Aplicações Web
tecnologias e soluções

Helder da Rocha
Consultor IT

Conteúdo

1. Introdução: arquitetura Web
2. Tecnologias de apresentação
   - HTML, CSS, XML, DHTML
3. Tecnologias interativas lado-servidor
   - CGI - Common Gateway Interface
   - Plug-ins e componentes no servidor (servlets, ISAPI)
   - Linguagens (*scripts*) de extensão do servidor (ASP, JSP)
4. Persistência
   - Cookies
5. Tecnologias interativas lado-cliente
   - Plug-ins e componentes no cliente (applets)
   - Linguagens (*scripts*) de extensão no cliente (JavaScript)
6. Comparativo: O que usar? Quando usar?

Objetivos

- Descrever e apresentar exemplos de cada tecnologia
- Discutir
  - o que usar
  - quando usar
  - vantagens e desvantagens

Arquitetura Web

- HTTP (RFC 2068)
  - Protocolo de transferência de arquivos
  - Sem estado (não mantém sessão aberta)

Arquitetura Web: servidor

- Papel do servidor (básico)
  - Gerenciar sistema virtual de arquivos e diretórios
    - mapear diretórios reais (ex: **c:\htdocs**) a diretórios virtuais (ex: **/**) usando notação de URI
  - Receber requisições HTTP do cliente, para
    - localizar arquivo
    - identificar tipo e tamanho do arquivo
    - construir cabeçalho (RFC 822) com informações sobre servidor e arquivo (tipo, tamanho, etc.)
  - Enviar arquivo (à saída padrão)
    - Código de resposta (200, 404, etc.)
    - Cabeçalho RFC 822 (cabeçalho e-mail) e dados

Arquitetura Web: cliente

- Papel do cliente (básico)
  - Enviar requisições a um servidor
    - Método HTTP (GET, POST, HEAD, ...)
    - URL absoluta (a partir da raiz do servidor)
    - Cabeçalhos RFC 822 (info sobre cliente)
    - Dados (se método for POST)
  - Processar respostas recebidas, e
    - separar cabeçalhos dos dados
    - identificar tipo de dados e aplicação que a executa
    - interpretar dados (ou repassar à outra aplicação)

Comunicação HTTP: detalhes

- Requisição do browser (uma imagem)

```
GET /images/minotaur.gif HTTP/1.1
User-Agent: Mozilla 4.05 [en] (Windows 95; I)
Cookies: querty=uiop; asdfg=hjkl; num=12345
```

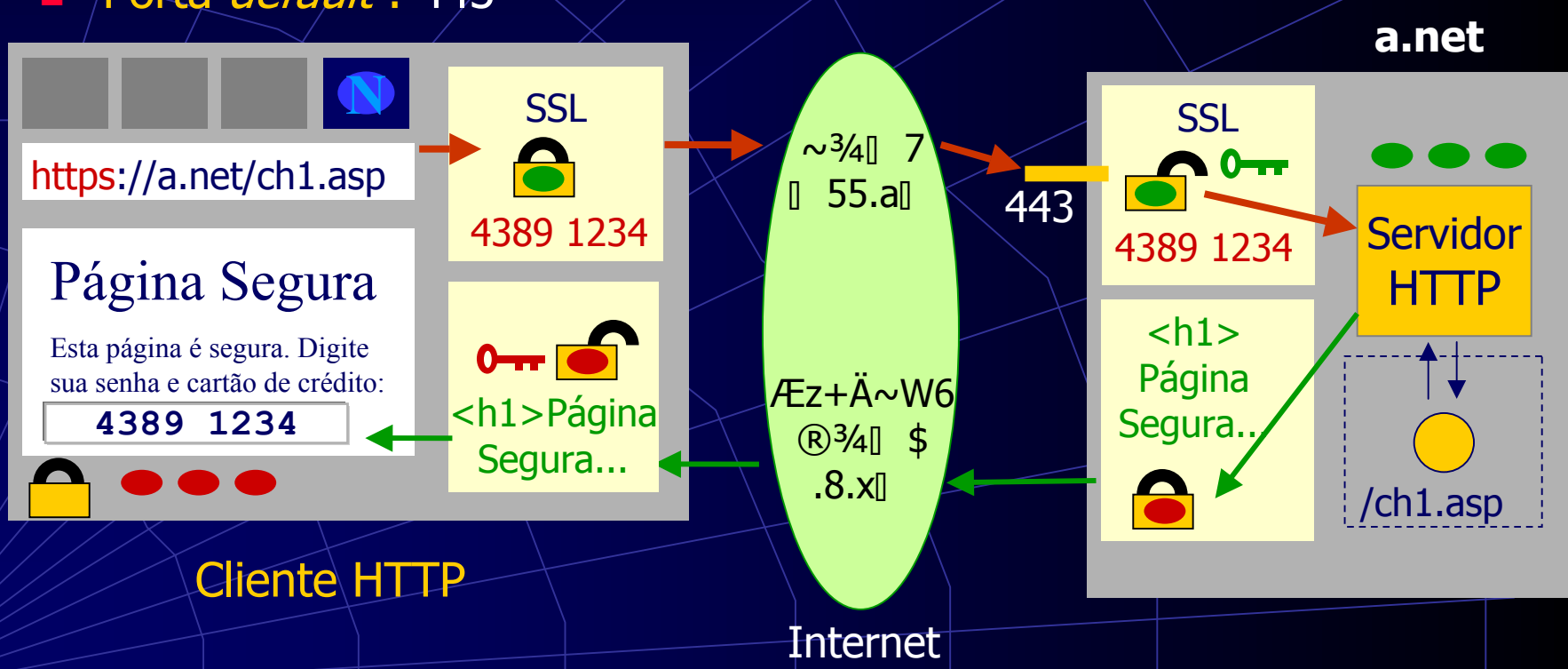
- Resposta do servidor

```
HTTP 1.1 200 OK
Server: Apache 1.02
Date: Friday, August 13, 1999 03:12:56 GMT-03
Content-type: image/gif
Content-length: 23779

!#GIF89~¾ 7
.55.a  6¤Ü4  ...
```

SSL - Secure Sockets Layer

- Camada adicional na comunicação HTTP que introduz criptografia da comunicação
- Tanto o browser quanto o servidor têm que suportar o recurso para que uma transação seja segura
- Porta *default* : 443



Serviço Web: funções

- Serviço de **informações**
  - finalidade: publicação de informações, multimídia
  - interatividade: limitada a hipertexto
  - tecnologias (passivas): HTML, folhas de estilo
- Serviço de **aplicações locais (rodam no cliente)**
  - finalidade: oferecer mais recursos interativos ao cliente
  - interatividade: limitada pelo cliente
  - tecnologias (ativas): JavaScript, applets Java, Flash, ActiveX
- Serviço de **aplicações cliente/servidor**
  - finalidade: oferecer interface para aplicações no servidor
  - interatividade: limitada pela aplicação e servidor Web
  - tecnologias (ativas): CGI, ASP, ISAPI, Servlets, JSP

Tecnologias de apresentação
(serviço de informações)

- HTML 4.0 (HyperText Markup Language)
  - Coleção de marcadores (SGML) usados para formatar texto:

