Next Generation Forms for PDF

John Brinkman

Principal Scientist, Adobe

September 2022

Outline

- **Overview**
- Responsive mode / Accessibility
- JSON
- Replace JavaScript
- Constraints
- Locale
- Date and Timezone
- Extensibility

High Level Goals

PDF Forms have not been enhanced in ~ 20 years (!)

1. Forms that work in classic mode and responsive mode (HTML)
2. Improved Accessibility
3. Remove JavaScript dependency, use declarative constructs

Status

- PDF Association WG meeting since November 2020
- Strawman proposals complete for all areas
  - https://github.com/pdf-association/pdf-forms
- Reference Implementation in progress

Incremental Approach

- Build on the basic structure of form definitions (`/Fields` array)
- Enhancements based on new properties and dictionaries
- A forms.next PDF loaded by a legacy viewer should be partially functional

Influencing Technologies

- HTML5 forms
- JSON data / JSON Schema
- ICU (International Components for Unicode)
- WAI-ARIA
  (Web Accessibility Initiative – Accessible Rich Internet Applications)

Outline

- Overview
- **Responsive mode / Accessibility**
- JSON
- Replace JavaScript
- Constraints
- Locale
- Date and Timezone
- Extensibility

Responsive mode + Accessibility

- Both are rooted in HTML best practices
- Both rely on a well-formed structure tree

Accessibility

Follow Web Content Accessibility Guidelines (WCAG)

- Field labelling
- Field grouping

Field Labeling

Simplest, most common case: One field, one label:


```
<label for="fname">First Name: </label>
<input id="fname" name="firstname">
```

HTML Label Element

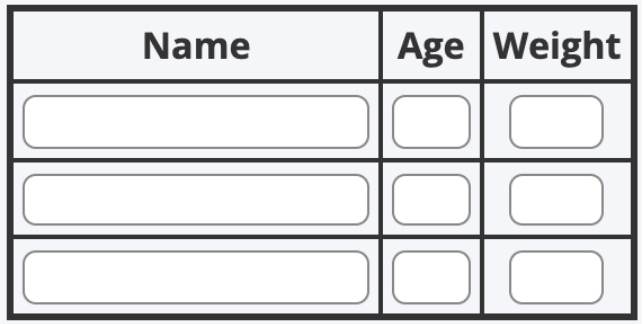
- Screen readers use label content to announce the field
- When a label receives focus, it passes the focus on to its associated control
- Clicking label will focus/activate the input element

New Label Widget

- Add a new annotation type for labels to mimic HTML label
- Label properties available in expressions
  - Label Value, Appearance, Visibility may all be modified

Special label cases:

One label per multiple fields:



Special label cases:

Multiple labels, one field:

Special label cases:

Descriptive label:

Special case handling

Add `aria-*` attributes to the structure tree:

- `aria-labelledby`
- `aria-describedby`

Field grouping

Accessibility Requirement:

- Radio buttons must be grouped in a <fieldset> element
- <fieldset> must have an associated <legend> element

Fieldsets not just for radio button groups – generic grouping mechanism

Field grouping in PDF

- PDF has always allowed nested fields
  - "terminal fields" are widgets
  - "non-terminal" fields are for grouping
- Non-terminal fields now referred to as fieldsets
- The top-level aggregating fieldset referred to as a formRoot
- Label widget also serves as a legend

Derivation Algorithm

Fields dictionary + Structure tree => HTML

formRoot	=>	<form>
"non-terminal fields"	=>	<fieldset> or <table> / <tr> / <section> …
Label annotations	=>	<label> / <legend> or <caption> / <h2>
Field widgets	=>	<input>, <select>, <textarea>, <button>
Field properties	=>	data-* attributes

HTML Derived from PDF

```
<</Fields [
 << /T (purchaseOrder)
  /FieldType (formRoot)
  /Kids [
  << /T (address)
    /Kids [
    << /FieldType /Legend
       /T (addrlabel) /V (Address) >>

    << /FieldType /Label
       /T (streetlabel) /V (Street Name)
       /For X Z R >>

    << /FieldType (text)
       /T (street)
       /V (Maple Street) >>
    ]
  ]
 >>
]>>
```

```
<form data-pdf-name="purchaseOrder">
  <fieldset data-pdf-name="address">
    <legend>Address</legend>

    <label
      for="purchaseOrder.address.street">
      Street Name</label>

    <pdf-field
      data-pdf-name="street"
      data-pdf-fieldtype="text"
      data-pdf-value="Maple Street"
    ></pdf-field>

  </fieldset>
</form>
```

