1 Привет, Мир!

Вы уже готовы написать свои первые строчки **кода**?
Сейчас-сейчас, только соберём парочку вещей в дорогу. За дело!

Компьютер, браузер, консоль

Первым делом вам понадобится **компьютер**: стационарный, ноутбук или Chromebook (но не смартфон или планшет!). Если у вас нет собственного компьютера, можете воспользоваться компьютером в библиотеке. Или можете прийти домой к другу, у которого есть компьютер. Ну или можете попробовать собрать компьютер из проволоки, светодиодов и кастрюли с остывшим пюре (не рекомендуется).

Далее: на компьютере должна быть установлена **операционная система**: Windows (если у вас PC), MacOS (если у вас компьютер от Apple), ChromeOS (если у вас Chromebook) или Linux (если вы компьютерный гик). Вы должны знать, какая именно операционка установлена у вас на компьютере.

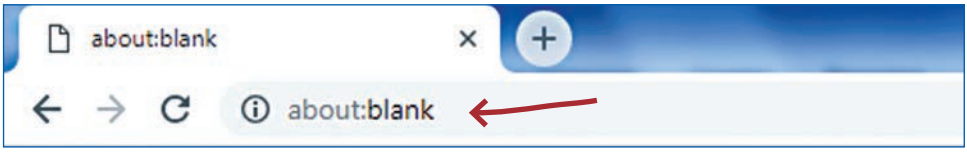
У вас должен быть установлен Google **Chrome**. Chrome — это **браузер**, то есть программа для просмотра веб-страниц. Вполне вероятно, что он уже установлен на вашем компьютере: поищите на рабочем столе иконку, похожую на разноцветный покебол[1].

Кстати, вы заметили, что некоторые слова выделены **жирным шрифтом**? Так я отмечаю важные для нашей работы слова, определение которых вы найдёте в глоссарии в конце книги. «Что же такое **глоссарий**?» — спросите вы. Глоссарий — это как словарь, только специальный, в котором собраны термины, использующиеся в этой книге. Короче, загляните в конец книги и посмотрите!

[1] Покебол (*англ.* — Pokéball) — мяч для ловли покемонов, монстров из одноимённой анимационной серии мультфильмов. — *Здесь и далее примеч. пер.*

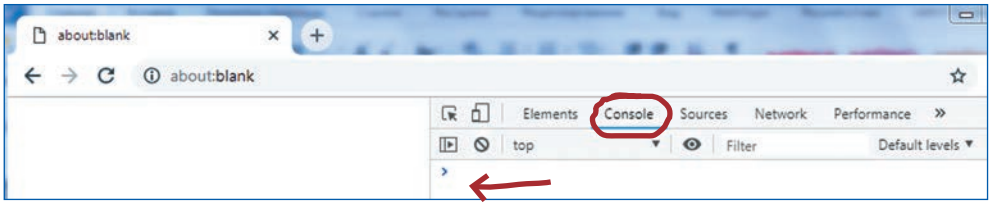
Если у вас всё же не установлен Chrome, то можете скачать его при помощи любого другого браузера (Firefox, Safari, Edge или IE), просто вбив в поисковой строке «скачать Chrome» и следуя дальнейшим инструкциям.

Отлично. После скачивания и установки Chrome откройте его и в адресной строке введите: `about:blank`. Нажмите ENTER (или RETURN, если у вас Mac). Браузер откроет новую, пустую страницу. Обратите внимание: клавиши, которые следует нажимать, также ВЫДЕЛЕНЫ в тексте, но их определений вы в глоссарии не найдёте, потому… ну, короче, потому что это просто кнопки на вашей клавиатуре.



Итак, пришло время познакомиться с консолью! **Консоль** является одним из мегасекретных инструментов Chrome, предназначенных специально для разработчиков. Раз вы теперь *начинающий программист-разработчик, ею можете пользоваться и вы!* Чтобы вызвать консоль, просто нажмите вместе клавиши CTRL, SHIFT (или же COMMAND и OPTION, если у вас Mac) и J. В будущем все подобные комбинации мы будем прописывать в виде CTRL+SHIFT+J (или COMMAND+OPTION+J для Mac соответственно).

После нажатия комбинации клавиш откроется новая строка меню, а в левом верхнем углу страницы вы увидите угловую скобку. Если же комбинация CTRL+SHIFT+J (или COMMAND+OPTION+J) по каким-то причинам *не работает*, вы можете войти в консоль, нажав на пустое место на странице правой кнопкой мыши, выбрать в контекстном меню пункт Inspect и затем нажать Console в строке меню (как показано на скриншоте ниже).