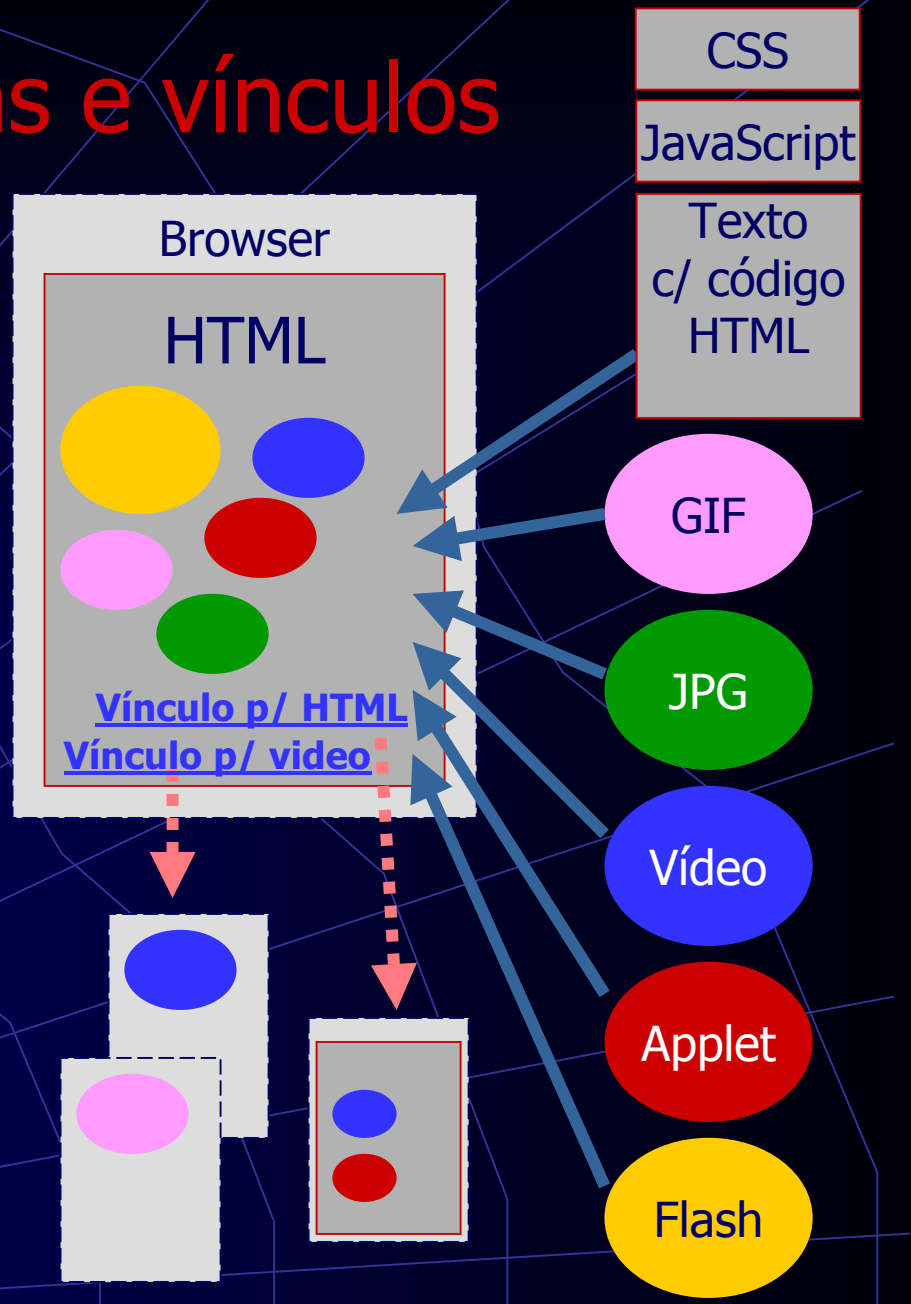
    <H2>Cabeçalho de Nível 2**</H2>**
    <P>Primeiro parágrafo**</P>**
  - Nada diz sobre aparência (browser é quem decide). Define *apenas* **estrutura** e **conteúdo**.
- CSS 2.0 (Cascading Style Sheets)
  - Lista de regras de apresentação para uma página ou todo um site (linguagem declarativa)
  - Depende da estrutura do HTML. Define **forma**.
- Padrões W3C (http://www.w3.org)

HTML - HyperText Markup Language

- Define a interface do usuário na Web
- Pode ser usada para
  - Definir a estrutura do texto de uma página (que o browser posteriormente formatará com uma folha de estilos)
  - Incluir imagens numa página
  - Incluir vínculos a outras páginas
  - Construir uma interface com formulários para envio de dados ao servidor
  - Servir de base para aplicações rodarem dentro do browser (applets Java, *plug-ins*, vídeos, etc.)
- Veja *exemplo* de HTML

Além das imagens e vínculos

- Cada página HTML pode estar interligada a outra página, ou a qualquer objeto como imagens, vídeos, applets, etc.
- → Vínculos internos: imagens, applets, que aparecem embutidas como parte da página
- ⇢ Vínculos externos: ativados em tempo real por eventos que provocam a transferência de imagens, programas, etc. que ocuparão toda a janela.



Interpretação do HTML

- HTML é texto. Instruções são marcadores entre "<" e ">"
- Elemento HTML geralmente *contém* texto, rotulado com sua função estrutural, por um par de marcadores. Marcador final começa com "</":
  - Um parágrafo

  mas às vezes marca um lugar (elemento vazio)
  - aqui haverá quebra
</br>de linha (ou
)
- HTML geralmente descreve estrutura
  - <h1>Isto **é um** cabeçalho de nível 1</h1>

  mas às vezes (raramente) descreve forma
  - O texto **está em** negrito
- *Atributos* mais freqüentemente descrevem forma:
  - <h1 align=center>Cabeçalho H1 no Centro</h1>

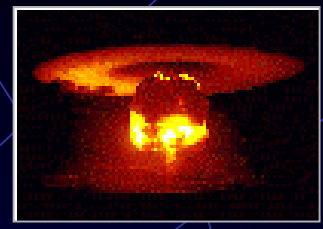
Estrutura HTML

```
<?xml version”1.0” ...?>  (ou <!doctype ... /dtd html 4/ ...>)
<html>
    <head>
        <title>
        (...)
    </head>
    <body>
        (...)
    </body>
</html>
```

Cabeçalho (meta-informação):
Informações sobre o conteúdo da página
Funções (JavaScript)
Propriedades de estilo (CSS1)

Corpo (informação):
Conteúdo da página
Links, imagens, applets, plug-ins

Sintaxe HTML



/bomb.gif

- Imagens

```
Eis a bomba:
<img src="http://a.b.com/bomb.gif"></img> aqui!
```

No browser:

Eis a bomba:  aqui!

- Vínculos

```
Vínculo para <a href="http://www.algumlugar.com">
algum lugar</a>.
```

No browser:

Vínculo para algum lugar.

