

Яндекс

Яндекс

# JavaScript: Базовые КОНСТРУКЦИИ ЯЗЫКА

# Условные операторы

Вставьте  
изображение

Сейчас мы умеем выполнять строки кода подряд, без исключения.

В алгоритме может возникнуть ситуация, когда мы, например, задаем вопрос пользователю, и, в зависимости от его ответа, что-то делаем или не делаем.

Для таких ситуаций существует условный оператор - **if**

# Угадай число

```
var guessed = Math.round(Math.random() * 100);  
var userNumber = +prompt('угадай число');  
  
if (userNumber === guessed) {  
    alert('Угадал!');  
} else {  
    alert('Не угадал!');  
}
```

# Последовательность исполнения

Подходим к оператору `if`, по мере выполнения скрипта

Видя `if`, смотрим в условие в круглых скобках, и вычисляем его

Вычислив, приводим к `boolean`, если это необходимо

Если в результате `true`, то выполняем код внутри первых фигурных скобочек `{}`. Код в скобках после `else` игнорируется.

Если `false` - то выполняем код внутри фигурных скобок после `else`, если `else` присутствует. Первые фигурные скобки игнорируются

# Без else

```
var guessed = Math.round(Math.random() * 100);  
var userNumber = +prompt('угадай число');  
  
if (userNumber === guessed) {  
    alert('Угадал!');  
}
```

# Без фигурных скобок

```
var guessed = Math.round(Math.random() * 100);  
var userNumber = +prompt('угадай число');  
  
if (userNumber === guessed) alert('Угадал!');
```



Поначалу, хорошая идея  
всегда ставить фигурные  
скобки

# Угадай число

```
var guessed = Math.round(Math.random() * 100);
```

```
var userNumber = +prompt('угадай число');
```

```
if (userNumber === guessed) {
```

```
    alert('Угадал!');
```

```
} else {
```

```
    alert('Не угадал!');
```

```
}
```



Покажем тут больше или  
меньше загаданное  
число

# Угадай число

```
var guessed = Math.round(Math.random() * 100);  
var userNumber = +prompt('угадай число');  
  
if (userNumber === guessed) {  
    alert('Угадал!');  
} else {  
    if (guessed > userNumber) {  
        alert('Не угадал, загаданное число больше!');  
    } else {  
        alert('Не угадал, загаданное число меньше!');  
    }  
}
```

# Угадай число

```
var guessed = Math.round(Math.random() * 100);  
var userNumber = +prompt('угадай число');  
  
if (userNumber === guessed) {  
    alert('Угадал!');  
} else if (guessed > userNumber) {  
    alert('Не угадал, загаданное число больше!');  
} else {  
    alert('Не угадал, загаданное число меньше!');  
}
```

# Угадай число

```
var guessed = Math.round(Math.random() * 100);  
var userNumber = +prompt('угадай число');  
  
if (userNumber === guessed) {  
    alert('Угадал!');  
}  
  
if (guessed > userNumber) {  
    alert('Не угадал, загаданное число больше!');  
}  
  
if (guessed < userNumber) {  
    alert('Не угадал, загаданное число меньше!');  
}
```

# Приведение типов

Вставьте  
изображение

Если вычислено не `boolean`,  
как понять, какому  
булевскому значению оно  
соответствует?

# Truly и Falsy

Тип данных	Truly	Falsy
Number	Все, кроме 0, включая отрицательные	0
String	Любые непустые строки	“
Array, Object	Любой	-
Null, Undefined	-	+



| Falsy - false, 0, "", null, undefined

# Циклы

Вставьте  
изображение

Бывает, что один и тот же блок кода хочется выполнить подряд несколько раз

Для этого существуют циклы, которых в любом уважающем себя языке будет несколько.

Разные циклы решают разные задачи, и имеют соответствующий синтаксис

Также циклы делятся на циклы с предусловием и постусловием

**for** - цикл для того, чтобы  
сделать заданное,  
перечислимое количество  
итераций

# Угадай число

```
var guessed = Math.round(Math.random() * 100);  
  
var userNumber;  
  
for (var i = 0; i < 3; i++) {  
    userNumber = +prompt('угадай число');  
    if (userNumber === guessed) {  
        alert('Угадал!');  
    } else {  
        if (guessed > userNumber) {  
            alert('Не угадал, загаданное число больше!');  
        } else {  
            alert('Не угадал, загаданное число меньше!');  
        }  
    }  
}  
  
alert('осталось ' + (2 - i) + ' попыток');  
}
```

# Последовательность исполнения

Подходим к оператору **for**, по мере выполнения скрипта

Видя **for**, перед первой операцией выполняем первую команду внутри круглых скобок, до ;

Затем, проверяем условие (вторая команда внутри круглых скобок, между ;)

Если условие `truly` - идем выполнять первую итерацию (код внутри `{}`).

