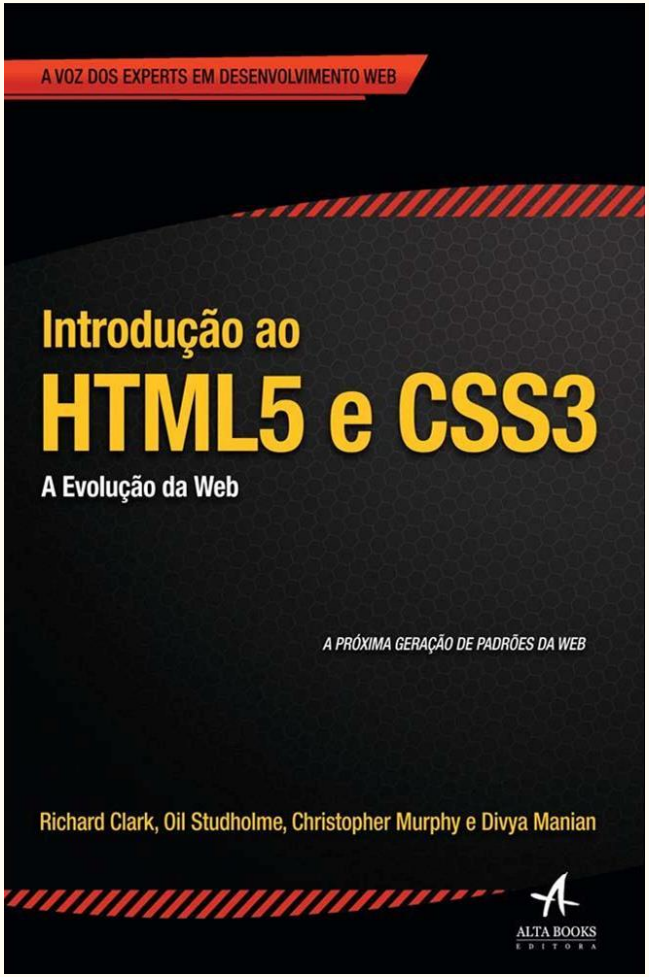
AJAX

Prof. Victor Farias

V 1.0

Referências

w3schools.com



JSON

JSON

- JSON é um estrutura de dados derivada da notação de objetos do JS
- Sintaxe:
  - Os dados estão dispostos em pares nome/valor
  - Os dados são separados por vírgula
  - Os objetos são colocados em chaves
  - Listas são colocadas entre colchetes

JSON Exemplos

- Exemplo par nome/valor: `"firstName":"John"`
  - Nomes JSON requerem aspas duplas
  - Valores JSON podem ser números, strings, booleanos, listas, objetos ou null
- Exemplo objetos: `{"firstName":"John", "lastName":"Doe"}`
- Exemplo lista:

```
"employees":[

    {"firstName":"John", "lastName":"Doe"},

    {"firstName":"Anna", "lastName":"Smith"},

    {"firstName":"Peter","lastName":"Jones"}

]
```

JSON - Como usar

- JSON é muito usado para ler dados do servidor e mostrar na página web
- Texto JSON para JS Object:

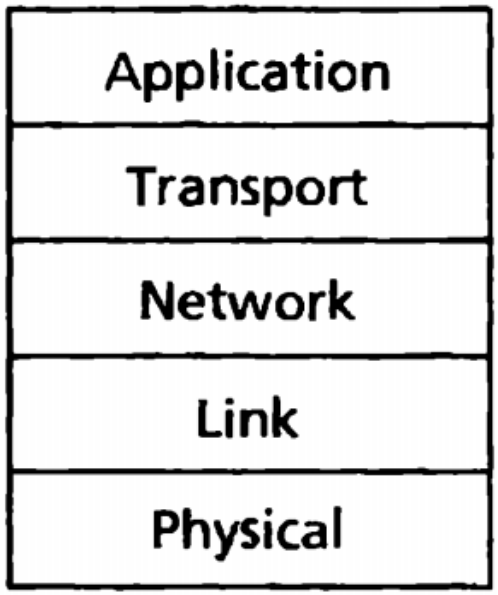
```
var text = '{ "employees" : [' +
'{ "firstName":"John" , "lastName":"Doe" },' +
'{ "firstName":"Anna" , "lastName":"Smith" },' +
'{ "firstName":"Peter" , "lastName":"Jones" } ]}';

var obj = JSON.parse(text); // Parsing JSON texto to JavaScript Object
```