(Nota: <img></img> segue notação HTML v4 compatível com XML)

CSS - Cascading Style Sheets

- Linguagem usada para definir folhas de estilo que podem ser aplicadas a todo o site.
- Cuida exclusivamente da aparência (forma) da página
- Permite posicionamento absoluto de textos e imagens, manipulação com fontes, cores, etc.
- Regras são colocadas em arquivo de texto .css:

```
H1, H2        {color: rgb(91.5%, 13%, 19%);
                font-size: 72pt; margin-top: 0.6cm}
P.citacao {font-family: Garamond, serif;
                font-weight: 300}
```

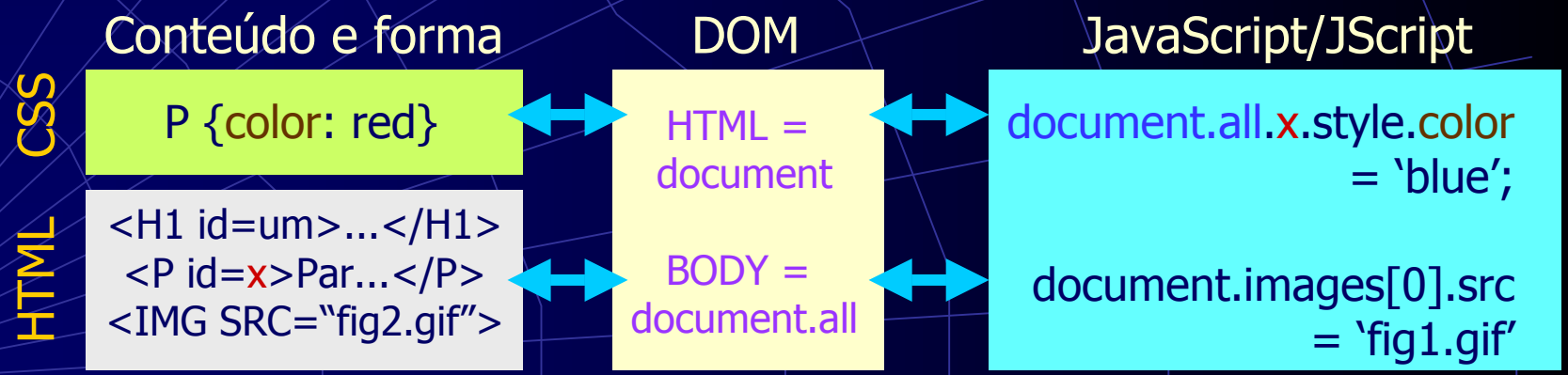
- Veja *exemplo* do uso de CSS

Limitações

- HTML e CSS são linguagens declarativas, interpretadas pelo browser, que definem apenas como a informação será organizada e apresentada.
- Não oferecem recursos de programação.
- Os formulários criados com HTML não fazem nada (eles precisam ser vinculados a uma aplicação)
- Não é possível construir aplicações Web interativas utilizando apenas CSS e HTML
- Dynamic HTML: solução para alguns problemas
  - apresentação + estrutura + interatividade

DHTML - Dynamic HTML

- Combinação de uma linguagem de programação (geralmente JavaScript) com HTML e CSS
  - Permite tratar eventos em qualquer lugar da página
  - Permite grande interatividade
  - Permite alteração dinâmica de conteúdo, estrutura e aparência
- **DOM - Document Object Model** é a ponte entre o HTML/CSS e a linguagem baseada em objetos



DHTML: Problemas

- **DOM** padrão (W3C) é ficção (ninguém adota), portanto, não existe um DHTML padrão
  - Cada fabricante de browser tem a sua versão
  - As versões são incompatíveis
    - Definem extensões ao HTML que não foram padronizadas:
      - <LAYER>, <ILAYER>, etc. (essenciais no DTHML da Netscape... Não fazem parte da especificação!
    - Utilizam sintaxe diferente e incompatível no DOM
  - É preciso identificar o browser antes de usar e...
  - É necessário escrever duas vezes mais código ou criar um conjunto de páginas para cada versão de browser

XML - e**X**tensible **M**arkup **L**anguage

- Meta-linguagem para definição de linguagens de marcação (??ML) utilizáveis na Web
- Permite definir novos marcadores, suas regras de uso, sintaxe, atributos, etc. Ex: 'dadosML':

```
<tabela nome="livros">
    <registro tipo="dados">
        <campo id="usuario">...</campo> <campo id="cod">...
... </registro>
</tabela>
```

- Cliente XML conhece o **DTD** (Document Type Definition) que especifica uma linguagem ??ML. A partir daí, pode entender páginas escritas em ??ML.

```
<!ELEMENT tabela ( registro+ )>
<!ELEMENT registro ( campo+ )>
<!ELEMENT campo (#PCDATA)>
```

XML

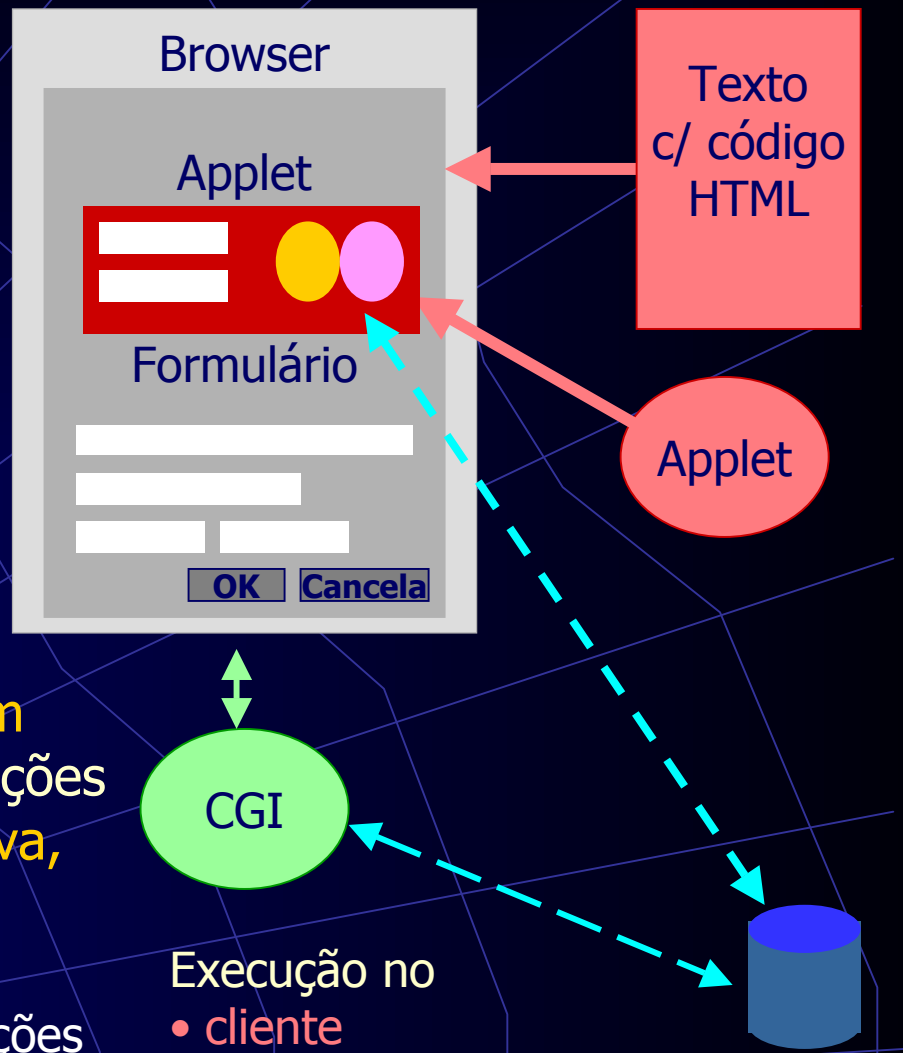
- A sintaxe de linguagens XML é semelhante à atual sintaxe do HTML. É, porém, mais exigente:
  - Elementos sempre têm marcador inicial ou final
    - <IMG></IMG>,
</BR> (ou <IMG/>,
)
  - Marcadores inicial e final devem ter caracteres em formato consistente (caixa-alta ou caixa-baixa)
  - Valores dos atributos exigem aspas
- Desenvolvimento XML possui duas etapas:
  - desenvolver o DTD (sintaxe XML/SGML)
  - desenvolver as páginas na linguagem definida pelo DTD
- HTML 4 (XHTML 1.0) é um caso particular de XML (usa o DTD HTML 4 que está embutido em todos os browsers)

