HTML5, CSS E JAVASCRIPT

Erick Eduardo Petrucelli

INFORMAÇÃO E COMUNICAÇÃO

NT EDITORA

HTML5, CSS E JAVASCRIPT

Erick Eduardo Petrucelli

INFORMAÇÃO E COMUNICAÇÃO

NT EDITORA

Autor

Erick Eduardo Petrucelli

Mestre em Engenharia de Produção, MBA em Gestão de Tecnologia da Informação, Tecnólogo em Processamento de Dados, Certificado MCP e MCPD. Atua com desenvolvimento de software há mais de 20 anos, principalmente com as tecnologias HTML, CSS, JavaScript, Node.js, .NET e SQL. Professor de Ensino Superior há mais de 10 anos. Coordenador do Curso Superior de Tecnologia em Sistemas para Internet da Faculdade de Tecnologia (Fatec) de Taquaritinga/SP. Eventualmente, palestra sobre desenvolvimento Web e sobre o mercado de TI. Desenvolvedor por vocação, nas horas vagas se mantém envolvido com o ecossistema open-source de desenvolvimento Web (principalmente com a plataforma Web, Vue.js e Node.js).

Design Instrucional
Vinicius Abreu

Projeto Gráfico
NT Editora

Revisão
Filipe Lopes
Valesca Fonseca

Capa
NT Editora

Editoração Eletrônica
Danilo Souza
Alice Leite
Ana Luísa Amaral

Ilustração
Valter Luis Corrêa

NT Editora, uma empresa do Grupo NT
SCS Quadra 2 – Bl. C – 4° andar – Ed. Cedro II
CEP 70.302-914 – Brasília – DF
Fone: (61) 3421-9200
sac@grupont.com.br
www.nteditora.com.br e www.grupont.com.br

Petrucelli, Erick Eduardo.
HTML5, CSS e JavaScript / Erick Eduardo Petrucelli – 1. ed. reimpr. – Brasília: NT Editora, 2019.
222 p. il. ; 21,0 X 29,7 cm.

ISBN 978-85-8416-622-0

1. HTML5 2. CSS 3. JAVASCRIPT
I. Título

ÍCONES

Prezado(a) aluno(a),
Ao longo dos seus estudos, você encontrará alguns ícones na coluna lateral do material didático. A presença desses ícones o(a) ajudará a compreender melhor o conteúdo abordado e a fazer os exercícios propostos. Conheça os ícones logo abaixo:

Saiba mais
Esse ícone apontará para informações complementares sobre o assunto que você está estudando. Serão curiosidades, temas afins ou exemplos do cotidiano que o ajudarão a fixar o conteúdo estudado.

Importante
O conteúdo indicado com esse ícone tem bastante importância para seus estudos. Leia com atenção e, tendo dúvida, pergunte ao seu tutor.

Dicas
Esse ícone apresenta dicas de estudo.

Exercitando
Toda vez que você vir o ícone de exercícios, responda às questões propostas.

Exercícios
Ao final das lições, você deverá responder aos exercícios no seu livro.

Bons estudos!

Sumário

APRESENTAÇÃO

Caro(a) estudante,

Seja bem-vindo(a) aos estudos de **HTML5, CSS e JavaScript!**

Vivemos em um mundo que parece cada vez mais dependente da Internet. Através dela, temos comunicação em tempo real, aplicações corporativas, comércio eletrônico, redes sociais, filmes, músicas, jogos, notícias e uma infinidade de outros benefícios. Padronizando como as informações seriam exibidas, surgiu a WorldWideWeb, termo em inglês para Rede Mundial de Computadores.

A primeira linguagem proposta para a W*eb* foi o HTML. Trata-se de uma linguagem de marcação, ou seja, colocamos "marcas" em um documento para definir a estrutura e o significado de cada informação. Da mesma forma que, ao escrever um texto, você define o que fica no título e o que fica em cada parágrafo, com o HTML faremos tais definições usando pequenos códigos para especificar o que é cada coisa (títulos, parágrafos, *links*, imagens, botões etc.). Ao longo dos anos, o HTML foi evoluindo e ganhando novas marcações para definir melhor as informações. Atualmente, estamos na versão HTML5.

Entretanto, o HTML não foi criado para formatação dos conteúdos, ou seja, nunca teve como objetivo questões visuais. Assim, a linguagem de estilização CSS foi proposta. Por meio dela, definiremos regras visuais para os conteúdos de nossas páginas, lidando com questões como cores, fontes, medidas e posicionamentos. Essa linguagem também foi evoluindo conforme as necessidades foram ficando mais refinadas e ganhou recursos bem interessantes para o *web design* moderno. Atualmente, estamos na versão CSS3.

Fechando o trio, temos o *JavaScript*, linguagem de programação criada em uma época em que a *Web* era completamente estática, ou seja, não dava para fazer nada além de ver conteúdos fixos navegando de uma página para outra. Ela é a única das três que é realmente uma linguagem de programação, ou seja, oferece todos os recursos para criar interações avançadas nas páginas, e foi padronizada internacionalmente através da especificação chamada ECMAScript.

Embora os navegadores tenham sido, por muito tempo, o ambiente exclusivo dessas linguagens *Web*, a evolução de tecnologias e plataformas trouxe um novo cenário no qual a *Web* pode ser usada como interface para criar modernas aplicações *mobile* e até mesmo tradicionais aplicações *desktop*. Hoje, essas três linguagens são totalmente padronizadas, amplamente utilizadas, possuem grande apelo no mercado de trabalho e permitem criar interfaces ricas e modernas para diversas plataformas com custo reduzido, sendo possível aproveitar o conhecimento para criar vários tipos de aplicações diferentes. Difícil pensar em linguagens mais interessantes para se aprender neste momento.

Bons estudos!

Erick Eduardo Petrucelli

1 COMEÇANDO NA PLATAFORMA *WEB*

Olá! Está preparado para conhecer a plataforma *Web*? Então, vamos começar!

Nesta primeira lição, vamos começar conhecendo os conceitos de Internet e de *Web*, pois as três linguagens que vamos estudar surgiram nesses ambientes e compõem a base da plataforma, apesar de atualmente não serem utilizadas somente na criação de *sites*. Traremos um pouco da história e da evolução dessas tecnologias e poderemos refletir juntos sobre algumas tendências para os próximos anos. Ao final, apresentaremos as ferramentas para preparar nosso ambiente de desenvolvimento para os estudos e as atividades práticas.

Objetivos

Ao finalizar esta lição, você deverá ser capaz de:

- compreender os conceitos de Internet e de *Web* e qual a relação entre eles;
- conhecer a história desde o surgimento da Internet até os dias atuais;
- antecipar-se sobre as principais tendências da *Web* para os próximos anos;
- preparar seu computador como um ambiente de desenvolvimento *Web*.

1.1 Entendendo a Internet e a *Web*

Você já misturou Internet e *Web* na mesma frase pensando ser a mesma coisa? Não se sinta mal, é uma confusão muito comum. Mas acredite, não são a mesma coisa!

Internet é o que permite a conexão em rede entre computadores e dispositivos de todo o mundo, enquanto a *Web* é uma camada por cima da própria Internet, a qual define mecanismos padronizados para transmitir e apresentar as informações. Vamos entrar em explicações mais detalhadas para fortalecer sua compreensão sobre os dois conceitos.

O conceito de Internet

A Internet é a própria rede mundial de computadores. Na verdade, não é uma única rede, mas um conjunto de redes do mundo todo, que se conectam e se comunicam entre si.

Computadores conectados a uma rede cabeada (cabos pretos da imagem)

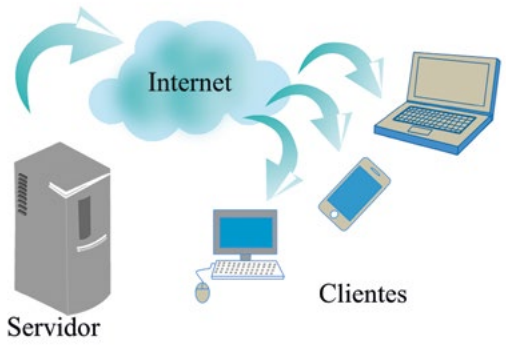
Você já viu uma rede de pesca? Todos aqueles fios se entrelaçando podem lembrar um pouco as redes de computadores (principalmente aquelas com fios bem desorganizados). As redes cabeadas continuam existindo, mas também temos as sem fios (como o famoso *Wi-Fi*). Enfim, a palavra rede fazia comparação com objetos do cotidiano e se tornou o padrão para expressar o conceito, mas nem faria tanto sentido atualmente.

Na Internet, formamos conceitualmente uma imensa teia de redes interligadas, mas é evidente que nem todos os computadores estão conectados fisicamente. Portanto, a Internet é um conjunto de protocolos (regras de comunicação) que permitem essa comunicação global entre redes diversas. Não é exagero dizer que, hoje em dia, é quase impossível se pensar em um computador ou em dispositivo móvel que não se conecte à Internet.

A arquitetura cliente-servidor

A Internet não surgiu do nada! Foi um advento a partir da conexão gradual de redes particulares, até que todas pudessem se comunicar. Em redes particulares (também chamadas de intranets), você podia se comunicar diretamente com cada máquina, usando, por exemplo, um número exclusivo para identificar cada uma delas e poder dizer qual gostaria de acessar.

Agora imagine, na Internet, se você tivesse de saber de cabeça milhares de dígitos de numeração de cada máquina específica desejada. Que loucura! Foi esse problema que inspirou a utilização da arquitetura cliente-servidor na Internet.