- JS Object para Texto JSON:

```
var obj = [{a:1,b:1,c:1}, {a:2,b:2,c:3}]
var json = JSON.stringify(obj);
```

Protocolo HTTP

Pilha de Protocolos na Internet



5 camadas

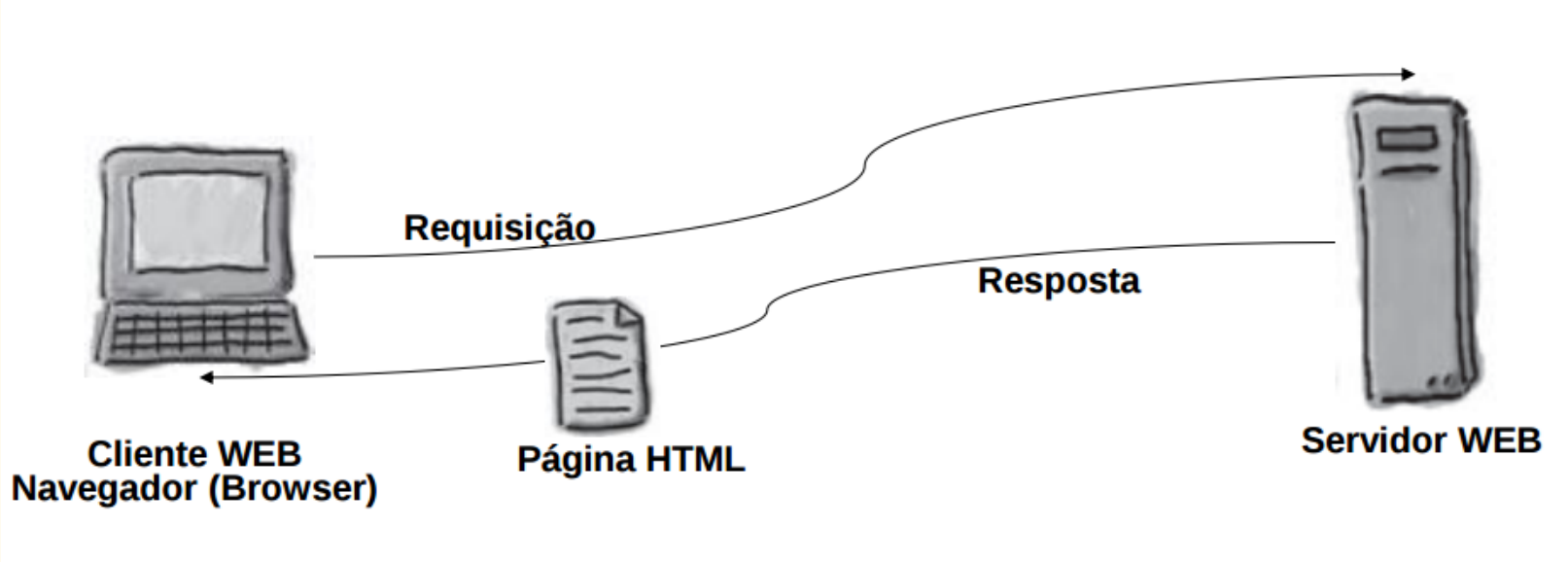
Camada	Protocolo
5.Aplicação	HTTP, SMTP, FTP, SSH, RTP, Telnet, SIP, RDP, IRC, SNMP, NNTP, POP3, IMAP, BitTorrent, DNS, Ping ...
4.Transporte	TCP, UDP, SCTP, DCCP ...
3.Rede	IP (IPv4, IPv6) , ARP, RARP, ICMP, IPSec ...
2.Enlace	Ethernet, 802.11 WiFi, IEEE 802.1Q, 802.11g, HDLC, Token ring, FDDI, PPP, Frame Relay,
1.Física	Modem, RDIS, RS-232, EIA-422, RS-449, Bluetooth, USB, ...

