


 → [Язык JavaScript](#) → [Основы JavaScript](#)

 30 марта 2022 г.

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 точку с запятой
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.

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
```

Подробности: [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`.

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.

Оператор нулевого слияния

Оператор `??` предоставляет способ выбора определённого значения из списка переменных. Результатом `a ?? b` будет `a`, если только оно не равно `null/undefined`, тогда `b`.

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.

Другие операторы

Существуют и другие операторы, такие как запятая.

Подробности: [Базовые операторы](#), [математика](#), [Операторы сравнения](#), [Логические операторы](#), [Оператор нулевого слияния \(??\)](#).

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.

Подробнее: [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 является строка, а не чис
6
7     case "18":
8         alert("А так работает!");
9         break;
10
11    default:
12        alert("Любое значение, неравное значению выше");
13 }

```

Подробнее: [Конструкция "switch"](#).

Функции

Мы рассмотрели три способа создания функции в JavaScript:

1. Function Declaration: функция в основном потоке кода

```

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

```

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 нужен return:  
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`.

Подробности: [Функции](#), [Стрелочные функции](#), [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.



Предыдущий урок

Следующий урок



Поделиться



Карта учебника

Комментарии

- Если вам кажется, что в статье что-то не так - вместо комментария напишите [на GitHub](#).
- Для одной строки кода используйте тег `<code>` , для нескольких строк кода — тег `<pre>` , если больше 10 строк — ссылку на песочницу ([plnkr](#), [JSBin](#), [codepen...](#))
- Если что-то непонятно в статье — пишите, что именно и с какого места.

Присоединиться к обсуждению...

ВОЙТИ С ПОМОЩЬЮ

ИЛИ ЧЕРЕЗ DISQUS ?

Имя

114

Share

Лучшие Новые Старые



Valeriy Albertovich

🕒 4 дня назад

Добро пожаловать @degen_dev0x7e7

0 0 Ответить



Denis Butyrskiy

🕒 17 дней назад

Ребят, всем привет! Изучаю JavaScript.

Главная цель ⚡ - «С нуля до Junior Frontend Developer за 7 месяцев». На протяжении этих 7 месяцев буду изучать JS и записывать еженедельные **видеоотчёты** об этом.

Всё здесь:

YouTube по названию канала **Denis Butyrskiy**

ТГ @itway_chat - обсуждаем всё, что касается обучения

Давайте объединяться, общаться, делиться друг с другом успехами и неудачами. Это сильно помогает пройти сложные моменты, уже проверено. Под каждым видео есть вся нужная информация 📌

0 0 Ответить



Джун на фронте

🕒 4 месяца назад edited

Вот уже как **300 дней** я веду блог прогресса...💪

За основу были взяты три кита разработки HTML, CSS, JS.

Приглашаю пройти этот путь вместе:

тг @divatoz или Джун на фронте!

0 8 Ответить



Джун на фронте

🕒 4 месяца назад

Как же так то ?

0 3 Ответить



Zipri

🕒 4 месяца назад

So Well

0 0 Ответить



Alexander Mandrov

🕒 4 месяца назад



Обстоятельства и желания сподвигли меня на создание собственного канала. Там я рассказываю о жизни и работе, о том, как стал программистом в 20 лет, кратко и по делу.

ТГ @unsleeping706

0 3 Ответить

A

Alex

🕒 4 месяца назад



Большое спасибо авторам за учебный материал!

5 0 Ответить

ДЯ

Дарья Янукович

🕒 5 месяцев назад



Привет!

В нашем сообществе множество разных и интересных статей, которые по настоящему помогают нашим подписчикам.

Совершенно недавно мы проводили интервью с фронтенд-разработчиков - ссылка в тлг

Также мы проводим осенью набор на наш новый курс)

У нас есть сообщество, в котором много разных задач и макетов для верстки)

Заходи к нам dh_jun

0 2 Ответить

ДЯ

Дарья Янукович

🕒 5 месяцев назад



Хай! Заходи к нам на тусовку разрабов в тлг - dh_jun

0 0 Ответить



Иван

🕒 5 месяцев назад



круто было б если в конце был какой тест на проверку усвоения материала.

Подскажите, где найти хорошие тесты или несложные практикум для закрепления данных тем? Спасибо!

7 0 Ответить

K

Konstantin

🕒 4 месяца назад

➔ Иван



codewars

4 0 Ответить



Фуга

→ Иван

🕒 5 месяцев назад



Поддерживаю.

1

0

Ответить



Vasya Kashapov

🕒 6 месяцев назад



Привет! Мы создали сообщество в котором обсуждаем решение задач, так же у нас есть ментор с несколькими годами опыта комерческой разработки которому ты можешь задать вопрос который тебя давно мучал. Если хочешь развиваться быстро - заходи @artistjs

0

0

Ответить



Никита Самарин

🕒 6 месяцев назад



13.08.22/Ok

0

0

Ответить



smcmdcs

🕒 6 месяцев назад



26.07.2022 3:35 есть

0

1

Ответить



Дима Дим

→ smcmdcs

🕒 6 месяцев назад



клоун?

3

1

Ответить



Nastya:3

→ smcmdcs

🕒 6 месяцев назад



И чо

0

0

Ответить



Viktor

🕒 7 месяцев назад



21.07.2022 20:26 Пройдено !

0

12

Ответить



Дима Дим

→ Viktor

🕒 6 месяцев назад



в спам

2

0

Ответить

Загрузить ещё комментарии

[Подписаться](#)

[Privacy](#)

[Не продавайте мои данные](#)