

# Drona Pay Private Limited- Assignment

~ Radhika Choudhary

- a. If the input string has only 3 characters, please check if it is a Pallindrome and give output "Pallindrome" or "Not a Pallindrome". If input string has more than 3 characters then output is "Large".

**Code:**

```
{
  "if": [
    {"===": [{"method": "length", "arguments":
[{"var": "input"}]}, 3]},
    {
      "if": [
        {"===": [{"var": "input"}, {"method":
"reverse", "arguments": [{"var": "input"}]}]},
        "Palindrome",
        "Not a Palindrome"
      ]
    },
    "Large"
  ] }
```

## Output:

# JsonLogic

Build complex rules, serialize them as JSON, share them between front-end and back-end

```
// Rule:
{
  "if": [
    { "==": [
      { "method": "length", "arguments": [{"var": "input"}], 3 }
    ]
  },
  { "if": [
    { "==": [
      { "var": "input", "method": "reverse", "arguments": [{"var": "input"}]
    },
    "Palindrome",
    "Not a Palindrome"
  ] }
]
}
```

OK

```
// Data:
"radar"
```

[Home](#)  
[Supported Operations](#)  
[Adding Operations](#)  
[Play with It](#)  
[Fizz Buzz](#)  
[Truthy and Falsy](#)

Parse JsonLogic in JavaScript

Parse JsonLogic in PHP

```
// Rule:
{ "==": [
  { "var": "input", "method": "reverse", "arguments": [{"var": "input"}]
},
"Palindrome",
"Not a Palindrome"
],
"Large"
}
}
```

OK

```
// Data:
"radar"
```

OK

Compute

```
// Output:
```

[Home](#)  
[Supported Operations](#)  
[Adding Operations](#)  
[Play with It](#)  
[Fizz Buzz](#)  
[Truthy and Falsy](#)

Parse JsonLogic in JavaScript

Parse JsonLogic in PHP

Parse JsonLogic in Python

Parse JsonLogic in Ruby

Parse JsonLogic in Go

OK

// Data:

"radar"

OK

Compute

// Output:

"Large"



- b. Based on 2 inputs values (transactionamount, transactioncount), we need you to return score = 0 if (totaltransactionamount < 10K, transactioncount < 5 or if either are NULL), if transactionamount >= 10K & transactioncount >=5 then score = 100.

**Code:**

```
{
  "if": [
    {
      "or": [
        {"<": [{"var": "transactionamount"},
10000]}},
        {"<": [{"var": "transactioncount"}, 5]}},
        {"===": [{"var": "transactionamount"},
null]}},
        {"===": [{"var": "transactioncount"},
null]}},
      ]
    },
    0,
    {
      "if": [
        {
          "and": [
            {">=": [{"var": "transactionamount"},
10000]}},
            {">=": [{"var": "transactioncount"},
5]}},
          ]
        },
        100,
        null
      ]
    }
  ]
}
```

## Output:

# JsonLogic

Build complex rules, serialize them as JSON, share them between front-end and back-end




```
// Rule:
{
  "if": [
    {
      "or": [
        { "<=": [{"var": "transactionamount", "10000"}],
        { "<=": [{"var": "transactioncount", "5"}],
        { "==" : [{"var": "transactionamount", null}],
        { "==" : [{"var": "transactioncount", null}]
      ]
    }
  ]
}
```

OK

```
// Data:
{
  "transactionamount": 10000,
  "transactioncount": 5
}
```


[> Home](#)  
[> Supported Operations](#)  
[> Adding Operations](#)  
[> Play with It](#)  
[> Fizz Buzz](#)  
[> Truthy and Falsy](#)

 Parse JsonLogic in JavaScript

 Parse JsonLogic in PHP

# JsonLogic

Build complex rules, serialize them as JSON, share them between front-end and back-end





```
// Rule:
},
0,
{
  "if": [
    {
      "and": [
        { ">=": [{"var": "transactionamount", "10000"}],
        { ">=": [{"var": "transactioncount", "5"}]
      ]
    }
  ]
}
```

OK

```
// Data:
{
  "transactionamount": 10000,
  "transactioncount": 5
}
```

[> Home](#)  
[> Supported Operations](#)  
[> Adding Operations](#)  
[> Play with It](#)  
[> Fizz Buzz](#)  
[> Truthy and Falsy](#)

 Parse JsonLogic in JavaScript

 Parse JsonLogic in PHP

// Rule:

```
}  
},  
100,  
null  
}  
}  
}
```

OK

// Data:

```
{  
  "transactionamount": 12000,  
  "transactioncount": 6  
}
```

OK

Compute

// Output:

- > Home
- > Supported Operations
- > Adding Operations
- > Play with it
- > Fizz Buzz
- > Truthy and Falsy

 Parse JsonLogic in JavaScript

 Parse JsonLogic in PHP

 Parse JsonLogic in Python

 Parse JsonLogic in Ruby

 Parse JsonLogic in Go

// Data:

```
{  
  "transactionamount": 12000,  
  "transactioncount": 6  
}
```

OK

Compute

// Output:

100

 Parse JsonLogic in PHP

 Parse JsonLogic in Python

 Parse JsonLogic in Ruby

 Parse JsonLogic in Go

 Parse JsonLogic in Java

 Parse JsonLogic in .Net

 Parse JsonLogic in C++

json-logic is maintained by Jeremy