Outline

- Overview
- Responsive mode / Accessibility
- **JSON**
- Replace JavaScript
- Constraints
- Locale
- Date and Timezone
- Extensibility

JSON data

- Acroforms: data interchange based on XML
- Forms.next: Import/export/submit data in JSON format
- Enhanced Submit()
  - Full control over method, headers, body
  - Allow data transformation on export/submit
- Data Import: Carry data that's not bound to fields. e.g., parts list

Data Model

```
/DataModel <<
  /taxrates <<
    /AL 0.4
    /AK 0.0
    /AZ 0.56
    /AR 0.65
    /CA 0.725
    /CO 0.29
  >>
  /orderNum ()
>>
```

```
/Fields
[
  << /T (address) /Kids [
    << /T (street) >>
    << /T (city) >>
    << /T (state) >>
    << /T (zipcode) >>
  ]>>
  << /T (orderNum) /V (1234)
]
```

Merged Data Model:

```
{
  "taxrates": {
    "AL:": 0.4,
    "AK:": 0.0,
    "AZ:": 0.56,
    "AR:": 0.65,
    "CA:": 0.725,
    "CO:": 0.29
  },
  "address": {
    "street": "",
    "city": "",
    "state": "",
    "zipcode": ""
  },
  "orderNum": "1234"
}
```

Enforce JSON Schema Definitions

Import/export/submit data matching JSON Schema definitions

1. Preserve hierarchy via data model and nested fields
2. Enforce value constraints (type, min, max etc.)
3. Relational validations (anyOf, oneOf, allOf) supported through various form design techniques

Outline

- Overview
- Responsive mode / Accessibility
- JSON
- **Replace JavaScript**
- Constraints
- Locale
- Date and Timezone
- Extensibility

JavaScript is Evil

- Running user-authored (untrusted) JavaScript in the browser
- Programmer-level skills for form authoring
- JS interpreter in all PDF run-times
- Subject to excesses and abuses

Need an Expression Grammar

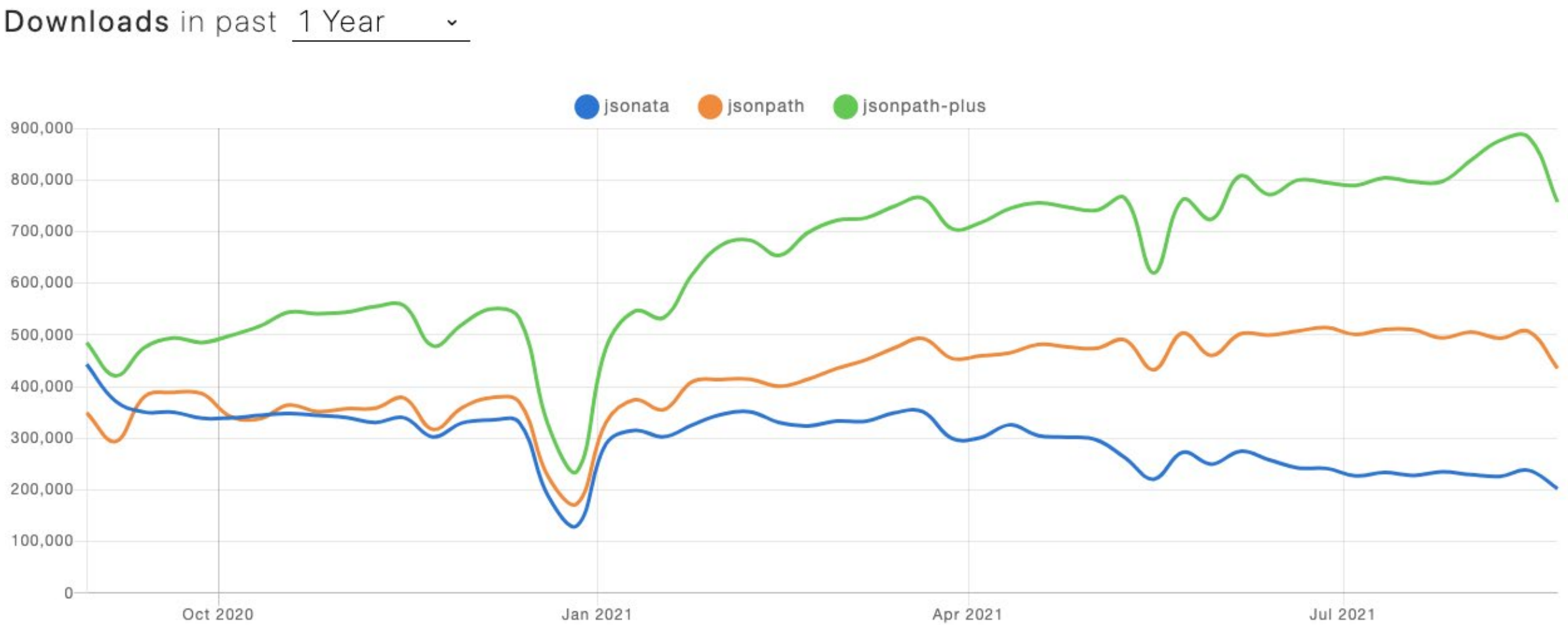
Model after spreadsheet formulas:

