**A história do JavaScript**

A história do JavaScript é, de muitas maneiras, a história da Internet moderna. Qualquer pessoa que se lembre das páginas da web estáticas, desajeitadas e de carregamento lento nos primeiros anos da Internet, ou mesmo que tenha visto um vislumbre delas através da Wayback Machine, tem uma noção de quanto o desenvolvimento web evoluiu desde os anos 1990. Elementos interativos como animação, formulários preenchíveis e até mesmo a capacidade de rolar através de artigos como este se devem em grande parte ao surgimento do JavaScript, uma linguagem de programação que foi inicialmente desenvolvida em 10 dias e concebida como um “coadjuvante” do Java.

Este artigo fará uma análise aprofundada de quão longe o JavaScript chegou, guiando os leitores através dos conceitos básicos, origens, esforços de padronização e estado atual da linguagem de programação mais popular do mundo.

**O que é JavaScript?**

JavaScript vs. HTML vs. CSS

JavaScript é uma linguagem de script do lado do cliente que permite a criação de elementos interativos em websites. Junto com HTML e CSS, JavaScript fornece um dos três blocos de construção que são usados para criar websites modernos. Um post do blog da Hubspot blog post fornece a seguinte distinção simples para entender os três conceitos:

HTML fornece a estrutura básica dos sites, que é aprimorada e modificada por outras tecnologias como CSS e JavaScript.

CSS é usado para controlar apresentação, formatação e layout.

JavaScript é usado para controlar o comportamento de diferentes elementos.

Em outras palavras, HTML fornece a estrutura para uma página da web, CSS adiciona elementos de design, e JavaScript permite a interatividade.

**Para que o JavaScript é usado?**

O site de codificação Skillcrush fornece uma excelente descrição dos usos para JavaScript. Ele observa que JavaScript é “usado para criar e controlar conteúdo dinâmico de website, ou seja, qualquer coisa que se move, atualiza ou muda na sua tela sem exigir que você recarregue manualmente uma página da web.” Entre outros elementos interativos, isso inclui:

Caixas de confirmação.

Chamadas para ação deslizantes.

Jogos baseados em navegador.

Animação e outros efeitos especiais.

Recursos de segurança como senhas.

Feeds de notícias que se atualizam automaticamente.

Desenvolvimento de aplicações móveis.

**Origens do JavaScript**

JavaScript 1.0 foi lançado em 1995, quando a Internet ainda era uma invenção relativamente nova. O primeiro navegador web, Mosaic, havia sido lançado apenas dois anos antes, e HTML era a única ferramenta para construir websites (na época, CSS ainda era uma proposta e não seria oficialmente lançado até 1996). Criar websites ainda era uma habilidade relativamente fácil de adquirir, algo que era acessível não apenas para engenheiros de software, mas também para desenvolvedores e designers inexperientes.

Marc Andressen, o fundador do recém-lançado navegador Netscape, queria expandir as capacidades do navegador adicionando elementos mais dinâmicos. Ao mesmo tempo, ele queria que esses elementos estivessem disponíveis para o novo e crescente mercado de desenvolvedores web amadores. Assim, nasceu a ideia de criar uma linguagem de script simples e dinâmica que um dia seria conhecida como JavaScript.

**Criação na Netscape**

Brendan Eich, o criador do JavaScript, foi contratado pela Netscape para criar uma linguagem de script baseada em navegador. De acordo com JavaScript: The First Twenty Years, um artigo publicado pela ACM co-escrito por Eich, esta tarefa foi complicada pela parceria da Netscape com a Sun Microsystems. A estratégia conjunta deles era integrar a linguagem de programação Java da Sun no Netscape 2.0 em uma tentativa de superar o Internet Explorer da Microsoft. Como resultado, “a rápida estratégia dentro da Netscape para escolher uma linguagem de script prejudicou severamente Scheme, Perl, Python, Tcl e Visual Basic como não viáveis devido a interesses comerciais e/ou considerações de tempo para o mercado. A única abordagem considerada viável… era projetar e implementar uma ‘pequena linguagem’ para complementar o Java”.

Como a versão beta do Netscape 2.0 estava programada para ser lançada em setembro de 1995, Eich teve que agir rapidamente. O protótipo inicial, então chamado “Mocha”, foi criado em 10 dias em maio de 1995, com trabalho adicional durante o verão para eliminar bugs, responder a solicitações de recursos e projetar APIs que permitiriam que o Mocha interagisse com o Netscape. Essas APIs estabeleceram as bases para o que eventualmente seria conhecido como o Document Object Model (DOM), uma interface chave para interagir com documentos HTML e XML.

