Code Challenge

Evaluador de Expresiones

En Yalo utilizamos muchas maneras para manejar el `input` de un usuario, nos referimos con `input`, al texto que un usuario final escribe en un chat, texto al cual nuestros chatbots deben reaccionar.

Los bots utilizan una máquina de estados para funcionar, esto les permite moverse del estado 1 al estado 2 o 3 , dependiendo del `input`. La parte en la que queremos enfocarnos en este momento es en la `condición` para movernos del estado 1 al 2.

Challenge

El challenge consiste en crear dos scripts capaces de tomar como input una **expresión** lógica o aritmética y por medio de una función poder evaluar la expresión y devolver un resultado false, true, error o next según la **expresión** que evaluó.

Aritmética

El script debe aceptar como input un objeto JSON que contenga los siguientes parámetros:

- ***`expression`***: La expresión que la función evaluará. (Ej. 180/(99**2))
- ***`save`***: El nombre de la variable en donde se almacenará el resultado de la evaluación
- ***`transitions.next`***: El ID del estado al cual se desea mover (estado siguiente) si el resultado de la evaluación de la expresión es exitoso, Ej: 1, 102, 55.
- ***`transitions.error`***: El ID del estado al cual se desea mover (estado siguiente) si la expresión es errónea. Ej: 1, 102, 55.

El script debe dar como output un objeto JSON que contenga los siguientes parámetros:

- ***`[ input.save ]`***: Valor recibido como parámetro en el objeto `input.save`. (Ej. “result, ”).
- ***`transition`***: Valor de la Transición correspondiente dependiendo del resultado de la evaluación (Ej. transition.next, transition.error).

▼ **Ejemplo 1**

```
// input
{
  "expression": "value/(99**2)",
  "save": "result",
  "transitions": {
    "next": 1,
    "error": 2
  },
  "context": {
    "value": 180
  }
}

// output
{
  "result": "0.01836547291",
  "transition": 1
}
```

▼ **Ejemplo 2**

```
// input
{
  "expression": "(str/2)",
  "save": "result",
  "transitions": {
    "next": 101,
    "error": 102
  },
  "context": {
    "str": "string-value"
  }
}

// output
{
  "result": "NaN",
  "transition": 102
}
```

▼ **Ejemplo 3**

```
// input
{
  "expression": "(10/2)",
  "save": "result",
  "transitions": {
```

```
    "next": 25,
    "error": 50
  },
  "context": {}
}

// output
{
  "result": "5",
  "transition": 25
}
```

Debe aceptar como mínimo las siguientes operaciones aritméticas:

- Addition `+`
- Substraction `-`
- Division `/`
- Power `**` or `^`
- Factorial `!`

Lógica

El script debe aceptar como input un objeto JSON que contenga los siguientes parámetros:

- ***`expression`***: La expresión que la función evaluará. (Ej. (foo*3)*32 > bar)
- ***`save`***: El nombre de la variable en donde se almacenará el resultado de la evaluación
- ***`transitions.isTrue`***: El ID del estado al cual se desea mover (estado siguiente) si el resultado de la evaluación de la expresión es verdadero, Ej: 1, 102, 55.
- ***`transitions.isFalse`***: El ID del estado al cual se desea mover (estado siguiente) si el resultado de la evaluación de la expresión es falso, Ej: 1, 102, 55.
- ***`transitions.isError`***: El ID del estado al cual se desea mover (estado siguiente) si la expresión es errónea. Ej: 1, 102, 55.

El script debe dar como output un objeto JSON que contenga los siguientes parámetros:

- ***`[ input.save ]`***: Valor recibido como parámetro en el objeto `input.save`. (Ej. “data, result, etc”)
- ***`transition`***: Valor de la Transición correspondiente dependiendo del resultado de la evaluación (Ej. transition.isTrue, transition.isFalse, transition.isError).

▼ **Ejemplo 1**

```
// input
{
  "expression": "(amount > min)",
  "save": "validAmount",
  "transitions": {
    "isTrue": 5,
    "isFalse": 10,
    "isError": 25,
  },
  "context": {
    "amount": 150,
    "min": 45,
  }
}

// output
{
  "validAmount": true,
  "transition": 5
}
```

▼ **Ejemplo 2**

```
// input
{
  "expression": "(age) >= 18)",
  "save": "adult",
  "transitions": {
    "isTrue": 15,
    "isFalse": 23,
    "isError": 45
  },
  "context": {
    "age": 15,
  }
}

// output
{
  "adult": false,
  "transition": 23
}
```

▼ **Ejemplo 3**

```
// input
{
  "expression": "(age) >= 18)",
  "save": "adult",
  "transitions": {
    "isTrue": 15,
    "isFalse": 23,
    "isError": 45
  },
  "context": {}
}

// output
{
  "adult": "Uncaught ReferenceError: age is not defined",
  "transition": 45
}
```

Debe aceptar como mínimo las siguientes operaciones lógicas:

- Greater Than `>`
- Greater Or Equal Than `>=`
- Less Than `<`
- Less Or Equal `<=`
- Equal `==`
- Not Equal `!=`
- Or `||`
- And `&&`

Condiciones:

- No utilizar la función `eval` de javascript.
- De preferencia utilizar NodeJS *`(no es obligatorio)`*.
- La solución debe contener ambos scripts de manera independiente `(ej. aritmetic.js & logic.js)`
- La solución debe ser entregada en un repositorio público de cualquiera de los siguientes controladores de versiones: `Github, Gitlab, Bitbucket`.
- La solución debe incluir dentro de sus pruebas `una expresión por cada tipo de operación aritmética y lógica`.
- La solución debe incluir dentro de sus pruebas `3 expresiones combinando 2 o más operadores aritméticos`.
- La solución debe incluir dentro de sus pruebas `3 expresiones combinando 2 o más operadores lógicos`.
- Tienes un máximo de `48 horas para realizar la entrega`, a partir del momento en que recibas este documento.