Кстати, если переместить курсор к краю светло-серой области, где он превратится в двойную стрелку, то, нажав и удерживая левую кнопку, можно растянуть окно консоли; обязательно сделайте это, чтобы у вас было больше места для введения кода.

Теперь, когда перед вами открыто окно консоли и вы растянули его удобным образом, нажмите мышкой рядом с угловой скобкой (>). В строке появится мигающий курсор, означающий, что можно начинать вводить текст. Сейчас вы наберёте свои первые строчки кода. Введите следующий текст (вместе с кавычками), не забыв точку с запятой «;» (находится чуть правее центра на клавиатуре) — важный символ, который будет постоянно заключать строку кода:

```
"Привет, Мир!";
```

Нажмите ENTER (или RETURN на Mac). Консоль вернёт вам в следующей строке ваш текст: `"Привет, Мир!"`. Обратите внимание: код, который вы должны будете ввести в консоль, `отображается синим цветом`, ответ же консоли `отображается красным`. Это сделано для того, чтобы вам легче было понять, что именно следует вводить в строку консоли. Позднее вы встретитесь также с примерами кода, `выделенного серым цветом`, — эти строки вам нужно будет лишь прочесть (а не вводить в строку).

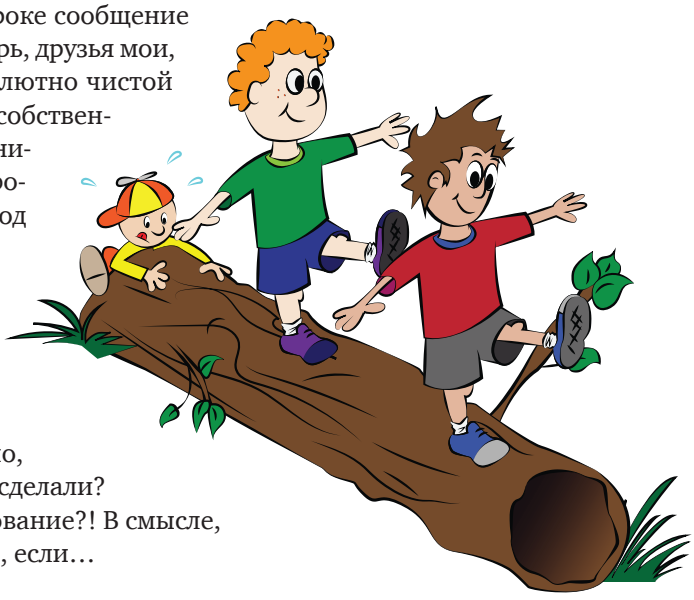
Теперь нажмите клавишу UP_ARROW (стрелка вверх); вы увидите свой текст `"Привет, Мир!";`. Переместите курсор в начало строки и наберите вместо этого текста следующий:

```
var приветствие = "Привет, Мир!";
```

и нажмите ENTER (помните, что ENTER всегда означает RETURN, если у вас Mac). В консоли вы увидите ответ: `undefined.` Всё идёт по плану! Теперь наберите:

```
приветствие;
```

и нажмите ENTER. Если вы всё сделали верно, то консоль вернёт в строке сообщение `"Привет, Мир!"`. Что ж, теперь, друзья мои, вы можете всегда и с абсолютно чистой совестью заявлять, что вы собственноручно написали и выполнили (хоть и совсем микрокрохотулечный, но всё же…) код на языке **JavaScript**!



Повторяй за мной!

Ну и… в чём же, собственно, смысл? Что мы только что сделали? И ЭТО и есть программирование?! В смысле, да кому вообще какое дело, если…

Таааак! Придержите коней! Очень рад, что спросили. Честное-пречестное! Я просто сгораю от нетерпения, чтобы всё это вам объяснить, но сперва давайте выполним маленькое упражнение на доверие. Мы сыграем в «повторяй за мной» — просто набирайте в консоли всё, что указано ниже (и `выделено синим цветом`!). Каждую строку завершайте нажатием клавиши ENTER и наслаждайтесь результатом:

```
5 + 8;
```

Консоль ответит вам: `13`

```
7 - 3 + 6;
```

И ответ: `10`

Далее наберите (обратите внимание на символ `*`, означающий *умножение*):

```
var x = 3; 4 * x;
```

И получите в ответе: `12`

Теперь:

```
x + x;
```

Ответ: `6`

Обратите внимание, что последнее вычисление было выполнено, используя полученные выше значения `x`. Едем дальше:

```
x = 200;
```

Мы присвоили `x` *новое* значение. Теперь дважды нажмём стрелку вверх, чтобы вернуться к:

```
x + x;
```

Нажав теперь ENTER, мы получим в ответ — `400`, поскольку значение `x` уже изменилось. Ладно, давайте ещё парочку, и я всё подробно объясню.

