GeekBrains

Основы программирования

Программа, переменные и типы данных

Привет!

Ты приступаешь к курсу "Основы программирования". Прежде чем начать, прочитай это сообщение.

Курс посвящен основам JavaScript, но, даже если ты планируешь изучать другой язык программирования, он поможет разобраться на базовом уровне в синтаксисе и заложит основы, которые помогут в дальнейшем.

- Курс состоит из пяти видео-уроков. Один урок - это видео, методичка, которая лучше поможет разобраться в том, о чем говорит спикер, и практическое задание.
- Ищи задание в методичке. Пожалуйста, пытайся одновременно с изучением материала делать задачки, которые мы предлагаем: так тебе будет проще взяться за практику.
- Все задачи решай в онлайн-редакторе: http://jsbasics.geekbrains.ru.
- Если вдруг что-то не получается и программа не работает - проверь внимательно написанный тобою код: возможно что-то пропущено или, наоборот, написано лишнее.
- После того, как разберешься в материале, приступай к практическому заданию. Оно поможет закрепить пройденное и проверить себя.

К сожалению, в этом курсе не предусмотрен преподаватель, который будет проверять домашнее задание и давать обратную связь. Если программа не работает, внимательно смотри, какую ошибку выдает редактор кода и пытайся ее найти и исправить. Если даже после нескольких попыток это тебе не удастся - пиши в поддержку.

Удачи!

На этом уроке

1. Узнаем что такое программа, из чего она состоит и как она выполняется;
2. Узнаем что такое переменная и какие типы данных она может хранить;
3. Научимся создавать и использовать переменные.

Оглавление

Введение

Что такое программа

Какие инструменты нужны для написания программы

Что такое JavaScript

Получение пользовательских данных и хранение их в переменной

Переменные

Вывод данных

Работа с переменными

Сложение, вычитание, умножение и деление

Комментарии

Пишем первую программу

Работа с графикой

Домашнее задание

    Дополнительно

Глоссарий

Дополнительные материалы

Используемые источники

Введение

Любой компьютер (смартфон, смарт-часы, современный телевизор, бортовой компьютер на МКС — все это компьютеры), как вы наверняка знаете, состоит из аппаратного обеспечения («железо») и программного обеспечения («софт»). Если быть точным, из железа, софта и данных. Железо — это нечто материальное, что можно потрогать, что имеет вес. Данные — это представление информации или просто информация. А вот софт — это то, что даёт компьютеру «жизнь», то, что оживляет экран компьютера (про смартфоны тоже не забываем), позволяет принтерам печатать, стиральным машинам стирать, роботам ходить, приложению DeliveryClub организовывать доставку еды к вам домой и так далее. Все эти три компонента друг без друга не имеют смысла. Без данных программе нечего будет вычислять, а без железа -- не на чем.

На этом курсе вы научитесь «оживлять» компьютер, то есть создавать программы.

Что такое программа

Компьютерная программа — это комбинация компьютерных инструкций и данных, позволяющая аппаратному обеспечению вычислительной системы выполнять вычисления или функции управления. Это определение из Википедии, а если говорить проще, то программа — это просто ***набор инструкций, которые работают с данными***.

Давайте посмотрим на простейшую программу:

```
let name = prompt('Укажите ваше имя');
console.log('Привет, ' + name + '!');
```

Листинг 1

Что делает эта программа? Даже человек, первый раз видящий какой-либо исходный код, скажет, что скорее всего, программа каким-то образом получает от пользователя имя и говорит пользователю: “привет”. Так оно и есть и мы можем это понять, не разбираясь в языках программирования, потому языки программирования как раз и созданы для того, чтобы нам было проще писать инструкции для компьютера и понимать уже написанные.

Какие инструменты нужны для написания программы

Для того, чтобы написать программу, достаточно ручки/карандаша и листок бумаги. Но для того, чтобы программу выполнить, нужен компьютер с установленным инструментарием, поддерживающим выполнение программы на том языке программирования, на котором написана ваша программа. А непосредственно писать текст программы на компьютере можно в обычном блокноте, но программисты для удобства используют редакторы кода или интегрированные среды разработки

программного обеспечения. Нам сейчас ни к чему забивать голову инструментарием, нам важнее понять основы программирования, поэтому мы будем использовать простейший инструмент, онлайн-редактор кода JavaScript, который может выполнить программу прямо в браузере: http://jsbasics.geekbrains.ru.