- Results of an expression populates current ~~cell~~ field
- Single statement: no looping, no variables
- Support hierarchical (JSON) data
- Replace cell addresses with field/data names
- Encapsulated: fields modify only themselves – not other fields

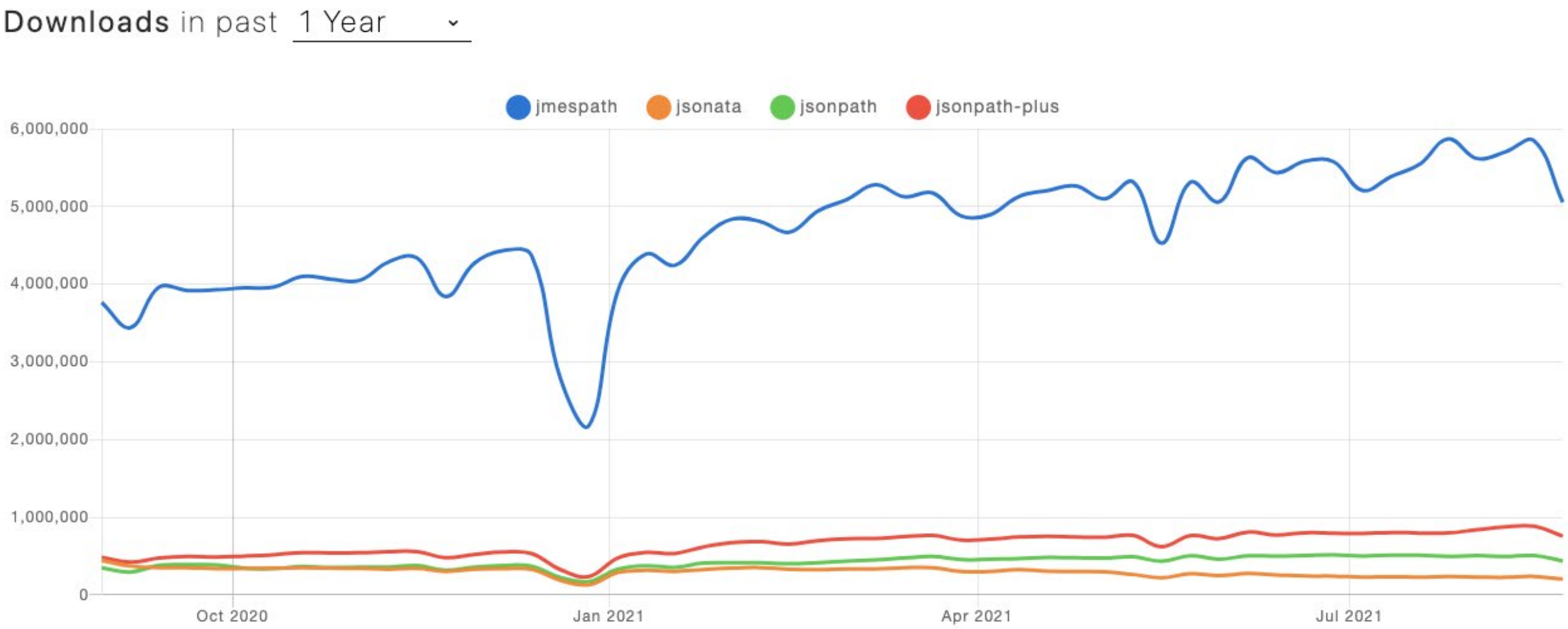
Starting point:

