

# **Программирование на стороне клиента**

Основы синтаксиса JavaScript

# Что такое JavaScript?

Изначально JavaScript был создан, чтобы сделать html-страницы «живыми».

Программы на этом языке называются скриптами. Они могут встраиваться в HTML и выполняться автоматически при загрузке веб-страницы.

Скрипты распространяются и выполняются, как простой текст. Им не нужна специальная подготовка или компиляция для запуска.

Сегодня JavaScript может выполняться не только в браузере, но и на сервере или на любом другом устройстве, которое имеет специальную программу, называемую «движком» JavaScript.

У браузера есть собственный движок, который иногда называют «виртуальная машина JavaScript».

## Что может JavaScript в браузере?

Современный JavaScript – это «безопасный» язык программирования. Он не предоставляет низкоуровневый доступ к памяти или процессору, потому что изначально был создан для браузеров, не требующих этого.

Возможности JavaScript сильно зависят от окружения, в котором он работает. Например, Node.js поддерживает функции чтения/записи произвольных файлов, выполнения сетевых запросов и т.д.

В браузере для JavaScript доступно всё, что связано с манипулированием веб-страницами, взаимодействием с пользователем и веб-сервером.

Например, в браузере JavaScript может:

- ✓ Добавлять новый HTML-код на страницу, изменять существующее содержимое, модифицировать стили.
- ✓ Реагировать на действия пользователя, щелчки мыши, перемещения указателя, нажатия клавиш.
- ✓ Отправлять сетевые запросы на удалённые сервера, скачивать и загружать файлы (технологии AJAX и COMET).
- ✓ Получать и устанавливать куки, задавать вопросы посетителю, показывать сообщения.
- ✓ Запоминать данные на стороне клиента («local storage»).

## **Чего НЕ может JavaScript в браузере?**

Возможности JavaScript в браузере ограничены ради безопасности пользователя. Цель заключается в предотвращении доступа недобросовестной веб-страницы к личной информации или нанесения ущерба данным пользователя.

- JavaScript на веб-странице не может читать/записывать произвольные файлы на жёстком диске, копировать их или запускать программы. Он не имеет прямого доступа к системным функциям ОС.
- Различные окна/вкладки не знают друг о друге. Иногда одно окно, используя JavaScript, открывает другое окно. Но даже в этом случае JavaScript с одной страницы не имеет доступа к другой, если они пришли с разных сайтов (с другого домена, протокола или порта).
- JavaScript может легко взаимодействовать с сервером, с которого пришла текущая страница. Но его способность получать данные с других сайтов/доменов ограничена. Хотя это возможно в принципе, для чего требуется явное согласие (выраженное в заголовках HTTP) с удалённой стороной. Опять же, это ограничение безопасности.

# **ОСНОВЫ JavaScript**



## HTML-тег <script>

Программы на JavaScript могут быть вставлены в любое место HTML-документа с помощью тега <script>

```
<!DOCTYPE HTML>
```

```
<html>
```

```
  <body>
```

```
    <p>Перед скриптом...</p>
```

```
    <script>
```

```
      alert( 'Привет, мир!' );
```

```
    </script>
```

```
    <p>...После скрипта.</p>
```

```
  </body>
```

```
</html>
```

# Внешние скрипты

Если у вас много JavaScript-кода, вы можете поместить его в отдельный файл. Файл скрипта можно подключить к HTML с помощью атрибута **src**:

```
<!DOCTYPE HTML>
<html>
  <head>
    <script src="script1.js"></script>
  </head>
  <body>
    <h1>Привет!</h1>
  </body>
  <script src="script2.js"></script>
</html>
```

# Структура кода

***Инструкции*** – это синтаксические конструкции и команды, которые выполняют действия.

Мы уже видели инструкцию `alert('Привет, мир!')`, которая отображает сообщение «Привет, мир!».

Инструкции могут отделяться точкой с запятой.

Обычно каждую инструкцию пишут на новой строке, чтобы код было легче читать:

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

## Комментарии

Со временем программы становятся всё сложнее и сложнее. Возникает необходимость добавлять комментарии, которые бы описывали, что делает код и почему.

Комментарии могут находиться в любом месте скрипта. Они не влияют на его выполнение, поскольку движок просто игнорирует их.

### Однострочные комментарии

Часть строки после `//` считается комментарием. Такой комментарий может как занимать строку целиком, так и находиться после инструкции.