Выполнив весь код, мы возвращаемся в заголовок цикла **for**. На сей раз будут выполнены третья команда, а затем проверено условие во втором блоке

# Сколько будет итераций и чему будет равен $i$ после цикла?

```
var guessed = Math.round(Math.random() * 100);

var userNumber;

for (var i = 0; i < 3; i++) {
  userNumber = +prompt('угадай число');
  if (userNumber === guessed) {
    alert('Угадал!');
  } else {
    if (guessed > userNumber) {
      alert('Не угадал, загаданное число больше!');
    } else {
      alert('Не угадал, загаданное число меньше!');
    }
  }
}

alert('осталось ' + (2 - i) + ' попыток');
```

Может не быть выполнено ни одной итерации, если перед первой в условии вычислено `falsy`



Цикл не «следит» за  
условием, а вычисляет его  
строго при возвращении к  
заголовку цикла

**| А сколько надо дать попыток,  
чтобы пользователь  
гарантированно мог отгадать  
число, следуя оптимальной  
стратегии?**

**while** - цикл для того, чтобы  
делать что-то «до тех пор,  
пока». Количество итераций  
заранее неясно.

# Угадай число

```
var guessed = Math.round(Math.random() * 100);  
  
var userNumber;  
  
while (userNumber !== guessed) {  
    userNumber = +prompt('угадай число');  
    if (userNumber === guessed) {  
        alert('Угадал!');  
    } else {  
        if (guessed > userNumber) {  
            alert('Не угадал, загаданное число больше!');  
        } else {  
            alert('Не угадал, загаданное число меньше!');  
        }  
    }  
}
```

**do ... while** - как **while**, только одна итерация обязательно выполняется. Почти не используется

**break** - экстренный выход из цикла

Автор

# Угадай число

```
var guessed = Math.round(Math.random() * 100);  
  
var userNumber;  
  
while (userNumber !== guessed) {  
    userNumber = +prompt('угадай число');  
    if (userNumber === guessed) {  
        alert('Угадал!');  
        break;  
    } else {  
        if (guessed > userNumber) {  
            alert('Не угадал, загаданное число больше!');  
        } else {  
            alert('Не угадал, загаданное число меньше!');  
        }  
    }  
}  
}
```



# «бесконечный» ЦИКЛ

Автор



# Угадай число

```
var guessed = Math.round(Math.random() * 100);  
var userNumber;  
  
while (true) {  
    userNumber = +prompt('угадай число');  
    if (userNumber === guessed) {  
        alert('Угадал!');  
        break;  
    } else {  
        if (guessed > userNumber) {  
            alert('Не угадал, загаданное число больше!');  
        } else {  
            alert('Не угадал, загаданное число меньше!');  
        }  
    }  
}
```



**цикл с присваиванием в  
условии**

# Угадай число

```
var guessed = 1 + Math.round(Math.random() * 99);  
var userNumber;  
  
while (userNumber = +prompt('угадай число')) {  
    if (userNumber === guessed) {  
        alert('Угадал!');  
        break;  
    } else {  
        if (guessed > userNumber) {  
            alert('Не угадал, загаданное число больше!');  
        } else {  
            alert('Не угадал, загаданное число меньше!');  
        }  
    }  
}
```

**continue** - переход к  
следующей итерации

# Функции

Вставьте  
изображение

# Функции

| Блок кода, который имеет имя,

# Функции

Блок кода, который имеет имя,

По этому имени функцию можно позвать в любой момент

# Обычное объявление функции

```
function hello() {  
    alert( 'привет' );  
}
```

```
hello();
```



# Присваивание безымянной функции в переменную

```
var hello = function() {  
    alert( 'привет' );  
}  
  
hello();
```

# Функции

Блок кода, который имеет имя,

По этому имени функцию можно позвать в любой момент

Функция возвращает результат

# Возвращение результата

```
var hello = function () {  
    return 'привет';  
}
```

```
alert(hello());
```

# Функции

Блок кода, который имеет имя,

По этому имени функцию можно позвать в любой момент

Функция возвращает результат

Функция принимает аргументы

# Использование аргументов

```
var hello = function(name) {  
    return 'привет, ' + name;  
}
```


```
alert(hello('dusty'));
```

**| аргументы по умолчанию**

# Использование аргументов

```
var hello = function(name = 'Гость') {  
    return 'привет, ' + name;  
}
```

```
alert(hello());
```



# Замыкания и доступ к внешним переменным