A arquitetura cliente-servidor se baseia em alguns computadores que assumem o papel de servidores, enquanto outros dispositivos exercem o papel de clientes. O servidor é um ponto focal da rede, ue recebe solicitações de informações, chamadas de requisições. Dependendo da informação, o servidor pode responder diretamente. Em outras situações, o servidor pode encaminhar a requisição para outros servidores, inclusive de outras redes.

Daí vem a ideia dos servidores *Domain Name System* (DNS), que são os sistemas de nomes de domínios. São servidores espalhados pelo mundo que possuem tabelas dos nomes de domínio já registrados. Por exemplo, ao acessar www.nteditora.com.br, suas requisições serão direcionadas a um servidor com um endereço 177.70.123.156. Seu dispositivo não precisa saber disso, ele só precisa saber o endereço de algum servidor DNS (os dispositivos clientes já possuem endereços de servidores DNS configurados por padrão).

Falando em endereços, aí está outra base da Internet: o protocolo TCP/IP. Na verdade, dois protocolos, concebidos para trabalharem juntos. O *Transmission Control Protocol* (TCP) é o protocolo de controle e transmissão em redes globais, o qual garante que os dados possam navegar em lotes sem se perderem. E o Internet *Protocol* (IP) é o protocolo dos endereços numéricos únicos. Cada servidor possui um IP exclusivo, para que possa ser encontrado pelos clientes. No exemplo do parágrafo anterior, 177.70.123.156 é o endereço IP exclusivo do servidor que hospeda o *site* da NT Editora, endereço esse que um servidor DNS vai informar ao seu navegador quando você tentar acessar.

Saiba mais

O tamanho do endereço IP tradicional (versão 4 do protocolo, conhecida como IPv4) não é suficiente para representar todos os dispositivos do mundo, por isso normalmente apenas servidores têm endereços fixos. Os clientes recebem endereços dinâmicos, cada vez que se conectam novamente à Internet. Para melhorar esse cenário, foi criado o IPv6, que teoricamente suporta um endereço exclusivo para cada dispositivo do mundo, mas a sua adoção vem ocorrendo lentamente desde 2000, por enquanto sem perspectivas de se tornar o único protocolo de endereços da Internet.

IP: *Internet Protocol*, termo em inglês para protocolo de Internet. É um protocolo fundamental da Internet, utilizado para o endereçamento durante a comunicação, por isso normalmente se utiliza o termo "endereço IP" para designar o número de identificação de um determinado dispositivo em uma rede.

A plataforma *Web*

Não entramos em muitos detalhes técnicos sobre a Internet, pois estariam mais relacionados aos conteúdos da área de redes. Vimos o suficiente para demonstrar como a estrutura de rede mundial da Internet é organizada. Mas para exibir o quê?

Podemos trafegar quaisquer dados pela Internet, o que potencialmente pode causar grande confusão e falta de padronização. Assim, foi necessário o estabelecimento de alguns padrões quanto ao formato de comunicação e de apresentação dos conteúdos que deveriam ser exibidos em qualquer dispositivo conectado. Vamos falar da história da *Web* no próximo tópico, mas o que interessa aqui é entender do que se trata.

Buscando resolver o problema de falta de padronização citado, surgiu a WorldWideWeb, termo que poderia ser traduzido como rede mundial de computadores. Atualmente, é mais comum chamar apenas de *Web*, para simplificar. Foi a partir desse nome que surgiu a famosa sigla WWW, existente no endereço de muitos *sites*, a qual já foi obrigatória e, mesmo não sendo mais, ainda permanece em grande parte deles.

Importante

Quando a *Web* foi concebida, originalmente era composta apenas pelo protocolo de transferência de hipertexto, o *Hypertext Transfer Protocol* (HTTP), e pela linguagem de marcação de hipertexto para a estruturação das páginas, o *Hypertext Markup Language* (HTML). Conforme a *Web* foi evoluindo, foi se transformando efetivamente em uma plataforma, composta por diversas tecnologias e linguagens.

Então, vamos dar uma olhada em um resumo das principais tecnologias e linguagens que compõem a plataforma *Web* nos dias atuais.

Navegadores

Os browsers, ou navegadores, não são oficialmente parte da plataforma *Web*. Muitos, inclusive, são iniciativas de empresas privadas. Mas não poderíamos deixar de citá-los, visto que são os programas que efetivamente permitem acessar a *Web*. Entre os mais relevantes, destaca-se o Google Chrome (utilizado por mais da metade dos usuários mundiais), o Mozilla Firefox (que vem perdendo usuários) e o Microsoft Edge (veio com o Windows 10 para substituir o limitado Internet Explorer). Em nichos mais específicos, temos o Apple Safari (dos sistemas macOS e iOS), o Opera (com uma parcela de mercado levemente abaixo do Firefox), além de outros específicos para dispositivos móveis.

HTTP

Já citamos esse protocolo, mas vale a pena reforçar que é a base da comunicação na *Web*. Através dele, as requisições aos servidores *Web* podem ser feitas de forma padronizada pelos navegadores. Os usuários finais dificilmente vão perceber o HTTP ali, mas qualquer comunicação que um navegador precise fazer passa por ele. Antigamente, os navegadores exibiam o protocolo no início dos endereços dos *sites*, mas acabou sendo removido da maioria. Atualmente, os navegadores exibem apenas o protocolo dos endereços que usam o HTTPS, uma variante do protocolo original, mas com mecanismos de criptografia para manter os dados seguros, que dificulta possíveis ataques baseados em visualizar as informações trafegadas.

HTML

Também já citamos essa linguagem de marcação, a base da apresentação dos conteúdos na *Web*. Vale enfatizar que não é uma linguagem de programação, seu foco é estruturar as páginas de forma padronizada, mas sem conceitos de lógica de programação envolvidos. Evidentemente, o HTML é um dos focos deste material, então teremos oportunidade de falar muito mais sobre ele posteriormente.

HTTP: *Hypertext Transfer Protocol*, termo em inglês para protocolo de transferência de hipertexto. É um protocolo fundamental da *Web*, estabelecendo as regras de comunicação entre um navegador e um servidor *Web*. É um protocolo de nível de aplicação, utilizando por baixo o protocolo TCP para a comunicação efetiva.

HTML: *Hypertext Markup Language*, termo em inglês para linguagem de marcação de hipertexto. É uma linguagem com foco na estruturação dos documentos a serem transmitidos na *Web*, sendo a base estrutural de todas as páginas dos *sites*.

Browser: termo em inglês para navegador, palavra comumente utilizada para *design*ar os *softwares* que têm a função principal de permitir a navegação na Web, apresentando as páginas.

CSS

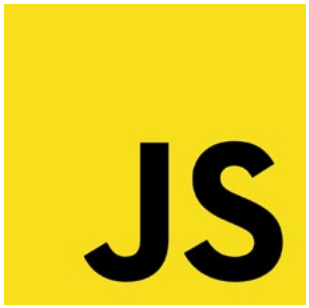
O *Cascading Style Sheets* (CSS) é um formato de arquivo de estilos, ou seja, permite definir visualmente a apresentação dos conteúdos estruturados pelo HTML em questões como cores, fontes, tamanhos, posicionamentos etc. Começou despretensiosamente permitindo apenas algumas definições, mas evoluiu tanto ao longo dos anos, a ponto de incorporar até questões relacionadas a animações em nossas páginas. Também teremos bastante espaço para falar de CSS no restante deste material.

CSS: *Cascading Style Sheets*, termo em inglês para folhas de estilo em cascata. É um formato de arquivo de estilos, utilizado em conjunto com o HTML para a definição do visual das páginas Web.

JavaScript

Como já dissemos, a *Web* surgiu como uma plataforma estática, ou seja, não havia muito o que fazer além de navegar para outras páginas. A iniciativa de transformar a *Web* em algo dinâmico fez surgir o *JavaScript*, a única tecnologia da plataforma que é efetivamente uma linguagem de programação, ou seja, que permite utilizar qualquer conceito de lógica de programação para criar códigos que façam operações dinâmicas em nossas páginas.

Graças ao *JavaScript*, podemos efetivamente criar interações empolgantes entre nossos usuários e nossas páginas. Então, ainda falaremos muito sobre isso no restante do material.



ECMAScript: nome oficial da especificação da linguagem comercialmente conhecida como JavaScript, originalmente padronizada em 1997 e, desde 2015, recebendo atualizações anuais.

Saiba mais

O nome JavaScript é, na verdade, marca registrada. Atualmente, pertence à Oracle. A linguagem foi padronizada internacionalmente em 1997, em uma especificação denominada ECMAScript. Na prática, todos continuam chamando de JavaScript.

Web APIs

No mundo da programação de computadores, o conceito de *Application Programming Interface* (API) se refere a mecanismos de comunicação entre linguagens e sistemas diversos. No cenário da *Web*, foram surgindo várias APIs para que o *JavaScript* pudesse acessar e manipular os conteúdos. Entre as APIs da plataforma *Web*, a mais importante é o Document Object Model (DOM) que estabelece um modelo de manipulação dos documentos HTML e seus estilos, a partir do *JavaScript*.

Atualmente, temos outras APIs, como a *Web*GL, para manipulação de gráficos tridimensionais, e a File API, para manipulação de arquivos em disco diretamente pelo navegador, entre outras. É válido ressaltar que não abordaremos todas as APIs e vamos, portanto, nos ater ao DOM, essencial para manipulação das páginas.

