Embora criado para ser leve, uma vez que é interpretado nativamente pelos navegadores, trata-se de uma

linguagem de programação completa e, como já mencionado, multiparadigma. Logo, seus códigos podem ser tanto estruturados quanto orientados a objetos. Além disso, permitem que bibliotecas, como **Jquery**, **Prototype** etc. sejam criadas a partir de seu core, estendendo assim a sua funcionalidade. Vejamos algumas características dessa linguagem:

JQUERY

Jquery é uma biblioteca Javascript rápida, pequena e rica em recursos que simplifica processos como a manipulação de documentos HTML, eventos, animação, além do AJAX (JQuery).

PROTOTYPE

Prototype é Framework Javascript de código aberto, modular e orientado a objetos que provê extensões ao ambiente de script do navegador, fornecendo APIs para manipulação do DOM e AJAX (PrototypeJs).

EVENTOS E MANIPULAÇÃO DOM

Essa linguagem oferece amplo suporte à manipulação de eventos relacionados a elementos HTML. É possível, por exemplo, utilizar um elemento `<button>` (botão) que, ao ser clicado, exiba uma mensagem na tela. Ou, ainda, aumentar o tamanho de uma fonte ou diminuí-lo.

MENSAGENS E ENTRADA DE DADOS

O Javascript possui suporte a funções nativas para a exibição de caixas de diálogo para entrada de dados ou exibição de mensagens, como alertas, por exemplo.

JSON

O JSON (JavaScript Object Notation) é um formato para troca de dados no Ambiente Web. Logo, é importante fazer menção a ele quando falamos das tecnologias do lado cliente, uma vez que tal formato está entre os mais utilizados no intercâmbio de dados com o lado servidor e entre diferentes sistemas, independentemente da linguagem utilizada nos mesmos. Embora relacionado ao Javascript, como o próprio nome diz, não é um formato exclusivo dessa linguagem e pode ser usado mesmo em sistemas onde esta não esteja presente.



Por IYIKON | Fonte: Shutterstock

COMO INSERIR O JAVASCRIPT NA PÁGINA WEB

A inserção do Javascript em uma página HTML é feita de forma bastante similar à da CSS. Dessa forma, é possível tanto inserir um código Javascript diretamente no documento – dentro da tag `<script>`, seja no `<head>` ou em qualquer outra parte – como através de arquivos externos com a extensão “.js”, devendo estes serem linkados na tag `<head>`. Vejamos alguns exemplos das formas de inclusão do Javascript:

★ EXEMPLO

```

<!DOCTYPE html>
<html>
  <body>
    <button onclick="funcaoJavascript()">Clique Aqui</button>

    <script>
      function funcaoJavascript(){
        alert("Você clicou no botão");
      }
    </script>
  </body>
</html>

```

```

<head>
  <script type="text/javascript" >
    window.onload = function(){
      alert("Bem-Vindo(a)");
    }
  </script>
</head>

```

```

<head>
  <script src="script.js"></script>
</head>

/*Conteúdo do arquivo script.js*/
function funcaoJavascript(){
  alert("Você clicou no botão");
}

```

📷 Figura 8 – Como inserir Javascript em páginas Web. Fonte: Paixão, 2020.

A primeira parte da Figura 8 mostra o fragmento de uma página HTML onde há um botão e um evento – onclick – que, ao ser disparado (no click do botão), chama a função “funcaoJavascript”, declarada no corpo do HTML, logo após a tag <button>. Tal função exibe um alerta na tela. A segunda parte da Figura mostra a declaração do Javascript dentro da tag <head>. Este código faz uso de um evento, o de carregamento da janela do navegador, para exibir uma alerta de boas-vindas. Por fim, na última parte da Figura vemos a declaração para a incorporação de um arquivo externo JS, assim como o código do arquivo.

BOAS PRÁTICAS RELACIONADAS AO JAVASCRIPT

1.

Assim como foi dito em relação à CSS, deve-se dar preferência à utilização de arquivos externos para a incorporação de Javascript.

Outra boa prática consiste em incorporar arquivos externos de Javascript ao final da página, após o fechamento da tag `<body>`. Com isso, temos ganho de desempenho, uma vez que o carregamento da página HTML é feito de cima para baixo. Logo, quanto mais código, CSS ou Javascript, existir no topo da página, dentro da tag `<head>`, mais demorado será o carregamento da página em si.

2.



