Encodings

MIME types and bytes

1's and 0's

- The Internet can only transfer bits

  - Copper: High/Low voltage

  - Fiber: Light/Dark

- All data sent over the Internet must be binary

- **How do we know what these 1's and 0's represent in HTTP?**

  - Encodings and MIME Types

Encoding Text

Text

- Only 1's and 0's can travel through the Internet
  - How do we send text?

ASCII

- Character encoding
  - Maps characters to numbers
  - Numbers are represented in bits
  - Bits are sent through the Internet
  - Numbers are mapped back to characters by the receiver

- ASCII uses 7 bit encodings

Dec	Hx	Oct	Char		Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL	(null)	32	20	040	 	Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH	(start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX	(start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX	(end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT	(end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ	(enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK	(acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL	(bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS	(backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB	(horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF	(NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT	(vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF	(NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR	(carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO	(shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI	(shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE	(data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1	(device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2	(device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3	(device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4	(device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK	(negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN	(synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB	(end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN	(cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM	(end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB	(substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC	(escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS	(file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174	|		
29	1D	035	GS	(group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS	(record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US	(unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

ASCII

- As a String:

  - "hello"

  - Programming language specific representation

- In Hex:

  - 68 65 6c 6c 6f

  - Need to encode the String into a byte representation

- In Binary:

  - 01101000 01100101 01101100 01101100 01101111

  - Send this over the Internet

HTTP Headers

HTTP Headers can only contain
ASCII characters

HTTP Headers

- When reading HTTP **headers** [And request/status lines]

  - Assume it is text encoded using ASCII

- The **body** of the request/response may be encoded differently

  - Read the headers to find the encoding for the body

Character Encodings

- ASCII can only encode 128 different characters

  - Decent enough for english text

  - **Un**usable for languages with different alphabets

- With the Internet, the world became much more connected

  - Too restrictive for each alphabet to have its own encoding

- How do we encode more characters with a single standard?

  - We need more bits!

  - Enter UTF-8

UTF-8

- The modern standard for encoding text

- Uses up to 4 bytes to represent a character

- If the first bit is a 0

  - One byte used. Remaining 7 bits are ASCII

  - All ASCII encoded Strings are valid UTF-8

Number of bytes	Bits for code point	First code point	Last code point	Byte 1	Byte 2	Byte 3	Byte 4
1	7	U+0000	U+007F	0xxxxxxx			
2	11	U+0080	U+07FF	110xxxxx	10xxxxxx		
3	16	U+0800	U+FFFF	1110xxxx	10xxxxxx	10xxxxxx	
4	21	U+10000	U+10FFFF	11110xxx	10xxxxxx	10xxxxxx	10xxxxxx

Source: Wikipedia

UTF-8

- If more bytes are needed:

  - Lead with 1's to indicate the number of bytes

  - Each continuation byte begins with 10

  - Prevents decoding errors

  - No character is a subsequence of another character

Number of bytes	Bits for code point	First code point	Last code point	Byte 1	Byte 2	Byte 3	Byte 4
1	7	U+0000	U+007F	0xxxxxxx			
2	11	U+0080	U+07FF	110xxxxx	10xxxxxx		
3	16	U+0800	U+FFFF	1110xxxx	10xxxxxx	10xxxxxx	
4	21	U+10000	U+10FFFF	11110xxx	10xxxxxx	10xxxxxx	10xxxxxx

Source: Wikipedia

Sending Data

- When sending Strings over the Internet

  - The Internet does not understand language-specific Strings

  - Always convert to bytes/bits before sending

  - **Encode** the String using UTF-8

- When receiving text over the Internet

  - It must have been sent as bytes/bits

  - Must convert to a language-specific String

  - **Decode** the bytes using the proper encoding

Sending Data

- Use the Python methods

  - .encode()

  - .decode()

- You can specify the encoding as an argument of these methods

  - Default is UTF-8 so providing no arguments will do what you want for this course

Content Length

- Content-Length header must be set when there is a body to a response/request

- Value is the number of **bytes** contained in the body

  - Bytes referred to as *octets* in some documentation

- If all your characters are ASCII

  - This is equal to the length of the String

- Any non-ASCII UTF-8 character uses >1 byte

  - **Cannot use the length of the String as your Content-Length!**

Content Length

- To compute the content length of a UTF-8 String
  - Convert to bytes first
  - Get the length of the byte array

MIME Types

MIME Types

- When an HTTP response [or request] contains a body, the body is an array of bytes

  - There's no restriction on the encoding used for these bytes (Cannot always assume ASCII/UTF-8)

- Set a Content-Type header to tell the browser what those bytes represent

  - Tells the browser how to read the body of your response

- This is the **MIME type** of the data

MIME Types

- MIME type

  - *Multipurpose Internet Mail Extensions*

  - Developed for email and adopted for HTTP

- Two parts separate by a /

  - <type>/<subtype>

- Common types

  - text - Data using a text encoding (eg. UTF-8)

  - image - Raw binary of an image file

  - video - Raw binary of a video

MIME Types

- Common Type/Subtypes
  - text/plain
  - text/html
  - text/css
  - text/javascript
  - image/png
  - image/jpeg
  - video/mp4
  - application/json

MIME Types

- Optional settings can be added to the Content-Type header

  - Separate options by a ;

  - Options are formatted as <name>=<value>

- Content-Type: text/html; charset=utf-8

  - The content is HTML encoded using UTF-8

  - *You must use this to tell the browser that you are using utf-8

  - Don't forget to add this exactly as it appears. One character difference will break your page

MIME Type Sniffing

- Modern browsers will "sniff" the proper MIME type of a response

  - If the MIME type is not correct, the browser will "figure it out"™ and guess what type makes the most sense

- Browsers can sometimes be wrong

  - Surprises when your site doesn't work with certain versions of certain browsers

- Best practice is to disable sniffing

- Set this HTTP header to tell the browser you set the correct MIME type

  - X-Content-Type-Options: nosniff

MIME Type Sniffing

- X-Content-Type-Options: nosniff

- Be sure to set this header properly!

  - Open the browser console

  - Check the headers of your response

  - Make sure this header was parsed by the browser

MIME Type Sniffing

- **Security concern:**

  - You have a site where users can upload images

  - All users can view these images

  - Instead of an image, a user uploads JavaScript that steals personal data

  - You set the MIME type to image/png

  - The browser notices something is wrong and sniffs out the MIME type of text/javascript and runs the script

  - **You just got hacked!**

- Solution:

  - X-Content-Type-Options: nosniff

MIME Types

- With the proper MIME types set through a Content-Type header

  - The browser will know how to parse and render the body of your HTTP response

- When receiving an HTTP request that contains a body

  - The Content-Type will be set to let our server know the MIME type

Sending Images

Sending Images

- Sometimes we want to send data that is not text

- Use different formats depending on the data

- To send an image

  - Read the bytes from the file

  - Send the bytes as-is

  - Content-Length is the size of the file

  - Set the Content-Type to image/<image_type>

Sending Images

- When sending images

  - Since the data is already in bytes when the file is read, no need to encode/decode

- **Never** try to read an image file as a string

- **Never** try to decode the bytes of an image into a string

- An image is not encoded using UTF-8

  - The bytes will not decode properly

Sending Images

- Don't overthink sending images

- Read the **bytes** of the file. That's the body of your response

  - In your language, you may have to specify that the file should be read as a byte array so your library doesn't decode it as text

- Set the Content-Length to the length of the byte array

- Set the appropriate MIME type in Content-Type

  - Ex: to send a .png the MIME type is "image/png"