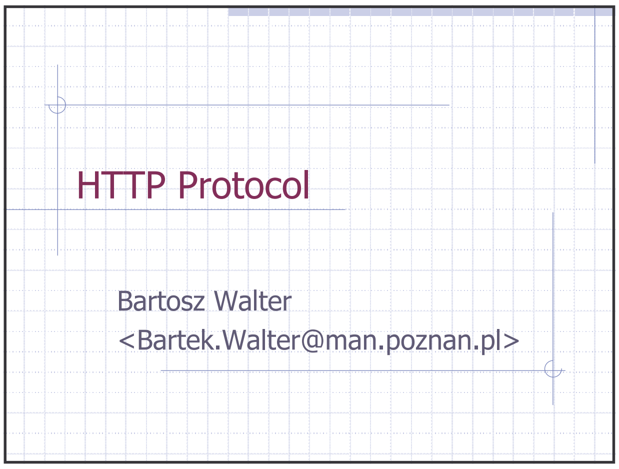
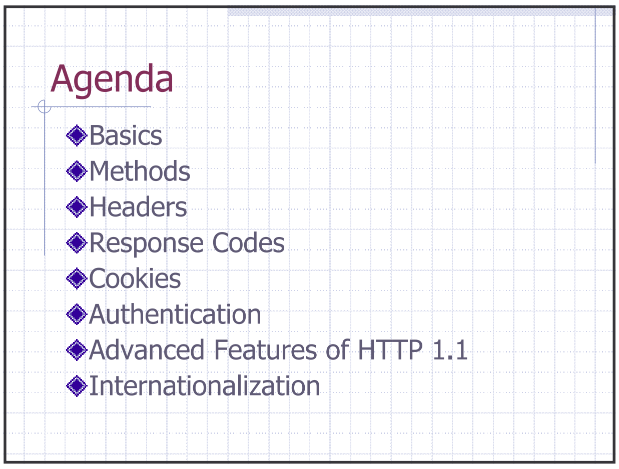
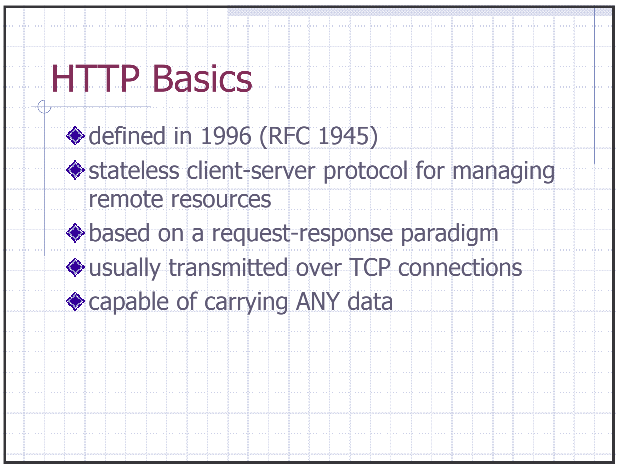