Наберите (символ `/` означает *деление*):

```
x / 100;
```

Ответ: `2`

Теперь:

```
(x - 50) / 5;
```

И получите ответ: `30`

```
(x - 50) / (5 - 1);
```

Ответ: `37.5`

```
"Хочу прожить до " + x + " лет! ";
```

И ответ: `"Хочу прожить до 200 лет!"`

```
"Хмм… хотя, пожалуй, и до " + (x / 2) + " было бы уже неплохо.";
```

Ответ: `"Хмм… хотя, пожалуй, и до 100 было бы уже неплохо."`

Ну как вам? Удивило ли вас что-нибудь? Я пока ещё ничего не объяснял, но думаю, что что-то вы уже начали понимать, верно?

Были ли у вас какие-нибудь ошибки? Надеюсь, что да. Ведь **разработчики**, пишущие на JavaScript, совершают ошибки в коде по десять раз на дню! Важнейшей чертой хорошего **программиста** является умение выявить собственные ошибки — их ещё называют **багами**[1] — и исправить (то есть раздавить) их. К этому мы ещё обязательно вернёмся; но сперва давайте выясним, из чего состоит ваш первый код на JavaScript — что означает и как интерпретируется (то есть понимается компьютером) каждая его часть.

Синтаксис

JavaScript, подобно множеству прочих **языков программирования**, обладает строгим **синтаксисом**. Синтаксис — это раздел грамматики, изучающий предложения и способы сочетания слов внутри предложения. Компьютеры не столь сообразительны, как люди: они не в состоянии «просто понять», что вы имели в виду. Машина сможет понять вас только в том случае, если вы будете общаться с ней строго в тех формах выражения, которые она от вас ожидает. Эта ожидаемая форма выражения и называется синтаксисом.

Мы сейчас ещё немного поработаем в консоли, но перед этим, пожалуйста, давайте убедимся, что вы уже точно умеете её вызывать. Итак, закроем окно Chrome (даже если вы специально оставили его открытым на будущее) и начнём сначала. Введём в адресной строке `about:blank`, затем при помощи комбинации CTRL+SHIFT+J (или, как вы помните, COMMAND+OPTION+J для Mac) вызовем консоль (обязательно запомните эти комбинации — вы будете ими пользоваться постоянно!). Должно открыться чистое, ожидающее ваших распоряжений окно консоли.

Рассмотрим пару примеров грамотного применения синтаксиса JavaScript. Перепечатывайте в консоль все строки кода, отмеченные синим (пожалуйста, набирайте код В ТОЧНОСТИ как указано далее) и в конце каждой строки нажимайте клавишу ENTER. Приступим:

```
var первая_часть_приветствия = "Привет,";
```

[1] От *англ.* bug — жук. По легенде, в 1945 году учёные Гарвардского университета, тестировавшие вычислительную машину Mark II Aiken Relay Calculator, нашли мотылька, застрявшего между контактами электромеханического реле, и вклеили насекомое в технический дневник с сопроводительной надписью: «First actual case of bug being found» (Первая фактическая ошибка (насекомое) была найдена). — *Примеч. ред.*

Объяснение: перед вами так называемые **предложения** JavaScript. Предложения зачастую заканчиваются точкой с запятой (`;`). Ключевое слово `var` в начале строки объявляет компьютеру (ну то есть технически, конечно, не напрямую компьютеру, а всего лишь встроенному в браузер **интерпретатору** JavaScript), что следующее за ним `первыйЧастьПриветствия` является **переменной**. Далее знак равенства (`=`) указывает компьютеру (тьфу… *интерпретатору*), что мы присваиваем значение `"Привет"` нашей переменной `первыйЧастьПриветствия`, чтобы в дальнейшем можно было бы производить с ней операции. Что-то многовато длинных-умных слов, да? Знаю-знаю, но, пожалуйста, постарайтесь их запомнить, потому что к ним мы будем обращаться практически постоянно. Если забудете, что какое-то там слово означает, помните, что всегда можно заглянуть в глоссарий в конце книжки!

```
let вторыйЧастьПриветствия = "Мир";
```

