JSONEX – JSON expression

JSON is great in representing static data however there are many cases that we need standard way to represent dynamic data in JSON itself

# Reserved names

name in JSON that use for operators

|  |  |
| --- | --- |
| **name** | **Operators or special variable** |
| and |  |
| or |  |
| equal |  |
| greater |  |
| less |  |
| lessOrEqual |  |
| greaterOrEqual |  |
| not |  |
| in |  |
| notin |  |
| name |  |
| value |  |

# Simple expression

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| operator | JSON expression | JSON data to evaluate | Result in native value e.g., true or false, 1,2, [1,2] | note |
| Reference a field value | {“name”:”data”} | {“data”:100} | 100 |  |
| A native value | 100  1.2  “hello world”  true  false  [1,2,3,4]  [1,2,”text”,null]  null |  | 100  1.2  “hello world”  true  false  [1,2,3,4]  [1,2,”text”,null]  null | string, number, boolean, null, array of values without type conversion no date value |
| equal | {“$data”:100} | {“data”:100} | true | data === 100  reference a property in JSON object using ${name} |
| equal null | {“$data”:null} | {“data”:null}  {“abc”:123} | true  true | data === null  $data == null since undefined (no “data” field)  == null  Use to check both a property in JSON data is store null value or not this property name |
| in – equal to a member | {“$gender”:[“male”, ”female”]} | {“gender”:”male”} | true | “male” in [“male”, “female”] |
| not | {“!$data”:100}} | {“data”:100} | false | data !== 100  reference a property in JSON object using !${name}  for native value (not array, object) |
| not null | {“!$data”:null } | {“data”:100}  {“data2”:100} | true  false | data != null  data != null => false, no data property  reference a property in JSON object using !${name} for null value |
| not in | {“!$age”:[15,25,35,45]} | {“age”:16 } | true | age not in [15,25,35,45] |

# Operators

## comparison operators

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| operator | JSON expression | JSON data to evaluate | Result in native true or false | note |
| equal | {“$data”:100}  {“equal”:[ {“type”:”date.weekday”, “name”:”$today”},  “Monday” ]  }  {“equal”:[ {“type”:”date.weekday”, “name”:”$today”},  {“type”:”date.weekday”, “name”:”birthday”},  “Thursday”]  }  {“equal”:[1,1,1]} | {“data”:100}  Today:”2022-10-06”  {“birthday”:”2022-10-13”}  Today:”2022-10-06” | True  false  true  true | data === 100  it’s Thursday  “Thursday” ==”Thursday” ==”Thursday”  The day of birthday and today are Thursday  1==1==1 |
| equal null | {“$data”:null} | {“data”:null}  {“abc”:123} | true  true | data === null  data === null since undefined (no “data” field)  === null |
| greater than | {“$age”: {“greater”:18}}  {“greater”:[18,25]}  {“greater”:[25,10,1]} | {“age”:25} | true  false  true | age > 25  18 > 25  25>10>1 |
| less than | {“$age”: {“less”:40}  {“less”:[25,10,11]} | {“age”:50} | false  false | Age < 50  25<10<11 |
| greater than or equal | {“$name”: {“greaterOrEqual”:”Apollo” }}  { “greaterOrEqual”:[25,10,10]} | {“name”:”apollo”} | false  true | name >= “apollo”  25>=10>=10 |
| less than or equal | {“$birthday”: {lessOrEqual:”2000-01-01” }}  { “lessOrEqual”:[25,50,10]} | {“birthday”:”1999-12-31”} | true  false | birthday >= “2000-01-01”  25<=50<=10 |
| between  Must always have 2 values in array | {“$BTSfare”:{“between”:[15,59]}} | {“BTSfare”:59} | true | BTSfare >=15 && BTSfare <=59 |
| in – equal to a member | {“$gender”:[“male”,”female”]}  {“in”:{“name”:”gender”,”values”:[ “male”,”female”]}  {“in”:{“value”:”male”,”values”:[ “male”,”female”]} | {“gender”:”male”}  {“gender”:”male”} | true  true  true | “male” in [“male”, “female”]  “male” =”male” |
| not null | {“!$data”:null } | {“data”:100}  {“data2”:100} | true  false | data != null  data != null => false, no data property |
| not in | {“!$age”:[15,25,35,45]}  {“notin”:{“name”:”age”,”values”: [15,25,35,45]}  {“notin”:{“value”:25,”values”: [15,25,35,45]} | {“age”:16 }  {“age”:16 } | true  true  false | age not in [15,25,35,45]  age != 15, age != 25, age != 35, age != 45  25 not in [15,25,35,45] |

