Лабораторная работа №6. Условные инструкции в JavaScript

Содержание:

На этом занятии мы изучим условные инструкции `if`, `if...else`, `switch` и тернарный оператор JavaScript, который очень часто используется в выражениях.

Условные инструкции и тернарный оператор

Условные инструкции – это инструкции языка JavaScript (ECMAScript), которые выполняют определенные действия в зависимости от значения некоторого условия.

Виды условных инструкций в JavaScript:

- `if` (с одной ветвью);
- `if...else` (с двумя ветвями);
- `else if...` (с несколькими ветвями);
- инструкция выбора `switch`.

Кроме условных инструкций в JavaScript имеется ещё тернарный оператор `?:`.

Инструкция if

Синтаксис:

```
// condition – это условие
if (condition) {
  // блок кода, который выполняется один раз, если condition истинно
}
```

Инструкция `if` начинается с ключевого слова `if`, затем после пробела идут круглые скобки, в которых находится условие и далее блок инструкций в фигурных скобках. При этом блок кода выполняется только в том случае, если условие, заданное в круглых скобках `()` истинно.

В качестве условия можно указывать любое выражение. Если выражение приводится к истине (то есть, если `Boolean(condition) === true`), то блок код выполняется. В противном случае нет.

Например:

```
if (true) {
  console.log('Привет, мир!');
}
```

В этом примере условие правдиво, а значит, что блок кода будет выполнен, и мы в консоли увидим сообщение «Привет, мир!». Этот код приведен в качестве примера, так как в таком виде он не имеет никакого смысла.

Если блок кода состоит из одной инструкции, то фигурные скобки можно опустить:

```
if (true) console.log('Привет, мир!');
```

Но фигурные скобки лучше указывать, так как это улучшает читаемость кода.

Написать программу, которая увеличит значение переменной `num` на `5`, если её значение больше `4`

Пример `if`, в условии которого используется оператор «НЕ»:

```
const article = {
  date: '18.05.2022'
}
if (!article.title) {
  console.log('Не указан заголовок!');
}
```

Здесь имеется объект `article`, содержащий одно свойство `date`. С помощью `if` мы выводем сообщение в консоль, если в объекте `article` нет свойства `title` или оно имеется, но его значение ложно.

Условие `!article.title` при приведении к булевому значению даст нам `true`. Так как свойства `title` нет в объекте `article`, то `article.title` вернёт `undefined`. А `!undefined` – это `true`.

Как только что выяснили условие правдиво, а следовательно, блок кода будет выполнен и мы в консоли увидим сообщение.

Инструкция if...else

Инструкция **if...else** состоит из 2 блоков кода. Если условие истинно, то выполняется первый блок кода, в противном случае – второй:

Синтаксис:

```
// condition - это условие
if (condition) {
  // блок кода, который выполняется, если условие истинно
} else {
  // блок кода, который выполняется, если условие ложно
}
```

В `if...else` всегда выполняется один из блоков. То есть в этой инструкции одновременно два блока выполнится не могут. При этом этот блок выполняется один раз. Если условие правдиво, то – первый блок, иначе – второй. После того как один из блоков выполнится интерпретатор JavaScript перейдёт к следующим за ним инструкциям.

Написать программу, которая выводит в консоль сообщение о том, является ли число четным или нет

Несколько условий else if

Если нужно проверить несколько вариантов условий, то можно использовать `else if`:

```
if (condition1) {
  // блок кода, который выполняется, если условие condition1 истинно
} else if (condition2) {
  // блок кода, который выполняется, если условие condition2 истинно
} else if (condition3) {
  // блок кода, который выполняется, если условие condition3 истинно
} else {
```

```
  // блок кода, который выполняется, если предыдущие условия ложны
}
```

Здесь всё как обычно:

- Если `condition1` правдиво, то выполнится первый блок кода. Остальные условия и блоки кода не будут даже просматриваться. На этом выполнение `if` закончится.
- Если же первое условие ложно, то мы переходим к `condition2`. Если `condition2` истинно, то выполнится второй блок кода и на этом выполнение инструкции `if` закончится.
- Если предыдущие условия ложны, то переходим к рассмотрению условия `condition3` и так далее.
- Если все условия ложны, то выполняем последний блок кода, который указан после `else` без дополнительных условий.

Используя `else if` вы можете комбинировать много разных условий и строить длинные цепочки.

Написать программу, которая выводит в консоль разные тексты, в зависимости от значения переменной: меньше нуля, равно нулю, больше нуля,

Кроме этого блок `else` не является обязательным:

```
const lang = 'ru';
if (lang === 'ru') {
  console.log('Это русский текст');
} else if (lang === 'en') {
  console.log('Это английский текст');
} else if (lang !== 'ru' || lang !== 'en') {
  console.log('Это не русский и не английский текст');
}
```

Очень часто `if` применяются внутри функций:

```
function greeting(time) {
  if (time > 18) {
    return 'Добрый вечер!';
  } else if (time > 11) {
    return 'Добрый день!';
  } else if (time > 4) {
    return 'Доброе утро!';
  }
  return 'Доброй ночи!';
}

greeting(10); // "Доброе утро!"
```

Эта функция будет возвращать разные приветствия в зависимости от времени суток. На вход она принимает один аргумент, который мы получаем с помощью параметра `time`. Далее в зависимости от значения параметра возвращаем ту или иную строку.

Условный тернарный оператор ?:

Тернарный оператор – это оператор JavaScript, который возвращает результат первого или второго выражения в зависимости от истинности условия.

Синтаксис:

```
// condition – условие
// expression1 – первое выражение
// expression2 – второе выражение
condition ? expression1 : expression2
```

Тернарный оператор является выражением, то есть он возвращает значение. У него три операнда: условие `condition`, первое выражения `expression1` и второе выражение `expression2`. Для разделения операндов используются знаки `?` и `:`.