### Многострочные комментарии

Такие комментарии начинаются косой чертой со звёздочкой `/*` и заканчиваются звёздочкой с косой чертой `*/`

```
// Этот комментарий занимает всю строку  
alert('Привет');
```

```
alert('Мир'); // Этот комментарий следует за инструкцией
```

```
/*  
Это - многострочный комментарий.  
*/  
alert('Привет');  
alert('Мир');
```

# Переменные

Переменная – это «именованное хранилище» для данных..

Для создания переменной в JavaScript используйте ключевое слово **let**.

```
let message;
```

Данная инструкция создаёт (другими словами: объявляет или определяет) переменную с именем «message».

Теперь можно поместить в неё данные, используя оператор присваивания =

```
let message;
```

```
message = 'Привет'; // сохранить строку
```

Строка сохраняется в области памяти, связанной с переменной. Мы можем получить к ней доступ, используя имя переменной:

```
let message;
```

```
message = 'Привет';
```

```
alert(message); // показывает содержимое переменной
```

Для краткости можно совместить объявление переменной и запись данных в одну строку:

```
let message = 'Привет'; // определяем переменную и присваиваем ей значение
```

```
alert(message); // Привет
```



## **var вместо let**

В старых скриптах вы также можете найти другое ключевое слово: `var` вместо `let`:

```
var message = 'Hello';
```

Ключевое слово `var` – почти то же самое, что и `let`. Оно объявляет переменную, но немного по-другому, «устаревшим» способом.

# Имена переменных

В JavaScript есть два ограничения, касающиеся имён переменных:

- ✓ Имя переменной должно содержать только буквы, цифры или символы \$ и \_.
- ✓ Первый символ не должен быть цифрой.

Если имя содержит несколько слов, обычно используется верблюжья нотация, то есть, слова следуют одно за другим, где каждое следующее слово начинается с заглавной буквы: `myVeryLongName`.

## ***Регистр имеет значение***

Переменные с именами apple и AppLE – это две разные переменные.

## ***Нелатинские буквы разрешены, но не рекомендуются***

Можно использовать любой язык, включая кириллицу или даже иероглифы.

Технически здесь нет ошибки, такие имена разрешены, но есть международная традиция использовать английский язык в именах переменных.

# Константы

Чтобы объявить константную, то есть, неизменяемую переменную, используйте **const** вместо **let**:

```
const myBirthday = '19.05.1985';
```

Переменные, объявленные с помощью **const**, называются «константами». Их нельзя изменить. Попытка сделать это приведёт к ошибке:

```
myBirthday = '02.10.2021';
```

# Типы данных

Значение в JavaScript всегда относится к данным определённого типа. Например, это может быть строка или число.

Переменная в JavaScript может содержать любые данные. В один момент там может быть строка, а в другой – число:

```
// Не будет ошибкой  
let message = "hello";  
message = 123456;
```

Языки программирования, в которых такое возможно, называются «динамически типизированными». Это значит, что типы данных есть, но переменные не привязаны ни к

# Число

```
let n = 123;  
n = 12.345;
```

Числовой тип данных (number) представляет как целочисленные значения, так и числа с плавающей точкой.

Кроме обычных чисел, существуют так называемые «специальные числовые значения», которые относятся к этому типу данных: Infinity, -Infinity и NaN.

Infinity представляет собой математическую бесконечность?. Это особое значение, которое больше любого числа.

NaN означает вычислительную ошибку. Это результат неправильной или неопределённой математической операции.



## BigInt

В JavaScript тип «number» не может содержать числа больше, чем  $(2^{53}-1)$  (т. е. 9007199254740991), или меньше, чем  $-(2^{53}-1)$  для отрицательных чисел. Это техническое ограничение вызвано их внутренним представлением.

Для большинства случаев этого достаточно. Но иногда нам нужны действительно гигантские числа, например, в криптографии.

Чтобы создать значение типа BigInt, необходимо добавить n в конец числового литерала:

```
// символ "n" в конце означает, что это BigInt  
const bigInt = 1234567890123456789012345678901234567890n;
```

# Строка

Строка (string) в JavaScript должна быть заключена в кавычки.

В JavaScript существует три типа кавычек.

Двойные кавычки: "Привет".

Одинарные кавычки: 'Привет'.

Обратные кавычки: `Привет`.

Двойные или одинарные кавычки являются «простыми», между ними нет разницы в JavaScript.