HTTP Protocol

Bartosz Walter
<Bartek.Walter@man.poznan.pl>

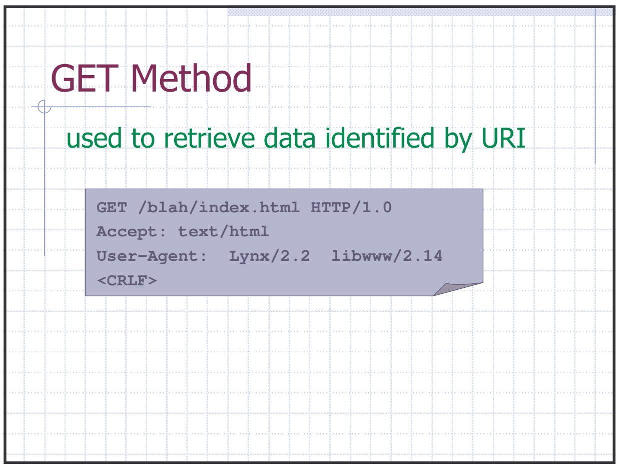


Agenda



HTTP Basics

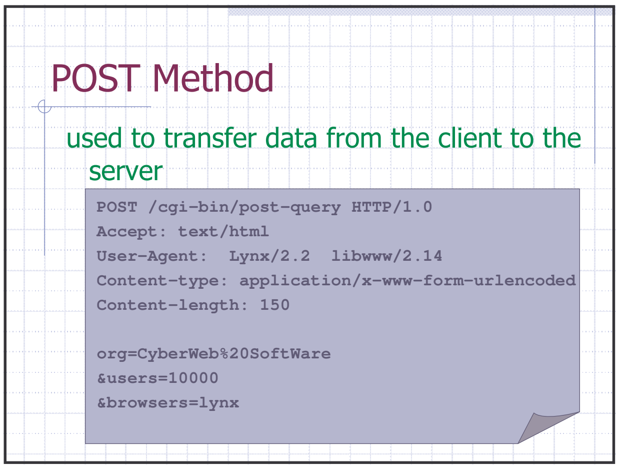
- defined in 1996 (RFC 1945)
- stateless client-server protocol for managing remote resources
- based on a request-response paradigm
- usually transmitted over TCP connections
- capable of carrying ANY data



GET Method

used to retrieve data identified by URI

```
GET /blah/index.html HTTP/1.0
Accept: text/html
User-Agent:  Lynx/2.2  libwww/2.14
<CRLF>
```

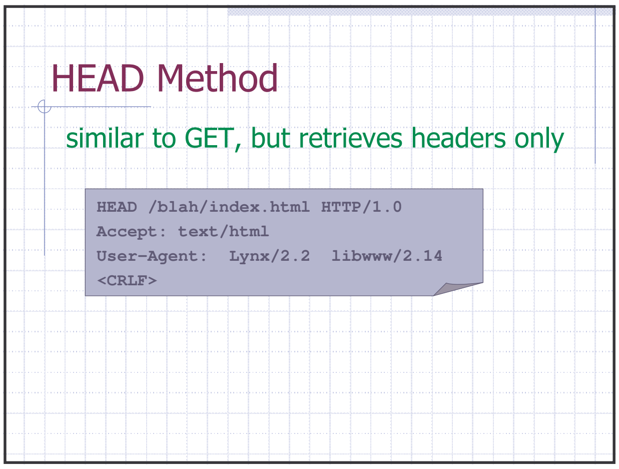


POST Method

used to transfer data from the client to the server

```
POST /cgi-bin/post-query HTTP/1.0
Accept: text/html
User-Agent:  Lynx/2.2  libwww/2.14
Content-type: application/x-www-form-urlencoded
Content-length: 150

org=CyberWeb%20SoftWare
&users=10000
&browsers=lynx
```



HEAD Method

similar to GET, but retrieves headers only

```
HEAD /blah/index.html HTTP/1.0
Accept: text/html
User-Agent:  Lynx/2.2  libwww/2.14
<CRLF>
```

PUT Method

requests that the object be stored under the supplied URI - thus allowing a client to write a file to a server

DELETE Method

Requests that the object be removed from the supplied URI - thus allowing a client to delete a file to a server.

Further the URI becomes invalid for subsequent requests.

OPTIONS Method

a way for a client to learn about the capabilities of a server without actually requesting a resource

for example, a proxy can verify that the server complies with a specific version of the protocol

Request

initial line
headers
empty line
body

```
GET /index.html HTTP/1.0
Host: www.wally.pl
User-Agent: MSIE/Mozilla
<CRLF>
<CRLF>
<data>
```