Por outro lado, deve-se tomar cuidado ao mover o Javascript para o final da página: caso algum elemento, evento ou funcionalidade da página dependa do código em questão, eles não serão executados, uma vez que foram carregados antes do Javascript. Nesse caso, pode ser adotada uma abordagem híbrida, onde os códigos Javascript dos quais a página dependa para funcionar corretamente fiquem no início e os demais, ao final.

ATENÇÃO

Assim como já foi falado em relação à CSS, é também recomendado minificar o arquivo .js a fim de otimizar o desempenho do carregamento da página.

Finalmente, cabe citar que, embora originalmente exclusivo do lado cliente, atualmente o Javascript também pode ser usado no lado servidor. O **node.js** é um exemplo de ambiente que

faz uso do Javascript no lado servidor.

NODE.JS

É um interpretador de JavaScript assíncrono com código aberto orientado a eventos, focado em migrar a programação do Javascript do cliente (frontend) para os servidores.

VERIFICANDO O APRENDIZADO

1. AO DESENVOLVERMOS UMA PÁGINA WEB DEVEMOS NOS PREOCUPAR NÃO SOMENTE COM O RESULTADO FINAL, MAS TAMBÉM EM UTILIZARMOS CORRETAMENTE CADA UMA DAS TECNOLOGIAS.

NESTE CONTEXTO, ASSINALE A OPÇÃO CORRETA QUANTO ÀS BOAS PRÁTICAS A SEREM SEGUIDAS:

- A)** Utilizar os elementos HTML corretamente, tendo em mente a semântica; separar as responsabilidades entre cada tecnologia; otimizar o tempo de carregamento das páginas; utilizar folhas de estilo e Javascript a partir de arquivos externos.
- B)** Deve-se evitar, sempre que possível, fazer uso de novas técnicas ou novas funcionalidades no que diz respeito às tecnologias Client Side. Isso porque as tecnologias Web já possuem uma especificação própria, antiga, e, por isso, não se adaptam bem com novos recursos.
- C)** A CSS possui um sistema de hierarquia, assim como o Javascript. Com isso, ao usarmos ambos no HTML, é recomendado usar estilos e scripts inline, já que facilitam o entendimento do comportamento e também visual do elemento ao qual foram aplicados.
- D)** Remover tanto a CSS quanto o Javascript internos para o final da página otimiza o desempenho e acelera o tempo de carregamento da página. Logo, esta é uma das práticas

mais recomendadas.

2. COMO VIMOS, CADA TECNOLOGIA DO LADO CLIENTE POSSUI SUA PRÓPRIA FUNÇÃO. LOGO, A RESPEITO DA SEPARAÇÃO DE FUNÇÕES E RESPONSABILIDADES, ASSINALE A ALTERNATIVA CORRETA:

A) O HTML cuida do conteúdo, o CSS, do layout/apresentação e o Javascript, do comportamento/interação. Com isso, ao não misturarmos as funções – embora seja possível –, obtemos vários benefícios, como o de separação de interesses e consequente facilidade para manter o código, uma vez que podemos ter diferentes pessoas trabalhando ao mesmo tempo em diferentes partes do site.

B) O HTML é a base, a principal tecnologia do lado cliente. Apenas utilizando HTML é possível criar uma página rica em conteúdo – já que as tags servem justamente para isso –, layout – já que tudo fica dividido na estrutura semântica do HTML – e interatividade, já que elementos como o Formulário existem justamente para isso.

C) Mais importante do que a preocupação com as funções de cada tecnologia é o resultado exibido no navegador. Logo, deve-se dar preferência ao resultado final, independente do que foi feito e de como foi feito, em termos de tecnologia, para se chegar a ele.

D) Entre as três tecnologias do lado cliente, CSS é a mais dispensável e menos importante, já que é possível cuidar de todo o layout e apresentação fazendo uso apenas de HTML.

GABARITO

1. Ao desenvolvermos uma página Web devemos nos preocupar não somente com o resultado final, mas também em utilizarmos corretamente cada uma das tecnologias.

Neste contexto, assinale a opção correta quanto às boas práticas a serem seguidas:

A alternativa **"A "** está correta.

Como frisado nas seções de Boas Práticas, o Ambiente Web está em constante evolução. Tal fator, somado aos princípios básicos como semântica e separação de responsabilidades definem o que são as boas práticas quanto às tecnologias Client Side.

2. Como vimos, cada tecnologia do lado cliente possui sua própria função. Logo, a respeito da separação de funções e responsabilidades, assinale a alternativa correta:

A alternativa **"A "** está correta.