- OpenFormula Level 1
- For JSON support, add a json-path derivative

json-path deriviatives



JMESPath


Downloads in past 1 Year
jmespath
jsonata
jsonpath
jsonpath-plus
6,000,000
5,000,000
4,000,000
3,000,000
2,000,000
1,000,000
0
Oct 2020
Jan 2021
Apr 2021
Jul 2021

JMESPath (https://jmespath.org/)

- Well-defined ABNF grammar
- Implementations available:
  - Python
  - PHP
  - Javascript
  - Ruby
  - Lua
  - Go
  - Java
  - Rust
  - DotNet
  - C++

json-formula

Mashup of JMESPath and Open Formula

Open-source implementation https://github.com/adobe/json-formula

json-formula functions

- abs
- avg
- caseFold
- ceil
- charCode
- codePoint
- contains
- date
- day
- decodeUrl
- decodeUrlComponent
- encodeUrl
- encodeUrlComponent
- endsWith
- entries
- exp
- false
- find
- floor
- fromEntries
- hour
- if
- join
- keys
- left
- length
- lower
- map
- max
- maxBy
- merge
- mid
- min
- minBy
- minute
- mod
- month
- notNull
- now
- null
- power
- proper
- replace
- rept
- reverse
- right
- round
- second
- sort
- sortBy
- split
- sqrt
- startsWith
- stdev
- stdevp
- substitute
- sum
- time
- toArray
- toNumber
- today
- tostring
- trim
- true
- trunc
- type
- unique
- upper
- values
- weekday
- year
- zip

```
{
  "address": {
    "street": "12 Oak St",
    "city": "San Jose",
    "state": "CA",
    "country": "USA",
    "phone": "1234561234"
  },
  "items": [
    {
      "desc": "pens",
      "quantity": 2,
      "price": 3.23
    },
    {
      "desc": "pencils",
      "quantity": 3,
      "price": 1.34
    }
  ],
  "tax": 1.13
}
```


address.street => 12 Oak St

```
{
  "address": {
    "street": "12 Oak St",
    "city": "San Jose",
    "state": "CA",
    "country": "USA",
    "phone": "1234561234"
  },
  "items": [
    {
      "desc": "pens",
      "quantity": 2,
      "price": 3.23
    },
    {
      "desc": "pencils",
      "quantity": 3,
      "price": 1.34
    }
  ],
  "tax": 1.13
}
```


```
substitute(address.street, 'Oak', 'Maple')
```