Tecnologias Interativas
(serviço de aplicações)

- Lado-servidor
  - CGI, plug-ins do servidor e componentes
  - Linguagens de extensão: ASP, JSP
- Persistência
  - Cookies
- Lado-cliente
  - Linguagens de extensão: JavaScript, VBScript
  - Plug-ins e componentes (applets, activeX)
  - Soluções integradas: DHTML

Servidor Web como repositório de aplicações

- O servidor Web pode servir de agente intermediário para aplicações rodando na máquina servidora
- Aplicações Web (no servidor) tem páginas HTML como interfaces
  - multiplataforma
  - uniformes
- Browsers modernos permitem que servidores enviem aplicações para execução no cliente (Java, ActiveX, scripts)
  - transformam browser em interface universal p/ aplicações



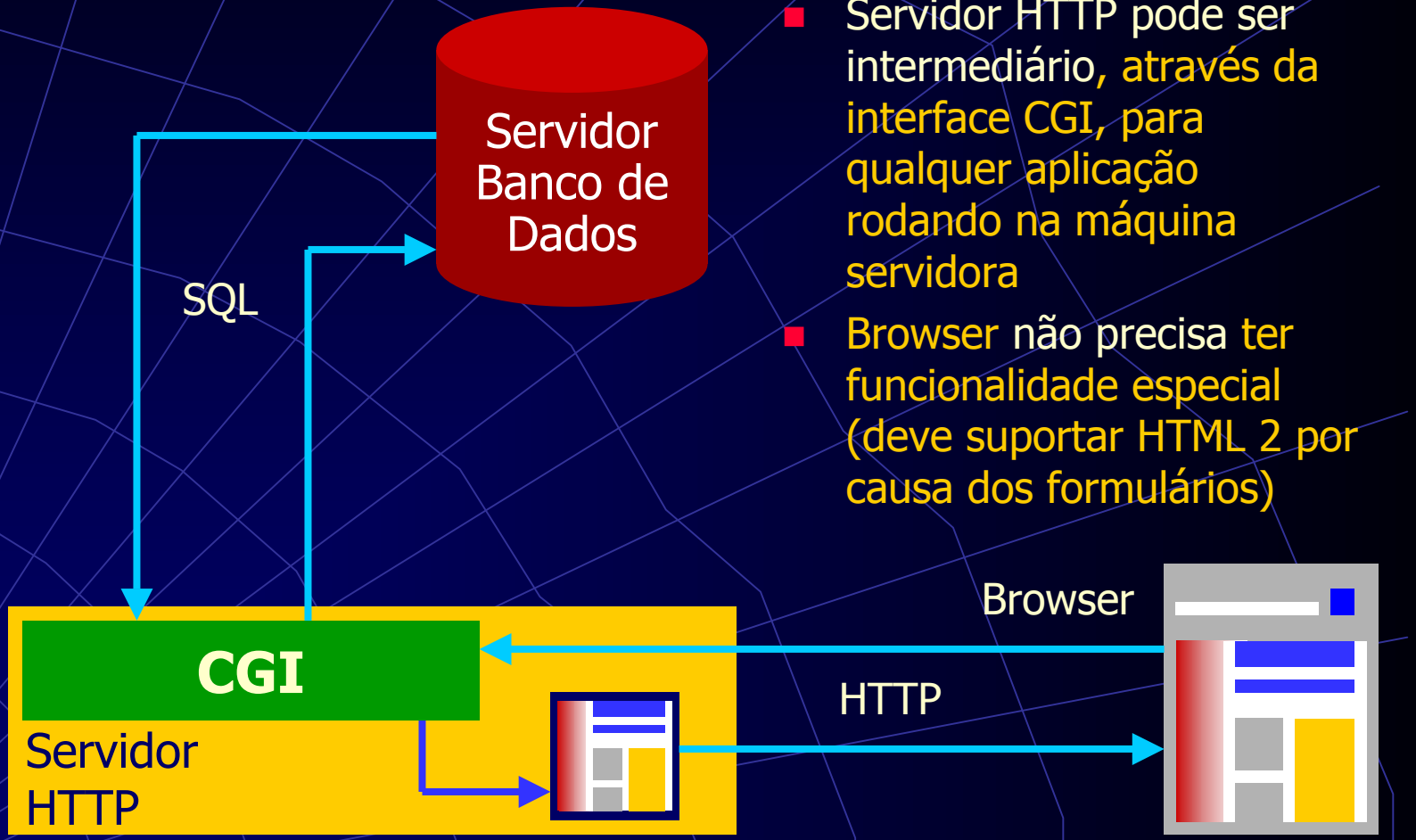
Tecnologias lado-servidor

- Tecnologias que estendem as funções básicas de um servidor HTTP:
  - CGI - Common Gateway Interface
  - Componentes: ISAPI, NSAPI, Apache API, Servlet API
  - *Scripts*: ASP, JSP, LiveWire (SSJS), Cold Fusion, PHP
  - Outras tecnologias proprietárias: Lotus (Domino), Oracle, etc.
- Rodam predominantemente do lado do servidor, portanto, não dependem de suporte por parte dos browsers
  - browsers fornecem apenas a interface do usuário (HTML, applet, plug-in) para serviço remoto
  - Recebem dados através de requisições HTTP (GET e POST)
  - Devolvem dados através de respostas HTTP

CGI - Common Gateway Interface

- Especificação que determina como construir uma aplicação que será executada pelo servidor Web
- Programas CGI podem ser escritos em qualquer linguagem de programação. A especificação limita-se a determinar os formatos de entrada e saída dos dados.
  - O que interessa é que o programa seja capaz de
    - obter dados de entrada a partir de uma requisição HTTP
    - gerar uma resposta HTTP incluindo os dados e parte do cabeçalho
  - Escopo: camada do servidor
    - não requer quaisquer funções adicionais do browser ou do protocolo HTTP

CGI: *Gateway* para aplicações



- Servidor HTTP pode ser intermediário, através da interface CGI, para qualquer aplicação rodando na máquina servidora
- Browser não precisa ter funcionalidade especial (deve suportar HTML 2 por causa dos formulários)

Aplicações CGI

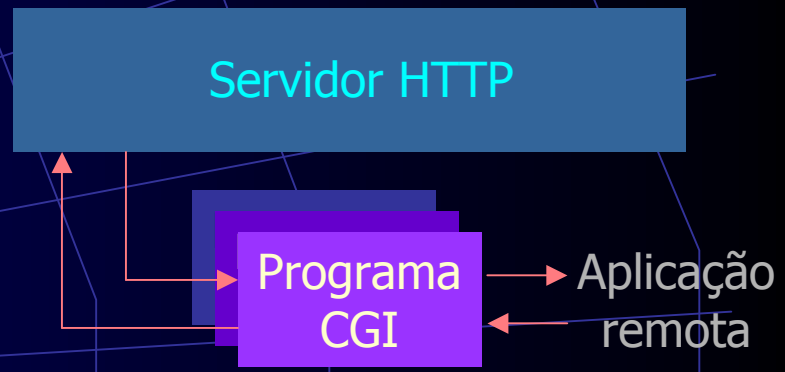
- CGI é a solução usada na maior parte das aplicações Web da Internet. Entre as aplicações mais populares implementadas com CGI estão:
  - Sistemas de busca: interface para Wide Area Information Service (WAIS) e sistemas proprietários
  - Envio de e-mail: interface para servidores SMTP
  - Acesso a dados em sistemas de arquivos
  - Acesso a banco de dados
    - textuais
    - relacionais, via ODBC, JDBC ou protocolos proprietários
    - orientados a objetos
  - Interfaces para servidores de videoconferência, voz, bate-papo
  - Interfaces para aplicações em outras plataformas