API: *Application Programming Interface*, termo em inglês para interface de programação de aplicações. É um termo genérico utilizado para descrever o conjunto de rotinas e padrões que um *software* oferece para utilização de suas funções por outros *softwares*. No cenário da plataforma Web, são oferecidas as Web- APIs para interação do código das páginas com o navegador e com recursos do dispositivo.

Exercitando

Das opções apresentadas abaixo, qual delas indica um protocolo da WorldWideWeb,?

a) HTTP.

b) TCP.

c) IP.

d) HTML.

Comentário: se você pensou na alternativa "a", parabéns! HTTP é o protocolo de comunicação da *Web*. TCP e IP são protocolos mais básicos, da própria Internet. E HTML não é um protocolo, é a linguagem de marcação das páginas *Web*.

DOM: *Document Object Model*, termo em inglês para modelo de objetos do documento. Estabelece o modelo de representação em memória dos documentos HTML e seus estilos, incluindo a API para manipulação destes a partir do JavaScript (ou, teoricamente, qualquer outra linguagem que pudesse ser executada em um navegador).

1.2 A breve história da Internet e da *Web*

Já conceituamos a relação entre Internet e *Web*, agora vale a pena abordar um pouco da história que nos trouxe até este ponto.

Não chamamos de "breve" ao acaso. A Internet ainda não completou quarenta anos, e a *Web* está na juventude, chegando aos trinta. Pode parecer um tempo razoável, mas significa que aproximadamente duas gerações humanas estão em contato com essas tecnologias desde cedo. Se considerarmos o tempo de existência de outras áreas de conhecimento da humanidade, é algo bem novo!

A criação da Internet

O dia 4 de outubro de 1957 iniciou-se com uma mudança definitiva no mundo: foi quando a União Soviética lançou com sucesso o primeiro satélite para a órbita da Terra. E não estamos falando apenas da tensão crescente que a Guerra Fria estava causando.

Esse fato, como era de se esperar, causou muita preocupação na outra potência da época, os Estados Unidos da América. Sentindo-se ameaçados, decidiram aprofundar seus investimentos em tecnologia, entre os quais na criação da Agência de Projetos Avançados de Pesquisa (ARPA), do Departamento de Defesa. Provavelmente o projeto mais famoso da agência (e certamente o mais usado) foi a criação da Internet. Vamos descobrir como?

Durante os anos de 1960, a agência estudou formas de conectar computadores em rede, com fins militares. Em 1969, ela chegou à primeira rede funcional, denominada Arpanet, que então serviu de inspiração para outras entidades. Por exemplo, para a rede britânica Janet, que conectou universidades do Reino Unido, e para a rede americana pública da CompuServe, possivelmente a primeira empresa privada a investir na área (foi ela também que criou, muitos anos depois, o famoso formato GIF de imagens animadas).

Saiba mais

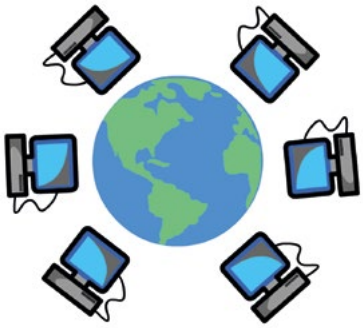
Inicialmente, a ARPA não estava tentando criar uma rede mundial de computadores, mas sim conectar vários computadores para armazenamento e recuperação rápida de informações militares. Curiosamente, não foi rápido, pois esse período durou de 1960, quando o primeiro artigo foi publicado, até 1969, quando a rede com apenas quatro computadores funcionou!

O surgimento de várias redes de computadores conectados entre si foi algo muito legal e promissor, mas ainda havia um problema sério. Faltavam protocolos (ou seja, regras de comunicação unificadas) para que redes diferentes pudessem "conversar" entre si.

Trabalhando em uma solução para isso, o engenheiro elétrico Robert Kahn da Arpa se juntou ao matemático e cientista da informação Vinton Cerf, da Universidade de Stanford. A pesquisa deles originou o protocolo TCP/IP, ue estabeleceu regras de comunicação baseadas em computadores atuando nas redes como servidores e outros como clientes.

A própria Arpa financiou o desenvolvimento do primeiro software que aplicou o protocolo e, em 1977, conseguiu realizar uma demonstração com três redes diferentes se comunicando. Em 1981, a especificação oficial do protocolo foi finalizada e, em 1982, todos os padrões anteriores da Arpanet foram migrados para o TCP/IP nas redes existentes. A partir disso, novas redes começaram a surgir, conectando-se às já existentes. Pronto, a Internet como conhecemos havia chegado e todas as redes do mundo poderiam se conectar!

Software: termo em inglês sem tradução direta, utilizado para *design*ar genericamente conjuntos de instruções que definem como um computador ou dispositivo eletrônico deve funcionar.



A criação da *Web*

Redes de computadores do mundo todo, interconectadas, formando uma gigantesca rede mundial. Se máquinas distantes finalmente poderiam "conversar", iriam "falar" o quê?

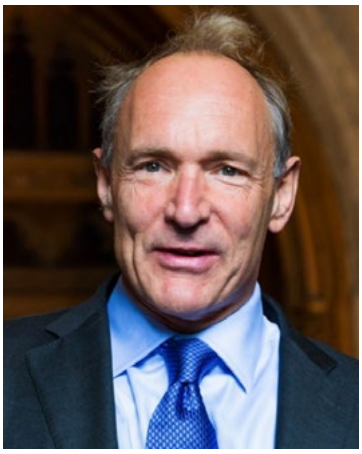
Importante

Tome os humanos como exemplo: conseguimos nos comunicar usando um mesmo canal de comunicação (normalmente a fala, emitida pela boca); mas precisamos de outros padrões para que essa comunicação faça sentido (o idioma escolhido, por exemplo). Da mesma forma, o protocolo TCP/IP definiu que computadores poderiam se comunicar, mas foi a *Web* que trouxe o protocolo HTTP e a linguagem HTML, padronizando a linguagem que usariam.

A intenção de criar padrões para a Internet é quase tão antiga quanto ela. Várias propostas de linguagens e protocolos surgiram durante os anos e, de alguma forma, inspiraram o trabalho do físico e cientista da computação Tim Berners-Lee, iniciado em 1980.

Na época, ele trabalhava na Organização Europeia para a Investigação Nuclear (CERN) e desenvolveu um projeto de associações entre informações, denominado ENQUIRE. A ideia era armazenar cada informação em uma "página", que deveria sempre ter "ligação" com outra página já existente. Após alguns anos fora, voltou ao CERN e começou a pensar sobre a apresentação das informações, muitas plataformas diferentes eram usadas no mundo e as informações das páginas precisariam ser exibidas da mesma maneira em todas.

Tim Berners-Lee, o criador da Web



Berners-Lee chegou a escrever, em 1989, uma proposta para um grande repositório de hiperligações entre páginas. Curiosamente, teve pouca repercussão. Mesmo assim, seu chefe, Mike Sendall, o motivou a desenvolver o sistema. Após considerar vários nomes, decidiu por WorldWideWeb, ou uma rede de alcance mundial, em tradução literal.

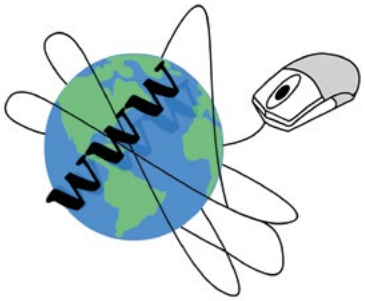
Saiba mais

Alguns nomes originalmente imaginados por Tim Berners-Lee poderiam soar bem egocêntricos e foram descartados, como *The Information Mine* (a abreviação seria TIM, o seu próprio prenome) ou *Mine of Information* (a abreviação seria MOI, que significa "eu", em francês). No fim, o nome WorldWideWeb, foi ótimo por ser abrangente e por soar bem. Mas, do ponto de vista técnico, foi uma escolha confusa, afinal, a verdadeira rede mundial de computadores é a própria Internet, não a *Web*.

As coisas começaram a avançar em 1990, quando seu colega de trabalho Robert Cailliau se empolgou com o projeto, reescreveu a proposta e conseguiu recursos no próprio CERN, tentou também buscar investimentos em conferências da área, porém, sem sucesso. Ainda assim, Berners-Lee terminou as ferramentas fundamentais da *Web*: o protocolo (HTTP), a linguagem para criar as páginas (HTML), o primeiro servidor (CERN httpd) e o primeiro navegador (simplesmente WorldWideWeb).

