**Engineer**

# JSON Schema

et sprog til at beskrive, hvordan JSON-data skal se ud

## Hvad det er

JSON Schema er et sprog til at beskrive, hvordan et stykke JSON skal se ud, og det er selv skrevet i JSON. En beskrivelse i sproget, et skema, opstiller reglerne: denne værdi er et objekt, den skal have disse egenskaber, den her er et helt tal på mindst én, den der en tekststreng på højst 200 tegn. Et program, der kaldes en validator, tager et skema og en instans, altså de data, der skal tjekkes, og svarer på, om instansen overholder skemaet, og hvis ikke, hvor og hvorfor. Projektet skriver navnet med stort S, JSON Schema, mens et enkelt dokument skrevet i sproget på engelsk blot hedder et schema, med lille s, og på dansk et skema.

Specifikationen vedligeholdes af JSON Schema-projektet på json-schema.org. Den aktuelle version er 2020-12 og den forrige 2019-09; tidligere versioner var nummererede drafts, som draft-04 og draft-07, og ældre værktøjer og artikler omtaler dem stadig ved de navne. Teksten består af to dele. Kernedokumentet definerer, hvordan et skema behandles, hvordan ét skema henviser til et andet, og medietypen `application/schema+json`, og et andet dokument, om validering, definerer de nøgleord, der tjekker værdier. De gældende dokumenter for 2020-12 er begge dateret 16. juni 2022.

Hvert nøgleord tilføjer én betingelse. `type` angiver en af seks primitive typer, null, boolean, object, array, number eller string, eller `integer`, der matcher ethvert tal, hvis brøkdel er nul. Tal tager `minimum`, `maximum` og `multipleOf`; tekststrenge tager `minLength`, `maxLength` og `pattern`, et regulært udtryk; arrays tager `minItems`, `maxItems` og `uniqueItems`; objekter tager `properties`, `required` og `additionalProperties`. Nøgleord som `allOf`, `anyOf`, `oneOf` og `not` kombinerer skemaer, og `$ref` peger på et skema, der er defineret et andet sted, ofte under `$defs` i samme dokument, så en form, der er skrevet én gang, kan bruges mange steder. Alt, hvad et skema ikke nævner, er tilladt. Det tomme skema, `{}`, accepterer enhver instans, der findes.

Et skema kan også angive, hvilken version af sproget det er skrevet i. Nøgleordet `$schema` rummer adressen på et metaskema, et skema, der beskriver gyldige skemaer i den version, og specifikationssiden bemærker, at metaskemaet validerer sig selv. For 2020-12 er adressen `https://json-schema.org/draft/2020-12/schema` .

Brug begrebet i en beslutning

Brug »Hvad er JSON Schema?« i en konkret beslutning. Vi kan knytte begrebet til det system, I bygger eller køber. Den del, brugerne aldrig ser: hvor data ligger, hvilke regler der gælder, og hvordan andre systemer må tale med den.

**Se vores ydelser** Kontakt os →

Undersøg antagelserne bag ordet

Brug forklaringen og det tilknyttede arbejde til at formulere spørgsmål til jeres projekt. Vi kan undersøge antagelserne sammen, før et værktøj vælges. Har designet og drevet 6 ETL/ELT-pipelines — 6 ETL-pipelines på tre databasemotorer

**Har designet og drevet 6 ETL/ELT-pipelines** Kontakt os →

## Hvorfor det betyder noget

JSON siger intet om betydning. Ethvert velformet dokument kan parses, uanset om programmet, der læser det, kan bruge det til noget. Uden en nedskrevet beskrivelse har hvert program, der læser et dokument, sin egen opfattelse af, hvad gyldigt er, spredt ud over betingelser i koden, og de opfattelser glider fra hinanden, indtil en forespørgsel fejler i produktion. Et skema samler reglerne ét sted i en form, som både mennesker og programmer kan læse, og det kan gemmes, versioneres, gennemgås og udgives som enhver anden fil.

Specifikationen beskriver sit formål bredt: JSON Schema er beregnet til validering, dokumentation, navigation via hyperlinks og styring af interaktionen med JSON-data, og valideringsordforrådet skal også give brugerflader, der arbejder med dataene, hints om dem. I praksis betyder det, at én beskrivelse kan afvise forkert input i kanten af en tjeneste, dokumentere et API, drive en formular og tjekke, at den JSON, en sprogmodel returnerer, har de felter, næste trin forventer.