Aplicações CGI

- Programas CGI podem ser escritos em qualquer linguagem. As linguagens mais populares são C e Perl.
- A linguagem usada deve ter facilidades para
  - Ler variáveis de ambiente (onde o servidor armazena informações passadas no cabeçalho da requisição).
  - Imprimir dados de 8 bits (e não apenas texto ASCII)
- Linguagens não recomendadas
  - Java: dificuldade de ler propriedades do sistema
  - MS-DOS: impossibilidade de gerar HTML
- Segurança depende do servidor *e* do código
- Veja um *exemplo* de acesso a banco de dados Access com CGI em Perl para Windows (ActivePerl)

CGI é prático... Mas ineficiente!

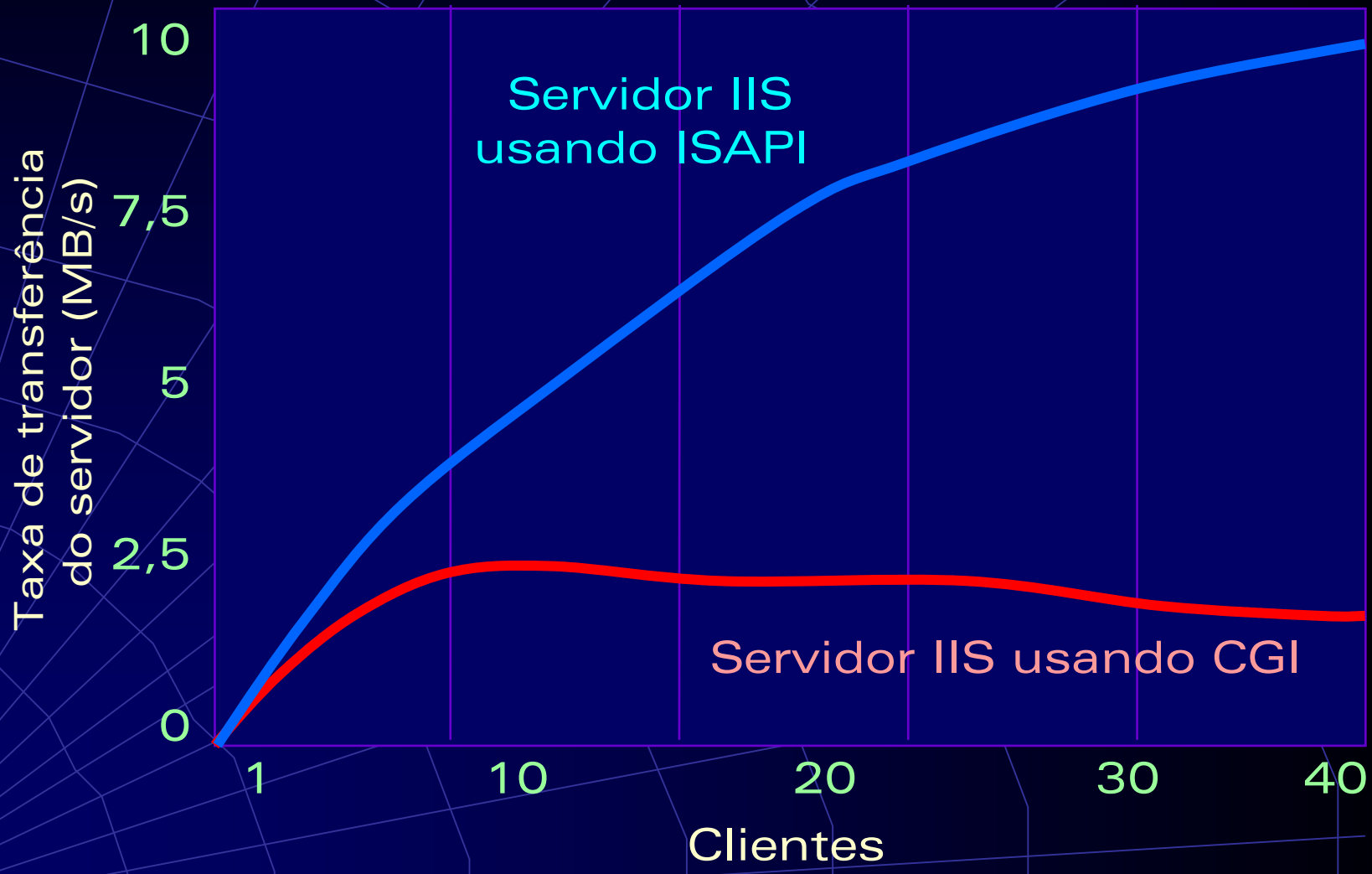
- A interface CGI requer que o servidor sempre execute um programa
  - Um novo processo do S.O. rodando o programa CGI é criado para cada cliente remoto que o requisita.
  - Novos processos consomem muitos recursos, portanto, o desempenho do servidor diminui bastante a cada novo cliente conectado.
- CGI roda como um processo externo, portanto, não tem acesso a recursos do servidor
  - A comunicação com o servidor resume-se à entrada e saída.
  - Não há compartilhamento de dados entre processos (exceto em disco).



Extensões do servidor

- Componentes construídos usando uma API de programação do servidor (plug-ins)
- Podem substituir totalmente o CGI, com vantagens:
  - Toda a funcionalidade do servidor pode ser usada
  - Lida com múltiplos clientes em processos internos (threads)
- Desvantagens:
  - Em geral dependem de plataforma, fabricante e linguagem
  - Soluções proprietárias
- Mais populares
  - ISAPI (Microsoft)
  - NSAPI (Netscape)
  - Apache Server API

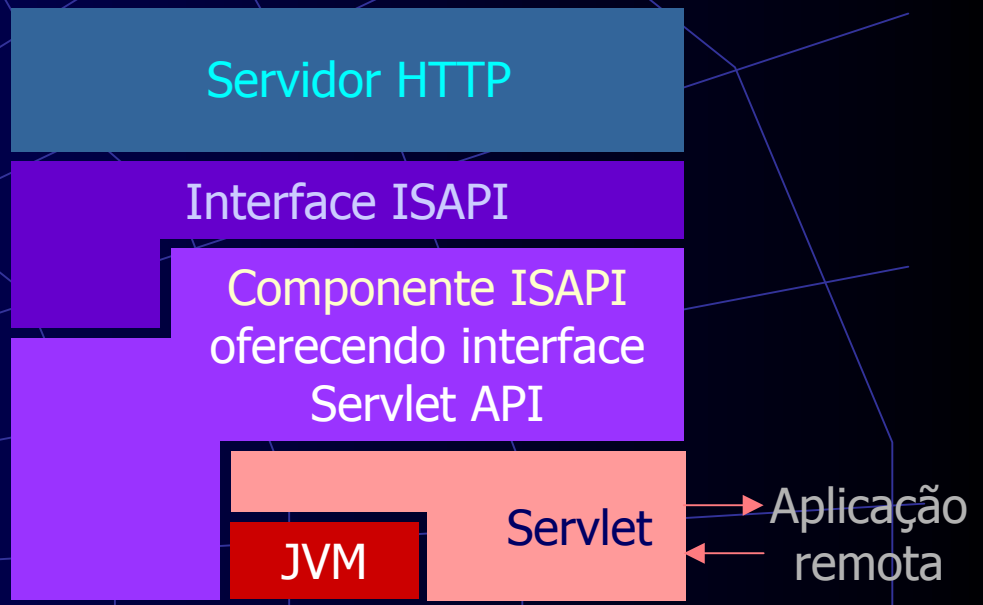
ISAPI vs. CGI



Fonte: PCMagazine La

Servlet API

- API independente de plataforma e praticamente independente de fabricante
- Componentes são escritos em Java e se chamam servlets
- Como os componentes ISAPI, rodam dentro do servidor, mas através de uma Máquina Virtual Java
- Disponível como 'plug-in' (componente API do servidor) para servidores que não o suportam
  - LiveSoftware JRun (servidores Unix, Windows e Mac)