As tecnologias do lado cliente foram desenvolvidas em momentos distintos, a começar pela HTML. Com isso, a partir do surgimento de novas necessidades, novas tecnologias, como a CSS e o Javascript foram desenvolvidas. A utilização em conjunto destas tecnologias, que se complementam, traz inúmeros benefícios, desde a otimização na criação das páginas ao resultado final.

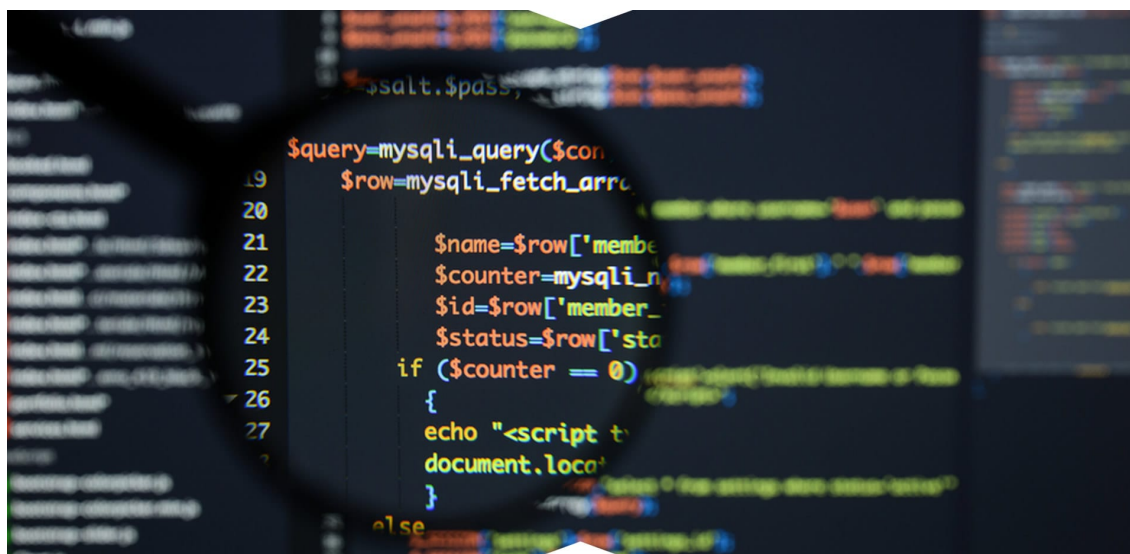
MÓDULO 4

⦿ Reconhecer as tecnologias do lado servidor

AS TECNOLOGIAS DO LADO SERVIDOR: PHP, PÁGINAS DINÂMICAS E ACESSO A DADOS

PHP

Uma das principais funções das linguagens de programação Server Side é permitir o acesso a informações armazenadas em Bancos de Dados. Uma vez que apenas utilizando HTML e Javascript isto não é possível, faz-se necessária a utilização de uma linguagem no lado servidor. Entre as diversas linguagens disponíveis no lado servidor estão o Java, o Python, o ASP, o .NET e o PHP – que conheceremos um pouco mais ao longo deste tópico.



Por Casimiro PT | Fonte: Shutterstock

O QUE É O PHP E PARA QUE SERVE

PHP (Hypertext Preprocessor) é uma linguagem de programação baseada em script, open source e destinada, sobretudo, ao desenvolvimento Web. Trata-se de uma linguagem criada para ser simples, tendo nascida estruturada e, posteriormente, adotado o paradigma de orientação a objetos – isto apenas 10 anos depois da sua criação.

FRISANDO O QUE JÁ FOI DITO ACIMA, O PRINCIPAL FOCO DO PHP SÃO OS SCRIPTS DO LADO SERVIDOR, DANDO SUPORTE A FUNÇÕES COMO COLETA E PROCESSAMENTO DE DADOS ORIUNDOS DE FORMULÁRIOS HTML, GERAÇÃO DE CONTEÚDO DINÂMICO COM O ACESSO A BANCOS DE DADOS, ENTRE OUTRAS. ALÉM DISSO, DO FOCO NOS SCRIPTS NO LADO SERVIDOR, É POSSÍVEL TAMBÉM UTILIZAR O PHP ATRAVÉS DE SCRIPTS EM LINHA DE COMANDO E NA CRIAÇÃO DE APLICAÇÕES DESKTOP (COM A UTILIZAÇÃO DA EXTENSÃO PHP-GTK), EMBORA NÃO SEJA A MELHOR LINGUAGEM PARA ISSO.

COMO O PHP FUNCIONA

O PHP é uma linguagem interpretada, ou seja, ela precisa “rodar” sobre um servidor Web. Com isso, todo o código gerado é interpretado pelo servidor, convertido em formato HTML e então exibido no navegador.

