JSON Schema

The good, the bad, the ugly

Maximilian Koegel (@mkoegel)

CC Image by liftarn

CC Image by wesd440

CC Image by RobynBroyles

What is JSON?

- Format for structured data
- Human- and machine readable
- It’s not XML (no namespaces, no markup semantics)
- De-facto web data interchange format
- Native JavaScript data format

Example:

```
{
    "firstName": "John",
    "lastName": "Doe",
    "age": 31
}
```

What is JSON Schema?

- Format to describe the structure of JSON instances
- A JSON Document
- Minimal Core define base concepts
- Extensions
  - Validation
  - Hyper-Schema
  - more to come?

```
{
	"title": "Example Schema",
	"type": "object",
	"properties": {
		"firstName": {
			"type": "string"
		},
		"lastName": {
			"type": "string"
		},
		"age": {
			"description": "Age in years",
			"type": "integer",
			"minimum": 0
		}
	},
	"required": ["firstName", "lastName"]
}
```

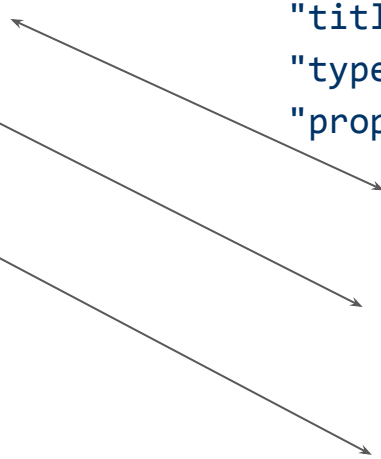
How does it look like?

Instance

```
{
    "firstName": "John",
    "lastName": "Doe",
    "age": 31
}
```

Schema

```
{
    "title": "Example Schema",
    "type": "object",
    "properties": {
        "firstName": {
            "type": "string"
        },
        "lastName": {
            "type": "string"
        },
        "age": {
            "description": "Age in years",
            "type": "integer",
            "maximum": 120
        }
    },
    "required": ["firstName", "lastName"]
}
```



Does it matter?

Yes! 😊

- Heroku (Heroku Platform API)
- OpenAPI/Swagger (Entity descriptions)
- Google Discovery API (Meta-Service for Google APIs)
- ...

How we got here

We're heavy users the Eclipse Modeling Framework (EMF).

EMF is a modeling framework and code generation facility for building tools and other applications based on a structured data model [which essentially is a subset of UML].

- Strong believe in benefits of structured data definitions
- Searched for structured data definitions for JSON
- Ended up embracing JSON Schema

→ JSON Schema allows us to re-use our expertise in the web

The Good

The good parts - Overview

- Interoperable/Polyglot
- Validation
- API Documentation
- Form generation
- Code generation
- Testing

Interoperable/Polyglot

- Usage of same data definition on Client/Server
- Language agnostic
- It is just JSON
- There is a JSON Schema for the JSON Schema
- Lots of libraries consuming JSON Schemata
- Good documentation

https://www.flickr.com/photos/wiredwitch/11330858474/

Validation

- Shared definition
- Unified format
- Cross-platform
- More readable than code
- Can be updated at runtime
- BUT: not all use-cases can be covered with a schema

https://openclipart.org/detail/170188/simple-tick-and-cross

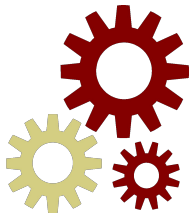
Form-based UIs

Interoperable/Polyglot
Validation
> Form-based UIs
API Documentation
Code generation
Testing

- Derive your UI from a Schema
- For mass deployment of forms, this approach is:
  - Faster (rapid prototyping + customization)
  - Less tedious to maintain
  - Less error-prone
- Examples:
  - Angular Schema Form (Angular)
  - JSON Forms (Angular)
  - react-jsonschema-form (React)