Response

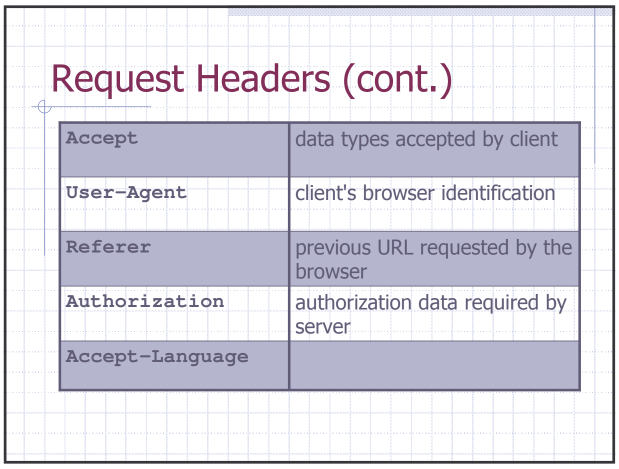
initial line
headers
empty line
body

```
HTTP/1.0 200 OK
Date: Sunday,
   25 November 2001
   18:42:05 GMT
Content-Type: text/html
Content-Length: 109

<data>
```

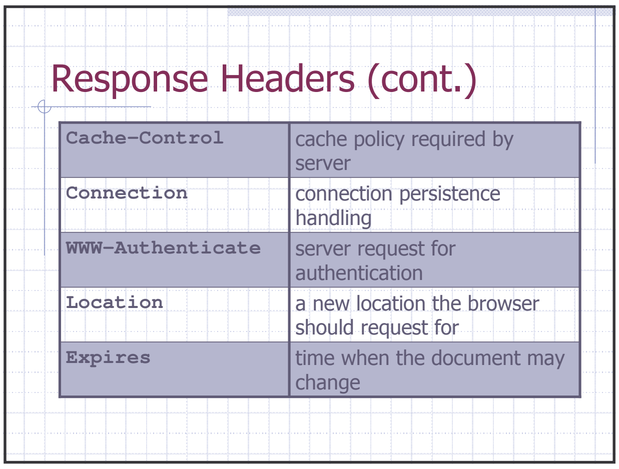
Headers in General

General form	`name: value`
Length span	usually single line (with exceptions)
Case sensitivity	not for names, allowed for values
Variety	16 defined in HTTP 1.0 46 defined in HTTP 1.1



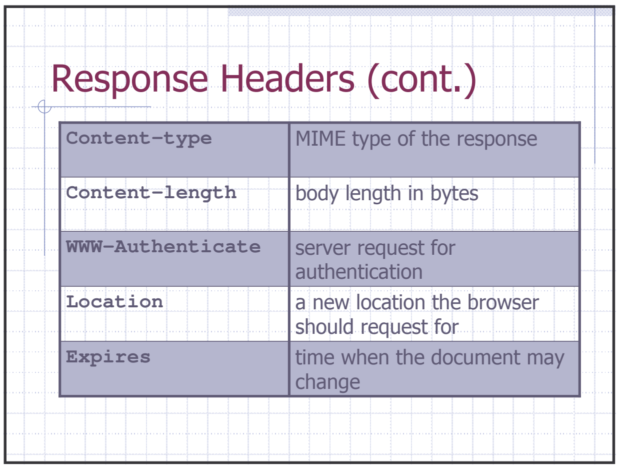
Request Headers (cont.)

`Accept`	data types accepted by client
`User-Agent`	client's browser identification
`Referer`	previous URL requested by the browser
`Authorization`	authorization data required by server
`Accept-Language`	



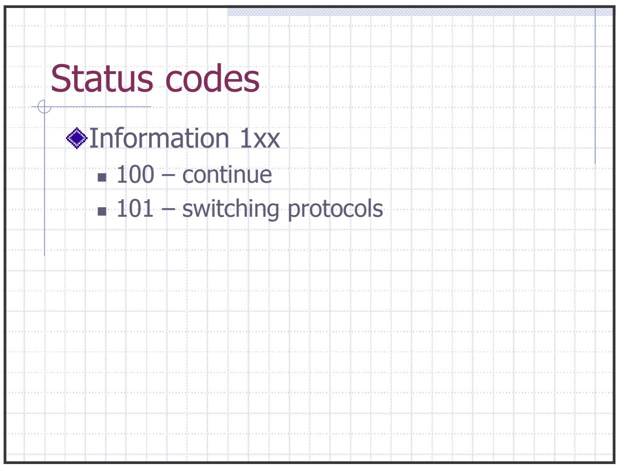
Response Headers (cont.)

