Character Encoding

Overzicht

Wat is Character Encoding?

Character encoding (karaktercodering) is een methode om tekens (zoals letters, cijfers en symbolen) te representeren als binaire data, zodat computers ze kunnen opslaan en verwerken. Computers werken intern met enen en nullen, en character encoding bepaalt hoe deze bits en bytes worden omgezet naar leesbare tekens.

Een character encoding definieert een mapping tussen binaire waarden en de symbolen die mensen begrijpen, zoals letters (A-Z), cijfers (0-9) en speciale tekens (€, @, ñ). Bekende encodings zijn ASCII, UTF-8, UTF-16 en ISO-8859-1.

Stel je voor: Je schrijft een brief aan een vriend, maar in plaats van gewone letters te gebruiken, vervang je elke letter door een nummer volgens een geheime code. Bijvoorbeeld:

A = 65 B = 66 C = 67 enzovoort...

Als je het woord "HALLO" wilt versturen, zet je het om in nummers:

H A L L O → 72 65 76 76 79

Je vriend weet dat hij deze nummers moet omzetten naar letters met dezelfde code om de boodschap te begrijpen.

Wat heeft dit met computers te maken? Computers begrijpen alleen getallen, geen letters. Om tekst op te slaan en weer te geven, gebruiken ze character encoding, zoals ASCII of UTF-8, waarin elke letter of symbool een uniek nummer krijgt.

Bijvoorbeeld:

- In ASCII is de letter "A" altijd 65 en "B" altijd 66.
- In UTF-8 kunnen we ook emoji's en andere talen coderen, zoals "é" of "你好".

- Als je een tekstbestand opent en de verkeerde encoding gebruikt, kan je computer de getallen verkeerd omzetten en krijg je rare tekens zoals "Ã©" in plaats van "é".

Waarom is Character Encoding relevant voor web- en software ontwikkelaars?

Als je character encoding niet begrijpt, krijg je rare tekens, gebroken tekst en zelfs beveiligingsproblemen. Als ontwikkelaar moet je ervoor zorgen dat alles – van je code tot je database en API's – dezelfde encoding gebruikt.

Enkele praktijkvoorbeelden

- Als een database in UTF-8 is ingesteld, maar een applicatie tekst opslaat met ISO-8859-1, kunnen bepaalde tekens (zoals "é" of "ü") verkeerd worden weergegeven. Dit kan problemen opleveren bij het tonen van data in een webinterface of het verwerken van gebruikersinvoer.
- Slechte handling van encoding kan leiden tot beveiligingsrisico's, zoals SQL-injection of Cross-Site Scripting (XSS). Als je niet goed controleert welke encoding je gebruikt, kunnen kwaadwillenden bijvoorbeeld Unicode-karakters misbruiken om je inputvalidatie te omzeilen.

Bits & Bytes

Zoals daarnet reeds aangehaald werd maken computers gebruik van het binaire talstelel om gegevens te verwerken, waaronder letters, cijfers en symbolen. Om character encoding goed te begrijpen, moeten we daarom eerst weten wat bits en bytes zijn en hoe ze zich verhouden tot het binaire talstelsel.

Wat is een bit?

Een bit is de kleinste eenheid van digitale informatie en kan twee waarden hebben:

- 0 (uit)
- 1 (aan)

Computers gebruiken bits om alles op te slaan en te verwerken, inclusief tekst, afbeeldingen en geluid.

Wat is een byte?

Een byte bestaat uit 8 bits en kan 256 verschillende waarden aannemen (2^8 = 256).

Bits	Decimale waarde
00000000	0
00000001	1
00000010	2
...	...
11111111	255

💡 **Waarom 8 bits?** Omdat een byte precies genoeg ruimte biedt voor ASCII-karakters (0-127 past in 7 bits, maar we gebruiken meestal 8 bits).

Voorbeeld: De letter A in ASCII wordt opgeslagen als 01000001 (65 in decimaal). De spatie ( ) is 00100000 (32 in decimaal).

Hoe verhouden bits en bytes zich tot elkaar?

Eenheid	Grootte
1 bit	0 of 1
1 byte	8 bits
1 kilobyte (KB)	1024 bytes
1 megabyte (MB)	1024 KB (1.048.576 bytes)
1 gigabyte (GB)	1024 MB (1.073.741.824 bytes)
1 terabyte (TB)	1024 GB (1.099.511.627.776 bytes)

Waarom 1024 en niet 1000? Omdat computers werken met binaire machten van 2, en 1024 is 2^{10}.

Waarom is dit relevant voor character encoding?

Omdat verschillende encodings een verschillend aantal bits en bytes per karakter gebruiken:

Encoding	Bits per karakter	Opslagruimte
ASCII	7 bits (gebruikt meestal 8 bits)	1 byte
Extended ASCII	8 bits	1 byte

UTF-8	8-32 bits	1-4 bytes
UTF-16	16-32 bits	2-4 bytes
UTF-32	32 bits	4 bytes