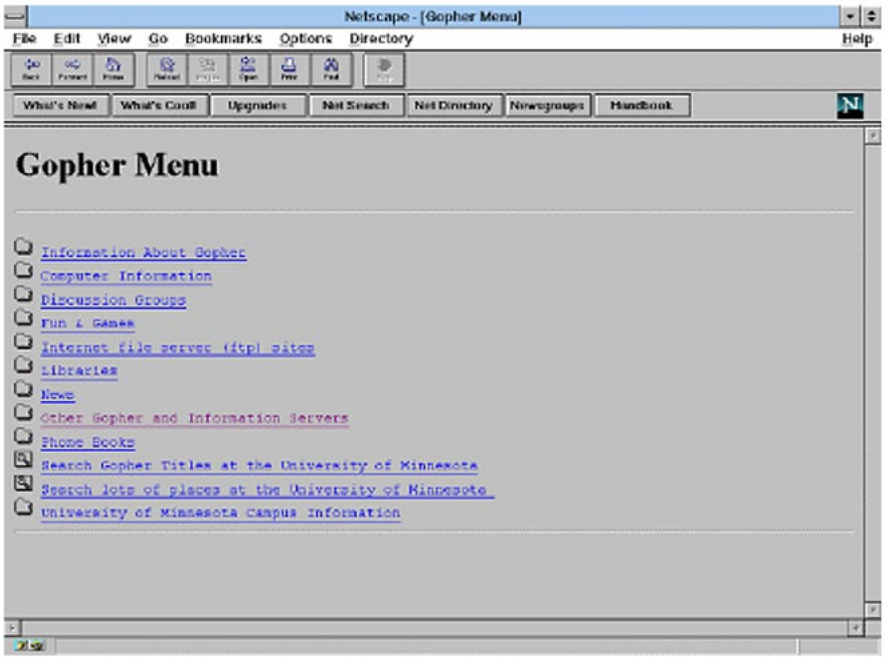
Saiba mais

A *Web* sempre foi pensada cautelosamente para manter a compatibilidade ao longo dos anos, mesmo com todas as evoluções. Embora os navegadores não tenham respeitado isso sempre (veremos mais sobre isso adiante), as tecnologias em si evoluíram sem estragar o que já existia. Quer um exemplo? As primeiras páginas *Web* criadas pelo próprio Tim Berners-Lee ainda estão disponíveis e você consegue acessá-las da mesma forma que faria no início dos anos de 1990, veja: https://www.w3.org/History/19921103-hypertext/hypertext/WWW/TheProject.html.



A tecnologia até caminhava bem, mas a adesão não. Ocorreu que, em 1991, cresceu rapidamente um sistema alternativo de recuperação de informações, o *Gopher*, criado pela Universidade de Minnesota, que fornecia um método de acessar *links* para arquivos em outros computadores conectados. Muitas instituições usavam o *Gopher* naquele período, então, os primeiros anos da *Web* foram bem discretos.

Servidor Gopher



Até que, em fevereiro de 1993, foi anunciado que seriam cobradas taxas de licenciamento para implementações de servidores *Gopher*. De imediato, muitas organizações se assustaram, começaram a buscar alternativas e acabaram se deparando com a *Web*. Aproveitando o novo interesse repentino, em 13 de abril de 1993, o CERN liberou o código-fonte da *Web* para o domínio público, dessa forma , qualquer pessoa poderia usar ou mesmo construir qualquer coisa com a plataforma, sem custos. Isso tranquilizou os possíveis novos adeptos e muitas organizações adotaram.

Você já tinha parado para pensar em como é maravilhoso poder usar as páginas publicadas na *Web*, e até mesmo criar as suas próprias, sem pagar taxas pelas tecnologias envolvidas? Agora você sabe como se tornou possível!

Pouco tempo depois, o Centro Nacional de Aplicações de Supercomputação (NCSA), órgão americano inserido na Universidade de Illinois, lançou um navegador *Web* que também era cliente *Gopher*, chamado *Mosaic*. Satisfazendo o novo público da *Web* junto ao público que ainda estava no *Gopher*, este navegador se tornou popular e, com ele, a *Web*.

Como a *Web* havia se tornado domínio público, Berners-Lee não queria que ela se perdesse em implementações variadas e conflitantes ou que não evoluísse. Então, em 1994, fundou o WorldWideWeb, *Consortium* (conhecido pela sigla W3C), com apoio do CERN, da Darpa (como passou a se chamar a Arpa desde 1996) e da Comissão Europeia. Com uma organização internacional sem fins lucrativos, começaram a padronizar as tecnologias.

W3C: *WorldWideWeb, Consortium*, principal organização de padronização da *Web*, formada por uma equipe dedicada e agregando centenas de membros de empresas da área, órgãos governamentais e outras organizações independentes.

Dessa forma, a *Web* se consolidou, livre, padronizada, universal - mas calma não é tão simples. A adoção das especificações não era forçada pelo W3C, uma vez que eram apenas recomendações. Na verdade, ainda é assim: os fabricantes devem seguir o W3C apenas se quiserem rotular seus produtos como compatíveis. Por muito tempo, não foi algo considerado valioso comercialmente, já que usuários e desenvolvedores não tinham muita noção da importância. Esse desinteresse pelos padrões até causou uma guerra comercial na área! Veremos no próximo tópico.

Exercitando

Considerando a história da criação da WorldWideWeb, quem projetou e desenvolveu as primeiras versões do protocolo HTTP e da linguagem de programação HTML?

a) Mike Sendall.

b) Vinton Cerf.

c) Robert Kahn.

d) Tim Berners-Lee.

Comentário: se você pensou na alternativa "d", parabéns! Tim Berners-Lee é considerado o criador da Web. Robert Kahn e Vinton Cerf foram autores do protocolo TCP/IP, uma das bases da Internet. Mike Sendall era chefe do Tim Berners-Lee e incentivou que ele continuasse o trabalho, mas não participou diretamente da criação.

1.3 A evolução das linguagens *Web*

Espero que você tenha gostado de conhecer a história da Internet e da *Web*. Por mais que seja passado, é muito bom entender como as coisas se tornaram o que são. Mas a criação e a popularização da *Web* e do W3C foi só o começo. Agora, vamos abordar o lado comercial da história: como fabricantes de navegadores influenciaram a área.

A guerra dos navegadores

No final de 1993, o mundo acordou para a Internet, graças à *Web*! Mas, não era muito comum ter acesso a ela. Poucos tinham computadores e a conexão era cara e lenta. Além disso, para ter acesso ao *Mosaic*, alguém precisaria copiá-lo para você. Afinal, não seria possível navegar na Internet para baixar o navegador sem um navegador.

Tentando popularizar os navegadores e espalhá-los pelo mundo, muitos projetos de universidades e de empresas surgiram. Um exemplo foi o navegador Opera, criado para fins de pesquisa pela empresa norueguesa de comunicações Telenor, em 1994.

Mas é claro que interesses comerciais surgiriam. Ainda naquele ano, Marc Andreessen e Jim Clark, membros do NCSA, saíram e fundaram, em 1994, a Netscape Corporation, lançando o Netscape Navigator, primeiro navegador comercial a conquistar grande mercado.

A criação desse navegador chamou atenção da gigante Microsoft, líder nos *softwares* para escritórios (com o pacote Office) e nos sistemas operacionais (com o MS-DOS e o Windows). Ela comprou a licença do Mosaic e usou como base para o primeiro Internet Explorer, em agosto de 1995. Isso iniciou uma competição acirrada entre Netscape e Microsoft, a Guerra dos Navegadores.

O foco da Netscape era atrair desenvolvedores, e o passo mais importante foi a criação de uma nova linguagem de programação para o navegador, chamada Mocha, depois renomeada para LiveScript e, finalmente, *JavaScript*, lançada em 1995. O nome e deu graças a um acordo comercial com a Sun Microsystems, criadora da linguagem Java, que não funcionava em navegadores, mas estava em alta no mercado de programação e, portanto, chamaria a atenção do público.

A abordagem da Microsoft, inicialmente, era criar recursos novos apenas para o HTML, mas inevitavelmente teve de reconhecer o poder de uma linguagem de programação xecutada diretamente no navegador. Basicamente, copiaram o concorrente e lançaram sua própria versão de *JavaScript*, com o nome de JScript, em agosto de 1996, uma vez que o nome original era marca registrada da Netscape.

Em meio à guerra, vale citar que a empresa Opera foi fundada em 1995, para cuidar especificamente do navegador. Manteve presença pequena, mas constante, tentando suportar padrões *Web* da melhor maneira possível naqueles tempos.

Importante

Durante a Guerra dos Navegadores, Microsoft e Netscape se concentraram tanto em novos recursos a uma velocidade estonteante (o primeiro esboço da linguagem Mocha ficou pronto em 10 dias!) que deixaram de corrigir problemas no que já tinham. Além disso, na corrida por criar recursos proprietários e alternativas que competiam diretamente com os recursos do outro navegador, cada um foi se tornando mais incompatível com os códigos criados pelo outro. Chegou ao ponto de os desenvolvedores terem de escolher entre fazer o mesmo trabalho duas vezes, com códigos diferentes ou, pior ainda, oferecer suporte a apenas um navegador e impedir que outros usassem seus *sites*!

Como já comentamos, as iniciativas de padronização do W3C não eram respeitadas, por não serem consideradas vantajosas comercialmente. Demorou até que a insatisfação dos desenvolvedores agitasse o mercado e começasse a mudar as coisas. Mas, naquela altura, a Guerra dos Navegadores já havia se encerrado. A Microsoft venceu "na marra", embutindo o Internet Explorer no seu sistema Windows. Por ter sido o sistema operacional mais usado do mundo por muitos anos, automaticamente ganhou quase todos os usuários. Curioso, não? Em uma guerra por inovação tecnológica, venceu a melhor jogada comercial.

Os padrões *Web* engatinhavam

Nos primeiros anos de existência, o W3C publicou especificações do HTML e do PNG (formato de imagens que se tornou muito popular na *Web*, anos depois). Era o começo da Guerra dos Navegadores e ficava claro o desdenho dos fabricantes para com os padrões.

Um exemplo famoso: Marc Andreessen passou alguns dias debatendo com o Tim Berners-Lee e com outros desenvolvedores e pesquisadores, em uma lista de discussão do W3C, sobre a necessidade de suportar imagens nas páginas. Sim, a primeira versão do HTML não previa imagens! Todos concordaram que era desejado, mas não se chegava a um consenso sobre como fazer. Semanas depois,

Andreessen voltou apenas para registrar que já haviam implementado, para a próxima versão do Navigator, da forma como ele próprio queria. De fato, a versão 2 do HTML, de 1995, foi uma corrida do W3C atrás do que os fabricantes estavam fazendo, para tentar oficializar tudo, obviamente sem sucesso.