'paper notes' by crisg

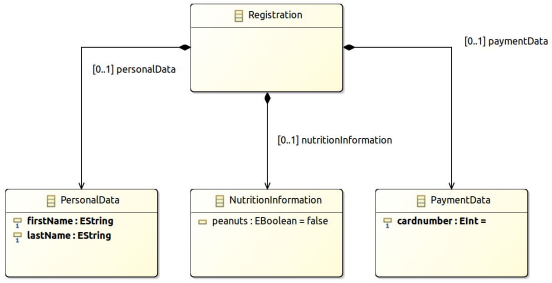
Declarative Data and UI Definition

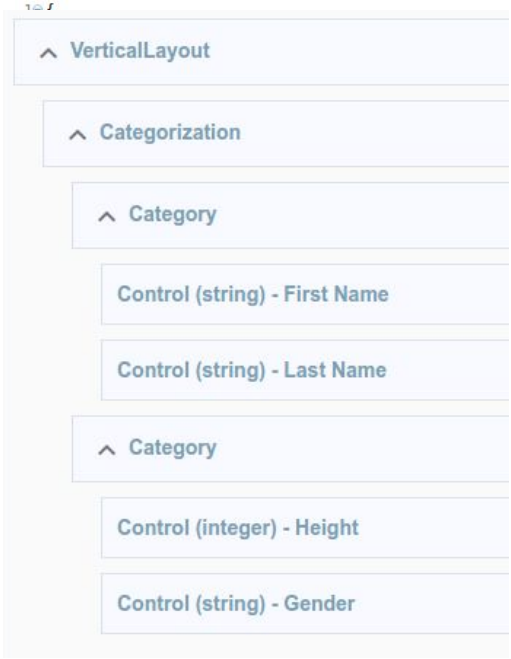


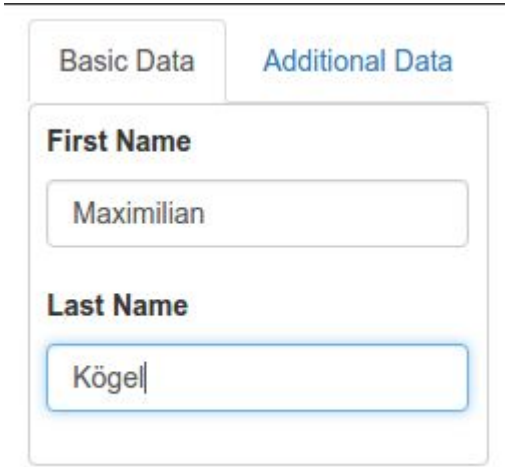
Renderer

JSON Schema + **UI Schema** = **User Interface**

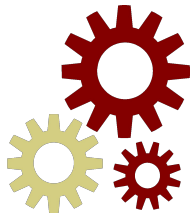
```
{
    "type": "object",
    "properties": {
        "firstName": {
            "type": "string"
        },
        "type": "integer"
        },
        "gender": {
            "type": "string",
            "enum": [
                "female",
                "male",
                "unkown"
            ]
        }
    }
}
```





