INVESTICE DO ROZVOJE VZDĚLÁVÁNÍ

Realizováno za finanční podpory ESF a státního rozpočtu ČR v rámci v projektu *Zkvalitnění a rozšíření možností studia na TUL pro studenty se SVP* reg. č. CZ.1.07/2.2.00/29.0011

JSON Schema

JSON Schema

- popisuje strukturu a hodnoty dat pro JSON
- umožňuje validaci dat
- schéma je také v JSON
- první verze 2007
- aktuální verze 2020-12
- hromada implementací
- json-schema.org

Schéma

- schéma je objekt
- prázdný objekt je korektní schéma, vyhoví každý syntakticky správný JSON soubor
  { }
- **type** určuje typ hodnoty
  { **"type": "number"** }
  20 OK 3.14 OK "tři" ne "3" ne { "x": 3 } ne
- lze i pole
  { "type": **["number", "string"]** }

Verze

- vyšlo několik verzí, aktuální 2020-12
- validátor může podporovat i starší
- je vhodné verzi vyznačit, jinak použije implicitní:

```
{
    "$schema": "https://json-schema.org/
                                draft/2020-12/schema",
    ...
}
```

Typy

- **string** řetězec znaků
- **number** číslo
- **integer** celé číslo
- **boolean** true nebo false
- **null** jen null
- **object** objekt
- **array** pole

Objekt

- **properties** definuje schémata pro jeho vlastnosti
  - klíč – název vlastnosti
  - hodnota – schéma pro její hodnotu
- ```
  { "type": "object",
    "properties": {
        "nazev": { "type": "string" },
        "cena": { "type": "number" },
        "pocet_kusu": { "type": "integer" }
    }
  }
  ```
  ```

Vlastnosti objektu

- výchozí ověřování dost volné: pokud obsahuje danou vlastnost, musí hodnota odpovídat jejímu schématu
- vlastnosti jsou nepovinné
- objekt může obsahovat i další (s libovolnými hodnotami – není pro ně schéma)
- { } OK
  { "cena": 3.50 } OK
  { "nazev": "Rohlík", "cena": 2.50, "akce": true } OK
  { "ulice": "Studentská 2" } OK
  { "nazev": 3.50 } ne

Povinné vlastnosti

- **required** určuje povinné vlastnosti
- hodnotou pole jejich názvů
- ```
{   "type": "object",
    "properties": {
        "nazev": { "type": "string" },
        "cena": { "type": "number" },
        "pocet_kusu": { "type": "integer" }
    },
    "required": ["nazev", "cena"]
}
```
```

Zákaz přidávání vlastností

- **additionalProperties** umožňuje zakázat
- ```
{   "type": "object",
    "properties": {
        "nazev": { "type": "string" },
        "cena": { "type": "number" },
        "pocet_kusu": { "type": "integer" }
    },
    "required": ["nazev", "cena"],
    "additionalProperties": false
}
```
```

Zobecněné vlastnosti (1)

- **patternProperties** používá pro jména vlastností regulární výrazy – vyhoví více jmen
- ```
{   "type": "object",
    "properties": {
        "nazev": { "type": "string" }
    },
    "patternProperties": {
        "^konzultant[0-9]*$": { "type": "string" },
        "^rok": { "type": "integer" },
    },
}
```
```

Zobecněné vlastnosti (2)

- **propertyNames** určuje syntax jmen, neřeší hodnoty
- **minProperties**, **maxProperties** omezují minimální a maximální počet vlastností
- např. nanejvýš 5 libovolných identifikátorů:

```
{   "type": object,
    "propertyNames": {
        "pattern": "^[A-Za-z_][A-Za-z0-9_]*$"
    },
    "maxProperties": 5
}
```

Omezení řetězců

- **minLength**, **maxLength** – minimální a maximální počet znaků

```
"nazev": { "type": "string", "maxLength": 50 }
```

- **pattern** – regulární výraz

```
"telefon": {
    "type": "string",
    "pattern": "^[0-9]{3} ?[0-9]{3} ?[0-9]{3}$"
}
```

Formát řetězce

- **format** – charakter hodnoty v řetězci
- **nepoužívá se pro validaci**, implementace ale může nabídnout konfigurační volbu, která validaci zapne
- cca 20 formátů: date, time, email, ipv4, ipv6, uri, json-pointer, regex, …
- "web": { "type": string", **"format": "uri"** }