Endereço IP e Portas

- Endereço IP
  - Identifica um host na rede
  - Cada interface de rede tem um IP
  - ex: 200.21.32.43
  - URLs são traduzidos em IP usando DNS (globo.com.br -> 186.192.90.5)
- Portas
  - Identificam os processos de origem e fim
  - Permite a comunicação de diversas aplicações na mesma máquina
  - Cada aplicação recebe e envia requisições por uma porta
  - ex: Servidor Web, por padrão, recebem requisições na porta 80

Arquitetura Cliente Servidor

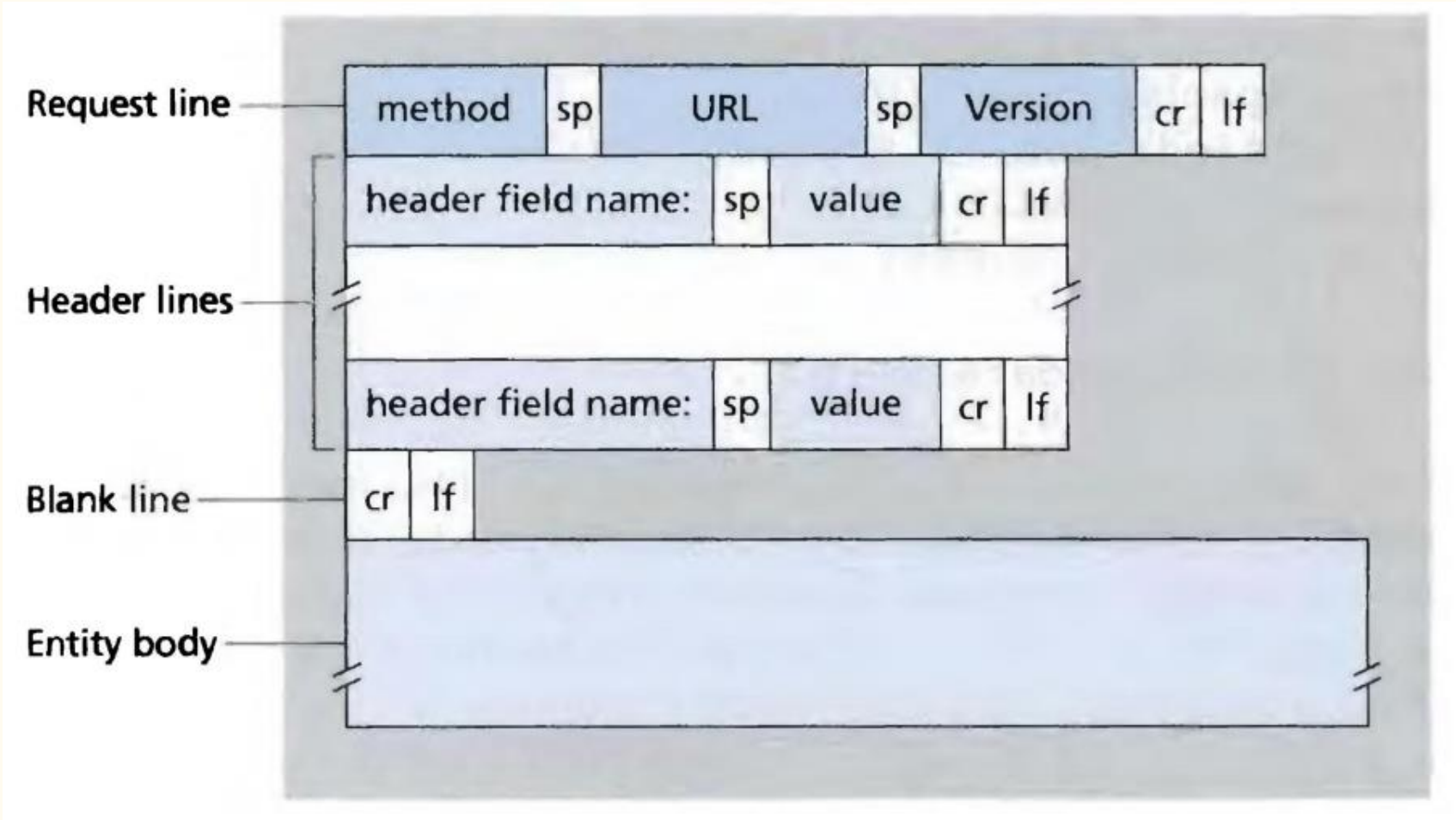
- Servidor
  - Aplicação que fornece serviço
  - Aceita requisições através da rede, em um porta conhecida, e retorna o resultado
- Cliente
  - Processo que requisita um serviço
  - Para receber resposta, o cliente aloca um porta arbitrária