Nos primeiros anos, a especificação HTML evoluiu rápido para tentar acomodar tudo o que os navegadores estavam criando. A proposta do HTML 3 foi promissora! Na verdade, virou recomendação em 1997, como HTML 3.2, por causa de alguns ajustes que adicionaram à especificação original. Nunca chegou a ser totalmente suportada nos navegadores.

Em 1998, o mercado de navegadores era dominado pelo Internet Explorer 4, com o Netscape Navigator 4 ainda mantendo boa base de usuários. Então, a Microsoft lançou uma prévia do Internet Explorer 5, implementando um novo modelo de HTML dinâmico e proprietário, ignorando completamente os padrões recentes. Com isso, os desenvolvedores precisariam aprender mais uma nova maneira específica de interagir com as páginas.

A história foi parecida com o CSS. Os problemas de estilização do conteúdo começaram a ficar evidentes, com cada navegador criando suas próprias alternativas. Inicialmente, inventavam recursos proprietários para o próprio HTML, porém voltados à estilização, que gerou grande confusão sobre o real objetivo da linguagem de marcação.

Em 1996, o CSS 1 foi completado, mas nenhum navegador o suportava. Demorou ao menos até 1999 para que as novas versões dos navegadores finalmente estivessem perto de suportar totalmente o CSS 1 (na verdade, ainda existiam falhas nas implementações, com cada navegador compreendendo de formas diferentes algumas partes da especificação). Mas, em 1998, o W3C já havia trazido a especificação do CSS 2. Apenas em 2000 os navegadores começaram a trazer partes da nova especificação, mas continuaram criando suas próprias interpretações para muitos dos recursos.

A recomendação HTML 4 foi publicada no final de 1999, com uma revisão denominada HTML 4.01 saindo em 2001. Esse foi o ano em que o famoso Internet Explorer 6 chegaria ao mercado, embutido no Windows XP, rapidamente dominando de vez a área. Essa versão do navegador da Microsoft se tornou um marco em ignorar padrões, mas vamos tentar fazer justiça: aquela era uma época em que os navegadores não recebiam novas versões rapidamente, muitas vezes emoravam anos para evoluírem, tornando impossível realmente seguirem novas recomendações lançadas tão rapidamente pelo W3C.

Saiba mais

Ainda que o Netscape Navigator tenha continuado no mercado por alguns anos, o fato é que, no início da década de 2000, o Internet Explorer já havia abocanhado quase todo o mercado. Podemos dizer que nenhuma das duas empresas dava muita importância para os padrões do W3C, mas o novo monopólio da Microsoft foi a parte mais prejudicial, uma vez que eles realmente não viam nenhuma vantagem comercial em reconstruir tudo o que o navegador deles já fazia apenas por conformidade com especificações.

Em paralelo, temos a complicada evolução do *JavaScript*. Já foi dito que, originalmente criado pela Netscape, ele gerou a réplica denominada JScript, da Microsoft. Tentando defender sua criação, a Netscape submeteu a linguagem para padronização através da organização internacional europeia ECMA (*European Computer Manufacturers Association*). Já que a palavra Java era marca registrada da Sun Microsystems, e *JavaScript* só poderia ser usado pela própria Netscape, a decisão mais simples foi renomear a nova especificação para ECMAScript, lançada em 1997. Observe que o W3C não tem relação com o *JavaScript*.

Enquanto a ECMA trabalhou pesado nas versões 2 e 3 da especificação (de 1998 a 1999, respectivamente), claramente a Microsoft não estava muito preocupada. Afinal, se nem o W3C tinha força suficiente para que os padrões HTML e CSS fossem adotados corretamente, imagina outra entidade que havia acabado de entrar no jogo.

A ascensão dos padrões *Web*

Gradativamente, os desenvolvedores foram se enchendo de complicações e começaram a formar grupos paralelos para ajudar a mostrar para o mundo que os padrões *Web* eram importantes e que não deveria usar um navegador que não os seguisse.

Quando a Netscape foi se aproximando do fim, parte dos desenvolvedores saíram e levaram muito da propriedade intelectual da empresa, fundando a Mozilla. Nos primeiros anos do navegador Firefox, a meta era suportar totalmente os padrões *Web* e convencer o máximo de público, o que seria muito vantajoso. Não foi rápido, mas o Firefox foi, aos poucos, crescendo em usuários e forçando a Microsoft a mudar seus conceitos.

Quando finalmente a Google resolveu entrar nesse mercado com o Chrome, em 2008, começamos a ver um fortalecimento ainda maior dos navegadores seguindo o W3C. Durante esses tempos confusos da década de 2000, o HTML 4.01 nunca vingou plenamente, o CSS 2.1 ficou anos tentando se estabelecer e o ECMAScript 4 acabou completamente abandonado. Sendo assim, os novos HTML 5, CSS 3 e ECMAScript 5 foram todos iniciados entre 2008 e 2009 e não demoraram tanto tempo para se tornarem os padrões realmente aplicados pelos navegadores. O segredo é que os próprios fabricantes de navegadores (inclusive a Microsoft) começaram a participar ativamente da evolução dos padrões, ou seja, ao invés de correrem para seguir recomendações, passaram a ajudar a criá-las.

Exercitando

A ECMA foi responsável por um fato de grande importância na evolução e no desenvolvimento da *Web*. Que fato foi esse?

a) Desenvolvimento do HTML.

b) Melhorias na linguagem CSS.

c) Criação de navegadores comerciais.

d) Padronização da linguagem *JavaScript*.

Comentário: se você pensou na alternativa "d", parabéns! A ECMA é a responsável pela especificação da linguagem, denominada ECMAScript, iniciada em 1997, que padronizou a linguagem *JavaScript*.

1.4 Tendências no desenvolvimento *Web*

Acabamos de passar por uma bela história da Internet aos padrões *Web*. Agora, vamos refletir sobre tendências para o futuro.

Já olhamos para o passado e, a partir de agora, exploraremos o presente durante todo o conteúdo restante. Mas como se preparar para o futuro em uma área que evolui tão rápido? A resposta mais coerente é estar atento às tendências que estão surgindo, confrontá-las com cada um dos conteúdos que estiver estudando e, assim, poder escolher a direção que prefere seguir.

Importante

Pode se acalmar: grande parte deste material é parte do futuro! As três linguagens que fundamentam a *Web* sofrem atualizações, mas você consegue imaginar um mundo sem Internet e sem *Web* nos próximos anos? Não faz sentido! Você realmente continuará aprendendo para se manter atualizado, mas as coisas não vão mudar completamente. É impossível pensar que o HTML, o CSS ou o JavaScript deixem de existir do subitamente.

Nos tópicos a seguir, veremos algumas das tendências perceptíveis para a *Web*. São coisas que já estão acontecendo, mas, ao mesmo tempo, são boas apostas para o futuro.

***Desktop*:** termo em inglês sem tradução direta, literalmente seria mesa de trabalho. Normalmente utilizado para *design*ar dispositivos e aplicações concebidos para computadores pessoais tradicionais, fixos em seus ambientes, bem como dispositivos que evoluíram para suportar recursos equivalentes, como laptops/notebooks.

Modernização do *design* e da interação

Pense na aparência das interfaces que você tem visto, em *sites*, aplicativos para dispositivos móveis e até aplicações desktop. É nítido que a simplicidade cresceu, tendência que chamamos de minimalismo no mundo do *design*.

Os usuários ficaram cansados de figuras animadas doidas, textos passeando pela tela, anúncios intermitentes e invasivos, conteúdos exagerados, imagens cheias de contrastes se sobrepondo, excessos de cores e gradientes, nada combinando entre si. Esse excesso de equívocos foi a marca visual do início da *Web* e durou tempo demais! Na verdade, é possível ver até hoje *sites* com visuais que parecem ter sido extraídos da *Web* de 1997.

Por isso, interfaces com *design* moderado são cada vez mais apreciadas e, inclusive, se tornam mais produtivas para o usuário. Não se trata de eliminar todos os efeitos; pelo contrário, se trata de escolher com cautela e utilizar cada recurso com uma razão coerente.

Por mais de cinco anos, o *flat design* se popularizou, principalmente por causa da variante chamada *metro design*, criada pela Microsoft para o Windows 8, que oferece uma experiência bem limpa, mas muito simplista e repetidamente utilizada, a ponto de enjoar os usuários.

Exemplo de flat design

Nos últimos anos, vem ocorrendo uma gradual aos gradientes, sombreamentos e cores mais saturadas. O estilo *material design*, criado pela Google para o Android e depois para vários de seus serviços *Web*, como Inbox, Gmail e YouTube, vem se espalhando e influenciando o *design* adotado por muitos *sites*. É provável que as interfaces continuem evoluindo nesse equilíbrio entre o minimalismo e os recursos visuais empolgantes.