Scripts (roteiros) de servidor

- Mais simples e mais práticos que ISAPI/NSAPI/Servlets
- Mais populares:
  - Microsoft Active Server Pages (ASP)
  - Java Server Pages (JSP)
  - Netscape Livewire (SSJS) e Allaire Cold Fusion
- Código ASP/VBScript/JavaScript (ou JSP/Java) é embutido em página HTML (e consumido pelo servidor):

```
<% ... %> <!-- ASP VBScript -->
<% if x > 0 then %> <h1>Maior</h1>
<% else %> <h1> Menor </h1><% end if %>
```

- Páginas são processadas e roteiros são executados pelo servidor, que os consome (nada chega ao browser)
  - browser recebe uma página HTML comum

Persistência de dados

- HTTP não preserva o estado de uma sessão
  - Eficiente para sistemas de informação
  - Ineficiente para várias aplicações Web
- Soluções
  - Extensões ao protocolo HTTP
    - Desvantagem: browser e servidor precisam suportar a extensão
  - Seqüência de páginas/aplicações (ASP, JSP, CGI)
    - Desvantagens: seqüência não pode ser quebrada; mesmo que página só contenha HTML simples, precisará ser gerada por aplicação
  - Cookies (informação armazenada no cliente)
    - Desvantagens: espaço e quantidade de dados reduzidos; browser precisa suportar a tecnologia

Cookies

- Padrão Internet (RFC) para persistência de informações
- Um cookie é uma pequena quantidade de informação que o servidor armazena no cliente
  - Par nome=valor. Exemplos: usuario=paulo, num=123
  - Escopo no servidor: domínio e caminho da página
  - Escopo no cliente: browser
  - Duração: uma sessão ou tempo determinado (cookies persistentes)
- Forma como são armazenados depende do browser. Restrições:
  - 2000 caracteres por cookie
  - 20 cookies por domínio
  - 300 cookies por browser

Gateway com Cookies

- Permite persistência de dados entre páginas
  - (HTTP não mantém sessão)
- Dados são guardados na máquina do cliente e recuperados quando necessários

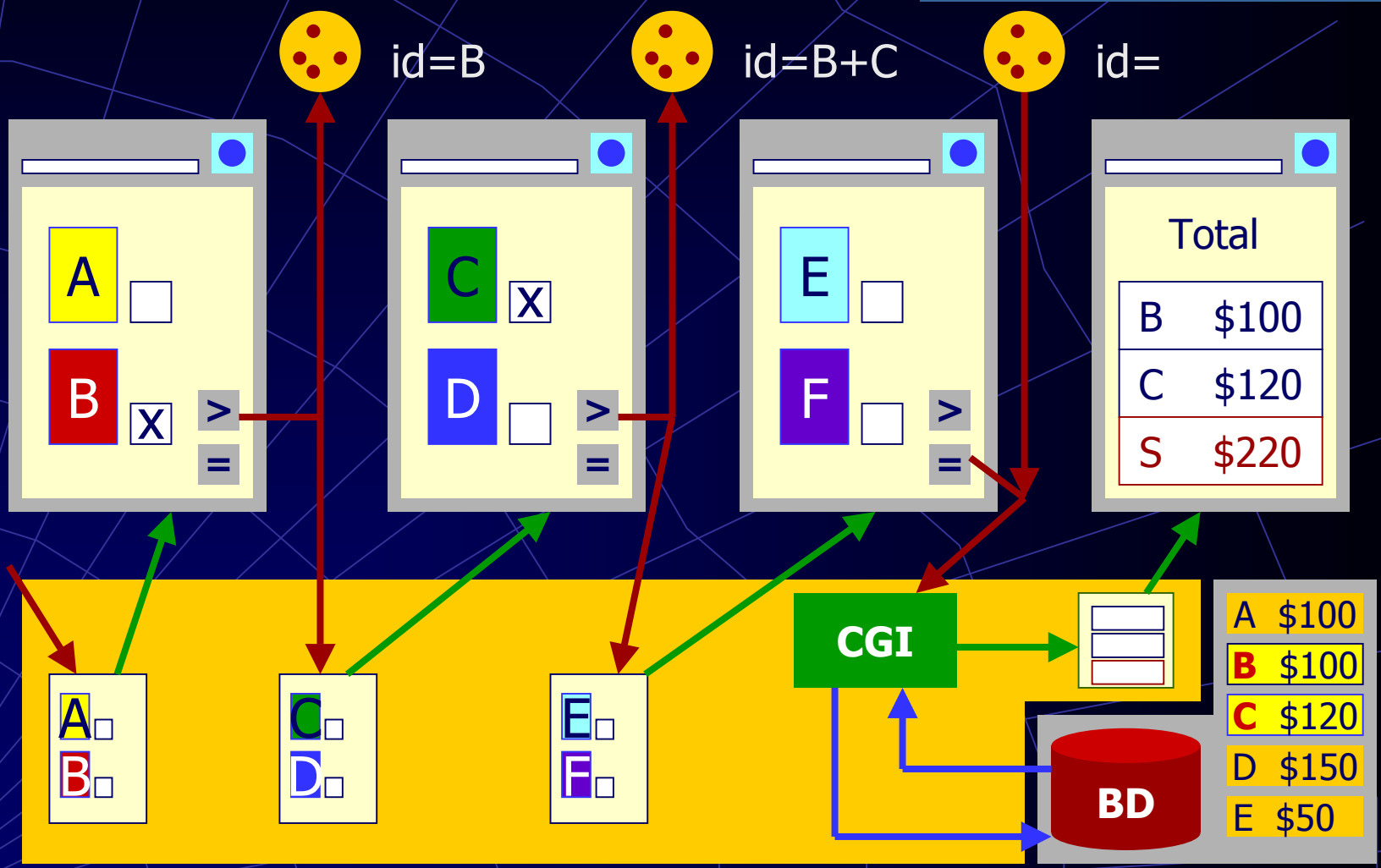
Carrinho de Compras

- Cookies são essenciais nas aplicações de comércio eletrônico
  - Mantém informações sobre o usuário entre páginas e sessões (códigos, senhas)
  - Mantém preferências do usuário entre páginas e sessões (opções do carrinho de compras)
- Cookies são apagados quando
  - a sessão acaba (o browser é fechado) quando não-persistentes
  - seu tempo de vida se esgota (expiration date) quando persistentes
  - quando redefinidos com mesmo nome e data de expiração no passado
- Segurança depende do servidor
  - Cookies marcados como 'secure' só são transferidos via conexão HTTP segura (SSL ou SHTTP)

