Калинин Станислав

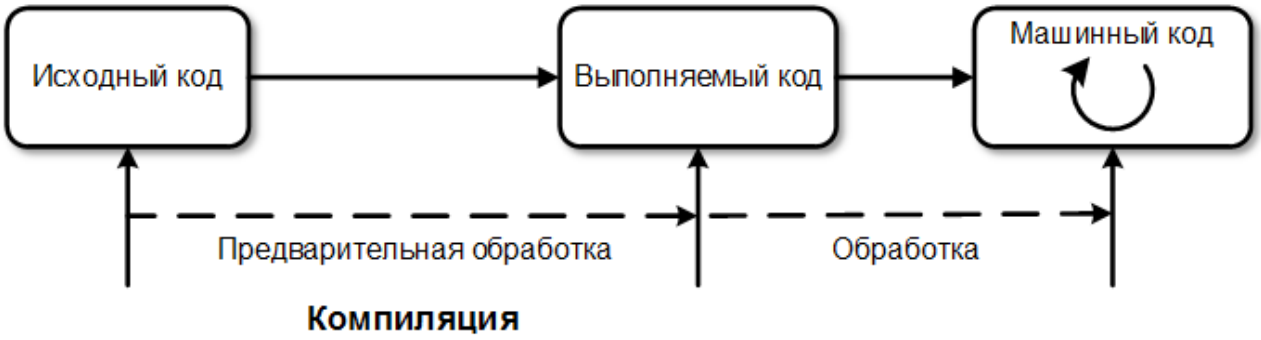
JS Homework #1

1 В чем отличие интерпретатора от компилятора?

Как Компилятор так и Интерпретатор имеют одно предназначение — конвертировать инструкции языка высокого уровня (как C или Java) в бинарную форму, понятную компьютеру. Это программное обеспечение, используемое для запуска высокоуровневых программ и кодов выполняемых различные задачи. Для разных высокоуровневых языков разработаны специфичные компиляторы/интерпретаторы. Не смотря на то что как компилятор так и интерпретатор преследуют одну и ту же цель, они отличаются способом выполнения своей задачи, то есть конвертирования высокоуровневого языка в машинные инструкции.

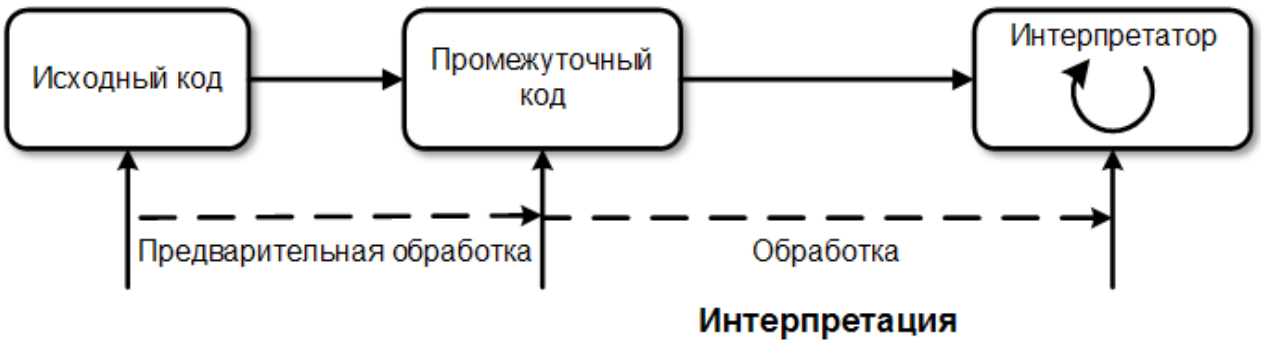
Компилятор

Компилятор транслирует высокоуровневый язык в машинный. Когда пользователь пишет код на языке высокого уровня, таком как Java, и хочет его выполнить, то прежде чем это может быть сделано, будет использован специальный компилятор разработанный для Java. Компилятор сначала сканирует всю программу, а потом транслирует ее в машинный код, который будет выполнен компьютерным процессором, после чего будут выполнены соответствующие задачи.



Интерпретатор

Интерпретаторы не очень сильно отличаются от компиляторов. Они также конвертируют высокоуровневые языки в читаемые машиной бинарные эквиваленты. Каждый раз когда интерпретатор получает на выполнение код языка высокого уровня, то прежде чем сконвертировать его в машинный код, он конвертирует этот код в промежуточный язык. Каждая часть кода интерпретируется и выполняется отдельно и последовательно, и если в какой-то части будет найдена ошибка, она остановит интерпретацию кода без трансляции следующей части кода.



