Конспект «Основы программирования на JavaScript»

Программа — это набор команд. JavaScript выполняет программу последовательно, команда за командой. Команды разделяются точкой с запятой `;`.

Консоль

Чтобы вывести информацию в консоль, используем команду `console.log`:

```
console.log(данные для вывода в консоль);
```

Эту команду можно использовать в любом месте программы и выводить в консоль результаты выполнения операций и текстовые подсказки. Текстовые подсказки, в отличие от результатов операций, нужно заключать в кавычки.

Комментарии

Комментарии не выводятся в консоль и не влияют на работу программы, но видны разработчику. Код внутри комментариев не выполняется. Обычно в них пишут поясняющие тексты для себя, или для других, или для себя в будущем.

Комментарии бывают двух типов: однострочные и многострочные:

```
// Эта строка кода не выполнится. Однострочный комментарий.

/*
Все эти строки кода не выполнятся.
Так как это многострочный комментарий.
*/
```

Типы данных

С разными типами данных можно производить разные действия, поэтому программисту важно знать, с чем он работает. В нашей консоли тип данных выводится в скобках, например `(String)` или `(Number)`.

Существуют простые и сложные типы данных. Простые:

— `number` — числа: целые и с точкой;

— `string` — строки;

— `boolean` — логические, или булевы, значения: `true` — «истина» и `false` — «ложь»;

— `undefined` — «не определено», англ.

Строки нужно оборачивать в кавычки: одинарные или двойные.

Сложные, или составные, типы содержат не одно, а несколько значений. Массив, `array`, хранит последовательность значений, и порядок этих значений важен. Объект, `object`, состоит из множества пар «ключ-значение», порядок этих пар не важен.

```
// Массив
[1, 2, 3, 4, 5]

// Объект
{month: 'june', day: 15}
```

Переменные

Переменная — просто название для данных, которое можно делать понятным для людей. Переменные упрощают работу с памятью: они «приклеиваются» к ячейкам памяти, как наклейка с названием приклеивается к папке с документами.

В JavaScript переменные можно создавать командой `let`, за которой следует имя переменной:

```
let имяПеременной;
```

Имя переменной можно записать по-разному. Два самых популярных способа: camelCase (верблюжья нотация) и snake_case (змеиная нотация). В первом случае все слова пишутся слитно и каждое слово, за исключением первого, начинается с большой буквы (`myNumber`, `userName`). Во втором случае все слова разделяются нижним подчёркиванием (`my_number`, `my_name`).

Имена переменных в JavaScript чувствительны к регистру: `myname` и `myName` — две разные переменные. Имя переменной может содержать буквы, цифры и знак подчёркивания, но оно не должно начинаться с цифры. Кроме того, в качестве имени переменной нельзя использовать ключевые слова, такие как `let` или `if`. Вот полный список этих ключевых слов.

После создания переменной её можно использовать в других командах, например, выводить в консоль:

```
// Обратите внимание, что кавычек нет!
console.log(имяПеременной);
```

Если обратиться к пустой переменной, то получим `undefined` — «не определено». Чтобы записать в переменную данные, ей их нужно *присвоить*. Для операции присваивания используется знак равенства:

```
let timeInHours;              // Объявляем переменную
console.log(timeInHours);     // Выведет: undefined

timeInHours = 2;              // Присваиваем одно значение
console.log(timeInHours);     // Выведет: 2

timeInHours = 'три часа';     // Присваиваем совершенно другое значение
console.log(timeInHours);     // Выведет: три часа
```

Команда `let` для создания каждой переменной используется всего один раз. Дальше мы обращаемся к переменной по её имени, без `let`. Если повторно задать значение переменной, то значение этой переменной изменится. Предыдущее значение при этом исчезнет. Это называется переопределением переменной.

Операции и операторы

Команды состоят из операций. `5 + 10;` — это операция. Она состоит из оператора, `+`, и двух операндов, `5` и `10`.

Оператор указывает, что произойдёт с операндами. Операции бывают унарными, бинарными и тернарными, в зависимости от количества операндов. Бинарные операции самые распространённые.

Над разными типами операндов можно производить разные операции, поэтому важно понимать, данные какого типа хранятся в переменных.

Порядок выполнения операций зависит от их приоритета. Если у операций одинаковый приоритет, они выполняются слева направо. Приоритет различных операторов можно посмотреть здесь.

Арифметические операции

Арифметические операции в JavaScript выполняются так же, как в математике: сначала умножение, потом сложение. Изменить порядок операций можно с помощью круглых скобок. Снова как в математике: выражение в скобках посчитается в первую очередь.

Сложение	+
Вычитание	-
Умножение	*
Деление	/

Конкатенация

Самая частая строковая операция — это «склеивание» строк, или конкатенация:

```
let name = 'Кекс';

'Инструктор' + 'Кекс';  // Результат: 'ИнструкторКекс'
'Инструктор ' + 'Кекс'; // Результат: 'Инструктор Кекс'
'Инструктор ' + name;   // Результат: 'Инструктор Кекс'
```

Конкатенация позволяет делать сообщения программ более информативными и «человечными».

Приведение типов

Что будет, если использовать операнды разного типа?

```
'Время, мин: ' + 50; // Результат: 'Время, мин: 50'
'2' * 50;            // Результат: 100
```

JavaScript попытается привести операнды к одному типу и выполнить операцию. Подходящий тип будет выбираться в зависимости от операции.

Плюс может быть знаком сложения или конкатенации, но так как один из операндов — строка, то сложение не подходит. Поэтому число `50` приводится к строке `'50'` и склеивается со строкой `'Время, мин: '`.

Звёздочка — это знак умножения, со строками она не используется. Поэтому JavaScript пытается превратить строку `'2'` в число, и ему это удаётся. Затем числа `2` и `50` перемножаются, и получается `100`.

Из-за того, что JavaScript умеет изменять тип операндов на лету, он называется языком со *слабой типизацией*.

Продолжить




Практикум

Курсы для новичков

Подписка

Профессии

Фронтенд-разработчик

JavaScript-разработчик

Фулстек-разработчик

Услуги

Работа наставником

Для вузов и колледжей

Для учителей

Курсы

HTML и CSS. Профессиональная вёрстка сайтов

HTML и CSS. Адаптивная вёрстка и автоматизация

JavaScript. Профессиональная разработка веб-интерфейсов

JavaScript. Архитектура клиентских приложений

React. Разработка сложных клиентских приложений

Node.js. Профессиональная разработка REST API

Node.js и Nest.js. Микросервисная архитектура

TypeScript. Теория типов

Алгоритмы и структуры данных

Паттерны проектирования

Webpack

Vite

Vue.js 3. Разработка клиентских приложений

Git и GitHub

Анимация для фронтендеров

Журнал

Справочник по HTML

Учебник по Git

Учебник по PHP

Информация

Об Академии

О центре карьеры

Остальное

Написать нам

Мероприятия

Форум

Акции

Отзывы о курсах

Соглашение

Конфиденциальность

Сведения об образовательной организации

Лицензия № Л035-01271-78/00176657

Участник