Det er også blevet det fælles grundlag, som andre specifikationer bygger på. OpenAPI 3.1 definerer sit Schema Object som en overmængde af JSON Schema 2020-12, så indholdet af forespørgsler og svar i en API-beskrivelse er beskrevet i JSON Schema med nogle få tilføjelser. MCP, protokollen, som AI-agenter finder og kalder værktøjer gennem, beskriver hvert værktøjs parametre med et `inputSchema` , der er et JSON Schema, og kan beskrive resultatet med et

`outputSchema`, som klienter bør validere strukturerede resultater mod. En fælles beskrivelse kan altså danne grundlag for en validator, et API-dokument og en agent.

## Sådan ser det ud i praksis

Skemaet nedenfor beskriver input til et værktøj, der booker en leveringstid. Det angiver den version, det er skrevet i, opregner fire egenskaber med deres regler, siger, hvilke tre der skal være til stede, og afviser enhver egenskab, det ikke opregner:

```
{
  "$schema": "https://json-schema.org/draft/2020-12/schema",
  "$id": "https://example.com/tools/book-slot.json",
  "title": "Book en leveringstid",
  "type": "object",
  "properties": {
    "order": { "type": "string", "pattern": "^[0-9]{4,8}$" },
    "date": { "type": "string", "format": "date" },
    "pallets": { "type": "integer", "minimum": 1, "maximum": 30 },
    "note": { "type": ["string", "null"], "maxLength": 200 }
  },
  "required": ["order", "date", "pallets"],
  "additionalProperties": false
}
```

En instans, der bryder reglerne:

```
{ "order": "4471", "date": "27/10/2026", "pallets": 0, "pallet": 2 }
```

Instansen fejler på to punkter. `pallets` ligger under minimum på 1, og `pallet`, en tastefejl for `pallets`, er en egenskab, skemaet ikke opregner. At `note` mangler, er ingen fejl, for `note` står ikke i `required`; var den sendt som `null`, ville det også gå igennem, da dens type tillader null. Datoen er den lærerige. Formatet `date` kræver året først, og `27/10/2026` er ikke skrevet sådan, men en validator, der kører med sine standardindstillinger, accepterer den, fordi specifikationen gør `format` til en annotation, der kun bliver tjekket, når det tjek er slået til. Slå det tjek til i en validator, der tjekker datoer, og den samme instans fejler på tre punkter.

Et værktøj beskrevet på den måde er netop emnet for siden om din software som værktøjer til AI-agenter her på sitet. Den beskriver, hvordan hver operation, en agent skal kunne udføre, som at booke en tid eller frigive en ordre, stilles til rådighed som et værktøj med typede input, fejlbeskeder, en agent kan handle på, og idempotente kald, så et nyt forsøg aldrig booker to gange. Beskrivelsen er et OpenAPI-dokument genereret fra koden, og en MCP-server stiller de samme operationer til rådighed. Sidens pointe er, at en agent stoler på et værktøjs beskrivelse, også når den er forældet, og en håndskrevet beskrivelse kan glide fra implementeringen. Derfor genereres beskrivelserne fra koden og udad, gennem databasen, API'et, beskrivelsen og klienten, med et tjek i hvert led, der får buildet til at fejle, når to af dem er uenige.

## Hvor det går galt

**En opregnet egenskab, der aldrig er påkrævet.** `properties` tjekker kun de egenskaber, der er til stede, så et objekt, der udelader dem alle, går igennem. Tilstedeværelse er et separat nøgleord, og udelades `required`, er intet påkrævet overhovedet. Skriv `required` ved siden af `properties` hver gang, og hav en testsag for hver påkrævet egenskab, hvor netop den egenskab er fjernet.

**Tastefejl, der slipper igennem som ekstra egenskaber.** Uden `additionalProperties` er enhver egenskab, skemaet ikke opregner, tilladt, så `pallet` går igennem, og det rigtige `pallets` mangler i al stilhed. Sættes nøgleordet til `false`, lukkes objektet, men nøgleordet ser kun de `properties` og `patternProperties`, der står i dets eget skemaobjekt. Et skema, der bygger på et andet gennem `allOf` eller `$ref` og derefter lukker sig selv på den måde, afviser netop de egenskaber, det har arvet. Version 2020-12 har `unevaluatedProperties` til det tilfælde, da det også medregner de egenskaber, der er tjekket gennem `allOf`, `$ref` og de andre nøgleord, der anvender skemaer på stedet. Brug `additionalProperties: false` på et skema, der står alene, og `unevaluatedProperties: false` på et, der er sammensat.