Protocolo HTTP

- HTTP = **H**ypertext **T**ransfer **P**rotocol ou Protocolo de Transferência de Hipertexto
- Protocolo usado para transferir dados na WEB
- Funcionamento:
  - O cliente envia uma requisição HTTP para o servidor
  - O servidor envia uma resposta HTTP ao cliente

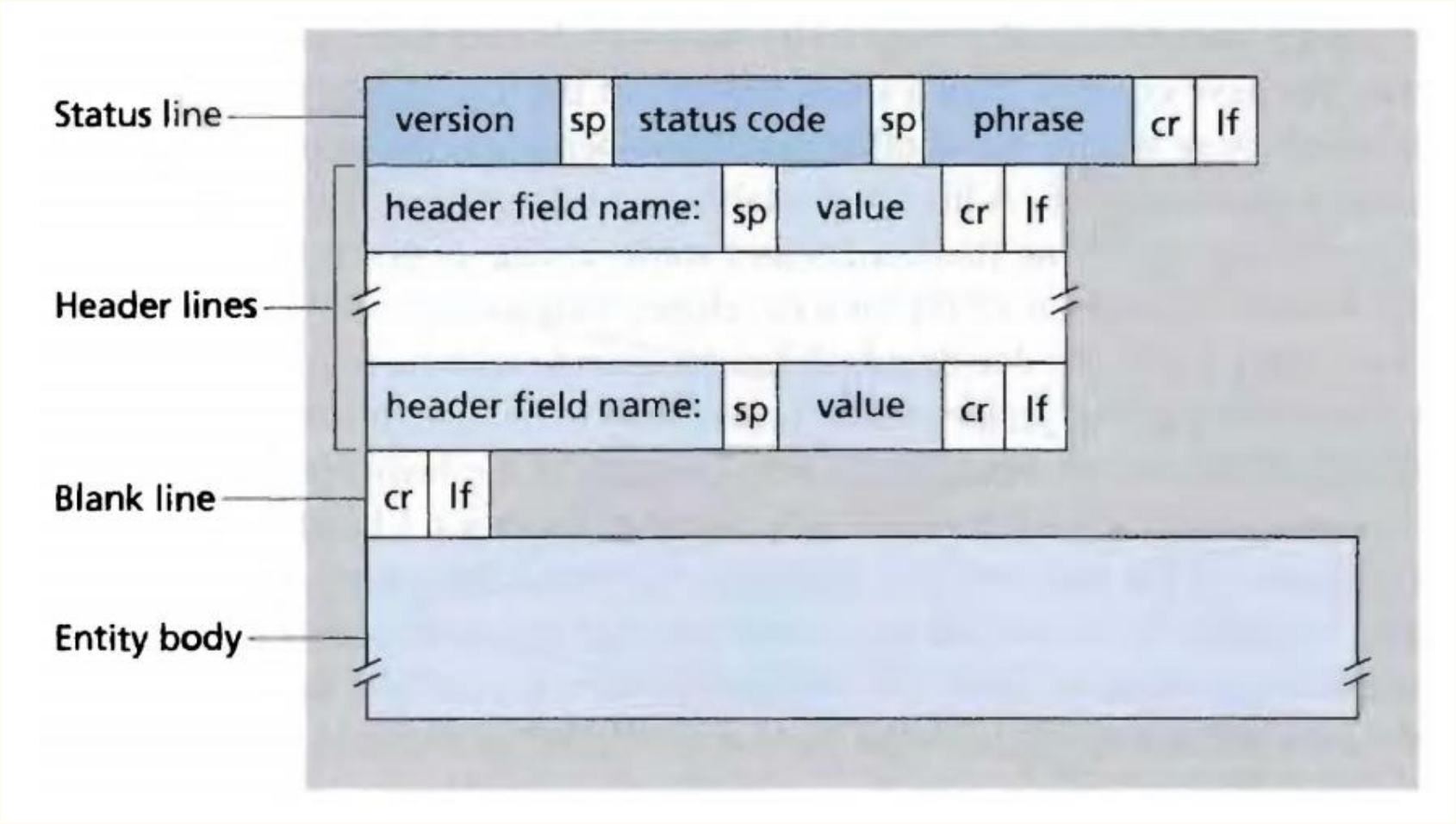
Composição da Requisição HTTP



Métodos de Requisição

- Métodos usados em requisições HTTP
  - GET – Solicita algum recurso
    - Dados são anexados à URL, ficando visíveis ao usuário
  - POST – Envia dados referentes ao recurso especificado para serem processados
    - Dados são incluídos no corpo do comando
  - PUT – Envia certo recurso
    - Em geral, é usado para atualizar dados
  - DELETE – Exclui o recurso
  - HEAD – Variação do GET em que o recurso não é retornado
    - Usado para obter metainformações por meio do cabeçalho da resposta, sem ter que recuperar todo o conteúdo.