Como a Auth0 observa em um blog post de 2017, o resultado final “parecia um Java dinâmico, mas por baixo era uma besta muito diferente: um filho prematuro de Scheme e Self” e foi apresentado sob o novo nome JavaScript “como uma linguagem de script para pequenas tarefas do lado do cliente no navegador, enquanto Java seria promovido como uma ferramenta maior e profissional para desenvolver componentes web ricos”.

**Lançamento inicial**

A disponibilidade do JavaScript na versão beta do Netscape 2.0 foi anunciada em um press release de dezembro de 1995, que descrevia o JavaScript como:

Projetado para criar aplicações centradas em rede.

Complementar e integrado ao Java.

Complementar e integrado ao HTML.

Aberto e multiplataforma.

Na documentação inicial, a Netscape descreve as principais capacidades do JavaScript para “reconhecer e responder a eventos do usuário como cliques do mouse, entrada de formulário e navegação de página”. Isso permitiu uma ampla gama de casos de uso, como:

Solicitar o número de telefone ou código postal de um usuário.

Alertar usuários sobre entradas de formulário inválidas.

Criar caixas de seleção.

Adicionar botões “voltar” para navegar facilmente pelos websites.

Preservar o estado entre páginas da web para rastrear ações do usuário.

Reproduzir animações, textos rolantes ou arquivos de áudio.

**ActiveX e NetscapeONE: Rumo ao desenvolvimento de aplicações:**

**JScript e ActiveX**

À medida que o interesse pelo JavaScript crescia, navegadores concorrentes como o Internet Explorer estavam sob pressão para permitir recursos de script semelhantes. Embora “Java” estivesse sob marca registrada pela Sun, JavaScript era aberto e livremente licenciado, permitindo que a Microsoft fizesse engenharia reversa de sua própria implementação, JScript, que foi lançada com o Internet Explorer 3.0 em agosto de 1996.

Além disso, o JScript poderia se integrar ao ecossistema mais amplo da Microsoft como parte de um conjunto de tecnologias conhecido como ActiveX. Como afirmado em um 1996 press release, o ActiveX foi projetado para “formar uma estrutura robusta para criar conteúdo interativo usando componentes de software, scripts e aplicações existentes” e permitiu que os usuários “integrassem aplicações em navegadores web para que os dados gerenciados por essas aplicações se tornassem acessíveis como páginas web”. Também permitiu a criação de aplicações interativas no lado do servidor através do suporte para JScript, bem como VBScript, uma linguagem de script derivada do Visual Basic da Microsoft.

**NetscapeONE e JavaScript 1.1**

Além da popularidade do JavaScript, um grande impulso para o lançamento do ActiveX pela Microsoft foi um projeto semelhante na Netscape projetado para expandir o HTML além do navegador e para o domínio do desenvolvimento de aplicações web. Este projeto, conhecido como NetscapeONE, permitiu que as páginas web não apenas apresentassem informações, mas servissem como contêineres para aplicações de negócios.

O NetscapeONE, bem como o Netscape 3.0, também incluía a versão 1.1 do JavaScript, que permitia a criação de bibliotecas JavaScript como páginas separadas em vez de inserir código JavaScript em arquivos HTML.

No entanto, diferenças entre a implementação do JavaScript pela Microsoft e o JavaScript 1.1 da Netscape resultaram em incompatibilidades que causavam falhas no código interativo dependendo do tipo de navegador ou, às vezes, até mesmo da versão do navegador. Para realmente permitir suporte multiplataforma, era necessária a padronização.

**Padronização e expansão**

Ao criar especificações sobre como as ferramentas tecnológicas devem ser implementadas, os órgãos de padronização permitem a interoperabilidade entre plataformas e dispositivos. A Netscape anunciou seus planos para padronizar o JavaScript através do órgão internacional de padrões ECMA em um 1996 press release. Como a ECMA não poderia usar o nome registrado “Java”, a linguagem foi padronizada sob o nome ECMAScript. A introdução do ECMAScript permitiu mais interoperabilidade entre navegadores, mas novos desenvolvimentos na história da Internet estavam impulsionando a demanda por expansões ainda mais ambiciosas.

