Введение в Javascript. Условные операторы

Frontend. Урок 19

Вопросы на повторение

1. Что такое оператор в JS?
2. Какие типы операторов есть в JS?
3. Как получить целую часть числа в JS?
4. Какие способы явных преобразований вы помните?

Постановка проблемы

Мы уже умеем выполнять определённые действия с данными. Однако, часто нам нужно выполнять разные действия в зависимости от условий. Каким образом мы можем программировать такие условия?

Условные операторы

Условные операторы позволяют выполнять определённые действия, основываясь на логике и условиях. В JavaScript используются ключевые слова if, else if и else. Пример:

```
let x = 10;
if (x > 5) {
  console.log("x больше 5");
} else {
  console.log("x меньше или равно 5");
}
```

Условные операторы

Также мы можем использовать все математические операторы сравнения: `>` - больше, `<` - меньше, `==` - проверка на равенство, `<=` - меньше либо равно, `>=` - больше или равно, `!=` - неравно

Это по сути своей бинарные логические операторы, которые возвращают значение true или false:

```
console.log(4 < 5)//true
console.log(4 > 5)//false
```

Условные операторы

Есть еще пара бинарных и один унарный операторы, которые позволяют нам собрать более сложное условие:

- `&&` - логическое И, оно объединяет условия (одно неверно = всё неверно)

```
let a = 4 > 5 //false
let b = 6 > 4 //true
let c = 2 == 2 //true
console.log(a && b) //false
console.log(b && c) //true
```

Условные операторы

- `||` - логическое ИЛИ, оно ищет “компромисс”, то есть, если хотя бы одно условие верно, значит всё сложное условие верно:

```
let a = 4 > 5 //false
let b = 6 > 4 //true
let c = 2 == 2 //true
let d = 1 != 1
console.log(a || b) //true
console.log(b || c) //true
console.log(a || d) //false
```

- `!` - унарный оператор НЕ, он просто всё отрицает (НЕ прадва = ложь, НЕ ложь =

```
let a = 4 > 5 //false
let b = 6 > 4 //true
console.log(!a)//true
console.log(!b)//false
```

Тернарный оператор

Также, иногда в коде вы можете встретить замену конструкции if else - еще один уже ТЕРНАРНЫЙ оператор там принимают “участие” 3 операнда:

```
let x = 10;
if (x > 5) {
  console.log("x больше 5");
} else {
  console.log("x меньше или равно 5");
}
// код выше можно заменить на код ниже
x > 5 ? console.log("x больше 5") : console.log("x меньше или равно 5")
```

Switch Case в Javascript

Switch - это конструкция в JavaScript, предназначенная для выбора действия из множества возможных вариантов. Это удобный способ заменить последовательность вложенных условий (if-else if-else). switch сравнивает выражение со списком вариантов и выполняет соответствующий блок кода для первого совпавшего варианта.

```
switch
(выражение) {
  case значение1:
    break;
  case значениеN:
    break;
  default:

}
```

Switch Case

Выражение: Значение, которое будет сравниваться с каждым вариантом.
case значение: Варианты, с которыми сравнивается выражение.
break: Ключевое слово, которое прерывает выполнение switch после выполнения соответствующего блока кода. Без break выполнение будет продолжено к следующему варианту, даже если условие не соответствует.
default: Блок кода, который выполняется, если ни один из вариантов не совпал.

Вопросы на закрепление

1) Что такое условия в JavaScript?
2) Где и для чего мы можем использовать условия?
3) Какие условные конструкции мы разобрали?

Практика А

ИСПОЛЬЗУЙТЕ КОНСТРУКЦИЮ SWITCH CASE

Помните, что когда мы просим ввести что-либо с клавиатуры, нам нужно преобразовать это к необходимому типу.

1. Попросите пользователя ввести два числа и оператор (сложение, вычитание, умножение, деление, возведение в степень, остаток от деления).
2. Используя условные операторы, выполните соответствующее математическое действие.
3. Выведите результат операции на экран.

Практика B

1. Изучите функцию Math.random() самостоятельно
2. Сгенерируйте случайное число от 1 до 100.
3. Предложите пользователю угадать это число, предоставив ему несколько попыток.
4. Используя условные операторы, предоставляйте подсказки (больше/меньше).
5. У пользователя одна попытка.

Практика С

Добавьте еще 2-3 попытки пользователю в практике B.
Выполнить это нужно в песочнице (https://runjs.co), после выполнения задания создать в папке урока файл taskA.js, в него вставить свой код и запушить в свой репозиторий для проверки.