`format`**, der intet tjekker.** Som standard er `format` en annotation, der kun beskriver, og ikke en assertion, der afviser: specifikationen kræver, at en validator, der kan tjekke formater, holder det tjek slået fra, medmindre der bliver bedt om det, og selv når det er slået til, må en validator lade ethvert format passere ukontrolleret. OpenAPI 3.1 følger samme regel og tilføjer, at hvor godt formater bliver tjekket, varierer mellem implementeringer. Slå formattjek til i den validator, I bruger, bevis det med en testsag, der burde fejle, og hvor et format betyder noget, men ikke kan tjekkes, så tilføj et `pattern` eller et tjek i koden.

**Mønstre, der matcher en del af en værdi.** Et `pattern` er ikke forankret, så `[0-9]{4}` accepterer `order 4471b`, fordi fire cifre optræder et sted i værdien; specifikationens eget eksempel er, at `es` matcher `expression`. Begynd mønstret med `^` og afslut det med `$`, når hele værdien skal matche. Mønstre følger ECMA-262-dialekten, og specifikationen anbefaler et lille sæt konstruktioner til mønstre, der skal opføre sig ens overalt, så hold jer til simple tegnklasser, kvantorer og ankre.

**Manglende, null og tom behandlet som ét.** En egenskab, der er fraværende, en egenskab sat til `null` og en tom tekststreng er tre forskellige instanser. `"type": "string"` afviser `null`, og en egenskab, der ikke står i `required`, må være fraværende. En klient, der sender `null` for at sige »ingen værdi« til et skema, der er skrevet til fravær, fejler på hver eneste af den slags forespørgsler. Beslut for hver egenskab, hvad hver af de tre betyder, og skriv det ned: `["string", "null"]`, når null bærer betydning, udeladelse fra `required`, når egenskaben er valgfri, og `minLength: 1`, når en tom tekststreng ikke er en værdi.

**Intet `$schema`, eller to versioner blandet.** Når roden af et skema ikke har noget `$schema`, overlader specifikationen opførslen til implementeringen, og hver validator anvender den version, den nu har som standard. Versionerne adskiller sig på steder, der betyder noget. I 2019-09 beskrev et array givet til `items` en tupel; i 2020-12 hører den opgave til `prefixItems`, og `items` har overtaget rollen fra `additionalItems`. Sæt `$schema` i hver rod, følg udgivelsesnoterne, når I skifter mellem versioner, og valider hvert skema mod dets metaskema i CI.

## Læs mere

- JSON Schema Specification — projektets oversigt over den aktuelle version, dens metaskemaer og udgivelsesnoterne for hvert tidligere draft.
- JSON Schema: A Media Type for Describing JSON Documents — Core 2020-12: hvordan skemaer behandles, `$schema`, `$ref` og `$defs`, og de nøgleord, der kombinerer skemaer.
- JSON Schema Validation: A Vocabulary for Structural Validation of JSON — nøgleordene for hver type værdi, `required` og reglerne for `format`.
- JSON Schema 2020-12 Release Notes — hvad der er ændret siden 2019-09, med `prefixItems` og `$dynamicRef` side om side med det, de afløste.
- OpenAPI Specification v3.1.1 — Schema Object som en overmængde af JSON Schema 2020-12, og `format` som annotation som standard.
- Model Context Protocol: Tools — hvordan et værktøj erklærer sine parametre og sit strukturerede output i JSON Schema.

Ordliste

Vælg ud fra jeres faktiske vilkår

Beskriv arbejdsgangen, de eksisterende grænseflader og de brugere, der har brug for ændringen. Vi kan definere en testbar leverance og en overdragelse, der gør teamet i stand til at vedligeholde løsningen.

**Se, hvad det koster** Kontakt os →

## Gør begreberne praktiske

Spørg, hvordan »Hvad er JSON Schema?« gælder i jeres organisation. Et konkret eksempel fra jeres nuværende system er nok til at begynde samtalen.

**Se vores ydelser**

welcome@engineer.company +45 50 10 61 84 København, Danmark

Tag denne side med dig

Visitkort

Lavet med kærlighed af D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Copyright © 2025 – nu · Engineer ApS · Alle rettigheder forbeholdes.

Udskrevet fra https://engineer.company/da/glossary/json-schema/