ETAPA 01

ETAPA 02

ETAPA 03

ETAPA 04

ETAPA 01

O PHP é uma linguagem interpretada, ou seja, precisa rodar sobre um servidor Web.

ETAPA 02

Com isso, todo o código gerado é interpretado pelo servidor.

ETAPA 03

Convertido em formato HTML.

ETAPA 04

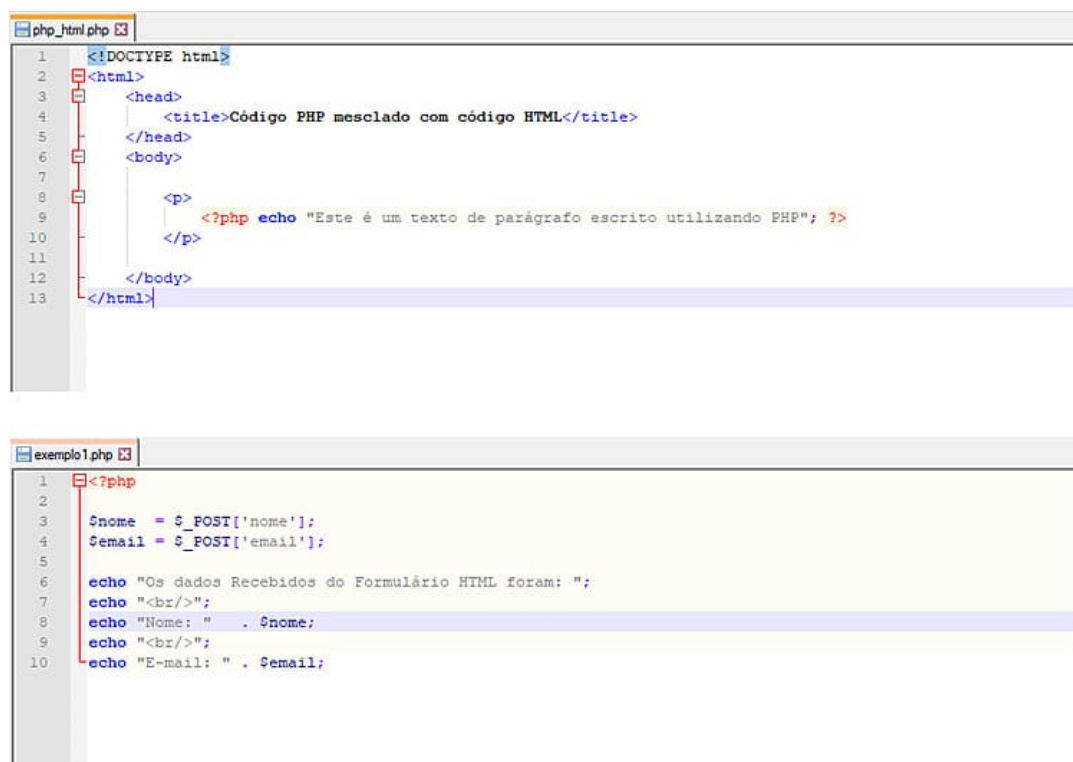
E, então, exibido no navegador.

LOGO, O CÓDIGO-FONTE NÃO PODE SER VISTO NO LADO CLIENTE, MAS APENAS O HTML GERADO.

Outra característica importante do PHP é poder ser utilizado na maior parte dos Sistemas Operacionais, assim como em vários servidores Web diferentes, como o Apache, o IIS e o Nginx, entre outros.

ANATOMIA DE UM SCRIPT PHP

Um script PHP é composto por código delimitado pelas tags `<?php` e `?>`. A última, de fechamento, não é obrigatória. Devido à sua simplicidade, um mesmo script PHP pode conter tanto código estruturado quanto orientado a objetos. Pode até conter código de marcação HTML. Neste último caso, o código próprio do PHP deverá ficar entre as tags de abertura e fechamento. A **Figura 9** mostra dois exemplos de código, um apenas em PHP e outro mesclado com HTML. Ao analisarmos os códigos, inicialmente é importante notar que ambos possuem a extensão “.php”. Outras extensões possíveis, mas atualmente em desuso, são “.php3”, “.php4”, “.phtml”.