Exemplo: Loja virtual

= Lê cookie, apaga-o e envia dados para CGI

> Guarda cookie e chama próxima página



Tecnologias lado-cliente

- Estendem a funcionalidade básica do browser (que é apresentação de informação)
- Permitem criar uma interface do usuário dinâmica
  - Tratamento de eventos
  - Alteração dinâmica do conteúdo ou da apresentação
  - Realização de cálculos e computação
  - Disposição de recursos não disponíveis no browser
- Principais tecnologias
  - Extensões do HTML (scripts): JavaScript, VBScript, linguagens proprietárias
  - Extensões do browser (componentes): Applets, ActiveX, Plug-ins

Páginas dinâmicas

- Aplicações no cliente transformam páginas HTML em interfaces dinâmicas
  - maior interatividade
  - interfaces mais amigáveis e mais poderosas
  - geralmente, menos segurança
- Principais soluções
  - **Componentes** (extensões do browser): applets Java, ActiveX, plug-ins Flash, plug-ins VRML, plug-ins de audio e vídeo
  - **Scripts** (extensões do HTML): VBScript, JavaScript

Applets Java

- Programas escritos na linguagem Java, especialmente criados para uso na Web
- Funcionam como extensão do browser
- Ocupam espaço determinado (como imagens)

```
<p>Este é uma applet<br/>
<applet code="form.class"
height="100" width="300">
</applet> para entrada
```

- São programas compilados (o código-fonte não é visível)
- Veja *exemplos* de Applets

http://www.jpj.nasa.gov/wits
Simulador do Mars Pathfinder na superfície de Marte (applet Java interativo)

Plug-ins e ActiveX

- Para fazer as mesmas coisas, pode-se usar plug-ins (Netscape), ActiveX (Microsoft) ou applets.
- Applets, escritos em Java, são independentes de plataforma
  - desvantagem: nivela a funcionalidade por baixo; restrições de segurança podem ser excessivas
- Plug-ins e Componentes ActiveX, geralmente escritos em C++ ou VB, dependem do browser e plataforma.
  - vantagem: mais flexibilidade, mais integração com S.O.; desvantagem: menos segurança
  - Passam a fazer parte do browser como componente (geralmente só são descarregados uma vez)
- Descritores HTML:
  - <applet></applet>, <embed></embed> e <object></object>

HTML 4

Scripts : extensões do HTML

- Forma mais flexível de estender o HTML
- Código geralmente fica visível na página:

```
<head>
<script language="JavaScript"><!--
    function soma(a, b) {
        return a + b;
    }
//--></script></head>
```

- Linguagens de roteiro *(script)* mais populares
  - VBScript: baseado na sintaxe do Visual Basic
  - JavaScript/JScript: sintaxe semelhante a de Java
- Código é interpretado diretamente pelo browser (e não por uma máquina virtual, como ocorre com os applets)

JavaScript e ECMAScript

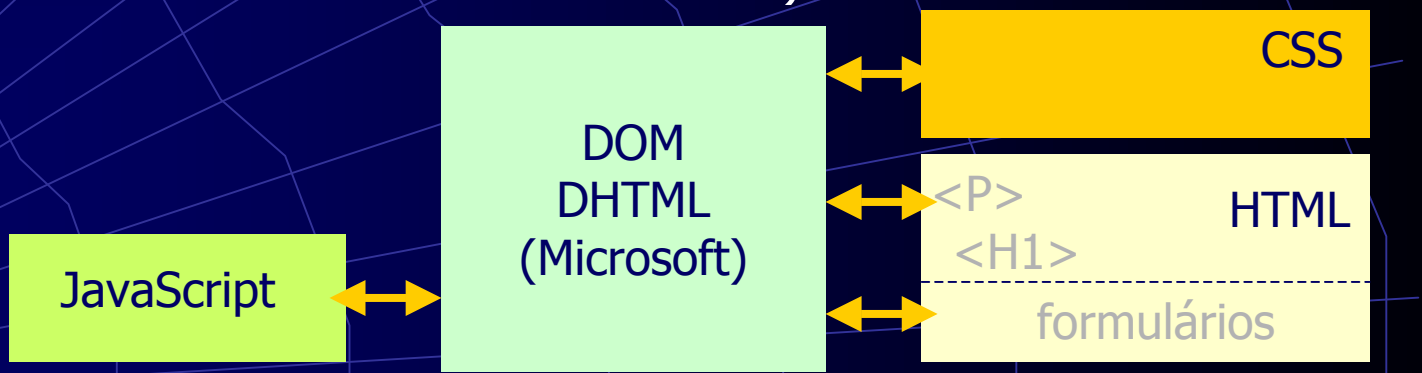
- JavaScript não é Java. Possui sintaxe semelhante, mas é interpretada, baseada em objetos e bem menor.
- JavaScript pode ser usada no browser ou no servidor. A linguagem possui duas partes
  - Núcleo (padrão ECMA chamado de ECMAScript)
  - Modelo de objetos do documento (quando usada no browser) ou do servidor (quando usada no servidor em tecnologias ASP, Livewire, etc.)
- O núcleo da linguagem define as estruturas de programação, sintaxe e objetos de propósito geral.
- Quando usada no browser, várias estruturas do HTML são acessíveis como 'objetos' JavaScript, permitindo que a linguagem os manipule.

JavaScript e DOM

- O Document Object Model do JavaScript mapeia algumas estruturas do HTML a objetos (variáveis) da linguagem
  - Propriedades dos objetos (e conseqüentemente dos elementos da página) poderão ser alteradas em tempo de execução
  - Mapeamento restringe-se a elementos de formulário, vínculos, imagens e atributos da janela do browser.
  - Permite validação de campos dos formulários, cálculos locais, imagens dinâmicas, abertura de novas janelas, controle de frames, etc.
  - Não mapeia parágrafos, títulos ou folhas de estilo

Dynamic HTML

- O Document Object Model do DHTML mapeia todos os elementos do HTML e folha de estilos, tornando-os acessíveis como objetos JavaScript
- Desvantagem: compatibilidade
  - A Microsoft utiliza um Document Object Model
  - A Netscape utiliza outro modelo...
  - A W3C tenta padronizar outro diferente dos demais (se assemelha mais ao da Microsoft)



Sumário

- Tecnologias de apresentação (cliente)
  - HTML, CSS, XML
- Tecnologias interativas (servidor)
  - CGI
  - APIs: ISAPI, Servlet API
  - Scripts: ASP, JSP
- Persistência
  - Cookies
- Tecnologias interativas (cliente)
  - Applets, Plug-ins e ActiveX
  - JavaScript e DHTML

Quando usar CGI

- Quando for necessário disponibilizar via browser, informações geradas ou disponíveis no servidor
  - Gateway para aplicações do servidor
  - Acesso a bancos de dados intermediado pelo servidor
  - Acesso a informações armazenadas no servidor
  - Geração de páginas de acordo com informações no servidor
- Quando o programa terá aplicação em servidores de fabricantes diferentes
- Pontos a observar
  - Roda fora do servidor (que inicia o processo no S.O.)
  - Desempenho cai de forma acentuada quando há muitos clientes (um processo para cada cliente)
  - Não aproveita recursos do servidor (*multithreading*, extensões, *drivers*)