=> 12 Maple St

```
{
  "address": {
    "street": "12 Oak St",
    "city": "San Jose",
    "state": "CA",
    "country": "USA",
    "phone": "1234561234"
  },
  "items": [
    {
      "desc": "pens",
      "quantity": 2,
      "price": 3.23
    },
    {
      "desc": "pencils",
      "quantity": 3,
      "price": 1.34
    }
  ],
  "tax": 1.13
}
```


```
items[*].price

    => [3.23,  1.34]
```

```
{
  "address": {
    "street": "12 Oak St",
    "city": "San Jose",
    "state": "CA",
    "country": "USA",
    "phone": "1234561234"
  },
  "items": [
    {
      "desc": "pens",
      "quantity": 2,
      "price": 3.23
    },
    {
      "desc": "pencils",
      "quantity": 3,
      "price": 1.34
    }
  ],
  "tax": 1.13
}
```


```
items[*].price * items[*].quantity

    => [ 6.46, 4.02 ]
```

```
{
  "address": {
    "street": "12 Oak St",
    "city": "San Jose",
    "state": "CA",
    "country": "USA",
    "phone": "1234561234"
  },
  "items": [
    {
      "desc": "pens",
      "quantity": 2,
      "price": 3.23
    },
    {
      "desc": "pencils",
      "quantity": 3,
      "price": 1.34
    }
  ],
  "tax": 1.13
}
```


```
sum(items[*].price * items[*].quantity)
  * tax

    => 11.8424
```

```
{
  "address": {
    "street": "12 Oak St",
    "city": "San Jose",
    "state": "CA",
    "country": "USA",
    "phone": "1234561234"
  },
  "items": [
    {
      "desc": "pens",
      "quantity": 2,
      "price": 3.23
    },
    {
      "desc": "pencils",
      "quantity": 3,
      "price": 1.34
    }
  ],
  "tax": 1.13
}
```


```
round(
 sum(
  items[*].price * items[*].quantity
 ) * tax, 2
)

    => 11.84
```

```
{
  "address": {
    "street": "12 Oak St",
    "city": "San Jose",
    "state": "CA",
    "country": "USA",
    "phone": "1234561234"
  },
  "items": [
    {
      "desc": "pens",
      "quantity": 2,
      "price": 3.23
    },
    {
      "desc": "pencils",
      "quantity": 3,
      "price": 1.34
    }
  ],
  "tax": 1.13
}
```


```
'(' & mid(address.phone, 0, 3) & ')' &
      mid(address.phone, 3, 3) & '-' &
      mid(address.phone, 6, 4)

   => (123)456-1234
```

OR:

```
address.phone |
'(' & mid(@, 0, 3) & ')' &
      mid(@, 3, 3) & '-' &
      mid(@, 6, 4)
```

json-formula added to forms

- json-formula expressions configured as event listeners
- Expression returns a scalar: assign field value (the common case)
- Expression returns a map/object: assign field properties

Common operations are easy to express

Complex operations are possible

Enhanced Event Processing

- Can now dispatch events
- New Custom Events
- Limited to forms-related events (app/document events not exposed)

Form Events

- load
- reset
- submit
- custom events

Form Events emit down via depth-first traversal

Cannot be cancelled

Field Events

Field Events

- focus
- blur
- change
- click
- format
- mouseEnter
- mouseLeave
- valid
- invalid
- keystroke

Most field events propagate:

- Bubble up
- Can stop propagation
- Can cancel

json-formula integration to Forms

- dispatchEvent()
- submit()
- import()
- export()
- format()
- parse()

Add shortcut symbols:

$ - Current field

$form - Root of the forms tree

Field properties:

```
firstName => 'John'
firstName.$value => 'John'
firstName.$required => true
```

Properties prefixed with "$" to disambiguate from child fields

Sample expressions

Assign field value using a change event handler expression:

```
sum(transactions[*].amount)
```

In context:

```
<<
  /T (subtotal)
  /Events <<
    /Change [(sum(transactions[*].amount))]
  >>
>>
```

Sample expressions

Assign value, and make negative values red:

```
{
  $value: sum(transactions[*].amount),
  $foregroundColor: if (sum(transactions[*].amount) < 0, '#F00', '#000')
}
```

OR:

```
sum(transactions[*].amount) |
  {
    $value: @,
    $foregroundColor: if (@ < 0, '#F00', '#000')
  }
```

Compare to JS

Combine uppercase first and last name in a calculation event:

```
var firstName = this.getField('firstName');
var lastName = this.getField('lastName');
event.value = firstName.value.toUpperCase() + ' ' + lastName.value.toUpperCase();
```

json-formula equivalent in a change event:

```
upper(firstName) & ' ' & upper(lastName)
```

Assign read-only and required

JavaScript

```
var paymentType = this.getField('paymentType');
var creditCardNo = this.getField('creditCardNo');
creditCardNo.readonly = paymentType.value !== 'creditcard';
creditCardNo.required = paymentType.value === 'creditcard';
```

json-formula

```
{
  $readonly: paymentType != 'creditcard',
  $required: paymentType == 'creditcard'
}
```