`Cache-Control`	cache policy required by server
`Connection`	connection persistence handling
`WWW-Authenticate`	server request for authentication
`Location`	a new location the browser should request for
`Expires`	time when the document may change



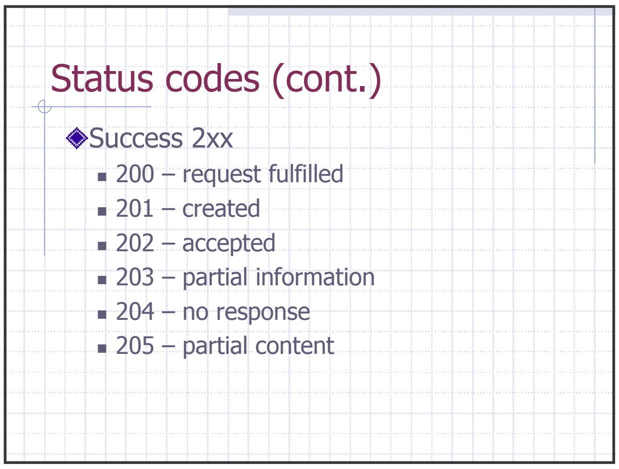
Response Headers (cont.)

`Content-type`	MIME type of the response
`Content-length`	body length in bytes
`WWW-Authenticate`	server request for authentication
`Location`	a new location the browser should request for
`Expires`	time when the document may change



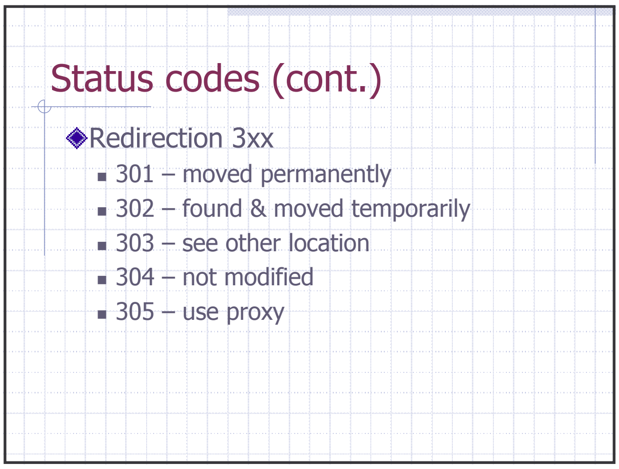
Status codes

- Information 1xx
  - 100 – continue
  - 101 – switching protocols



Status codes (cont.)

- Success 2xx
  - 200 – request fulfilled
  - 201 – created
  - 202 – accepted
  - 203 – partial information
  - 204 – no response
  - 205 – partial content



Status codes (cont.)

- Redirection 3xx
  - 301 – moved permanently
  - 302 – found & moved temporarily
  - 303 – see other location
  - 304 – not modified
  - 305 – use proxy

Status codes (cont.)

- Client-originated errors 4xx
  - 400 – bad request syntax
  - 401 – unauthorized
  - 402 – payment required
  - 403 – forbidden
  - 404 – not found
  - 405 – method not allowed

Status codes (cont.)