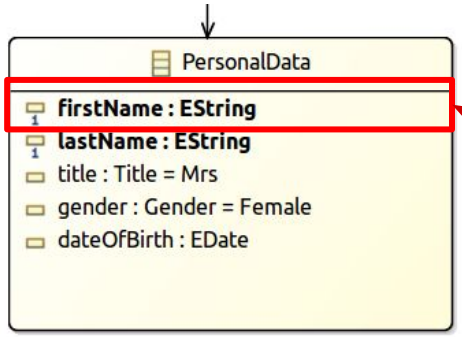


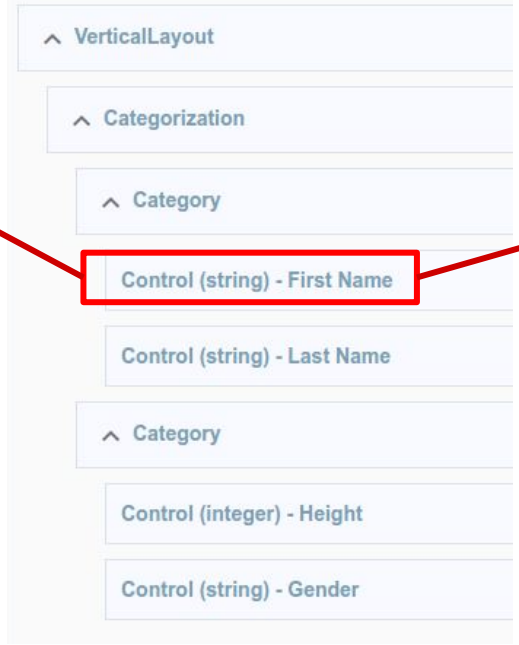
Declarative Databinding

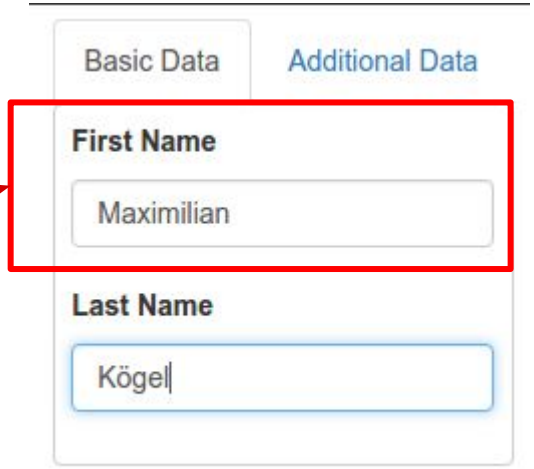


Renderer

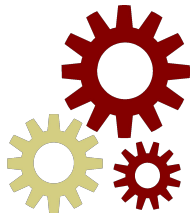
JSON Schema + **UI Schema** = **User Interface**





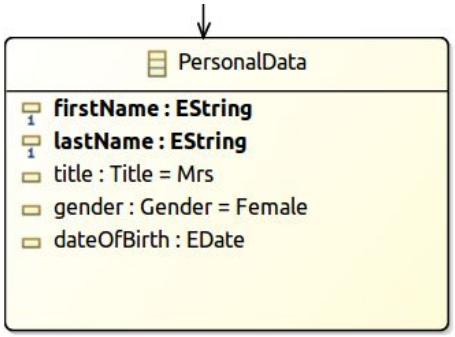


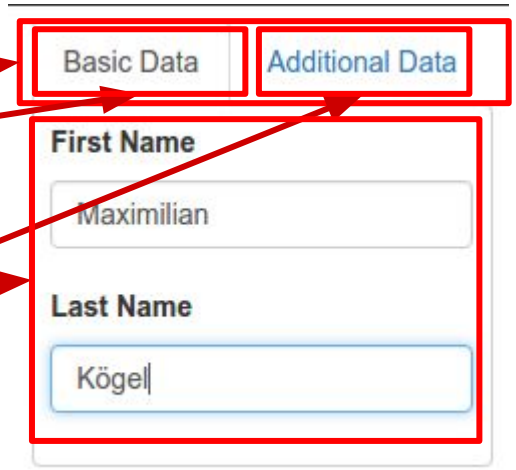
Declarative Layouts



Renderer

JSON Schema + **UI Schema** = **User Interface**





API Documentation

- Probably most prominent use: Swagger/OpenAPI
- Describes RESTful API
- Also acts as a Sandbox for API exploration
- Enables generation of stubs (see “Code generation”)
- OpenAPI/Swagger usage:
  - gettyimages
  - github
  - gmail
  - Youtube
  - Instagram
  - Slack
  - Trello
  - xkcd

HTTP://IMGS.XKCD.COM/COMICS/PHILOSOPHY.PNG

xkcd OpenAPI/Swagger Schema (1/2)

Interoperable/Polyglot
Validation
Form-based UIs
> API Documentation
Code generation
Testing

```
[...] // General API info like basePath, host, etc.
"paths": {
    "/info.0.json": { ... },
    "/{comicId}/info.0.json": {
      "get": {
        "description": "Fetch comics and metadata  by comic id.\n",
        "parameters": [
          {
            "in": "path",
            "name": "comicId",
            "required": true,
            "type": "number"
          }
        ],
        "responses": {
          "200": {
            "description": "OK",
            "schema": { "$ref": "#/definitions/comic" }
          }
        }
      }
    }
  }
[...]
```

OPEN {API} INITIATIVE

xkcd OpenAPI/Swagger Schema (2/2)

Interoperable/Polyglot
Validation
Form-based UIs
> API Documentation
Code generation
Testing

