# JSON Schema

Kris Zyp
SitePen

# JSON Schema

- JSON-based Schema for JSON data
- Structure definition
- Validation
- Structured documentation
- Hyperlink relation definitions

# JSON Schema (cont.)

- Core Schema validates instances and allowed values
- Key validating attributes:
  - **`type`** - represents the value's required type (object, string, etc.)
  - **`properties`** - key/value object of fields and types
  - **`minimum/maximum`** - ensures the minimum and maximum value of the data
  - **`items`** - references the child objects of an array
  - **`pattern`** - a pattern by which a value must match
  - **`extends`** - should be another schema which will provide a base schema which the current schema will inherit from.

# JSON Schema (cont.)

- Core Schema validates instances and allowed values
- Key documentation attributes:
  - **`title`** - brief title of property
  - **`description`** - full explanation of property
  - **`default`** - default value that should be used for property

# Sample: JSON Schema

```
// Schema file:
{"description":"A person",
   "type":"object",
   "properties":
      {"name": {"type":"string"},
         "age" : {"type":"integer",
         "maximum":125
}}}

// Instance of schema file:
{"name":"Kris",
   "age":30
}
```

# Sample: JSON Schema (cont.)

```
// schema:
{
    "type":"object",
    "properties":
    // union type, string OR number allowed
    {"age": {"type":["string","number"]},
        "gender" : {"type":"string",
        // A value of "male" or "female" is allowed
            "enum":["male","female"]
}}}

// instance:
{"age":31,
    "gender":"male"
}
```

# Sample: JSON Schema (cont.)

```
// schema:
{"type":"array",
   "items":[{"type":"string"},{"type":"number"}]
}

// instance:
["foo",44]
```

# Sample: dojox.json.schema (cont.)

```
// Create a schema by which to validate data
var schema = {
    type: "object",
    properties: {
        posts: {
            type: "object",
            items: {
                type: "object",
                properties: {
                    id: { type: "number" },
                    avatar: { type: "string" },
                    username: { type: "string" }
                }
            }
        }
    }
};
// Describes the following object
var objectToValidate = {
    posts: [
        { id: 5, avatar: "albert.jpg", username: "AlbertEinstein" },
        { id: 3, avatar: "copernicus.jpg", username: "Copernicus" }
    ]
};
// dojox.json.schema validates the data against the schema
var results = dojox.json.schema.validate(objectToValidate, schema); // results.valid =
true
```

# JSON Hyper Schema

- Defines the properties of a JSON structure that represent links or references within data structures
- Allows for truly RESTful interchange
- Can be referenced from instances by Link headers or media type parameters
- Flexible hyperlinking, data can be any structure, and still have hyperlink data embedded
  - Allows for more compact representations
  - Works with existing data structures without modification

# Sample: JSON Hyper Schema

```
// Sample book data
Content-Type: application/json; schema=http://www.book-warehouse.com/book-schema
[
    { "title": "Oliver Twist", "price": 16.99 },
    { "title": "Robinson Crusoe", "price": 15.99 }
]

// The book schema
Content-Type: application/schema+json; schema=http://json-schema.org/hyper-schema
{
    "properties": {
        "title": { "type": "string", "description": "The title of the book" },
        "price": { "type": "number", "description": "The price of the book in US" }
    }
}

// Extend the schema
{
    "extends": { "$ref": "http://book-warehouse.com/book-schema" },
    "properties": {
        "ukPrice": { "$ref": "http://uk-books.com/schema/book#properties.price" }
    }
}
```

# Sample: JSON Schema

```
// Schema file:
{
   "description":"A person",
   "type":"object",
   “links”: [
     {“href”: “{id}”, “rel”: “self”},
     {“href”: “{spouseId}”, “rel”: “spouse”}
   }]
}}}

// Instance of schema file:
{
   "name":"Kris",
   "age":30,
   ”id”: “kris”,
   ”spouseId”: “nikki”
}
```

# JSON Pointers

- Describes complex object graphs
  - Circular and Multiple references
- Describe references between messages
- JSON Pointers is hyperlinking applied to JSON
  - URL-based for cross-site references
- A specific form of hyperlinking in JSON (can be described with a hyper schema)
- The hyperlinking format for JSON Schema

# JSON Referencing Example

```
// ID-Based
var me = {
    "name": "Kris",
    "father": { "name": "Bill", "wife": { "$ref": "2" } },
    "mother": { "name": "Karen", "id": "2" }
};

// Path-Based
var me = {
    "name": "Kris",
    "father": { "name": "Bill", "wife": { "$ref": "#mother" } },
    "mother": { "name": "Karen" }
};

// Relative URLs
{
    "id": "2",
    "amount": 5.99,
    "purchasedBy": { "$ref": "/Person/1" }
}
```

# JSON Schema

- JSON-based Schema for JSON data
- Structural definition with same-format syntax
- Well integrated with documents in web context
- Extensive libraries