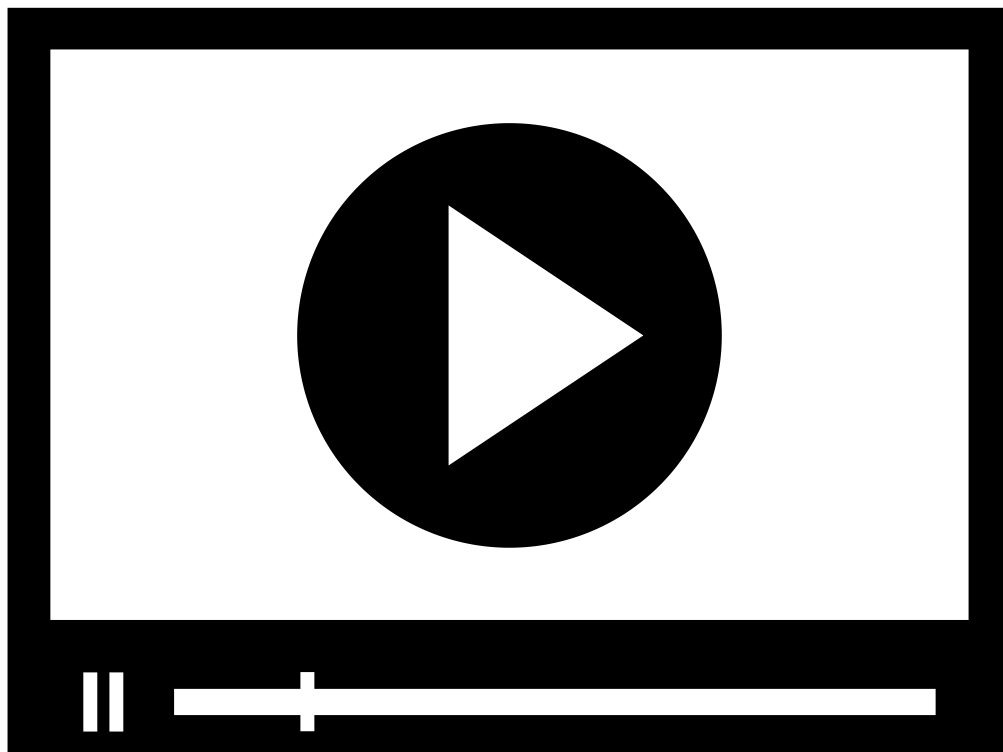


📷 Figura 9 – Exemplos de Código PHP. Fonte: Paixão, 2020.

A seguir, no primeiro código, temos as tags de um arquivo HTML comum, com exceção do código inserido dentro das tags `<?php` e `?>`. Aqui temos a função “echo”, que serve para

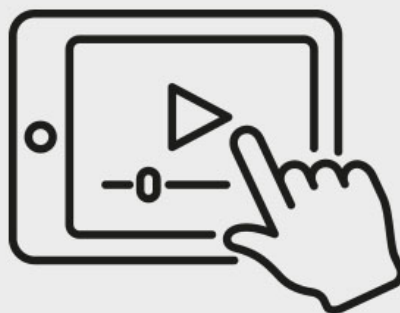
imprimir algo na tela, associado a uma frase. Quando visualizado no navegador, o código será renderizado como HTML normal. Caso exibamos a fonte, só será possível ver as tags HTML e o conteúdo, sem o código PHP em questão.

Na segunda imagem, temos um exemplo de código onde são definidas duas variáveis, \$nome e \$email, que recebem dois valores enviados de um formulário HTML, através do método POST. Daí a utilização do array superglobal \$_POST – cujos índices ‘nome’ e ‘email’ correspondem ao atributo ‘name’ dos campos input do formulário. A seguir, é utilizada a função “echo” para a impressão de uma frase e do conteúdo das variáveis recebidas. Repare ainda na utilização, mais uma vez, de uma tag html, a
, em meio a código PHP.



Assista a esse vídeo e conheça um exemplo de código PHP.

Para assistir a um vídeo sobre o assunto, acesse a versão online deste conteúdo.



SINTAXE

Assim como fizemos com o Javascript, veremos a seguir um resumo sobre a sintaxe do PHP:

VARIÁVEIS

As variáveis em PHP são criadas com a utilização do símbolo de cifrão (\$). Além disso, PHP é **case sensitive**. Logo, tomando como exemplo as variáveis mostradas na Figura 9, \$nome e \$email, ambas são diferentes das variáveis \$Nome e \$Email.

TIPOS DE DADOS

O PHP é uma linguagem fracamente tipada. Logo, embora possua suporte para isto, não é necessário definir o tipo de uma variável em sua declaração.