- Server-originated errors 5xx
  - 500 – internal server error
  - 501 – facility not supported
  - 502 – service overload
  - 503 – service unavailable
  - 504 – gateway timeout

Cookies

- short data exchanged by parties
  - name=value format
  - persistence control
  - stored by client
- Cookies over HTTP

```
Set-Cookie: NAME=VALUE; expires=DATE;
Cookie: NAME1=OPAQUE_STRING1;
```

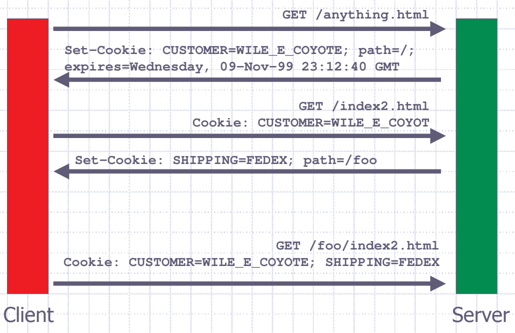
Cookie's Attributes

- name,
- value,
- expiration date of the cookie,
- path the cookie is valid for,
- domain the cookie is valid for,
- need for a secure connection to exist to use the cookie.

Operations on Cookies

- reset a cookie
  - either set its value to null
  - or set the expiration date in the past
- check whether cookies are accepted
  - set a cookie (1st request)
  - retrieve it (2nd request)

Example of a Cookie Transaction



```
GET /anything.html
Set-Cookie: CUSTOMER=WILE_E_COYOTE; path=/;
expires=Wednesday, 09-Nov-99 23:12:40 GMT
GET /index2.html
Cookie: CUSTOMER=WILE_E_COYOT
Set-Cookie: SHIPPING=FEDEX; path=/foo
GET /foo/index2.html
Cookie: CUSTOMER=WILE_E_COYOTE; SHIPPING=FEDEX
```

Basic Authentication

HTTP has a built-in authentication mechanism

```
⇒ GET /index.html HTTP/1.0
⇐ WWW-Authenticate .... realm:
⇒ GET ... Authorization J987kl8SAl
⇐ 401 HTTP/1.0 Unauthorized
```

```
user:password ⇒ (Base64) ⇒ J987kl8SAl
```

New Features in HTTP/1.1

- multiple transactions over single persistent connection
- cache support
- multiple hosts over single IP
- chunked encoding

Persistent Connections

Allows for sending multiple request & responses over single connection

```
HTTP/1.1 200 OK
Date: Fri, 31 Dec 1999 23:59:59 GMT
Content-Type: text/plain
Content-Length: 10
Connection: keep-alive <or closed>

abcdefghij
```

Cache Control

Allows for sending multiple request & responses over single connection

```
GET /index.html HTTP/1.1
Host: www.host1.poznan.pl
If-Modified-Since:Fri, 31 Dec 1999 23:59:59 GMT
<CRLF>
```

```
HTTP/1.1 304 Not Modified
Date: Fri, 31 Dec 1999 23:59:59 GMT
<CRLF>
```

Multiple Hosts over Single IP

Allows for sending multiple request & responses over single connection

```
GET /index.html HTTP/1.1
Host: www.host1.poznan.pl
<CRLF>
```

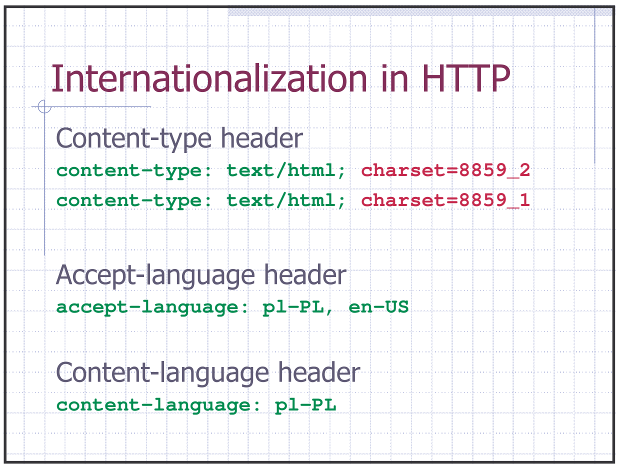
```
GET /index.html HTTP/1.1
Host: www.host2.poznan.pl
<CRLF>
```