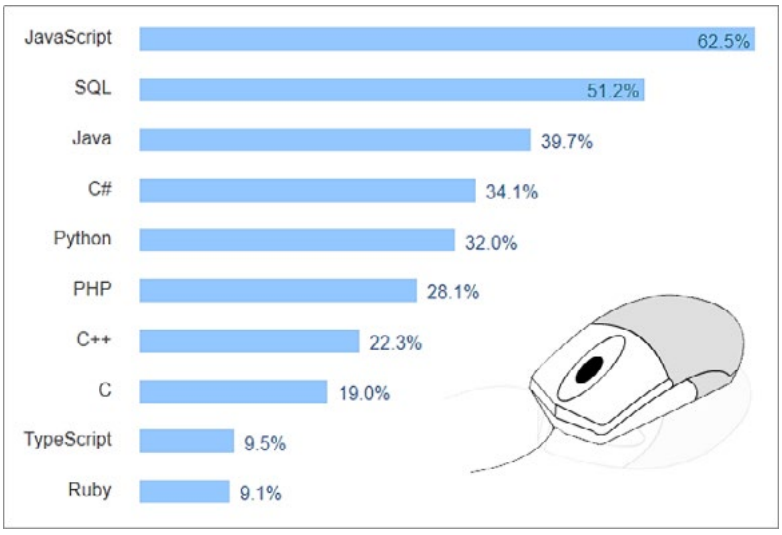
Também se observa um crescimento rápido das tendências de Motion UI (movimentos de interface de usuário), através da utilização de transições e animações sutis, leves e fluidas, permitindo adicionar mais personalidade aos visuais minimalistas. Unindo esta tendência a recursos mais recentes do CSS3, podemos direcionar a atenção dos nossos usuários para os elementos mais importantes, conforme navegam e interagem com a interface.

Existem várias maneiras empolgantes de dar vida a um *site* minimalista, e o CSS evoluiu na última década para incrementar as opções e facilitar as coisas. Tornar a presença na *Web* mais agradável e memorável para os usuários é uma aposta definitiva para o futuro. Não vamos nos aprofundar tanto em questões de *design*, mas recomendamos que você fique de olho nisso para extrair o melhor de seu HTML e CSS: a mais pura arte digital!

Muito JavaScript, bibliotecas e *frameworks*

O *JavaScript* chegou a um ponto que, de certa forma, domina o mundo da programação. E, já que estamos falando em tendências, não é difícil apostar que a linguagem permanecerá no topo das pesquisas de uso e com grande relevância no mercado, por muito tempo.

Dez linguagens de programação mais populares do mundo em 2017



Para apoiar o desenvolvimento e deixar as coisas mais produtivas, o mercado também tem abraçado o uso de bibliotecas e dos chamados *frameworks* (a tradução literal seria "estrutura de trabalho", o que não ajuda muito no entendimento). Ambos são conjuntos de códigos especializados e

flexíveis, criados e testados por vários desenvolvedores, às vezes independentes, às vezes com empresas comerciais por trás. Podemos entender *framework* como um grupo de bibliotecas, ou mesmo uma única biblioteca grande que força determinada estrutura para o projeto, enquanto bibliotecas simples são mais independentes.

Há várias opções para o *JavaScript*, algo que virou costume há mais de dez anos com a famosa biblioteca jQuery. Também há *frameworks* CSS, que trazem estilos para *layouts* elaborados e padrões visuais consistentes, tornando mais produtiva a criação das interfaces.

Saiba mais

Na área do JavaScript, vemos competição acirrada entre o Angular (da Google) e o React (do Facebook), com o Vue (de desenvolvedores independentes) crescendo como outra boa opção. Na área do CSS, destacamos o famoso Bootstrap (do Twitter), o Foundation (da agência de *design* Zurb) e o mais recente Bulma (de desenvolvedores independentes), mas há outros menos conhecidos que podem se destacar.

É viável afirmar que bibliotecas e *frameworks* são tendências para o futuro, contudo, é impossível acertar quais vão continuar crescendo e quais vão cair em desuso. Uma piada recorrente entre os desenvolvedores é que você não consegue contar os dias desde a última biblioteca lançada, todo dia tem uma nova. Como se manter atualizado? Acompanhando tendências e estudando um pouco do que começa a despontar. Mas, não se assuste, ninguém aprende todos. E não precisa.

Mais *bots* com inteligência artificial

Talvez você tenha percebido que os *chatbots* (termo que poderia ser traduzido como "robôs de bate-papo"), baseados em inteligência artificial, estão crescendo em popularidade desde 2016.

De forma resumida, são programas desenvolvidos para simular interação de bate-papo com humanos, da forma mais natural possível. Não são robôs de verdade, fisicamente falando, são códigos que simulam a compreensão e a linguagem humana, por isso estão na área de inteligência artificial. Começamos a vê-los em páginas de empresas no Facebook, WhatsApp e até em alguns *sites*. E pode esperar para se deparar cada vez mais com eles!

É uma boa aposta considerar que os *chatbots* continuarão a evoluir, ajudando a agilizar a comunicação *online*, simplificando (e barateando) o atendimento a clientes, possivelmente até aumentando vendas e fidelização. O trabalho em *call centers* pode deixar de existir em alguns anos, não acha? Para estar pronto para este futuro, é interessante que as empresas saibam utilizar tais tecnologias em seus produtos e serviços.

A "Internet das coisas" cada vez em mais coisas

Também conhecida pela sigla IoT (do inglês Internet *of Things*), essa é uma tendência de interligar a Internet às coisas tradicionalmente desconectadas. Podemos citar como exemplo relógios e eletrodomésticos, assim como carros e equipamentos industriais. Na verdade, essa ideia se relaciona a qualquer coisa que possa se beneficiar de um controle a distância ou de coleta de dados *online*.

São tantas possibilidades: sua TV transmitindo e sendo controlada pela Internet; sua geladeira avisando que está esvaziando e encomendando o que falta em mercados *online*; seu carro em contato com o fabricante, otimizando a operação e avisando sobre possíveis problemas mecânicos; indústrias produzindo em tempo real quando um pedido entra, sem estoque excessivo ou atrasos. Poderíamos passar horas listando ideias!

As soluções de IoT entram na vida das pessoas gradualmente e estamos os aproximando de uma era de dispositivos inteligentes. Muitos deles, além de se comunicarem pela Internet, podem precisar da *Web* como sua interface, o que nos direciona ao assunto que veremos agora.

Web como interface multiplataformas

Já contamos a história: a *Web* surgiu padronizando a apresentação das informações na Internet, com linguagens criadas para execução em navegadores, mas, ao longo dos anos, elas foram se expandindo e migrando para outras plataformas, bem como outras plataformas foram apreciando a evolução e as incorporando entre suas opções para o desenvolvedor. Veja a seguir exemplos de plataformas que estão abraçando a *Web* de alguma forma.

- Com o Node.js, a linguagem JavaScript chegou ao mundo dos servidores *Web*, permitindo programar essa camada com a mesma linguagem que usamos nas interfaces.
- Plataformas como Unreal Engine 4 e Unity permitem criar jogos 2D e 3D de nível profissional usando JavaScript como linguagem de programação.
- Desde o Windows 8 (e ampliando no Windows 10), podemos criar aplicativos completos com HTML, CSS e JavaScript para esses sistemas operacionais.
- Android e iOS oferecem suas próprias linguagens nativas. Mas projetos como Apache Cordova e PhoneGap permitem criar aplicações híbridas, feitas com HTML, CSS e JavaScript, mas que rodam nos dispositivos por cima de uma leve camada nativa.
- Alguns "torcem o nariz" para aplicações híbridas (eventualmente, o desempenho delas pode ser ruim), mas projetos como React Native, NativeScript e Weex permitem usar JavaScript para construir aplicações que viram aplicativos realmente nativos no final.
- As tradicionais aplicações *desktop* obrigam muitos desenvolvedores a aprender linguagens mais antigas - algo que é desmotivador. Mas projetos como Electron, NW.js e Vuido permitem a mesma coisa explicada acima, mas para sistemas Windows, macOS e Linux.
- Até mesmo plataformas comuns para estudos com IoT, como Arduino e Raspberry Pi, podem ser programadas com extensões que permitem o controle por JavaScript.

Depois disso tudo, alguma dúvida de que a *Web* está se tornando multiplataformas, quase universal? Onde não está integralmente, está ao menos através do *JavaScript*. Chegamos à tendência mais importante: aprender a programar para *Web* e abrir todas as outras portas!

Exercitando

Sobre as tendências no desenvolvimento da *Web*, assinale a alternativa incorreta.

a) A *Web* está se expandindo como a interface de muitas plataformas.

b) Entre HTML, CSS e *JavaScript*, é provável que alguma linguagem deixe de existir em breve.

c) Códigos interagirão mais naturalmente com humanos, graças à inteligência artificial.

d) Novas bibliotecas e *frameworks JavaScript* e CSS devem continuar surgindo.

Comentário: se você pensou na alternativa "b", parabéns! HTML, CSS e *JavaScript* formam a base da plataforma *Web*, além de evoluírem e ganharem novos recursos frequentemente, o que afasta a ideia de que poderiam deixar de existir – pelo menos pela próxima década.

1.5 Preparando o ambiente de desenvolvimento

Vamos começar a preparar nosso ambiente de desenvolvimento, ou seja, ajustar o nosso computador para desenvolver com HTML, CSS e *JavaScript*! Já citamos que a *Web* é uma plataforma aberta, e isso pode ser interpretado de duas formas diferentes.

A primeira forma refere-se a ninguém ser proprietário das tecnologias, pois o HTML, o CSS e o *JavaScript* são linguagens padronizadas internacionalmente e totalmente livres, não havendo preocupação com qualquer tipo de taxa sobre a utilização. E não só as linguagens! O HTTP também é padronizado internacionalmente e livre, inclusive a sua evolução mais recente, o HTTP/2, suportado por todos os navegadores modernos desde 2015. Mas a *Web* não é feita só de textos e *links* há muito tempo, outros formatos livres emergiram para atender às necessidades, como o *Portable Network Graphics* (PNG), o *Scalable Vector Graphics* (SVG), o *Web Picture* (*Web*P) para imagens e o *Web Media* (*Web*M) para vídeos.