## Logical operators

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| operator | JSON expression | JSON data to evaluate | Result in native true or false | note |
| and | {and:[true,true,true,{“$data”:100}]} | {“data”:100} | True | true && true && true && true |
| or | {or:[false,false,false,{“$data”:100}]} | {“data”:100} | true | false || false || false || true |
| not | {“!$data”:100}}  {“not”:{“name”:”is\_good”}}  {“not”:{“$data”:100}}  {“not”:[true,false,1,2,0,null]}  {“not”:{“name”:”data” } } | {“data”:100}  { “is\_good”:false }  {“data”:100}  {“data”: [true,false,1,2,0,null]} | false  true  false  [false, true, false, false, true, true]  [false, true, false, false, true, true] | data !== 100  !false  !(data === 100)  [!true, !false, !1, !2, !0, !null]  [!true, !false, !1, !2, !0, !null] |

# TYPE Conversion

## Boolean

Boolean native value still represents as true, false

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| operator | JSON expression | JSON data to evaluate | Result in native true or false | note |
| return boolean | {“type”:”boolean”  “name”: “is\_thai” } | {“is\_thai”: true}  {“is\_thai”: false}  {“is\_thai”: 1}  {“is\_thai”: 0}  {“is\_thai”: “text”} | true  false  true  false  false | 1 == true  0 == true  “text”== true |
| return boolean  value | {“type”:”boolean”, “value”: true }  {“type”:”boolean”, “value”: false }  {“type”:”boolean”, “value”: 0 }  {“type”:”boolean”, “value”: 1 }  {“type”:”boolean”, “value”: 11.1 }  {“type”:”boolean”, “value”: null } |  | true  false  false  true  false  false | 0 == true  1 == true  11.1 == true  null == true |
| not the day in week | {“not”:{“equal”:[  {“type”:”date.weekday”}, ”name”:”birthday”},  “Monday”  ]}} | {“birthday”:”2022-10-06” } | true | Birthday is not Monday.  It’s “Thursday”. |