Chunked Transfer-Encoding

Allows for sending partitioned responses

```
HTTP/1.1 200 OK
Date: Fri, 31 Dec 1999 23:59:59 GMT
Content-Type: text/plain
Transfer-Encoding: chunked

1a; ignore-stuff-here
abcdefghijklmnopqrstuvwxyz
10
1234567890abcdef
0
some-footer: some-value
another-footer: another-value
<CRLF>
```



Internationalization in HTTP

Content-type header

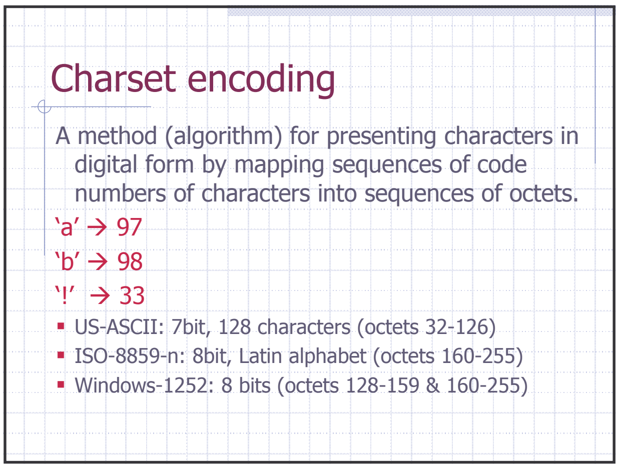
```
content-type: text/html; charset=8859_2
content-type: text/html; charset=8859_1
```

Accept-language header

```
accept-language: pl-PL, en-US
```

Content-language header

```
content-language: pl-PL
```



Charset encoding

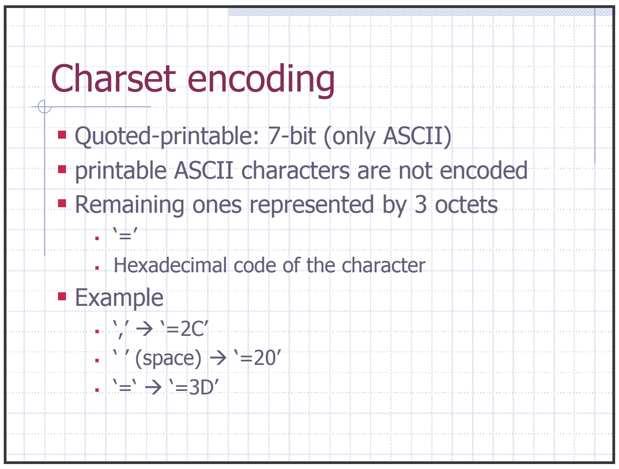
A method (algorithm) for presenting characters in digital form by mapping sequences of code numbers of characters into sequences of octets.

'a' → 97

'b' → 98

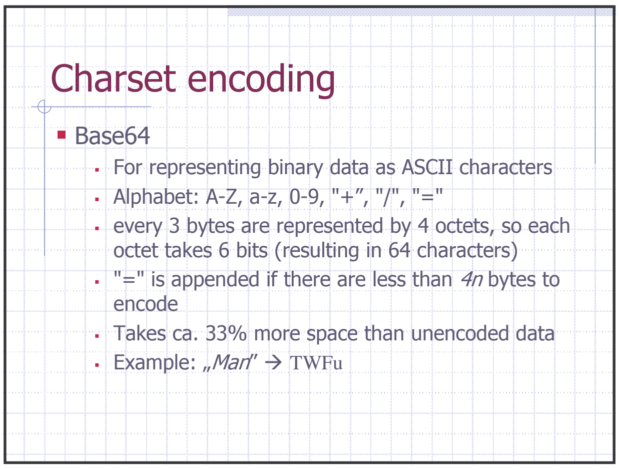
'!' → 33

- US-ASCII: 7bit, 128 characters (octets 32-126)
- ISO-8859-n: 8bit, Latin alphabet (octets 160-255)
- Windows-1252: 8 bits (octets 128-159 & 160-255)