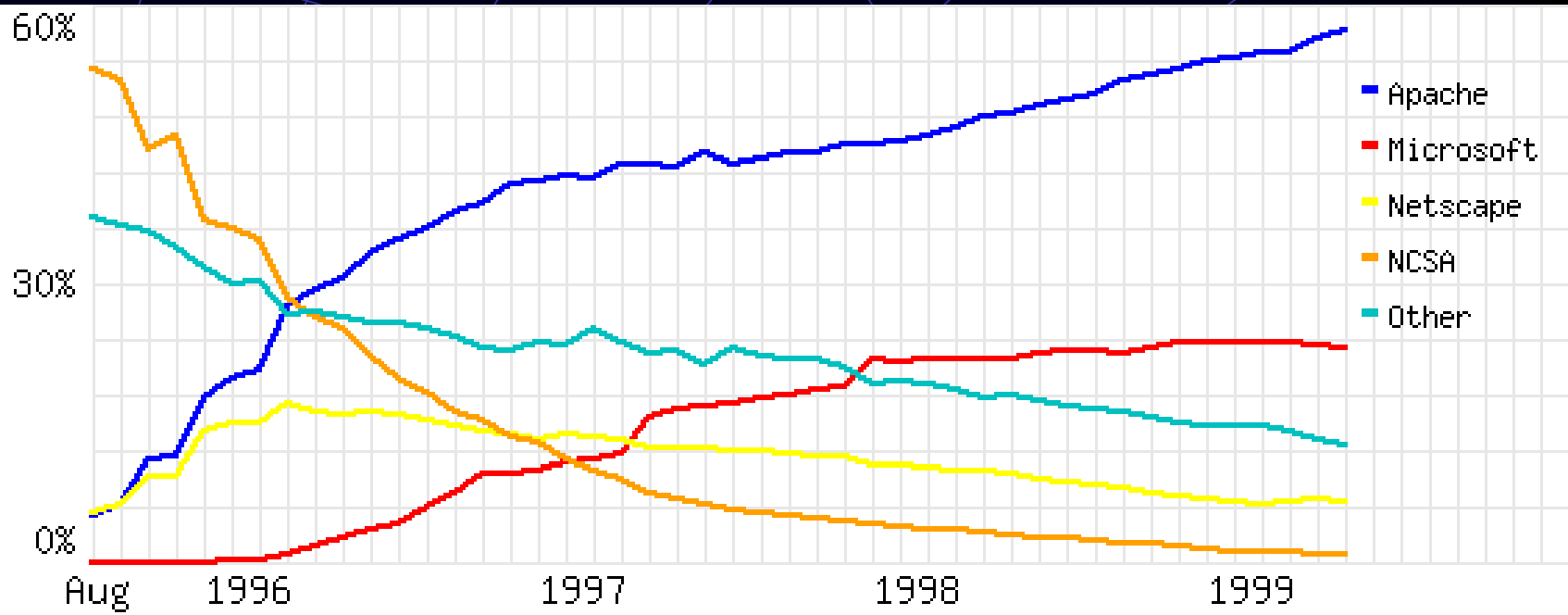
Quando usar Servlets/ISAPI

- Quando for necessário oferecer serviços como os que são oferecidos por CGI de forma mais eficiente
  - Acesso a bancos de dados por múltiplos clientes simultâneos
  - Realização de múltiplas operações no servidor
  - Extensão de recursos já disponíveis no servidor
- Pontos a observar
  - ISAPI é exclusivo de servidores Microsoft ou compatíveis. Os componentes (DLLs) são dependentes de plataforma.
  - Servlets rodam através de plug-in em todos os principais servidores. Os componentes (arquivos CLASS) são independentes de plataforma (escritos em Java)
  - Componentes precisam ser compilados e instalados no servidor. Alterações podem exigir o re-início do servidor.

Quando usar JSP/ASP

- Quando for necessário gerar páginas dinâmicas a partir de informações no servidor (ex: banco de dados)
  - Java compilado on-the-fly (JSP) ou JavaScript/VBScript interpretado (ASP)
  - Programa destino de formulário e ao mesmo tempo meta-página HTML (desenvolvimento simplificado)
  - Integração com servlets (JSP) ou ADO/ISAPI (ASP)
- Quando for necessário oferecer serviços simples do tipo CGI ou Servlet/ISAPI com manutenção simplificada
- Ideal para rotinas simples, geração de HTML, seqüências de páginas/aplicações
- Pontos a observar
  - Desempenho menor que servlets/ISAPI (porém, em média, melhor que CGI, já que novos processos não são abertos.

Qual servidor?



Servidor	Maio 1999	Percentual
Apache	3.097.993	57.22 %
Microsoft-IIS/PWS	1.244.808	22.99 %
Netscape-Enterprise/FTrack	356.207	6.58 %

Fonte: www.netcraft.com/survey (Universo: **5.414.325** sites)

Quando usar JavaScript/DHTML

- Para automação de operações do browser
  - Validação de campos de entrada de dados
  - Realização de cálculos baseados em fórmulas
  - Quaisquer operações que necessitem programação mas que não dependam de informações (em tempo real) no servidor.
- Pontos a observar
  - O cliente deve suportar a versão de JavaScript utilizada, caso contrário
    - aparecerão mensagens de erro (s/ sentido para o usuário)
    - a navegação poderá ser impossibilitada
    - a visualização de informações poderá ser prejudicada
  - É preciso criar alternativas (de acessibilidade) caso o público-alvo do *site* utilize browsers antigos

Quando usar Applets

- Para acrescentar recursos à interface oferecida pelo browser
  - Implementar suporte a protocolos proprietários
  - Implementar suporte a formatos de mídia proprietários
  - Melhorar a interatividade com mais eventos, recursos de manipulação gráfica, desenhos e animações
  - Oferecer uma interface externa à janela do browser
- Pontos a observar
  - O cliente deve possuir uma Máquina Virtual Java compatível com a versão de Java com a qual foi compilado o Applet, caso contrário o Applet não funcionará.
  - Existem diferenças entre plataformas que podem fazer com que o applet não funcione da forma esperada.
  - Evite que a navegação do *site* dependa de applets

Quando usar Cookies

- Quando for necessário preservar informações entre páginas
  - Comércio eletrônico (carrinho de compras)
  - Formulários cumulativos de múltiplas páginas
  - Navegação personalizada
- Quando for necessário que informações persistam por mais de uma sessão
  - Páginas personalizadas com preferências do usuário
  - Páginas dependentes da freqüência de acessos do usuário
- Pontos a observar
  - O cookie só existe para um determinado browser e cliente
  - O cookie é definido para um domínio e caminho (dentro do domínio) específicos. Fora desse escopo, ele não existe
  - Cookies definidos como 'seguros' só podem ser lidos quando a transferência é feita via servidor seguro (SHTTP, SSL)

Fontes de referência

- World Wide Web Consortium. HTML, CSS, DOM, HTTP, Cookies, CGI, XML. http://www.w3.org
- Jennifer Niederst. *Web Design in a Nutshell*. O'Reilly and Associates, 1999.
- Netscape. JavaScript, DHTML, Cookies, NSAPI, Plug-ins http://developer.netscape.com
- Sun. Java e Applets http://java.sun.com. Servlets e JSP, JRun. http://jserv.java.sun.com
- Microsoft. ASP, ISAPI, PWS, ActiveX, JScript, VBScript. http://www.microsoft.com/sitebuilder
- ECMA. ECMAScript (JavaScript). http://www.ecma.org

Aplicações Web

tecnologias e soluções

Maio 1999

F I M

Helder da Rocha

http://www.ibpinet.net/helder/helder.html

hlsr@uol.com.br