Dependency Tracking

- Runtime tracks which fields are referenced in an expression (dependents)
- Any change in a field triggers a change event on its dependents
- No need to explicitly "recalculate all"

Outline

- Overview
- Responsive mode / Accessibility
- JSON
- Replace JavaScript
- **Constraints**
- Locale
- Date and Timezone
- Extensibility

Constraints

Borrow from:

- JSON Schema validations
- HTML5 form validations

- Data Type
- Format (date / email / image …)
- Numeric range (min/max)
- Maximum/minimum Length
- Regex match
- Step
- Enum
- Required
- Size of array
- Unique array
- Attachment size, mime type
- json-formula expression

Verify 'age' field is a positive integer

JavaScript validation script:

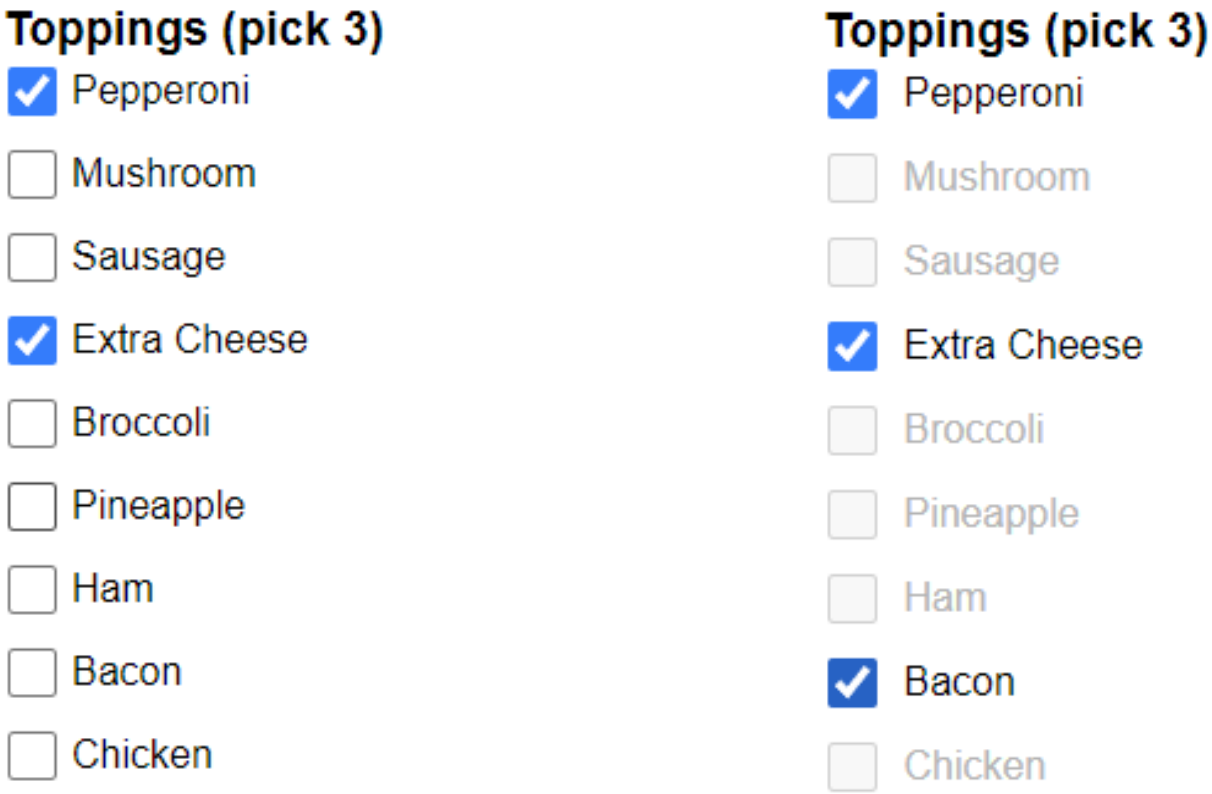
```
var val = parseFloat(event.value);
if (val < 0 || Math.ceil(val) !== val) {
  app.alert('Age must be a whole number greater than zero ');
  event.rc= false;
}
```

Declarative Constraints:

```
/Constraints <<  /Type (integer)  /Min 0 >>
/ValidationMessage 'Age must be a whole number greater than zero'
```