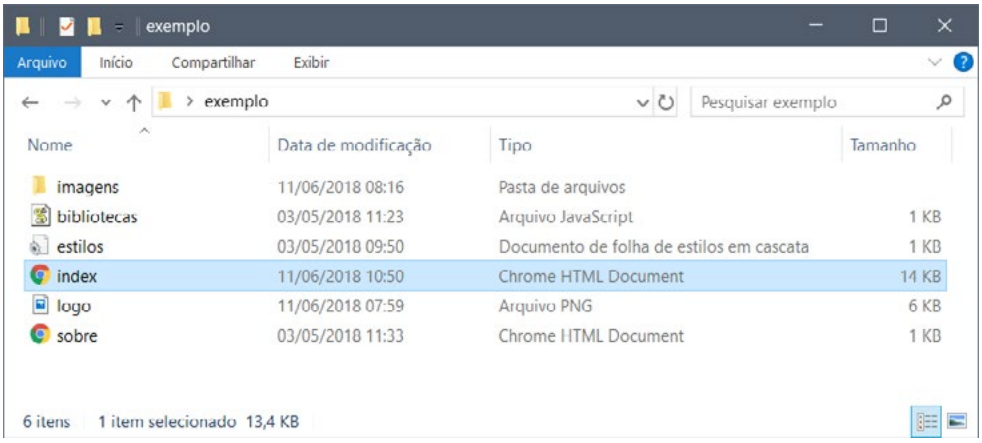
Saiba mais

Alguns formatos comuns na *Web* não foram sempre livres, como o GIF e o JPG. Hoje ambos são padronizados internacionalmente, mas já foram alvo de vários processos sobre quebra de patentes. No caso do GIF, as possíveis patentes expiraram em 2004, sendo seguro utilizar desde então. No caso do JPG, guerras de patentes ocorreram por toda a década passada e duraram até 2013, quando teoricamente a última patente venceu. Se mais nenhuma empresa aparecer alegando que possui patentes infringidas por esses padrões ainda válidas, podemos finalmente dizer que eles são livres para utilização.

A segunda forma de interpretar a afirmação inicial deste tópico refere-se ao fato de que as três linguagens fundamentais da *Web* são, na verdade, arquivos de texto simples, livremente editáveis por qualquer *software* de edição de textos. Vamos entender melhor?

Para isso, é necessário falar sobre extensões. Quando você olha para um arquivo em uma pasta, observa que ele tem um ícone (exceto alguns tipos de arquivos que exibem uma miniatura do conteúdo). Quando você usa um duplo-clique para executar o arquivo, ele abre automaticamente em um programa recomendado. O sistema operacional sabe qual deve ser o programa e qual deve ser o ícone correto para aquele arquivo específico. Como? Através do conceito de extensão, que são alguns caracteres no final do nome do arquivo, após um ponto. Mas, se você abrir uma pasta qualquer no computador agora mesmo para olhar as tais extensões, é possível que não veja nada especial.

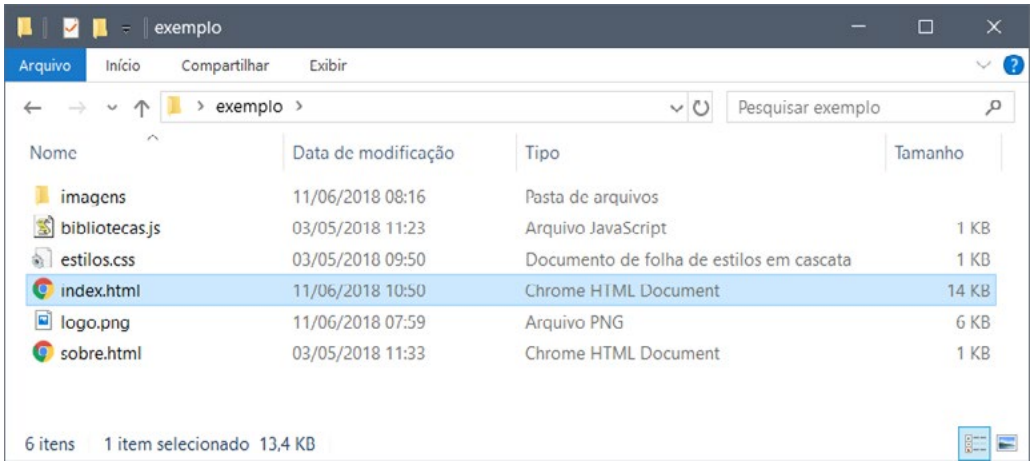
Conjunto de arquivos no Windows Explorer sem exibição de suas extensões



Acredite, as extensões estão aí e são essenciais! Um passo fundamental para preparar qualquer ambiente de desenvolvimento é configurá-lo para que extensões sejam exibidas. Afinal, é importante ter o controle sobre os arquivos que você está criando.

No Windows Explorer, acesse o menu "Arquivo" e selecione "Opções". Na tela que se abre, acesse a aba "Modo de Exibição". Procure na lista a opção "Ocultar as extensões dos tipos de arquivo conhecidos". Desmarque essa opção e clique em "OK" para confirmar as alterações. Imediatamente, você verá todas as extensões na pasta aberta.

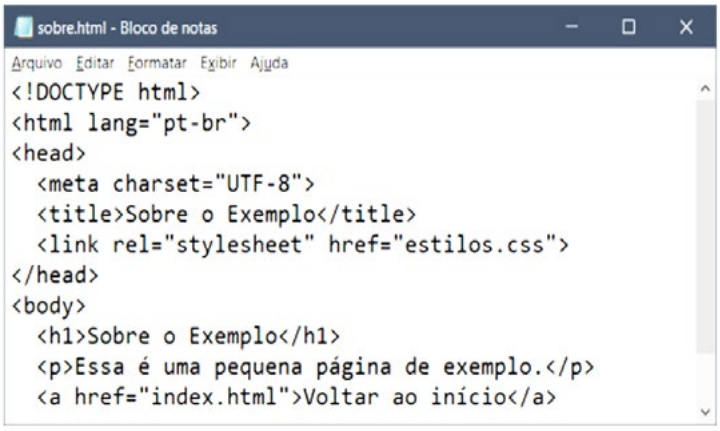
Conjunto de arquivos no Windows Explorer com a exibição de suas extensões



Ótimo, agora podemos ter certeza das extensões de cada arquivo! Ainda assim, é possível forçar a exibição em outros programas. Isso não costuma servir para muita coisa em arquivos proprietários; mas, no caso das linguagens abertas, é algo muito revelador.

Ao clicar com o botão direito em um arquivo HTML, acesse a opção "Abrir com" e escolha "Bloco de notas" na lista (ou procure-o através de "Escolher outro aplicativo", se ele não estiver visível na lista inicial), conseguimos ver todo o código do arquivo escolhido.

Exemplo de página HTML aberta através do "Bloco de notas" do Windows



A mesma experiência serve para CSS e JS (a extensão usada para os arquivos *JavaScript*). Vamos reforçar: a extensão do arquivo direciona a abertura para algum programa previamente instalado, sendo o navegador padrão do computador o programa indicado para arquivos HTML (nas figuras anterio-

res, observa-se que o Google Chrome é o navegador padrão no computador utilizado). Mas qualquer editor de textos (como o próprio "Bloco de notas") serve para abrir tais arquivos e, até mesmo, editar completamente seus códigos.

Mas não é por isso que vamos passar nossos dias no Bloco de notas, provavelmente, o editor de textos mais simples que existe! Programadores profissionais preferem editores dedicados à programação, que colorem partes do código de acordo com a função, ajudam na manipulação dos arquivos, autocompletam comandos, sugerem correções etc.

Outras plataformas e linguagens de programação costumam estar vinculadas a determinados ambientes de desenvolvimento, com as chamadas *Integrated Development Environment* (IDE) específicas para cada uma delas. Uma IDE não é apenas um editor de código, é um conjunto de programas integrados, o que oferece os recursos para abordar todas (ou quase todas) as etapas do desenvolvimento de um *software* na plataforma em questão.

Algumas IDEs famosas são: *Microsoft Visual Studio* (para a plataforma .NET); *Eclipse*, *NetBeans* e *IntelliJ* (para a plataforma Java); *Android Studio* (para a plataforma Android); e *Xcode* (para as plataformas macOS e iOS). Existem várias outras, com diversos focos, muitas abordando mais de uma linguagem. Dessa forma, muitas IDEs permitem manipular arquivos da *Web* e, de fato, qualquer uma delas seria mais produtiva do que o "Bloco de notas". Mas o foco das IDEs costuma ser muito mais abrangente e, na maioria das vezes, elas acabam sendo pesadas demais em processamento e memória, sem necessidade.

Por isso, no cenário da *Web*, tem se consolidado o uso de editores de código especializados nas três linguagens fundamentais, evidentemente muito melhores do que o editor de texto padrão do Windows, mas muito mais leves e focados do que as IDEs. Eles têm evoluído tanto que, em alguns casos, disponibilizam recursos que nem as IDEs tradicionais fazem.

Importante

Existem vários editores de código, alguns inclusive bem parecidos. Entre os mais conhecidos e usados atualmente, temos: *Sublime Text*, *Notepad++*, *Atom*, *Brackets* e *Visual Studio Code* (não confundir com a IDE denominada apenas *Visual Studio*). Com tantas opções, ficaria difícil acertar um que fosse o ideal para todos. Mas vale a pena adotarmos um padrão, então o escolhido para nosso estudo é o *Visual Studio Code*.