Os tipos de dados disponíveis em PHP são: Booleanos, inteiros, números de ponto flutuante, strings, arrays, iteráveis (iterables), objetos, recursos, NULL e call-backs.

OPERADORES CONDICIONAIS

O PHP tem suporte às condicionais if, else, if e else ternários, if else e switch.

LAÇOS DE REPETIÇÃO

Estão disponíveis os laços: For, foreach, while e do-while.

FUNÇÕES E MÉTODOS

O PHP possui uma grande quantidade de funções e métodos nativos.

CASE SENSITIVE

Sensível a letras maiúsculas e minúsculas. Ou seja, faz diferença quando utilizamos uma e outra – conforme o exemplo fornecido, onde as letras “e” e “E” estão em minúsculo e maiúsculo, respectivamente, em cada nomeação.

INCLUSÃO DE SCRIPTS DENTRO DE SCRIPTS

O PHP permite a inclusão de um script dentro de outro script. Isso é muito útil, sobretudo se pensarmos no paradigma de orientação a objetos, em que temos, em um programa, diversas classes, codificadas em diferentes scripts. Logo, sempre que precisarmos fazer uso de uma dessas classes, de seus métodos ou atributos, basta incluí-la no script desejado. Para incluir um script em outro o PHP disponibiliza algumas funções:

Include

Require

Include once

Require_once

ACESSO AO SISTEMA DE ARQUIVOS

Através do PHP é possível ter acesso ao sistema de arquivos do servidor Web. Com isso é possível, por exemplo, manipular arquivos e diretórios, desde a simples listagem à inclusão ou exclusão de dados.



Por hanss | Fonte: Shutterstock

PÁGINAS DINÂMICAS

Se fôssemos implementar em uma página Web tudo o que estudamos até aqui, teríamos uma página HTML básica, com um pouco de interação no próprio navegador, graças ao Javascript, e também com um pouco de estilo, este devido ao CSS.

Além disso, já sabemos que é possível enviar dados do HTML para o PHP através de um formulário. Mas e agora? Qual o próximo nível? O que fazer a seguir? A resposta para essas duas perguntas está no que abordaremos a seguir: **páginas dinâmicas** e acesso a dados.

PÁGINAS DINÂMICAS

Para prosseguirmos, é importante definirmos o que são páginas dinâmicas. A melhor forma de fazer isso, porém, é definindo o que seria o seu antônimo, ou seja, as páginas estáticas. HTML + Javascript + CSS, sem conexão com uma linguagem de programação, formam o que podemos chamar de páginas estáticas. Embora seja até possível termos um site inteiro composto por páginas estáticas, isso seria muito trabalhoso e também nada usual.

★ EXEMPLO

Imagine um site que tenha, por exemplo, dez páginas. Agora imagine que esse site tenha a mesma estrutura visual, o mesmo cabeçalho, menu, rodapé e outros pontos em comum. Pense em um blog, por exemplo, onde o que muda são os conteúdos dos posts. No site estático, teríamos que escrever dez diferentes arquivos HTML, modificando o conteúdo em cada um deles, diretamente nas tags HTML e só conseguiríamos reaproveitar os estilos e a interatividade de navegador utilizando CSS e Javascript externos. Entretanto, todo o conteúdo precisaria ser digitado e muito código HTML, repetido. Todo esse trabalho nos ajuda a entender o que são páginas estáticas.

Ainda utilizando o exemplo de um blog, imagine que você deseja receber comentários em seus posts, deseja que seus visitantes possam interagir com você e vice-versa. Como fazer isso? A resposta, como você já deve imaginar, é: páginas dinâmicas.

A combinação das tecnologias do lado cliente com as tecnologias do lado servidor produzem as páginas dinâmicas. Nelas, é possível receber as informações provenientes do cliente, processá-las, guardá-las, recuperá-las e utilizá-las sempre que desejarmos. E não é só isso: Podemos guardar todo o conteúdo do nosso blog no banco de dados. Com isso, teríamos apenas uma página PHP que recuperaria nosso conteúdo no banco e o exibiria no navegador. A Tabela 2 apresenta um pequeno resumo comparativo entre as páginas estáticas e dinâmicas.

	Páginas estáticas	Páginas dinâmicas
Inclusão/Alteração/Exclusão de conteúdo	Manualmente, direto no código HTML	Automaticamente através de scripts no lado servidor, como PHP
Armazenamento do conteúdo	Na própria página HTML	Em um banco de dados

 Tabela 2 – Comparativo entre páginas estáticas e dinâmicas. Fonte: Paixão, 2020.