Что такое JavaScript

JavaScript — это язык программирования, на котором мы будем писать программы на этом курсе. Языков программирования много всяких разных: Си, Python, Java, PHP, Rust, Scala, Kotlin, Go и много-много других. Но нас сейчас интересует JavaScript, во-первых, потому что он нам пригодится в дальнейшем, во-вторых, потому что он достаточно прост в изучении.

Получение пользовательских данных и хранение их в переменной

Вернёмся к тексту программы (Листинг 1), а именно рассмотрим первую его строку:

```
let name = prompt('Укажите ваше имя');
```

В самом начале мы видим ключевое слово **let**, оно используется для объявления переменных. **name** — это название переменной, его мы должны придумать сами. Дальше идёт знак **=**, который отвечает за присвоение (назначение) переменным какого-то значения. Потом идёт вызов функции **prompt**, в которую передаётся текст, говорящий пользователю что мы от него хотим. Как было сказано, **prompt** — это функция, но поскольку мы пока ещё не знаем что такое функции, нам проще рассматривать весь вызов prompt('Укажите ваше имя') как инструкцию, которую на русский можно было перевести так: «Покажи пользователю диалоговое окно с текстом “Укажите ваше имя” и запроси у пользователя ввод данных». Строка кода заканчивается точкой с запятой.

Теперь у нас есть вся информация для того, чтобы прочитать эту строку кода: объявлена переменная name, которой присвоено значение, полученное от пользователя. Самым неясным для нас моментом пока остаётся **переменная**.

Переменные

Давая вольное определение программе, мы сказали, что это набор инструкций, которые что-то делают с данными. Так вот, переменные в программе — это инструмент хранения данных. Предположим, что мы собираем о пользователе какую-то информацию:

```
let name = 'Василий';
```

```
let age = 33;
let isStudent = true;
```

Что мы на текущий момент можем сказать об этом коде? Мы уже знаем, что let отвечает за объявление переменных. Значит, в этом коде объявлено три переменных, name, age и isStudent.

Мы также видим, что переменным присвоены значения. Переменной, которую мы назвали **name** (имя) присвоено **текстовое** значение 'Василий' (**строковый тип**). Что такое текст мы прекрасно понимаем, нам знаком этот тип данных. А теперь мы знаем как в программе объявлять текст: достаточно поместить его в одинарные **'** или двойные **"** кавычки.

Дальше мы видим как создаётся переменная, которой мы придумали название **age** (возраст), и ей присваивается **числовое** значение 33 (**числовой тип**). Что такое число нам тоже понятно и объявлять числа в программе даже проще, чем текст: не нужны никакие кавычки.

А вот что происходит на следующей строке, сказать пока не так просто. Мы видим, что создаётся переменная **isStudent** (можно перевести как «является ли студентом», да английский более лаконичен, поэтому он и используется в программировании) и ей присваивается какой-то **true**. Дело в том, что **true** — это булево значение (**логический тип**). В дополнительных материалах к этому уроку будет ссылка на материалы по алгебре логики, её ещё называют булевой алгеброй. Не поленитесь, потратьте десять минут, чтобы освежить в памяти эти знания из школьного курса математики. Но сейчас нам про булевы значения нужно знать только то, что они бывают двух типов: **true** (правда) и **false** (ложь). Самые простой и понятный аналог это выключатель света в комнате, он может быть только в одном из двух положений: включен (**true**) и выключен (**false**). Итак, переменной **isStudent** присвоено значение **true**, из чего мы можем сделать вывод, что пользователь является студентом, если бы переменной было присвоено значение **false**, мы бы сказали, что пользователь НЕ является студентом.

Итак, мы рассмотрели три простых типа данных: строка, число и булев тип. Оставим пока переменные и перейдём ко второй строке программы (Листинг 1).

```
console.log('Привет, ' + name + '!');
```

Вывод данных

Если бы программа не могла как-то сообщить о результатах своей работы (вывести данные), она была бы бесполезна. Один из способов, которым на языке JavaScript можно вывести данные, является вызов функции **console.log**(). Она может выводить что угодно:

```
console.log('Привет, Мир!');
console.log(20);
```

```
console.log(false);
```

Работа с переменными

Вернёмся ко второй строке Листинга 1, давайте обратим внимание на эту операцию: 'Привет, ' + name + '!'. Мы понимаем что здесь формируется строка “Привет, [имя пользователя]!”. И формируется она сложением (оператор **+**) строки “Привет, ” с переменной name и затем со строкой “!”. Мы можем сделать, как минимум, два вывода: во-первых, строки можно складывать, во-вторых, переменные могут участвовать в операциях сложения. Этот же код мы могли бы записать так:

```
let hello = 'Привет, ';
let exclamation = '!';
let result = hello + name + exclamation;
console.log(result);
```

Несмотря на то, что программа решает ту же самую задачу, что и до этого, изменился наш, как программиста, подход к решению. Мы вынесли строки в отдельные переменные, результат сложения строк тоже присвоили переменной, которую назвали **result** (результат), а затем вызвали **console.log** и передали в вызов переменную **result**. Этот пример хорошо показывает, что мы вольны объявлять переменные тогда, когда они нам нужны и когда нам это удобно, кроме того, мы можем передавать переменные в вызываемые функции.