Обратные же кавычки имеют расширенную функциональность. Они позволяют нам встраивать выражения в строку, заключая их в `${}.`

# Булевый (логический) тип

Булевый тип (boolean) может принимать только два значения: true (истина) и false (ложь).

Такой тип, как правило, используется для хранения значений да/нет: true значит «да, правильно», а false значит «нет, не правильно».

Например:

```
let nameFieldChecked = true; // да, поле отмечено  
let ageFieldChecked = false; // нет, поле не отмечено
```

## Значение «null»

Специальное значение `null` не относится ни к одному из типов, описанных выше.

Оно формирует отдельный тип, который содержит только значение `null`:

```
let age = null; // значение переменной age неизвестно
```

В JavaScript `null` не является «ссылкой на несуществующий объект» или «нулевым указателем», как в некоторых других языках.

Это просто специальное значение, которое представляет собой «ничего», «пусто» или «значение неизвестно».

## Значение «undefined»

Специальное значение `undefined` также стоит особняком. Оно формирует тип из самого себя так же, как и `null`.

Оно означает, что «значение не было присвоено».

Если переменная объявлена, но ей не присвоено никакого значения, то её значением будет `undefined`:

```
let age;
```

```
alert(age); // выведет "undefined"
```

Технически мы можем присвоить значение `undefined` любой переменной:

```
let age = 123;
```

```
// изменяем значение на undefined  
age = undefined;
```

```
alert(age); // "undefined"
```

...Но так делать не рекомендуется. Обычно `null` используется для присвоения переменной «пустого» или «неизвестного» значения, а `undefined` – для проверок, была ли переменная назначена.

## Объекты и символы

Тип **object** (объект) – особенный.

Все остальные типы называются «примитивными», потому что их значениями могут быть только простые значения (будь то строка, или число, или что-то ещё). В объектах же хранят коллекции данных или более сложные структуры.

Тип **symbol** (символ) используется для создания уникальных идентификаторов в объектах.

# Итого

В JavaScript есть 8 основных типов.

`number` для любых чисел: целочисленных или чисел с плавающей точкой; целочисленные значения ограничены диапазоном  $\pm(2^{53}-1)$ .

`bigint` для целых чисел произвольной длины.

`string` для строк. Строка может содержать ноль или больше символов.

`boolean` для `true/false`.

`null` для неизвестных значений – отдельный тип, имеющий одно значение `null`.

`undefined` для неприсвоенных значений – отдельный тип, имеющий одно значение `undefined`.

`object` для более сложных структур данных.

`symbol` для уникальных идентификаторов.



**Взаимодейств  
ие  
с  
пользователем**

## alert

С этой функцией мы уже знакомы. Она показывает сообщение и ждёт, пока пользователь нажмёт кнопку «ОК».

Например:

```
alert("Привет");
```

Это небольшое окно с сообщением называется **модальным окном**. Понятие модальное означает, что пользователь не может взаимодействовать с интерфейсом остальной части страницы, нажимать на другие кнопки и т.д. до тех пор, пока взаимодействует с окном. В данном случае – пока не будет нажата кнопка «ОК».

## **prompt**

Синтаксис: `result = prompt(title, [default]);`

Этот код отобразит модальное окно с текстом, полем для ввода текста и кнопками ОК/Отмена.

Пользователь может напечатать что-либо в поле ввода и нажать ОК. Введённый текст будет присвоен переменной `result`. Пользователь также может отменить ввод нажатием на кнопку «Отмена» или нажав на клавишу `Esc`. В этом случае значением `result` станет `null`.

Например:

```
let age = prompt('Сколько тебе лет?', 100);
```

```
alert(`Тебе ${age} лет!`); // Тебе 100 лет!
```

## confirm

Синтаксис: **result = confirm(question);**

Функция `confirm` отображает модальное окно с текстом вопроса `question` и двумя кнопками: ОК и Отмена.

Результат – `true`, если нажата кнопка ОК. В других случаях – `false`.

Например:

```
let isStudent = confirm("Вы студент?");
```

```
alert( Student ); // true, если нажата ОК
```

Все эти методы являются модалными: останавливают выполнение скриптов и не позволяют пользователю взаимодействовать с остальной частью страницы до тех пор, пока окно не будет закрыто.

На все указанные методы распространяются два ограничения:

1. Расположение окон определяется браузером. Обычно окна находятся в центре.
2. Визуальное отображение окон зависит от браузера, и мы не можем изменить их вид.