

Раздел

[Основы JavaScript](#)

Навигация по уроку

Структура кода

Строгий режим

Переменные

Взаимодействие с посетителем

Операторы

Циклы

Конструкция «switch»

Функции

Далее мы изучим больше

Комментарии

Поделиться

[Редактировать на GitHub](#)[🏠 → Язык программирования JavaScript](#)
[→ Основы JavaScript](#) 1-го октября 2020

Особенности JavaScript

Давайте кратко повторим изученный материал и отметим наиболее «тонкие» моменты.

Структура кода

Инструкции разделяются точкой с запятой:

```
1 alert('Привет'); alert('Мир');
```



Как правило, перевод строки также интерпретируется как разделитель, так тоже будет работать:

```
1 alert('Привет')
2 alert('Мир')
```



Это так называемая «автоматическая вставка точки с запятой». Впрочем, она не всегда срабатывает, например:

```
1 alert("После этого сообщения ждите ошибку")
2
3 [1, 2].forEach(alert)
```



Большинство руководств по стилю кода рекомендуют ставить точку с запятой после каждой инструкции.

Точка с запятой не требуется после блоков кода {...} и синтаксических конструкций с ними, таких как, например, циклы:

```
1 function f() {
2     // после объявления функции необязательно ставить точ
3 }
4
5 for(;;) {
6     // после цикла точка с запятой также необязательна
7 }
```

...Впрочем, если даже мы и поставим «лишнюю» точку с запятой, ошибки не будет. Она просто будет проигнорирована.

Подробнее: [Структура кода](#).

Строгий режим

Чтобы по максимуму использовать возможности современного JavaScript, все скрипты рекомендуется начинать с добавления директивы "use strict".

```
1 'use strict';
2
3 ...
```

Эту директиву следует размещать в первой строке скрипта или в начале тела функции.

Раздел

Основы JavaScript

Навигация по уроку

Структура кода

Строгий режим

Переменные

Взаимодействие с посетителем

Операторы

Циклы

Конструкция «switch»

Функции

Далее мы изучим больше

Комментарии

Поделиться



Редактировать на GitHub



Без "use strict" код также запустится, но некоторые возможности будут работать в «режиме совместимости» со старыми версиями языка JavaScript. Нам же предпочтительнее современное поведение.

Некоторые конструкции языка (например, классы, которые нам ещё предстоит изучить) включают строгий режим по умолчанию.

Подробнее: [Строгий режим — "use strict"](#).

Переменные

Можно объявить при помощи:

- `let`
- `const` (константа, т.е. изменению не подлежит)
- `var` (устаревший способ, подробности позже)

Имя переменной может включать:

- Буквы и цифры, однако цифра не может быть первым символом.
- Символы `$` и `_` используются наряду с буквами.
- Иероглифы и символы нелатинского алфавита также допустимы, но обычно не используются.

Переменные типизируются динамически. В них могут храниться любые значения:

```
1 let x = 5;  
2 x = "Вася";
```

Всего существует 8 типов данных:

- `number` для целых и вещественных чисел,
- `bigint` для работы с целыми числами произвольной длины,
- `string` для строк,
- `boolean` для логических значений истинности или ложности: `true/false`,
- `null` – тип с единственным значением `null`, т.е. «пустое значение» или «значение не существует»,
- `undefined` – тип с единственным значением `undefined`, т.е. «значение не задано»,
- `object` и `symbol` – сложные структуры данных и уникальные идентификаторы; их мы ещё не изучили.

Оператор `typeof` возвращает тип значения переменной, с двумя исключениями:

```
1 typeof null == "object" // ошибка в языке  
2 typeof function(){} == "function" // именно для функций
```

Подробнее: [Переменные](#), [Типы данных](#).

Взаимодействие с посетителем

В качестве рабочей среды мы используем браузер, так что простейшими функциями взаимодействия с посетителем являются:

`prompt(question, [default])`

Задаёт вопрос `question` и возвращает то, что ввёл посетитель, либо `null`, если посетитель нажал на кнопку «Отмена».

`confirm(question)`

Задаёт вопрос `question` и предлагает выбрать «ОК» или «Отмена». Выбор возвращается в формате `true/false`.

`alert(message)`

Выводит сообщение `message`.

Раздел

Основы JavaScript

Навигация по уроку

Структура кода

Строгий режим

Переменные

Взаимодействие с посетителем

Операторы

Циклы

Конструкция «switch»

Функции

Далее мы изучим больше

Комментарии

Поделиться



Редактировать на GitHub



Все эти функции показывают *модальные окна*, они останавливают выполнение кода и не позволяют посетителю взаимодействовать со страницей, пока не будет дан ответ на вопрос.

Например:

```
1 let userName = prompt("Введите имя", "Алиса");
2 let isTeaWanted = confirm("Вы хотите чаю?");
3
4 alert( "Посетитель: " + userName ); // Алиса
5 alert( "Чай: " + isTeaWanted ); // true
```



Подробности: [Взаимодействие: alert, prompt, confirm.](#)

Операторы

JavaScript поддерживает следующие операторы:

Арифметические

Простые `*` `+` `-` `/`, а также деление по модулю `%` и возведение в степень `**`.