Нужно понимать, что одну и ту же переменную нельзя объявить дважды, например, так делать нельзя, это вызовет ошибку:

```
let text = 'Какая-то текстовая строка';
let text = 'Другая текстовая строка';
```

Зато вы можете задать уже существующей переменной другое значение:

```
let text = 'Какая-то текстовая строка';
text = 'Другая текстовая строка';
```

Сложение, вычитание, умножение и деление

Уже знакомый нам оператор сложения (**+**) можно использовать и с числами:

```
let a = 2;
```

```
let b = 3;
console.log(a + b);
```

Нет ничего удивительного в том, что к числам можно применять и другие арифметические операции, для этого используются: оператор вычитания (**-**), оператор умножения (*****) и оператор деления (**/**):

```
let a = 2;
let b = 3;
console.log(a + b); // 5
console.log(a - b); // -1
console.log(a * b); // 6
console.log(a / b); // 0.6666666666666666
```

Заметьте, что в коде появились комментарии.

Комментарии

В JavaScript комментарий можно разместить одним из двух способов:

```
/* Это многострочный комментарий,
он очень удобен, когда комментарий длинный
и занимает несколько строк */

// А это однострочный комментарий

// Впрочем, если вашему комментарию нужно
// несколько строк, то можно использовать
// и однострочные комментарии, это дело вкуса
```

Комментарии предназначены для программиста, компьютер, при выполнении программы, игнорирует комментарии. У комментариев может быть много применений. Например, комментарии помогают описать участки кода, пояснить что происходит в конкретном месте, отметить места, на которые разработчику стоит обратить внимание в дальнейшем.

Теперь у нас есть достаточно знаний для того, чтобы написать простую программу, получающую от пользователя два числа и выводящую результат умножения одного на другое.

Пишем первую программу

Получать пользовательский ввод мы умеем. Нам необходимо создать переменную и задать ей значение, полученное от пользователя. Давайте назовём переменную **a**. Используя функцию prompt запросим у пользователя первое число:

```
let a = prompt('Введите первое число');
```

Это первая строка нашей программы. На второй строке совершим такое же действие, только переменную назовём b, ну и покажем пользователю соответствующий текст запроса, теперь наша программа выглядит так:

```
let a = prompt('Введите первое число');
let b = prompt('Введите второе число');
```

Для того, чтобы умножить одну переменную на другую, необходимо воспользоваться оператором умножения (*), сама операция умножения будет выглядеть так a * b, давайте запишем это в третью строку программы:

```
let a = prompt('Введите первое число');
let b = prompt('Введите второе число');
a * b;
```

Несмотря на то, что наша программа выполняет все нужные вычисления, мы лишены возможности увидеть результат. Давайте сохраним результат в переменную **result**:

```
let a = prompt('Введите первое число');
let b = prompt('Введите второе число');
let result = a * b;
```

Что же, интересующий нас результат находится в переменной result, всё что нам нужно, это вывести эту переменную пользователю. Мы знаем, что вывод осуществляется с использованием **console.log**:

```
let a = prompt('Введите первое число');
let b = prompt('Введите второе число');
let result = a * b;
console.log(result);
```

Программа готова. Давайте обратим внимание вот на что: переменная result создаётся только для того, чтобы сразу же быть переданной в **console.log**. Мы можем это с лёгкостью избежать:

```
let a = prompt('Введите первое число');
let b = prompt('Введите второе число');
console.log(a * b);
```

Теперь наша программа стала короче и даже понятнее — перед глазами нет переменной result, от которой было мало пользы. Аналогично мы можем избавиться от переменных a и b:

```
console.log(prompt('Введите первое число') * prompt('Введите второе число'));
```

Теперь мы вообще не используем переменные, а результаты вызова prompt сразу перемножаются и передаются в console.log. Несмотря на то, что программа стала значительно короче, гнаться за краткостью тоже не стоит. Согласитесь, этот вариант гораздо сложнее читать, чем предыдущий. Так что давайте остановимся на самом оптимальном варианте:

```
let a = prompt('Введите первое число');
let b = prompt('Введите второе число');
console.log(a * b);
```

Обратите внимание.