Um desses desenvolvimentos foi o surgimento do AJAX, uma técnica de desenvolvimento que usa várias tecnologias, incluindo JavaScript, para permitir programação assíncrona. Com isso, as páginas web podiam alterar dinamicamente o conteúdo sem recarregar a página inteira e, como resultado, sem interromper a experiência do usuário. Embora o post inicial de blog de 2005 propondo o AJAX o apresentasse principalmente como uma forma de melhorar a experiência do usuário reduzindo o tempo que os usuários estão “esperando o servidor fazer algo”, suas ramificações incluem muitos recursos amplamente utilizados na web hoje. Isso inclui sugestões de pesquisa que mudam conforme você as digita no Google até barras de status, notificações, feeds de notícias e outros elementos que se atualizam automaticamente.

**Node.js e JavaScript do lado do servidor**

Ao longo dos anos 2000, a introdução e o rápido crescimento da Internet sem fio, dispositivos móveis conectados à Internet, computação em nuvem e mídias sociais estavam impulsionando o uso da Internet a novos patamares, exigindo cada vez mais capacidade para atender a esse tráfego. Em 2009, a frustração com a incapacidade dos servidores HTTP Apache de lidar com requisições simultâneas na casa das dezenas e centenas de milhares inspirou a criação do Node.js.

Node.js é um ambiente de tempo de execução que move o JavaScript para fora do navegador, permitindo que o JavaScript seja executado em qualquer lugar onde o Node.js esteja instalado, usando o V8 Engine do Google para compilar código JavaScript em código de máquina. Como observado em JavaScript in Plain English, “isso mudou a linguagem JavaScript para sempre e ajudou lentamente a transformá-la em mais uma linguagem de programação e menos uma linguagem de script” e permitiu que desenvolvedores frontend escrevessem código do lado do servidor sem aprender uma nova linguagem. Além disso, o Node.js aumenta a capacidade do servidor usando um loop de eventos e E/S assíncrona para lidar com muitas solicitações simultâneas ao mesmo tempo.

JavaScript hoje

Hoje, JavaScript é consistentemente classificado como a linguagem de programação mais usada por desenvolvedores web, de acordo com GitHub. A partir deste ano, as pesquisas da W3Tech notes observam que JavaScript é usado para programação do lado do cliente por 97,1% de todos os websites, incluindo alguns dos websites mais populares do mundo, como Google, YouTube, Facebook e Amazon.

Os últimos anos trouxeram alguns dos desenvolvimentos mais emocionantes no JavaScript até o momento, incluindo a introdução de:

Async and await, um recurso JavaScript que simplifica a programação de código assíncrono.

Webassembly, uma linguagem complementar ao JavaScript que permite que código escrito em diferentes linguagens seja executado em velocidade quase nativa.

Typescript, uma variante do JavaScript que verifica erros em um programa antes da execução.

Embora o JavaScript tenha sido projetado rapidamente e destinado a pequenas tarefas de design frontend, agora é um ecossistema rico e maduro que transformou a Internet moderna com sites dinâmicos e aplicações web.

**Edge Functions com suporte a JavaScript**

Aqui na Azion, um de nossos principais objetivos é tornar a programação o mais simples e flexível possível. É por isso que o JavaScript, uma linguagem com suporte ubíquo e criada com facilidade de uso em mente, é uma ótima opção para Edge Functions, um novo produto que traz computação serverless para o edge da rede.

Nossa implementação de JavaScript usa o padrão ECMA, incluindo suporte para promessas com async/await e planos para suportar WebAssembly em breve. Com Edge Functions, os desenvolvedores podem construir e executar funções orientadas a eventos no edge da rede, usando a API JavaScript Runtime. As Edge Functions são executadas em milissegundos no node de edge mais próximo do usuário final e escalam automaticamente, eliminando a necessidade de provisionar ou gerenciar recursos. Como resultado, o custo e o uso de recursos são drasticamente reduzidos, e os desenvolvedores podem passar mais tempo focando no frontend de sua aplicação, garantindo a melhor funcionalidade e experiência do usuário possível.

Para ler mais sobre Edge Functions, visite a página do produto ou crie uma conta gratuita para começar a programar funções orientadas a eventos em JavaScript hoje.

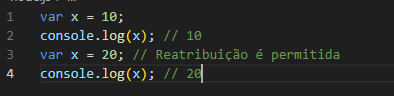
**Principais Diferenças entre var, let e const:**

Em JavaScript, as palavras-chave var, let e const são usadas para declarar variáveis, mas possuem diferenças importantes em seu comportamento

**Var:**

Escopo: Possui escopo de função (se declarada dentro de uma função) ou global (se declarada fora de uma função).

Hoisting: Variáveis declaradas com var são içadas (hoisted) para o topo do seu escopo, mas com valor undefined até que a linha de código seja executada.

Reatribuição: É possível reatribuir valores à variável declarada com var.