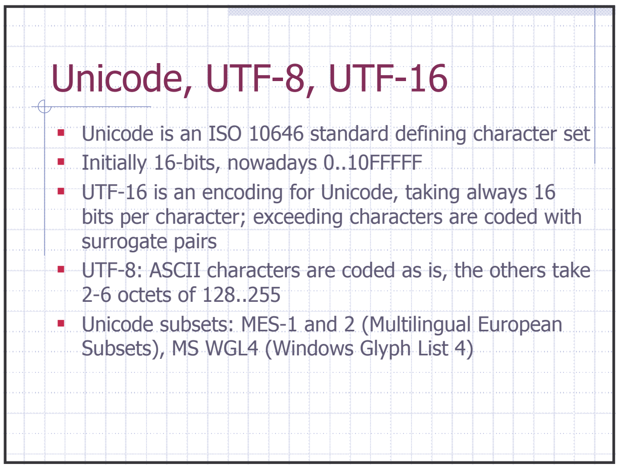
Charset encoding

- Quoted-printable: 7-bit (only ASCII)
- printable ASCII characters are not encoded
- Remaining ones represented by 3 octets
  - '='
  - Hexadecimal code of the character
- Example
  - ',' → '=2C'
  - ' ' (space) → '=20'
  - '=' → '=3D'



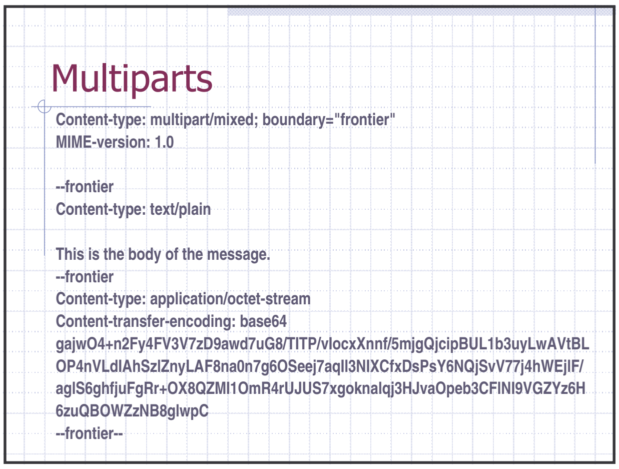
Charset encoding

- Base64
  - For representing binary data as ASCII characters
  - Alphabet: A-Z, a-z, 0-9, "+", "/", "="
  - every 3 bytes are represented by 4 octets, so each octet takes 6 bits (resulting in 64 characters)
  - "=" is appended if there are less than *4n* bytes to encode
  - Takes ca. 33% more space than unencoded data
  - Example: „*Man*" → TWFu



Unicode, UTF-8, UTF-16

- Unicode is an ISO 10646 standard defining character set
- Initially 16-bits, nowadays 0..10FFFFF
- UTF-16 is an encoding for Unicode, taking always 16 bits per character; exceeding characters are coded with surrogate pairs
- UTF-8: ASCII characters are coded as is, the others take 2-6 octets of 128..255
- Unicode subsets: MES-1 and 2 (Multilingual European Subsets), MS WGL4 (Windows Glyph List 4)



Multiparts

```
Content-type: multipart/mixed; boundary="frontier"
MIME-version: 1.0

--frontier
Content-type: text/plain

This is the body of the message.
--frontier
Content-type: application/octet-stream
Content-transfer-encoding: base64
gajwO4+n2Fy4FV3V7zD9awd7uG8/TITP/vlocxXnnf/5mjgQjcipBUL1b3uyLwAVtBL
OP4nVLdlAhSzlZnyLAF8na0n7g6OSeej7aqIl3NlXCfxDsPsY6NQjSvV77j4hWEjlF/
aglS6ghfjuFgRr+OX8QZMl1OmR4rUJUS7xgoknalqj3HJvaOpeb3CFlNl9VGZYz6H
6zuQBOWZzNB8glwpC
--frontier--
```

Next week...

Common Gateway Interface

Thank you!