 **Atenção!** Para visualização completa da tabela utilize a rolagem horizontal

Outra importante característica de um site dinâmico é possibilitar a utilização de ferramentas de gestão de conteúdo (CMS) para manter as informações do site sempre atualizadas. Com isso, depois de pronto, não será mais necessário mexer nos códigos-fonte, basta acessar a ferramenta e gerenciar o conteúdo através dela. Já no site estático, será preciso modificar diretamente o código HTML, tornando necessário alguém com conhecimento técnico para isto.

ACESSO A DADOS

Como mencionado anteriormente, o Ambiente Web é composto por tecnologias que rodam do lado cliente e do lado servidor. Complementando o que vimos até aqui, temos ainda, do lado

servidor, **o banco de dados**. De forma resumida, podemos defini-lo como um repositório onde diversas informações podem ser armazenadas e posteriormente recuperadas. Para realizar a gestão destes dados, existem os **SGBD**, ou Sistemas Gerenciadores de Bancos de Dados.

SGBD

Há diversos tipos de SGBD, para as mais variadas necessidades, com opções gratuitas ou pagas. Entre os gratuitos, dois são comumente utilizados em conjunto com o PHP: MySQL e PostgreSQL.



Por Tommy Lee Walker | Fonte: Shutterstock

Se, por um lado, o SGBD é responsável por montar a estrutura do banco de dados – entre outras funções –, por outro lado, para recuperarmos uma informação guardada em um banco de dados e exibi-la em uma página Web, é necessário utilizar uma linguagem do lado servidor, como o PHP. Em outras palavras, não é possível acessar o banco de dados utilizando apenas HTML ou mesmo Javascript. Sempre será necessária a utilização de uma linguagem server side para o acesso aos dados.

FORMAS DE ACESSO A DADOS

A partir das tecnologias vistas até aqui, há algumas formas de acessar os dados guardados em um banco de dados.

A PARTIR DO HTML

Uma das maneiras mais comuns de enviar e recuperar dados a partir do HTML é fazendo uso de formulários. Com eles é possível submetermos nossos dados para uma linguagem no lado servidor/PHP. Este, então recebe as informações e as armazena no banco de dados. Da mesma forma acontece o caminho inverso. Podemos ter um formulário em nossa página HTML que solicite dados ao PHP e este as envie de volta, após recuperá-las do banco de dados.

Vale lembrar ainda o que vimos sobre o PHP: Ele permite a utilização de códigos HTML diretamente em seus scripts. Logo, uma página Web feita em PHP pode recuperar dados do banco de dados toda vez que é carregada. É isso o que acontece na maioria dos sites. Cada página visualizada é composta por conteúdo armazenado em banco de dados e código HTML produzidos por uma linguagem do lado servidor. Com isso, cada página que abrimos em sites dinâmicos implica em uma chamada/requisição ao lado servidor – script e banco de dados.

A PARTIR DO JAVASCRIPT

O Javascript possui, essencialmente, duas formas para se comunicar com linguagens do lado Servidor: Através das APIs (Application Programming Interface) XMLHttpRequest e Fetch API. A primeira é amplamente utilizada, sendo a forma mais comum de realizar esta comunicação. É normalmente associada a uma técnica de desenvolvimento Web chamada AJAX. Já a segunda é mais recente e oferece algumas melhorias, embora não seja suportada por todos os navegadores.

A comunicação em ambas consiste em, através de algum evento no navegador – que normalmente é originada em uma ação disparada pelo usuário –, enviar uma requisição ao lado servidor, como recuperar algum dado, por exemplo, tratar o seu retorno e o exibir na tela. Isso tudo sem que seja necessário recarregar toda a página.

VERIFICANDO O APRENDIZADO

1. O PHP É UMA LINGUAGEM DE SCRIPT, ALTAMENTE ADAPTÁVEL À HTML E QUE LHE POSSIBILITA INTERATIVIDADE E DINÂMICA. ASSINALE

A ALTERNATIVA CORRETA QUANTO A ESTA AFIRMAÇÃO:

- A)** É possível criar um script PHP que faça acesso a banco de dados utilizando apenas código HTML.
- B)** Para recuperar informações de um banco de dados, a HTML precisa fazer uso do PHP, seja diretamente – a partir de algum elemento próprio, – ou através de Javascript.
- C)** Como o PHP é altamente adaptável à HTML e esta ao Javascript, um script escrito nesta última linguagem pode recuperar informações acessando diretamente o banco de dados.
- D)** O PHP é altamente adaptável à HTML. Logo, assim como a HTML, um script PHP é renderizado diretamente pelo navegador.