Resposta HTTP



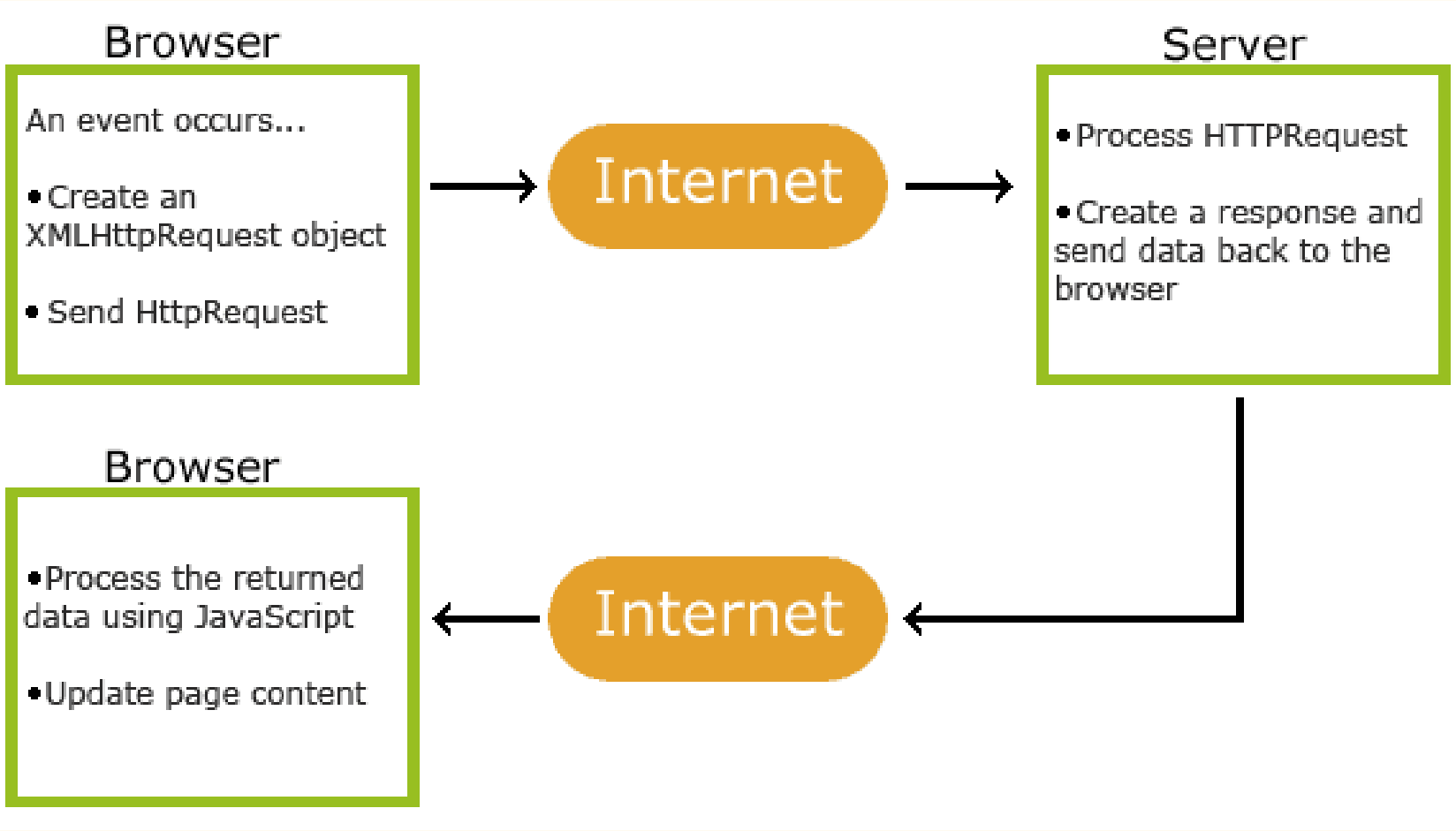
Código de Status

- O código de status é formado por três dígitos e o primeiro dígito representa a classe que pertence classificada em cinco tipos:
  - 1xx: Informational (Informação) – utilizada para enviar informações para o cliente de que sua requisição foi recebida e está sendo processada
  - 2xx: Success (Sucesso) – indica que a requisição do cliente foi bem sucedida
  - 3xx: Redirection (Redirecionamento) – informa a ação adicional que deve ser tomada para completar a requisição
  - 4xx: Client Error (Erro no cliente) – avisa que o cliente fez uma requisição que não pode ser atendida
  - 5xx: Server Error (Erro no servidor) – ocorreu um erro no servidor ao cumprir uma requisição válida
- O protocolo HTTP define somente alguns códigos em cada classe, mas cada servidor pode definir seus próprios códigos

AJAX

AJAX

- Como AJAX é possível
  - Atualizar uma página web sem recarregar a página
  - Requisitar dados ao servidor
  - Receber dados do servidor
  - Enviar dados para servidor
- AJAX não é uma linguagem de programação
- AJAX = **A**synchronous **J**avaScript **A**nd **X**ML
- AJAX é uma combinação de
  - objeto XMLHttpRequest para requisitar dado do servidor
  - JavaScript e HTML DOM para mostrar o dado
- Embora pareça que AJAX só pode transmitir XML, ele também pode transportar texto pleno ou JSON


Browser
An event occurs...
• Create an XMLHttpRequest object
• Send HttpRequest
Internet
Server
• Process HTTPRequest
• Create a response and send data back to the browser
Internet
Browser
• Process the returned data using JavaScript
• Update page content

Objeto XMLHttpRequest

- Todos os navegadores modernos suportam XMLHttpRequest
- O objeto XMLHttpRequest é usado para trocar dados com o servidor
- Protocolo HTTP
- Criar objeto XMLHttpRequest
  - var xhttp = new XMLHttpRequest();
- Fazer requisição
  - xhttp.open("GET", "ajax_info.txt", true);
  - xhttp.send();

Métodos do objeto XMLHttpRequest

Method	Description
open(*method*, *url*, *async*)	Specifies the type of request *method*: the type of request: GET or POST *url*: the server (file) location *async*: true (asynchronous) or false (synchronous)
