Лабораторная работа «Введение в JavaScript/PHP»

I

*Выполняется в среде c9 (логин kodent, пароль Qwerty.123, проект gossoudarev/**js-study1**, папка tag, в папке со своей фамилией)*

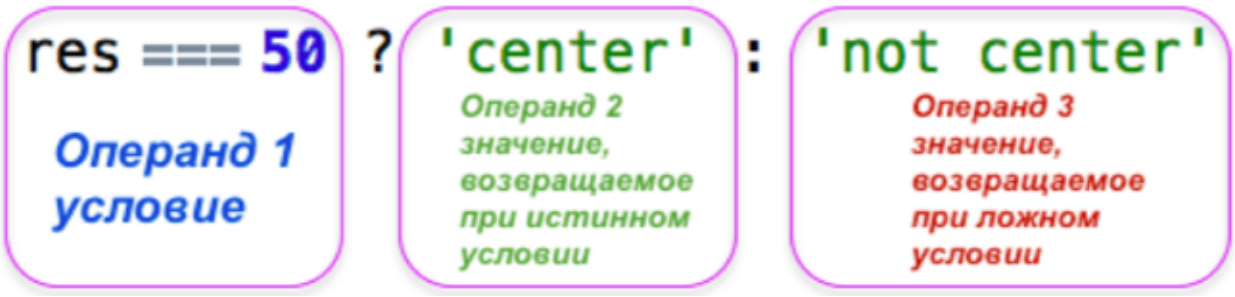
Тернарный оператор и ветвление

A. Пусть у нас есть выражение, генерирующее случайное число от 0 до 100. Используем тернарный оператор для того, чтобы идентифицировать, выпало ли число 50, находящееся в середине этого диапазона:

```
const
  a = 0,
  b = 100,
  res = Math.floor(a + Math.random() * (b - a + 1));

const middle = res === 50 ? 'center' : 'not center';

console.log(middle);
```



Тернарный оператор называется так потому, что имеет три операнда, между которыми располагаются его части (вопросительный знак и двоеточие).

Рассмотрите код в строках 11-13 на странице https://kodaktor.ru/ternary

```
const age = 17;
const restricted = ( age < 18 ) ? 'yes' : 'no' ;
Out.log( restricted );
```

И доработайте его так, чтобы переменная restricted принимала не одно из двух, а одно из трёх различных значений: (а) значение yes при значении переменной age меньше 18 (б)

значение notsure при значении переменной age равном 18 (в) значение no в противном случае

Б. Так как в JavaScript существуют значения, которые нестрого равны друг другу при неявном приведении типов к логическому (они приводятся к false и называются falsy, «ложностные»), а одно из этих значений ещё и не равно самому себе, то нужен способ отличать их друг от друга.

Функция isNaN тоже занимается неявным приведением:

```
> isNaN()
true
> isNaN('p')
true
```

(при этом отметим, что Math.sqrt(-1) не приводится к NaN, а в точности есть NaN, так же как литерал значения NaN, выглядящий в программе как **NaN**).

С использованием *операторов* напишите тернарный оператор, возвращающий:

'=NaN', если тестируемое значение **в точности есть** NaN,

'=null, если если тестируемое значение в точности есть null,

'=undefined, аналогично,

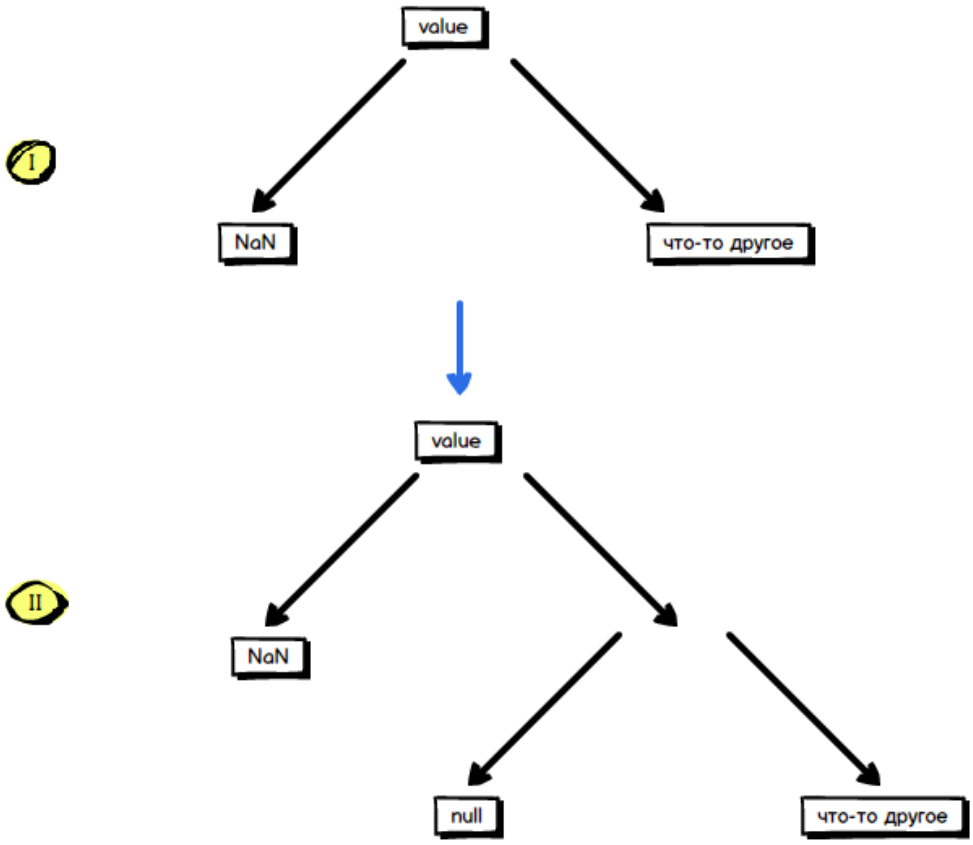
'=0', аналогично.

'=""', в случае пустой строки

и

'=false' в случае значения Boolean False.

Для этого поэтапно спроектируйте дерево вида



II

Полнота

Исполнитель [алгоритмов] называется полным по Тьюрингу, если с его помощью можно реализовать любую вычислимую функцию, т.е. если он совпадает «по диапазону» задач с машиной Тьюринга (и прочими формализмами)
Строго говоря, ни один вычислитель не будет полным без бесконечного объема памяти, однако этим требованием можно пренебречь, если есть средства для обращения к памяти, не ограниченные ее объемом – например, «бесконечное» наращивание доступной памяти по мере требования.
Рассмотрим «шесть волшебных символов»: Шесть волшебных символов: [](}+!
(пара квадратных скобок, пара круглых скобок, плюс и восклицательный знак).

Можно ли «создать» полный по Тьюрингу язык, используя только эти символы?

Пусть нам известно, что:

- ![] есть false
- !false есть true
- +true есть число 1
- x+y есть сумма чисел x и y
- пробел может иметь значение

Запишите число 2 с помощью символов [ ] + !

III

Выясните, как можно получить промежуточный код, в который движок JavaScript компилирует предложенный ему текст программы. Изучите код, получаемый в результате компиляции объявления/присваивания переменной.

IV

Выполните часть I на языке PHP, не рассматривая undefined и NaN.