Checkbox Groups

- User may select up to <n> values
- Once limit is reached, unselected checkboxes are read-only
- Result is returned as an array



Checkbox Groups

- Each checkbox shares the same name
- Type is an array
- MaxItems specified
- Each checkbox populates the array with its enumerated value (/Opt)

```
<<
  /T (toppings)
  /FT /Btn
  /FieldType /Checkbox
  /Constraints <<
    /Type ([string])
    /MaxItems 3
  >>
  /Opt [(Chicken)]
>>
```

Outline

- Overview
- Responsive mode / Accessibility
- JSON
- Replace JavaScript
- Constraints
- **Locale**
- Date and Timezone
- Extensibility

Locale/Language support

- Language is an inherited property for fields
- Display numbers, currency and date/time in locale-sensitive formats:
  - Short/long dates
  - Currency
  - Radix
  - Thousands separators

Formatting Patterns

Based on International Components for Unicode(ICU)

Implementations:

- C (ICU4C)
- Java (ICU4J)
- Javascript: Intl.DateTimeFormat, Intl.NumberFormat are compatible with ICU

Formatting Patterns

New `/Patterns` dictionary:

- Format to display value of a field (without focus)
- Format to edit the field (with focus)
- Format to serialize data (submit/export)

Date/number formats

Value	Format	Language	Result
1234.56	number|currency/USD	en-US	$1,234.56
1234.56	number|currency/EUR	de-DE	1.234,56 €
2022-03-21	date|full	en-CA	Monday, March 21, 2022
2022-03-21	date|yyyy.MM.dd G 'at' HH:mm:ss zzz	en-CA	2022.03.21 AD at 12:08:56 PDT

Outline

- Overview
- Responsive mode / Accessibility
- JSON
- Replace JavaScript
- Constraints
- Locale
- **Date and Timezone**
- Extensibility

Time Zone Support

- date and datetime are supported field formats
- timezone property added to fields/fieldsets

```
<<
  /T (appointment)
  /Constraints << /Format (datetime) >>
  /Timezone (America/Los_Angeles)
  /V (2022-09-12T16:30:00)
>>
```

- Runtime processes dates as UTC values
- Timezone setting used for input and display

Time Zone Support

Europe/Berlin

07.09.22, 18:06:05 MESZ

```
<<
  /T (appointment)
  /Lang (de-DE)
  /Pattern << /Format (date|short long) >>
  /TimeZone (Europe/Berlin)
>>
```

America/Toronto

2022-09-07, 12:06:05 p.m. EDT

```
<<
  /T (appointment)
  /Lang (en-CA)
  /Pattern << /Format (date|short long) >>
  /TimeZone (America/Toronto)
>>
```

America/Los_Angeles

9/7/22, 9:06:05 AM PDT

```
<<
  /T (appointment)
  /Lang (en-US)
  /Pattern << /Format (date|short long) >>
  /TimeZone (America/Los_Angeles)
>>
```

Outline

- Overview
- Responsive mode / Accessibility
- JSON
- Replace JavaScript
- Constraints
- Locale
- Date and Timezone
- **Extensibility**

Custom properties

- Arbitrary custom properties defined in /Properties dictionary
- read/write access from json-formula

e.g.

- Translated strings for labels, validation messages
- Centralize lists to populate combo-boxes
- Conversion rates
- ...

Custom Widgets

- `/FieldSubtype` property allows new widget types
- `/FieldType` and `/FT` must also be populated to provide fall-back behavior for viewers that do not support the custom widget

```
<<
 /T (percentage)
 /FieldSubtype (ADBE:RangeSlider)
 /FieldType /Text /FT /Tx
 /Constraints <<
  /Min 0 /Max 100 /Step 5 /Type (Integer)
 >>
>>
```

Extensibility

Out-of-the-box functionality doesn't meet all requirements

Extension mechanisms to fill gaps:

- Data model content
- Custom field properties
- Custom Events
- Custom Widgets
- Host application response to events

Summary

- Easier to author
- Filling Experience
  - Accessible
  - Responsive
  - Internationalization
- Interoperable
  - data manipulation
  - data exchange
- Extensible