Ниже перечислены ***главные отличия*** между компилятором и интерпретатором:

- Интерпретатор берет одну инструкцию, транслирует и выполняет ее, а затем берет следующую инструкцию. Компилятор же транслирует всю программу сразу, а потом выполняет ее.

- Компилятор генерирует отчет об ошибках после трансляции всего, в то время как интерпретатор прекратит трансляцию после первой найденной ошибки.
- Компилятор по сравнению с интерпретатором требует больше времени для анализа и обработки языка высокого уровня.
- Помимо времени на обработку и анализ, общее время выполнения кода компилятора быстрее в сравнении с интерпретатором.

2 Что такое EcmaScript? Почему он так называется и кто занимается его развитием?

JavaScript создан в 1995 г. в компании Netscape. По инициативе компании Netscape была проведена стандартизация языка ассоциацией ECMA, которая занимается стандартизацией информационных технологий. Так появился стандарт ECMAScript, сокращенно его называют ES. Стандарт описывается документом ECMA-262.

Отличие JavaScript от ECMAScript

ECMAScript это стандарт, а JavaScript его реализация. ECMAScript говорит как должно быть, а JavaScript выполняет то, что говорит ECMAScript.

Кроме JavaScript существуют другие реализации стандарта: SpiderMonkey, V8 и ActionScript. Стоит отметить, что реализация JavaScript может не соответствовать стандарту, в чём убедились сотни тысяч разработчиков при работе с Internet Explorer старых версий.

Ecma International — основанная в 1961 году ассоциация, деятельность которой посвящена стандартизации информационных и коммуникационных технологий. Изначально ассоциация называлась ECMA — European Computer Manufacturers Association, однако она сменила название в 1994 году в связи с глобализацией её деятельности. Вследствие

этого название Ecma перестало быть аббревиатурой и больше не пишется заглавными буквами.

3 Какие операции можно выполнять с NaN и Infinity?

NaN

В JavaScript имеется предопределённая глобальная переменная NaN. Она хранит специальное значение NaN (NaN сокращение от англ. Not a Number – не число). Эта переменная доступна только для чтения.

Значение NaN используется для обозначения математической ошибки, которая возникает в том случае, если математическая операция не может быть совершена:

```
1	var a = 10, b = "текст";
2
3	document.write(a - b);  // Операнд не может быть преобразован в число
4	document.write(0 / 0);  // Деление нуля на нуль
```

Значение NaN обладает одной особенностью: оно не равно никакому значению, в том числе и другому NaN:

```
1	NaN == NaN   // false
```

Чтобы определить, является ли значение переменной значением NaN следует выполнить проверку на неравенство x != x. Эта проверка вернёт true тогда и только тогда, когда x имеет значение NaN:

```
1    var x = NaN;
2
3    alert(x != x);   // true
```

Если вычисление возвращает значение NaN, это значение не может использоваться ни в каких дальнейших вычислениях, потому что значение NaN не имеет числового представления и любая арифметическая операция с NaN всегда возвращает NaN:

```
1    var num = NaN / 10;   // NaN
```

Infinity

В JavaScript имеется предопределённая глобальная переменная Infinity. Она хранит специальное значение обозначающее бесконечность – Infinity. Эта переменная доступна только для чтения.

Значение Infinity можно получить:

- В результате деления числа на 0.
- Если результат вычислений, не попадает в допустимый диапазон чисел JavaScript. Любое отрицательное число, которое не может быть представлено, считается отрицательной бесконечностью (-Infinity), а положительное – положительной бесконечностью (Infinity).

```
1    alert(123 / 0);     // Infinity
2    alert(-12345 / 0);  // -Infinity
3    alert(1e500);       // Infinity
4    alert(1 - 1e500);   // -Infinity
```

Если вычисление возвращает одну из бесконечностей, это значение не может использоваться ни в каких дальнейших вычислениях, потому что значение Infinity не имеет числового представления и любая арифметическая операция с бесконечностью всегда возвращает бесконечность:

```
1    alert(20 + Infinity);   // Infinity
```

4 Опишите все типы данных в JS.

В JavaScript типы данных можно разделить на две категории: простые (их также называют примитивными) типы и составные (их также называют ссылочными или объекты).

К категории простых типов относятся:

- **string** - текстовые строки (обычно их называют просто - строки)
- **number** - числа
- **boolean** - логические (булевы) значения

Так же к простым типам относятся два специальных значения:

- **null**
- **undefined**

К составным типам данных относятся:

- **function** - функции
- **array** - массивы
- **object** - объекты