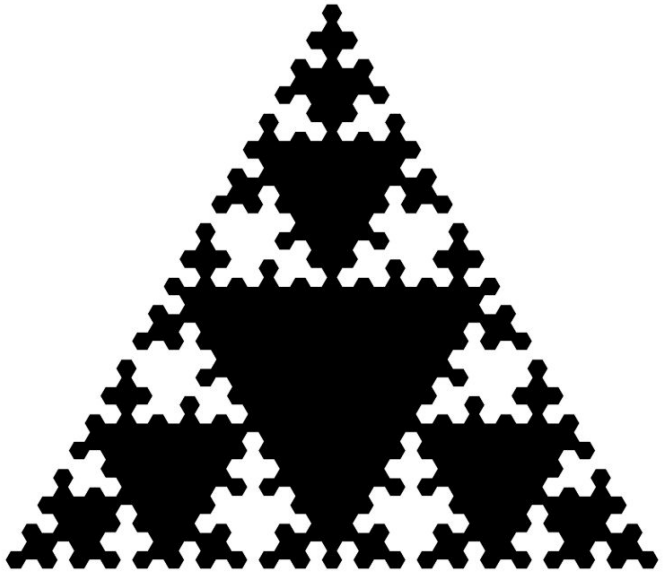
```
"definitions": {
    "comic": {
      "properties": {
        "alt":        { "type": "string" },
        "day":        { "type": "string" },
        "img":        { "type": "string" },
        "link":       { "type": "string" },
        "month":      { "type": "string" },
        "news":       { "type": "string" },
        "num":        { "type": "number" },
        "safe_title": { "type": "string" },
        "title":      { "type": "string" },
        "transcript": { "type": "string" },
        "year":       { "type": "string" }
      },
      "type": "object"
    }
  }
}
```

OPEN {API} INITIATIVE

Code generation

- Generate language-specific types
- Allows for rapid prototyping/scaffolding
- Examples
  - Jsonschema2pojo (Java POJOs)
  - dtsgenerator (Typescript Stubs)
  - Heroics (Ruby HTTP Client API)

Interoperable/Polyglot
Validation
Form-based UIs
API Documentation
> Code generation
Testing



https://openclipart.org/detail/50317/sierpinskis-triangle

Testing

- Generate random data to test your business logic
- Useful for testing logic of services
- JSON Schema Faker (JS)
- Schematic Ipsum (Browser)

https://openclipart.org/detail/226237/car-accident-sign

The Bad

The Bad (meta)

- It's still a draft (v6 now)
- See https://github.com/json-schema/json-schema/issues/167

BUT:

→ Widespread use of v4/new draft

→ Active Github community (new draft v6, new web page etc.)

CC Image by Merlin2525

The Bad - Implementation

- No uniform error reporting format (fge vs is-my-json-valid)

```
[ {
  "level": "error",
  "schema": {
    "loadingURI": "#",
    "pointer": ""
  },
  "instance": {
    "pointer": ""
  },
  "domain": "validation",
  "keyword": "required",
  "message": "object has missing required properties ([\"lastName\"])",
  "required": [ "firstName", "lastName" ],
  "missing": [ "lastName" ]
} ]
```

VS.

```
[ { "field": 'data.lastName', "message": 'is required' } ]
```

- Plethora of validators for Javascript (which one to choose?) => ajv
- Writing schemas
  - happens mostly by hand
  - tedious & error-prone

The Ugly

The Ugly (v6)

- Sometimes unclear semantics
  - additionalProperties in combination with other keywords
    - https://github.com/json-schema/json-schema/issues/116
    - https://github.com/json-schema/json-schema/issues/240
  - Also, conflicting schemas that never validate
- Confusing parts
  - scope refining mechanics via id, relative URLs
    - https://github.com/json-schema/json-schema/issues/17
    - https://github.com/json-schema/json-schema/issues/16
  - Behaviour on missing type keyword
  - ...
- Overview

Conclusion and Take-aways

Warning: Our opinion ahead

- JSON Schema is a schema (meta-description) for JSON Instances
- JSON Schema is important, widely adopted and enables many applications
- You should definitely consider using JSON Schema if...
  - your application uses lots of JSON
  - your application needs to validate JSON often
  - your application uses multiple technology stacks
- To get started read (see refs slide): JSON Schema Introduction by Space Telescope
- Slides and Demo: https://goo.gl/VUGmdp

CC Image by gnokii

Thank you!

Questions?

References

- Main Site: http://json-schema.org/
- Github Main Page: https://github.com/json-schema/json-schema
- Best Guide: https://spacetelescope.github.io/understanding-json-schema/index.html
- Validators:
  - https://github.com/epoberezkin/ajv (JS)
  - https://github.com/geraintluff/tv4 (JS)
  - Benchmark: https://github.com/ebdrup/json-schema-benchmark (JS)
  - https://github.com/daveclayton/json-schema-validator (Java)
  - https://github.com/eclipsesource/play-json-schema-validator (Play Scala)
- Extensive list of software: http://json-schema.org/implementations.html