Условие – это тоже выражение. Если условие истинно, то тернарный оператор возвращает результат первого выражения `expression1`. В противном случае, то есть, если условие ложно, то результат второго выражения `expression2`.

Тернарный оператор очень часто используется там где нужно получить значение и использовать его дальше. Например, это значение можно присвоить переменной.

```
const value = 10;
const result = value > 10 ? 'Число больше 10!' : 'Число равно или меньше 10!';
console.log(result); // "Число равно или меньше 10!"
```

В этом **примере** мы объявили переменную `value` и присвоили ей число `10`. Так как условие у нас ложно (то есть 10 не больше 10), то тернарный оператор вернёт нам результат вычисления второго выражения, то есть строку «Число равно или меньше 10!». После этого данная строка будет присвоена переменной `result`, которую мы создали с помощью ключевого слова `const`. После этого значение этой переменной будет выведено в консоль.

В JavaScript допустимы множественные тернарные операторы (`?:`):

```
const dayNumber = new Date().getDay();

day =
  (dayNumber === 0) ? 'Воскресенье' :
    (dayNumber === 1) ? 'Понедельник' :
      (dayNumber === 2) ? 'Вторник' :
        (dayNumber === 3) ? 'Среда' :
          (dayNumber === 4) ? 'Четверг' :
            (dayNumber === 5) ? 'Пятница' :
              (dayNumber === 6) ? 'Суббота' : 'Неизвестный день недели';

console.log(`Сегодня ${day.toLowerCase()}.`);
```

Вышеприведённый пример, но с использованием множественной записи инструкции `if...else`:

```
const dayNumber = new Date().getDay();

if (dayNumber === 0) {
  day = 'Воскресенье';
} else if (dayNumber === 1) {
  day = 'Понедельник';
} else if (dayNumber === 2) {
  day = 'Вторник';
} else if (dayNumber === 3) {
  day = 'Среда';
} else if (dayNumber === 4) {
  day = 'Четверг';
} else if (dayNumber === 5) {
  day = 'Пятница';
} else if (dayNumber === 6) {
  day = 'Суббота';
} else {
  day = 'Неизвестный день недели';
}

console.log(`Сегодня ${day.toLowerCase()}.`);
```

Инструкция switch

Инструкция **switch** предназначен для выполнения одного варианта инструкций из нескольких в зависимости от значения выражения. Выбор того или иного варианта определяется посредством строгого равенства результата выражения значению случая (**case**).

Синтаксис инструкции **switch**:

```
// expression – выражение
switch (expression) {
  case valueA:
    // действия, которые будут выполнены, если expression === valueA
    break; // прерываем дальнейшее выполнение switch
  case valueB:
    // действия, которые будут выполнены, если expression === valueB
    break; // прерываем дальнейшее выполнение switch
  // ...
  case valueN:
  // действия, которые будут выполнены, если expression === valueN
    break; // прерываем дальнейшее выполнение switch
  default:
    // действия по умолчанию, если expression не равно valueA, valueB, ..., valueN
}
```

Ключевое слово **default** является необязательным. Оно используется, когда необходимо задать инструкции, которые нужно выполнить, если результат выражения будет не равен ни одному значению варианта (**case**).

Инструкция **break** является необязательной. Она предназначена для прерывания выполнения инструкции **switch** и передачи управлению инструкции, идущей после него.

Например, выведем сообщение в консоль браузера о количестве конфет:

```
const countCandyBoys = 1;
const countCandyGirls = 2;
let message;
switch (countCandyBoys + countCandyGirls) {
  case 1:
    message = 'Одна конфета';
    break;
  case 2:
  case 3:
    message = 'Две или три конфеты';
    break;
  case 4:
    message = 'Четыре конфеты';
    break;
  default:
    message = 'Не одна, не две, не три и не четыре конфеты';
}
// выведем сообщение в консоль
console.log(message);
```

В вышеприведенном примере вычисленное выражение равно 3. Следовательно, будет выполнены инструкции `message = 'Две или три конфеты'` и `break`. Инструкция `break` прервёт дальнейшее выполнение инструкции **switch** и передаст управление инструкции, идущей после него, т.е. `console.log(message)`. Она выведет в консоль сообщение **«Две или три конфеты»**.

Написать программу, которая примет от пользователя число от 1 до 7 и выведет в консоль соответствующий числу день недели: 1 – понедельник, 2 – вторник, ..., 7 – воскресенье.

Пример, в котором не используется инструкция **break**:

```
const result = 'success';

switch (result) {
  case 'success':
    console.log('Успех!');
  case 'invalidCaptcha':
    console.log('Неверная капча!');
  default:
    console.log('Ошибка!');
}
```

В этом примере выражение инструкции **switch** равно **success**. Следовательно, будет выполнена инструкция `console.log('Успех!')`, которая выведет сообщение **«Успех!»**

в консоль. Но так как после неё нет инструкции `break`, то выполнение скрипта будет продолжено в следующем варианте. Таким образом, инструкции будут выполняться до тех пока пока на пути не встретиться `break` или не будет достигнут конец инструкции **switch**. В результате выполнения этого примера в консоль будут выведены 3 сообщения: **«Успех!»**, **«Неверная капча!»** и **«Ошибка!»**.

В некоторых случаях может требоваться именно такое поведение, но не в этом. Здесь просто допущена ошибка.

Исправленный вариант примера:

```
const result = 'success';

switch (result) {
  case 'success':
    console.log('Успех!');
    break;
  case 'invalidCaptcha':
    console.log('Неверная капча!');
    break;
  default:
    console.log('Ошибка!');
}
```