Бинарный плюс `+` объединяет строки. А если одним из операндов является строка, то второй тоже будет конвертирован в строку:

```
1 alert( '1' + 2 ); // '12', строка
2 alert( 1 + '2' ); // '12', строка
```



Операторы присваивания

Простые `a = b` и составные `a *= 2`.

Битовые операции

Битовые операторы работают с 32-битными целыми числами на самом низком, побитовом уровне. Подробнее об их использовании можно прочитать на ресурсе [MDN](#) и в разделе [Побитовые операторы](#).

Условный оператор

Единственный оператор с тремя параметрами: `cond ? resultA : resultB`. Если условие `cond` истинно, возвращается `resultA`, иначе – `resultB`.

Логические операторы

Логические `И` `&&`, `ИЛИ` `||` используют так называемое «ленивое вычисление» и возвращают значение, на котором оно остановилось (не обязательно `true` или `false`). Логическое `НЕ` `!` конвертирует операнд в логический тип и возвращает инвертированное значение.

Сравнение

Проверка на равенство `==` значений разных типов конвертирует их в число (за исключением `null` и `undefined`, которые могут равняться только друг другу), так что примеры ниже равны:

```
1 alert( 0 == false ); // true
2 alert( 0 == '' ); // true
```



Другие операторы сравнения тоже конвертируют значения разных типов в числовой тип.

Оператор строгого равенства `===` не выполняет конвертирования: разные типы для него всегда означают разные значения.

Значения `null` и `undefined` особенные: они равны `==` только друг другу, но не равны ничему ещё.

Операторы сравнения больше/меньше сравнивают строки посимвольно, остальные типы конвертируются в число.

Раздел

[Основы JavaScript](#)

Навигация по уроку

Структура кода

Строгий режим

Переменные

Взаимодействие с посетителем

Операторы

Циклы

Конструкция «switch»

Функции

Далее мы изучим больше

Комментарии

Поделиться



[Редактировать на GitHub](#)

Другие операторы

Существуют и другие операторы, такие как запятая.

Подробности: [Базовые операторы](#), [математика](#), [Операторы сравнения](#), [Логические операторы](#).

Циклы

- Мы изучили три вида циклов:

```
1 // 1
2 while (condition) {
3     ...
4 }
5
6 // 2
7 do {
8     ...
9 } while (condition);
10
11 // 3
12 for(let i = 0; i < 10; i++) {
13     ...
14 }
```

- Переменная, объявленная в цикле `for(let...)`, видна только внутри цикла. Но мы также можем опустить `let` и переиспользовать существующую переменную.
- Директивы `break/continue` позволяют выйти из цикла/текущей итерации. Используйте метки для выхода из вложенных циклов.

Подробности: [Циклы while и for](#).

Позже мы изучим ещё виды циклов для работы с объектами.

Конструкция «switch»

Конструкция «switch» может заменить несколько проверок `if`. При сравнении она использует оператор строгого равенства `===`.

Например:

```
1 let age = prompt('Сколько вам лет?', 18);
2
3 switch (age) {
4     case 18:
5         alert("Так не работает"); // результатом prompt яв.
6
7     case "18":
8         alert("А так работает!");
9         break;
10
11     default:
12         alert("Любое значение, неравное значению выше");
13 }
```

Подробности: [Конструкция "switch"](#).

Функции

Мы рассмотрели три способа создания функции в JavaScript:

- Function Declaration: функция в основном потоке кода

```
1 function sum(a, b) {
2     let result = a + b;
3 }
```

```
4   return result;
5 }
```

Раздел

Основы JavaScript

Навигация по уроку

Структура кода

Строгий режим

Переменные

Взаимодействие с посетителем

Операторы

Циклы

Конструкция «switch»

Функции

Далее мы изучим больше

Комментарии

Поделиться



Редактировать на GitHub



2. Function Expression: функция как часть выражения



```
1 let sum = function(a, b) {
2   let result = a + b;
3
4   return result;
5 };
```

3. Стрелочные функции:

```
1 // выражение в правой части
2 let sum = (a, b) => a + b;
3
4 // многострочный код в фигурных скобках { ... }, здесь
5 let sum = (a, b) => {
6   // ...
7   return a + b;
8 }
9
10 // без аргументов
11 let sayHi = () => alert("Привет");
12
13 // с одним аргументом
14 let double = n => n * 2;
```

- У функций могут быть локальные переменные: т.е. объявленные в теле функции. Такие переменные видимы только внутри функции.
- У параметров могут быть значения по умолчанию: `function sum(a = 1, b = 2) { ... }`.
- Функции всегда что-нибудь возвращают. Если нет оператора `return`, результатом будет `undefined`.

Подробнее: [Функции](#), [Функции-стрелки](#), [основы](#).

Далее мы изучим больше

Это был краткий список возможностей JavaScript. На данный момент мы изучили только основы. Далее в учебнике вы найдёте больше особенностей и продвинутых возможностей JavaScript.

Проводим [курсы по JavaScript и фреймворкам](#). ✕



Комментарии

перед тем как писать...