2. AS PÁGINAS DINÂMICAS, AO CONTRÁRIO DAS PÁGINAS ESTÁTICAS, PROVEEM DINAMISMO AO AMBIENTE WEB. NESTE CONTEXTO, ASSINALE A OPÇÃO CORRETA:

- A)** Uma página Web completa só pode ser produzida com a utilização de páginas dinâmicas.
- B)** As páginas dinâmicas são, resumidamente falando, uma forma de interação entre um usuário e uma página HTML. Logo, uma página que faz uso de Javascript é uma página dinâmica.
- C)** A única vantagem, de fato, de se utilizar páginas dinâmicas é guardar os dados do site em um lugar mais seguro.
- D)** A utilização de linguagens de programação Server Side é a principal característica de uma página dinâmica.

GABARITO

1. O PHP é uma linguagem de script, altamente adaptável à HTML e que lhe possibilita interatividade e dinâmica. Assinale a alternativa correta quanto a esta afirmação:

A alternativa **"B "** está correta.

O PHP é uma linguagem Server Side, utilizada, sobretudo, para criar páginas dinâmicas, em conjunto com a HTML e demais tecnologias do lado cliente. Embora muito adaptável à HTML, o PHP é uma linguagem de programação completa, que possui uma sintaxe específica, assim como funções e métodos nativos que lhe possibilitam o acesso tanto ao sistema de arquivos quanto à diferentes bancos de dados.

2. As páginas dinâmicas, ao contrário das páginas estáticas, proveem dinamismo ao Ambiente Web. Neste contexto, assinale a opção correta:

A alternativa "D " está correta.

Nas páginas dinâmicas, todo o conteúdo de um site pode ser gerenciado automaticamente através de scripts que rodam no servidor.

CONCLUSÃO

CONSIDERAÇÕES FINAIS

Ao longo deste tema, vimos como o modelo Cliente x Servidor, inicialmente restrito às redes internas das empresas, evoluiu tanto nos diferentes modelos de camadas quanto na migração para a internet. Novas tecnologias foram criadas no lado cliente – linguagens de marcação; folhas de estilo; linguagens de script – e também do lado servidor – linguagens de script; bancos de dados; páginas dinâmicas –, criando o Ambiente Web como o conhecemos atualmente, caracterizado pelas tecnologias que formam sua base e pela preocupação com a melhor experiência possível para os usuários que dele fazem uso.

Para ouvir um *podcast* sobre o assunto, acesse a versão online deste conteúdo.



Podcast

REFERÊNCIAS

BENYON, D. **Interação Humano-Computador**. São Paulo: Pearson Prentice Hall, 2011.

GLOBAL STATS. **StatCounter**: Screen Resolution Stats Brazil.

KNIGHT, K. **Responsive Web Design**: What It Is And How To Use It. In: Smashing Magazine. Responsive Design.

MOZILLA. **MDN Web Docs**: CSS Preprocessor.

PAIXÃO, A. **Notas de Aula sobre Ambiente Web Cliente x Servidor e as Tecnologias do professor Alexandre Paixão**. Disponível sob licença Creative Commons BR Atribuição – CC BY, 2020.

SILVA, M. S. **Web Design Responsivo**: Aprenda a criar sites que se adaptam automaticamente a qualquer dispositivo, desde desktop até telefones celulares. São Paulo: Novatec, 2014.

WROBLEWSKI, L. **Mobile First**.

W3SCHOOLS. **CSS Reference**.

Leia o livro ***Design de Interação: Além da interação homem-computador***, escrito por Jennifer Preece, Yvonne Rogers e Helen Sharp, da editora John Wiley e Sons.

Leia o livro ***Interação humano-computador***, escrito por Simone Diniz Junqueira Barboza e Bruno Santana da Silva, da editora Elsevier.

Visite o tópico "**CSS Units**" no site do W3C e conheça a lista completa das unidades de medidas da CSS. Conforme mencionado, a CSS possui uma série de unidades de medidas para expressar tamanho, sendo esse associado a propriedades como "width", "margin", entre outros. Estão disponíveis unidades de tamanho absoluto, como centímetros, milímetros, pixels etc. e também unidades de tamanho relativo, como "em", "ex", entre outros.

Assim como a maioria das tecnologias, a CSS possui um guia de referência oficial. Como esse guia é mantido pelo W3C, entre no site para acessá-lo e conhecer a sua especificação e todos os detalhes a ela relacionados.

CONTEUDISTA

Alexandre de Oliveira Paixão

 **CURRÍCULO LATTES**