Допустим вы написали вот такую программу.

```
let a = prompt('Введите первое число');
let b = prompt('Введите второе число');
console.log(a + b);
```

И при запуске ввели первое число 1 и второе число 1. После этого вы обнаружите, что в консоль выводится не сумма чисел 1 + 1 = 2, а почему-то некая их “склейка” 1 + 1 = 11. Почему так произошло?

Дело в том, что при выполнении функции prompt() JavaScript считает что вы ввели текст, и тогда делает все правильно “1” + “1” = “11”, мы склеиваем две строки в одну, то есть ошибки никакой нет. А что делать если мы хотим вводить именно числа?

В таком случае, необходимо воспользоваться функцией parseInt(), которая преобразует текст в число. В коде это выглядит следующим образом:

```
let a = parseInt(prompt('Введите первое число'));
let b = parseInt(prompt('Введите второе число'));
console.log(a + b);
```

Теперь в консоль отпечатается число 2. Чтобы понять что изменилось, давайте посмотрим в каком порядке JavaScript будет выполнять первые две строки.

1. `let a = parseInt(prompt('Введите первое число'))`

   В начале выполнится prompt() и запросит у вас ввод текста.

2. `let a = parseInt(“1”)`

Представим что вы ввели 1, тогда на месте prompt() для JavaScript в эту строку подставится “1”, и этот текст будет отдан функции parseInt(), которая преобразует его в число 1

3. И третьим этапом в переменную a запишется число 1
   `let a = 1`
4. Ну а дальше будет выполнена остальная часть программы

Следует учесть, что если вы введете не число, а какой-то текст, то функция parseInt() не сможет выполнить преобразование такой строки в число и выдаст ошибку.

Работа с графикой

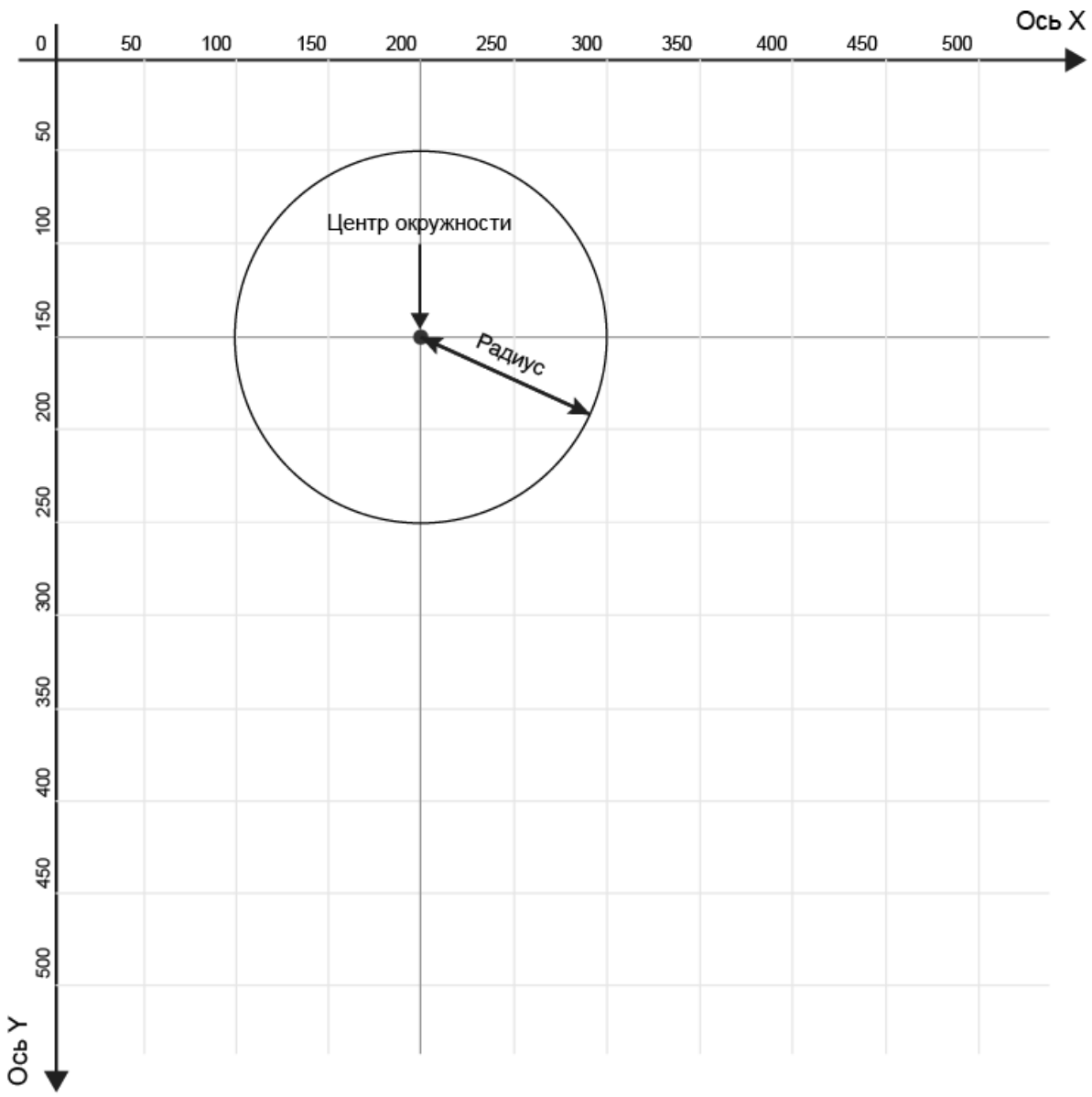
Для того, чтобы показать вам, что программа позволяет взаимодействовать с пользователем не только с помощью текстовых сообщений, мы сделали поддержку простых функций для рисования в нашем онлайн-редакторе.