Výčet hodnot

- **enum** – hodnotou pole přípustných hodnot
- nemusí být jen řetězce, lze i míchat různé typy
- "fakulta": {
  "enum": ["FS", "FT", "FP", "EF", "FUA", "FM", "FZS"]
  }
  "znamenko": {
  "enum": [-1, 0, 1]
  }

Pevná hodnota

- **const** – hodnotou musí být uvedená hodnota
- použitelnost pro data je problematická, ale poslouží v podmínkách
- "pi": {
  "const": 3.14
  }

Omezení čísel (1)

- **minimum**, **exclusiveMinimum** – nejmenší hodnota
- **maximum**, **exclusiveMaximum** – největší hodnota
- např. pěticiferné PSČ:

```
"psc": {
    "type": "integer",
    "minimum": 10000,
    "maximum": 99999
}
```

Omezení čísel (2)

- **multipleOf** – hodnota musí být násobkem tohoto čísla
- např. cena na dvě desetinná místa:

```
"cena": {
    "type": "number",
    "multipleOf": 0.01
}
```

Pole

- **items** definuje typ jeho položek
- např. pole celých čísel z intervalu 0 až 100

```
{
    "type": "array",
    "items": {
        "type": "integer",
        "minimum": 0,
        "maximum": 100
    }
}
```

Počet položek

- **minItems** minimální, **maxItems** maximální
- např. šachovnice 8×8 celých čísel:

```
{   "type": "array",
    "items": {
        "type": "array", "items": { "type": "integer" },
        "minItems": 8, "maxItems": 8
    },
    "minItems": 8,
    "maxItems": 8
}
```

Unikátní hodnoty

- **uniqueItems** – musí být hodnoty v poli unikátní?
- např. unikátní čísla v poli:

```
{   "type": "array",
    "items": {
        "type": "integer"
    },
    "uniqueItems": true
}
```

Obsahuje pole hodnotu?

- **contains** – jakou hodnotu má obsahovat
- **minContains** – minimální počet výskytů (1)
- **maxContains** – maximální počet výskytů (∞)
- např. pole libovolných hodnot, ve kterém je alespoň 5 nezáporných čísel:

```
{   "type": "array",
    "contains": { "type": "number", "minimum": 0 },
    "minContains": 5
}
```

Pole s pevnou strukturou (1)

- **prefixItems** předepisuje typy počátečních položek, pole schémat, záleží na pořadí
- např. pole s adresou – ulice (řetězec), PSČ (číslo) a město (řetězec):

```
{   "type": "array",
    "prefixItems": [
        { "type": "string", "maxLength": 50 },
        { "type": "integer" },
        { "type": "string" }
    ]
}
```

Pole s pevnou strukturou (2)

- za počátečními položkami mohou následovat další
- **items** umožňuje definovat schéma pro jejich hodnoty
  - "items": false zakáže
  - "items": { "type": "integer" } povolí jen čísla
- lze omezit počet (maxItems)
- mají-li položky pevně daný význam, bývá vhodnější objekt než pole

Kombinování schémat

- hodnotou pole schémat:
- **allOf** hodnota musí vyhovět všem
- **anyOf** alespoň jednomu
- **oneOf** právě jednomu
- hodnotou jedno schéma:
- **not** nesmí vyhovět

Příklady kombinací (1)

- číslo dělitelné 3 a 5:

```
{   "type": "number",
    "allOf": [
        { "multipleOf": 3 },
        { "multipleOf": 5 }
    ]
}
```

Příklady kombinací (2)

- pole kladných čísel a řetězců s nanejvýš 10 znaky:

```
{   "type": "array",
    "items": {
        "anyOf": [
            { "type": "number", "exclusiveMinimum": 0 },
            { "type": "string", "maxLength": 10 }
        ]
    }
}
```

Podmíněné vlastnosti

- **dependentRequired** – pokud objekt obsahuje určitou vlastnost, musí obsahovat i jiné
  - klíče – podmiňující vlastnosti
  - hodnoty – pole jmen povinných vlastností
- např. obsahuje-li objekt číslo OP, musí obsahovat i vydavatele a datum vydání

```
"dependentRequired": {
    "cisloOP": [ "vydavatelOP", "datumVydaniOP" ]
}
```