O *Visual Studio Code* é um editor moderno, criado pela Microsoft, totalmente livre (seu código-fonte é aberto para qualquer um contribuir). É fortemente otimizado para as linguagens *Web*, com destaque para seus recursos relacionados ao *JavaScript*, alguns dos melhores existentes. Além disso, é totalmente personalizável com complementos, podendo ganhar novos recursos (inclusive outras linguagens) de forma simples. E uma curiosidade: o próprio editor é construído em HTML, CSS e *JavaScript*! Seria muito interessante usar um editor criado com as mesmas linguagens que você está estudando, não é mesmo?

Para começar a utilizar, basta acessar o site oficial https://code.visualstudio.com/ e escolher a a opção de *download*. O *site* costuma detectar e oferecer corretamente a versão para o seu sistema operacional, mas também permite baixar manualmente outras versões. Aliás, é um projeto muito ativo e recebe novas atualizações ao menos uma vez por mês, se atualizando automaticamente depois que você fizer a primeira instalação. Durante este material, usamos a versão para Windows, mas, de forma geral, apenas a instalação é levemente diferente em Linux ou macOS, caso você prefira utilizar outro sistema operacional.

Para completar um ambiente de desenvolvimento *Web*, seria impossível não falarmos dos navegadores, afinal, precisaremos deles para testar as páginas. Por ser o mais utilizado mundialmente, escolhemos o Google Chrome (tenha cuidado de mantê-lo atualizado, para garantir máxima compatibilidade possível com novidades).

Dicas

Desenvolvedores *Web* profissionais sempre testam suas criações em vários navegadores, de acordo com a fatia de mercado de cada um. Portanto, se possível, mantenha também Microsoft Edge e Mozilla Firefox atualizados em seu ambiente.

Exercitando

Quais das características a seguir é um recurso comumente encontrado em editores de código utilizados no desenvolvimento de projetos para a Web?

a) Edição de fotografias com aplicação de filtros e efeitos profissionais.

b) Construção de plantas arquitetônicas baixas ou tridimensionais e fotos realistas.

c) Editoração de planilhas com fórmulas, somas automáticas e gráficos.

d) Coloração de trechos de código de acordo com a linguagem em questão.

Comentário: se você pensou na alternativa "d", parabéns! É um recurso típico colorir cada trecho de código de acordo com as especificidades da linguagem de programação que se está utilizando. As alternativas adicionais são características de *softwares* diferentes, de nenhuma forma relacionados a editores de códigos.

Resumindo

Nesta primeira lição, você conheceu os conceitos de Internet e de *Web*, bem como a relação entre eles. Também pôde acompanhar um resumo dos principais fatos históricos desde o surgimento da Internet e da *Web*, além da criação e da evolução das linguagens HTML, CSS e *JavaScript*. Também apresentamos algumas tendências da *Web* para os próximos anos e o convidamos a refletir sobre quais caminhos seguir nessa área. Por fim, você aprendeu como configurar seu computador como um ambiente de desenvolvimento *Web*, com as ferramentas fundamentais que vamos utilizar.

Veja se você se sente apto a:

- explicar os conceitos de Internet e de *Web* e qual a relação entre eles;
- destacar os principais fatos históricos do surgimento da Internet até os atuais;
- discutir sobre as principais tendências da *Web* para os próximos anos;
- preparar seu computador como um ambiente de desenvolvimento *Web*.

Parabéns, você finalizou esta lição!

Agora responda às questões ao lado.

Exercícios

Questão 1 - No âmbito da ciência da computação, protocolos são regras de comunicação, padrões que garantem que diferentes plataformas computacionais possam interagir entre si. Qual das opções abaixo refere-se a um protocolo da WorldWideWeb?

a) IP.

b) TCP.

c) HTTPS.

d) CSS.

Questão 2 - A seguir, você encontrará afirmações sobre a Internet e a Web. Analise-as com atenção e julgue os itens em certo (C) ou errado (E).

a. ( ) Originalmente, a Web foi concebida para ser baseada apenas no protocolo de transferência, o HTTP, e a linguagem de marcação de hipertexto, o HTML.

b. ( ) Os navegadores, também conhecidos pelo termo em inglês *browsers*, são parte oficial da plataforma Web e só podem ser lançados se autorizados pelo W3C.

c. ( ) O protocolo HTTPS é uma variante do HTTP tradicional, mas com uma camada adicional de segurança seguindo o protocolo SSL/TLS, com criptografia dos dados trafegados e validação de autenticidade através de certificados digitais.

d. ( ) Sobre a arquitetura cliente-servidor da Web, ela está associada ao fato de que todas as máquinas conectadas automaticamente disponibilizam seus arquivos para todos.

Questão 3 - Assinale a alternativa que apresenta apenas linguagens criadas para a *Web*, ainda que tenham evoluído para outras plataformas. Pode considerar tanto as linguagens de marcação e estilização quanto as linguagens de programação.

a) HTML e JavaScript.

b) JavaScript e HTTP.

c) Delphi e CSS.

d) Java e Delphi.

Questão 4 - Existem diversos editores de código, alguns focados na Web, outros mais genéricos, alguns gratuitos e outros pagos. Não podemos confundir editores de código com IDEs, pois estas costumam ser maiores, normalmente vinculadas a alguma plataforma de programação específica e, em muitos casos, mais pesadas durante a execução. Entre alguns dos editores mais conhecidos, com foco em desenvolvimento Web, podemos destacar:

I. Eclipse

II. Atom

III. Visual Studio Code

IV. Android Studio

V. Sublime Text

É correto afirmar que:

a) as afirmativas I, II e III estão corretas.

b) as afirmativas III e IV estão corretas.

c) as afirmativas II, III e V estão corretas.

d) todas as afirmativas estão corretas.

Questão 5 - Qual das alternativas a seguir apresenta passos corretos para habilitar a exibição de extensões de arquivos conhecidos em sistemas Windows?

a) No Windows Explorer, acesse o menu "Arquivo" e escolha "Opções". Na tela que se abre, acesse a aba "Modo de Exibição". Procure a opção "Ocultar as extensões dos tipos de arquivo conhecidos", desmarque-a e clique em "OK".

b) No Visual Studio Code, acesse o menu "File" e escolha "Open Folder". Na tela que se abre, navegue pelos diretórios até encontrar a pasta desejada. Selecione a pasta, escolha a codificação "UTF-8" e clique em "OK".

c) No Windows Explorer, acesse o menu "Arquivo" e escolha "Ajuda". Na tela que se abre, digite sua dúvida no buscador "Bing". Espere até que o buscador entenda corretamente seu problema e faça as configurações necessárias em seu sistema.

d) O sistema operacional Windows não permite a exibição de extensões dos arquivos conhecidos, mas o Visual Studio Code exibe as extensões mesmo assim. De qualquer forma, recomenda-se fortemente utilizar um macOS com o VS Code.

Questão 6 - Analisar tendências pode soar como meros tiros no escuro. Prever o futuro é sempre suscetível a erros, mas, observando como algumas coisas estão despontando, é possível ter uma boa margem de segurança sobre o que esperar para os próximos anos. Qual das afirmações não é uma tendência realmente esperada no desenvolvimento Web?

a) Novas bibliotecas e frameworks JavaScript e CSS devem continuar surgindo.

b) Códigos interagirão mais naturalmente com humanos, graças à inteligência artificial.

c) Entre HTML, CSS e JavaScript, é provável que algum deixe de existir em breve.

d) A Web está se expandindo como a interface de muitas plataformas.

Questão 7 - A primeira rede de computadores funcional, conhecida como Arpanet, é um feito atribuído à Agência de Projetos Avançados de Pesquisa (Arpa), do Departamento de Defesa dos Estados Unidos da América. Sobre esse feito, assinale a alternativa incorreta.

a) Foi iniciada de forma bem simples, com apenas quatro computadores conectados à rede.

b) Serviu de inspiração para redes de outras entidades, como universidades.

c) Não tinha objetivos militares, almejando-se apenas elevar o conhecimento humano.

d) O protocolo TCP/IP foi iniciado por um dos engenheiros da Arpa, mas acabou não sendo uma criação exclusiva e tal agência.

Questão 8 - A linguagem de programação para navegadores Web conhecida como JavaScript, graças a um acordo comercial da Netscape com a Sun Microsystems na época de seu lançamento, não pôde manter esse nome quando foi submetida para padronização. Qual organização padronizou internacionalmente tal linguagem, em 1997?

a) ECMA.

b) Microsoft.

c) Netscape.

d) W3C.

Questão 9 - Associe as alternativas, de acordo com as respectivas definições de termos importantes utilizados no contexto do desenvolvimento *Web*.

a) Formato de arquivo de estilos para definições de visual.

b) Linguagem de marcação para estruturar as páginas *Web*.

c) Linguagem de programação nativa nos navegadores *Web*.

d) Protocolo de comunicação entre navegadores e servidores.

( ) JavaScript

( ) HTTP

( ) CSS

( ) HTML

Questão 10 - No dia a dia, o nome JavaScript se mantém inabalável, mas, oficialmente, apenas a empresa Oracle poderia utilizar o nome, por ter adquirido a Sun Microsystems. Ou seja, mesmo quando a linguagem foi padronizada, em 1997, o nome já era registrado. Por isso, o nome oficial da especificação internacional que padroniza o JavaScript é:

a) JScript.

b) Web Script.

c) W3C Script.

d) ECMAScript.

Gabarito: 1c; 2 CECE; 3a; 4c; 5a; 6c; 7c; 8a; 9 CDAB; 10a;