send()	Sends the request to the server (used for GET)
send(*string*)	Sends the request to the server (used for POST)

Method	Description
setRequestHeader(*header*, *value*)	Adds HTTP headers to the request *header*: specifies the header name *value*: specifies the header value

Exemplo Requisições

```
// Requisição GET
var xhttp = new XMLHttpRequest();
xhttp.onreadystatechange = function() {
  if (this.readyState == 4 && this.status == 200) {
           document.getElementById("demo").innerHTML = this.responseText;
  }
};
xhttp.open("GET", "ajax_info.txt", true);
xhttp.send();
```

- É possível modificar esse código para:
  - fazer uma requisição post
    - `xhttp.open("POST", "ajax_test.asp", true);`
  - adicionar cabeçalho
    - `xhttp.setRequestHeader("Content-type", "application/x-www-form-urlencoded");`
  - adicionar dados no corpo da mensagem
    - `xhttp.send("fname=Henry&lname=Ford");`

Resposta do Servidor

- A propriedade **readyState** representa o status do objeto XMLHttpRequest
- A propriedade **onreadystatechange** define a função a ser executada com o **readyState** mudar
- A propriedade **status** e statusText contém o código de retorno da resposta e o texto associado ao status do retorno respectivamente

Property	Description
onreadystatechange	Defines a function to be called when the readyState property changes
readyState	Holds the status of the XMLHttpRequest. 0: request not initialized 1: server connection established 2: request received 3: processing request 4: request finished and response is ready
status	200: "OK" 403: "Forbidden" 404: "Page not found" For a complete list go to the Http Messages Reference
statusText	Returns the status-text (e.g. "OK" or "Not Found")

Resposta do Servidor

Propriedades para recuperar corpo da mensagem

Property	Description
responseText	get the response data as a string
responseXML	get the response data as XML data

Métodos para recuperar cabeçalho da resposta

Method	Description
getResponseHeader()	Returns specific header information from the server resource
getAllResponseHeaders()	Returns all the header information from the server resource

Perguntas?

Prof. Victor Farias