## Date

Date type in JSON expression

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| case | JSON expression | JSON data to evaluate | Result always in date instance | note |
| return date | {“type”:”date”,  “name”:”birthday”}  {“type”:”date”,  “value”:”$today”}  {“type”:”date”,  “value”:”$now”} | {“birthday”:”2002-01-01” }  {“birthday”: 1665033868536}  {“birthday”: 0}  {“aday”:1000}  {“birthday”:”Not a day”}  today:”2022-10-06”  now:”2022-10-06 13:01:01.599” | “2002-01-01 00:00:00.000”  “2022-10-06T05:24:28.536Z”  “1970-01-01T00:00:00.000Z”  “1970-01-01T00:00:00.000Z”  “Invalid Date”  ”2022-10-06”  Return special field today’s value  ”2022-10-06 13:01:01.599”  Return special field now’s value | date will treat as local time e.g. “2002-01-01” will not treat as 00:00 in UTC time as Date in Javascript  “$today”can be used to reference today date without time,  “$now” can be used to reference current datetime.  Both cannot use outside of “type”:”date” e.g., {“$today”:100}  Mean reference value of JSON property name “today” |
| return date value | {“type”:”date”,  “value”:”2022-01-01”}  {“type”:”date”,  “value”: 1665033868536}  {“type”:”date”,  “value”: “Not a date”} |  | “2002-01-01 00:00:00.000”  {“birthday”: 1665033868536} | See [JavaScript Date Objects (w3schools.com)](https://www.w3schools.com/js/js_dates.asp)  Throw error, does not allow value that cannot convert to date |
| Modify date | {  “type”:”date”,  “name”:”birthday”,  “add”:{  “day”:15,  “year”:1,  “hour”:1  }  }  {  “type”:”date”,  “name”:”birthday”,  “subtract”:{  “day”:15,  “year”:1,  “hour”:1  }  } | {“birthday”:”2002-01-01” }  {“birthday”:”2002-01-16 05:00” } | “2003-01-16 01:00:00.000”  “2001-01-01 04:00” | Support  “day”, “month”, ”year”, “hour”, “minute”, “second” |
| Before | {“$expire\_date”: {“greater”:{“type”:”date”, “value”:”$today”}  }  }  {“$expire\_date”:  {  “greater”:  {“type”:”date”,  “value”:”$today”,  “add”:{  “day”:15  }  }  }  } | {“expire\_date”:”2222-12-01” }  today:”2022-10-06”  {“expire\_date”:”2022- 10-15” }  today:”2022-10-06”  today + 15: “2022-10-21 | true  False | Before expire date  Today < expire\_date – 15  == expire\_date > today + 15 |
| within 15 days before expired | {“$expire\_date”:  {  “between”: [  {“type :”date”, “value”:”$today”  },  {type:”date”,  “value”:”$today”,  “add”:{  “day”:15  }  ]  }  }  } | {“expire\_date”:”2022- 10-15” }  {“expire\_date”:”2022- 10-01” }  {“expire\_date”:”2022- 10-21” }  today:”2022-10-06”  between: [“2022-10-06”, “2022-10-21”] | true  false  true | expired before today  start of the period |

## Part of date

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| case | JSON expression | JSON data to evaluate | Result in number except weekday (string) | note |
| day in week  1-7  Sunday=1  Monday=2  …  Saturday=7 | {type:”date.day”,  “name”:”birthday”} | {“birthday”:”2002-01-01” }  {“birthday”:”Not a day”} | 2  NaN | date will treat as local time e.g. “2002-01-01” will not treat as 00:00 in UTC time as Date in Javascript |
| Weekday  “Sunday”, “Monday”,  “Tuesday”,  …  “Saturday” | {“type”:”date.weekday”,  “name”:”birthday”} | {“birthday”:”2022-10-06” }  {“birthday”:”Not a day”} | “Thursday”  null |  |
| Date of month | {“type”:”date.date”,  “name”:”birthday”} | {“birthday”:”2022-10-06” }  {“birthday”:”Not a day”} | 6  NaN |  |
| Month of year | {“type”:”date.month”,  “name”:”birthday”} | {“birthday”:”2022-10-06” }  {“birthday”:”Not a day”} | 10  NaN |  |
| Date of month | {“type”:”date.year”  “name”:”birthday”} | {“birthday”:”2022-10-06” }  {“birthday”:”Not a day”} | 2022  NaN |  |
| Hour of day | {“type”:”date.hour”  “name”:”birthday”} | {“birthday”:”2022-10-06 21:00” }  {“birthday”:”Not a day”} | 21  NaN |  |
| Minute of hour | {“type”:”date.minute”  “name”:”birthday”} | {“birthday”:”2022-10-06 21:00” }  {“birthday”:”Not a day”} | 00  NaN |  |
| Second of minute | {“type”:”date.secord”  “name”:”birthday”} | {“birthday”:”2022-10-06 21:00:12” }  {“birthday”:”Not a day”} | 12  NaN |  |
| Birthday is 25th | {“target\_date”:{“type”:”date.date”  “name”:”birthday”}} | {target\_date:25,  “birthday”:”2022-10-06 21:00:12”} | false |  |
| Every birthday’s month | {“equal”:[{type:”date.month”, “name”:”birthday”}, {“type”:”date.month”, “value”:”$today”}]} |  |  |  |