Консоль ответит нам: `undefined`

Объяснение: в этом предложении мы также произвели операцию **присваивания** значения. Ключевое слово `let` делает то же самое, что и `var`. Впрочем, некоторые различия между ними, конечно, есть, но их подробное описание сейчас было бы излишне, так что ладно. Просто имейте в виду, что несмотря на то, что по большей части мы здесь будем использовать `let`, не менее часто при дальнейшем изучении JavaScript вы встретитесь и с `var`. В нашем предложении мы присвоили новой переменной `вторыйЧастьПриветствия` **строку** (`"Мир"`). Слово `"Мир"` называется строкой, поскольку заключено в кавычки (об этом мы подробно поговорим в одной из следующих глав). Ещё вы, наверное, обратили внимание, что в ответ консоль выдала `undefined`. Не стоит волноваться из-за этого — подобный ответ всего лишь означает, что вы пока не определили, что именно должен вернуть (RETURN) вам в качестве ответа интерпретатор JavaScript. Вроде того, что вы бы, например, сказали интерпретатору: «Слушай, запиши-ка вот это, чтобы в будущем мы что-нибудь с этим могли сделать». Интерпретатор послушно исполнил ваше приказание и теперь терпеливо дожидается дальнейших инструкций.

Кстати, если во время перепечатывания кода отсюда в консоль вы опечатались где-то и произошли какие-нибудь странные ошибки и всё такое, то просто обновите страницу Chrome (значок обновления похож на кольцо со стрелочкой; находится в верхней части окна браузера) и попробуйте проделать всё заново. В будущем вы научитесь решать подобные проблемы и другими способами, но пока довольно и этого. Теперь наберём следующий код:

```
let одиночныйПробел = " ";
```

Ответ: `undefined`

Объяснение: наше предложение присвоило переменной `одиночныйПробел` значение строки (содержащей один пробел, заключённый в кавычки). Обратите внимание: *при операции присваивания слева от знака равенства всегда должна*

быть лишь одна переменная. Например, `let x = 2 + 2;` является для JavaScript грамотным предложением, в то время как `2 + 2 = let x;` или, скажем, `2 + 2 = x;` — *нет*.

```
let полноеПриветствие = перваяЧастьПриветствия + одиночныйПробел +
вораяЧастьПриветствия;
```

Ответ: `undefined`

Объяснение: это очередное присваивание. Кстати говоря, вы заметили, что внутри всех переменных — `перваяЧастьПриветствия`, `вораяЧастьПриветствия`, `одиночныйПробел` или `полноеПриветствие` — слова пишутся с прописной буквы? А что между словами нет пробелов? Это вовсе не случайно! В именах переменных не должно быть пробелов, а сами эти имена, по-хорошему, должны быть написаны «верблюжьимРегистром» (**camelCase**). Этот стиль подразумевает, что имя переменной начинается со строчной буквы, а каждое следующее слово (или аббревиатура), входящее в имя переменной, начинается с прописной. Своё название этот стиль получил оттого, что прописные буквы в середине выглядят будто верблюжьи горбы. Если вы пока не поняли — просто продолжайте читать: вы столько раз ещё увидите перед собой «верблюжийРегистр», что в один момент горбы просто появятся, будто всегда там и были.

```
полноеПриветствие = полноеПриветствие + "!!";
```

Ответ: `"Привет, Мир!!"`

Объяснение: в данной операции мы добавляем новую строку (`"!!"`) в конец уже знакомой нам переменной `полноеПриветствие`, присваивая затем всю эту удлинённую строку нашей переменной. Вы заметили отсутствие `let`? Как так вышло? А так вышло оттого, что `let` используется, когда мы создаём (ещё говорят — *объявляем*) новую переменную. В нашем же случае мы не создали никакой новой переменной. Переменная `полноеПриветствие` уже существует, так что для работы с ней нет необходимости прибегать к `let`.

```
полноеПриветствие;
```

Ответ: `"Привет, Мир!!"`

Объяснение: раз уж мы не совершили нового присваивания или ещё чего, компьютеру (да что ж такое! — *интерпретатору*!) уже есть что нам вернуть (RETURN) в ответ! Как если бы вы обратились к интерпретатору со словами: «Слушай, а теперь выдай-ка (RETURN) мне значение той переменной, помнишь?» Соответственно, консоль и выдала вам в ответ нынешнее значение переменной `полноеПриветствие`.