Podmíněné podschéma

- obecnější mechanismus **dependentSchemas** – přítomnost vlastnosti aktivuje část schématu
- vlastnosti se definují, až když se vyskytne klíčová vlastnost
- umožňuje, aby se jejich definice v různých variantách dat lišily

Podmíněné podschéma

- alternativa k předchozímu:
- **"dependentSchemas": {**

```
    "cisloOP": {
        properties": {
            "vydavatelOP": { "type": "string" },
            "datumVydaniOP": { "type": "string" }
        },
        "required": ["vydavatelOP", "datumVydaniOP"]
    }
}
```

if-then-else

- **"if": { *podmínka* },**
  "then": { *schéma1* },
  "else": { *schéma2* }
- platí-li podmínka, musí platit schéma1 (schéma2 se ignoruje), jinak musí platit schéma2 (schéma1 se ignoruje)
- chybějící then nebo else se bere jako true

Příklad if-then-else

- schéma se mění podle typu kontaktu

```
{   "type": "object",
    "properties": {
        "kontakt": { "enum": [ "telefon", "mail" ] }
    },
    "if": {"properties": { "kontakt": { "const": "mail" } } },
    "then": { "properties": { "mail": { "type": "string" } } },
    "else": {
        "properties": { "telefon": { "type": "integer" } } },
    "unevaluatedProperties": false
}
```

Přidávání vlastností

- additionalProperties je příliš přísné, akceptuje jen vlastnosti ze stejného podschématu
- v předchozím příkladu nepřijme vlastnosti přidané v if-then-else
- **unevaluatedProperties** bere v úvahu i vlastnosti z vnořených podschémat
- podobně u pole **unevaluatedItems**

Nečekané splnění podmínky

- chybí-li vlastnost posuzovaná v podmínce, bere se podmínka jako splněná
- nebude-li objekt obsahovat kontakt, přidá se vlastnost mail
- řešení: required do podmínky
-
```
"if": {
    "properties": { "kontakt": { "const": "mail" } },
    "required": [ "kontakt" ]
}
```

Více větví

- kombinace allOf a if-then bez else

```
"allOf": [
    {   "if": {  "properties": { "kontakt": { "const": "mail" } } },
        "then": { "properties": { "mail": { "type": "string" } } } },
    {   "if": {  "properties": { "kontakt": { "const": "telefon" } },
                 "required": [ "kontakt" ] },
        "then": { "properties": { "tel": { "type": "integer" } } } },
    {   "if": {  "properties": { "kontakt": { "const": "voip" } },
                 "required": [ "kontakt" ] },
        "then": { "properties": { "voipID": { "type": "string" } } } }
]
```

Identifikátor schématu

- **$id** – hodnotou je URI
- silně doporučeno **absolutní URI**, relativní se posuzuje vůči URI, ze kterého bylo schéma získáno
- ```
  {   "$id": "https://tul.cz/schema/datum",
      "type": "object",
      "properties": { "den": { "type": "integer" },
          "mesic": { "type": "integer" },
          "rok": { "type": "integer" } },
      "required": [ "den", "mesic", "rok" ]
  }
  ```
  ```

Odkaz na schéma

- **$ref** – hodnotou je URI odkazovaného schématu
-
```
{   "$id": "https://tul.cz/schema/osoba",
    "type": "object",
    "properties": {
        "jmeno": { "type": "string" },
        "narozen": { "$ref": "/schema/datum" }
    }
}
```

Lokální definice

- standardně vkládány do **$defs**
- odkaz: **#/$defs/*název***
- ```
{   "properties": {
        "tel": { "$ref": "#/$defs/telcislo" }
    },
    "$defs": {
        "telcislo": { "type": "string",
                "pattern": "^[0-9]{3} ?[0-9]{3} ?[0-9]{3}$" }
    }
}
```
```

JSON Pointer

- RFC 6901
- identifikuje konkrétní hodnotu v JSON
- kroky oddělované lomítky
- krokem je **název vlastnosti** v objektu nebo **index** položky v poli
- často ve fragmentu URI (za znakem #)
- #/$defs/telcislo
  #/sachovnice/2/5

Informační vlastnosti

- nevyužívají se pro validaci
- **title** – název
- **description** – popis
- **default** – výchozí hodnota, jen informativní (příklad)
- **examples** – pole příkladů platných hodnot
- **readOnly**, **writeOnly** – instrukce pro API
- **deprecated** – zastaralé, nepoužívat