Эти функции не являются стандартными для JavaScript, они используются только в рамках этого курса для ознакомления с основами программирования графики.

Сегодня мы рассмотрим функцию drawCircle, которая рисует круг:

```
drawCircle(200, 150, 100, 'black');
```

Первое число, которое мы передаём в функцию — это координата по оси X. А второе число — координата по оси Y. Эти координаты сообщают функции drawCircle где должен быть центр круга, который мы собираемся нарисовать. Третьим аргументом функции является радиус окружности. Четвёртый аргумент — название цвета, которым мы хотим нарисовать круг, на английском языке. Координатная сетка при этом выглядит так:



Оригинал: https://drive.google.com/file/d/1EdAQAJX3LMFV-YDD0J_VBhzXHFqv0dIs/view?usp=sharing

Исходник: https://drive.google.com/file/d/1mdFHJxv39GwghDRHDM8MS3I0IXDoEQBA/view?usp=sharing

Домашнее задание

Необходимо написать программу, которая запрашивает у пользователя его имя, год рождения и выводит в консоль текст, например «Иван, уже в следующем году вам будет 40 :-(». Для того, что вычислить сколько лет будет пользователю в следующем году, необходимо от текущего года отнять год рождения и добавить единицу. Каждая значимая строка программы должна сопровождаться комментарием.

Дополнительно

Написать программу, которая рисует смайлик (один чёрный круг, и два белых круга поверх чёрного — глаза, дополнительно можно нарисовать зрачки).

Глоссарий

- Компьютер — устройство или система, способная выполнять заданную, чётко определённую, изменяемую последовательность операций. Это чаще всего операции численных расчётов и манипулирования данными, однако сюда относятся и операции ввода-вывода;
- Компьютерная программа — комбинация компьютерных инструкций и данных, позволяющая аппаратному обеспечению вычислительной системы выполнять вычисления или функции управления;
- Исходный код — текст компьютерной программы на каком-либо языке программирования или языке разметки, который может быть прочтён человеком;
- Интегрированная среда разработки — комплекс программных средств, используемый программистами для разработки программного обеспечения;
- Язык программирования — формальный язык, предназначенный для записи компьютерных программ;
- JavaScript — мультипарадигменный язык программирования высокого уровня;
- Переменная — поименованная, либо адресуемая иным способом область памяти, адрес которой можно использовать для осуществления доступа к данным и изменять значение в ходе выполнения программы;
- Строковый тип — тип данных, значениями которого является произвольная последовательность (строка) символов алфавита;
- Числовой тип — тип данных, значениями которого являются как целочисленные значения, так и числа с плавающей точкой;
- Логический тип — примитивный тип данных в информатике, принимающий два возможных значения, иногда называемых истиной (true) и ложью (false).

Дополнительные материалы

- Конспект школьного урока по алгебре логики — https://resh.edu.ru/subject/lesson/5426/conspect/163619/;
- Переменные в JavaScript — https://learn.javascript.ru/variables.

Используемые источники

- https://wikipedia.org