

Javascript

Foi originalmente implementada como parte dos navegadores web para que scripts pudessem ser executados do lado do cliente e interagissem com o usuário sem a necessidade deste script passar pelo servidor, controlando o navegador, realizando comunicação assíncrona e alterando o conteúdo do documento exibido, porém os mecanismos JavaScript agora estão incorporados em muitos outros tipos de software host, incluindo servidores em servidores e bancos de dados da Web e em programas que não são da Web, como processadores de texto e PDF, e em tempo de execução ambientes que disponibilizam JavaScript para escrever aplicativos móveis e de desktop, incluindo widgets de área de trabalho.

Os termos Vanilla JavaScript e Vanilla JS se referem ao JavaScript não estendido por qualquer estrutura ou biblioteca adicional.

Embora existam semelhanças entre JavaScript e Java, incluindo o nome da linguagem, a sintaxe e as respectivas bibliotecas padrão, as duas linguagens são distintas e diferem muito no design; JavaScript foi influenciado por linguagens de programação como Self e Scheme.

Desde 1996, o servidor da Web do IIS tem suportado a implementação do JavaScript - JScript do lado do servidor - em páginas ASP e .NET. Desde meados da década de 2000, foram introduzidas implementações adicionais de JavaScript no lado do servidor, como o Node.js em 2009.

.NET. Com o tempo, ficou claro que a Microsoft não tinha intenção de cooperar ou implementar o JavaScript adequado no Internet Explorer, mesmo que eles não tivessem uma proposta concorrente e tivessem uma implementação parcial no lado do servidor.

O resultado foi a proliferação de estruturas e bibliotecas abrangentes, práticas de programação JavaScript aprimoradas e aumento do uso de JavaScript fora dos navegadores da Web, conforme observado pela proliferação de plataformas JavaScript do lado do servidor.

Aplicações como Gmail tomam vantagem disso: muito da lógica da interface do usuário escrita em JavaScript, e o JavaScript envia requisições de informação, tais como o conteúdo de um correio eletrônico, para o servidor.

Uma JavaScript engine interpreta código fonte JavaScript e o executa de forma adequada.

No script acima, vimos que existe uma tag chamada `<noscript>`, ela está neste código HTML porque um meio de acessibilidade com o cliente, fazendo com que seu texto seja renderizado pelo navegador quando o JavaScript estiver desativado.

Para lidar com essas diferenças, programadores JavaScript com frequência tentam escrever códigos que conformam com o padrão comum a maioria dos navegadores; não sendo possível isso, tentam escrever de maneira ad-hoc um código que verifique a presença de certos recursos e que se comporte de maneira adequada caso tais recursos não estejam disponíveis.

Para suportar tais usuários, programadores web tentam criar páginas que sejam robustas a agentes que não suportem o JavaScript da página.

Uma abordagem alternativa que muitos acham preferível a página se desenvolvida por primeiro a partir de tecnologias básicas que funcionem em todos os navegadores, e então aprimorá-la para os usuários que possuam JavaScript.

Não se recomenda que o código JavaScript de uma página seja totalmente dependente do eventos provenientes do mouse já que usuários que não conseguem ou optam por não usar o mouse não estarão aptos a colher os benefícios de tal código.

"Sequestro JavaScript" um tipo de ataque CSRF no qual uma tag `<script>` no site do atacante explora uma página no lado da vítima que retorna informação privada tal como JSON ou JavaScript.

Isso torna o JavaScript um vetor teoricamente viável para um cavalo de Tróia, embora cavalos de Tróia em JavaScript sejam incomuns na prática.

Além do software de computador nativo, há o ambiente de desenvolvimento integrado JavaScript online, que possui recursos de depuração que são gravados em JavaScript e criados para serem executados na Web.