JS, Основы, которые пригодятся

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 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»).

- Полная интеграция с HTML/CSS.
- Простые вещи делаются просто.
- Поддерживается всеми основными браузерами и включён по умолчанию.

Редакторы кода

Термином IDE (Integrated Development Environment, «интегрированная среда разработки») называют мощные редакторы с множеством функций, которые работают в рамках целого проекта. Как видно из названия, это не просто редактор, а нечто большее

Консоль разработчика

Код уязвим для ошибок. И вы, скорее всего, будете делать ошибки в коде… Впрочем, давайте будем откровенны: вы *точно* будете совершать ошибки в коде. В конце концов, вы человек, а не робот.

Но по умолчанию в браузере ошибки не видны. То есть, если что-то пойдёт не так, мы не увидим, что именно сломалось, и не сможем это починить.

Для решения задач такого рода в браузер встроены так называемые «Инструменты разработки» (Developer tools или сокращённо — devtools).

Пробежимся по легкому

1) для чего нужна точка с запятой?
2) как оформляются комментарии в коде?
3) что такое console.log()?

Переменные

Переменная – это «именованное хранилище» для данных. Мы можем использовать переменные для хранения товаров, посетителей и других данных.

Для создания переменной в JavaScript, используйте ключевое слово `let`.

```
let message;
```

Константы

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

Наименование

Несколько хороших правил:

- Используйте легко читаемые имена, такие как `userName` или `shoppingCart`.
- Избегайте использования аббревиатур или коротких имён, таких как `a`, `b`, `c`, за исключением тех случаев, когда вы точно знаете, что так нужно.
- Делайте имена максимально описательными и лаконичными. Примеры плохих имён: `data` и `value`. Такие имена ничего не говорят. Их можно использовать только в том случае, если из контекста кода очевидно, какие данные хранит переменная.
- Договоритесь с вашей командой о используемых терминах. Если посетитель сайта называется «user» тогда мы должны назвать связанные с ним переменные `currentUser` или `newUser` вместо того, чтобы называть их `currentVisitor` или `newManInTown`.

Типы данных

- `number` для любых чисел: целочисленных или чисел с плавающей точкой.
- `string` для строк. Строка может содержать один или больше символов, нет отдельного символьного типа.
- `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.

Условные операторы

Иногда нам нужно выполнить различные действия в зависимости от условий.

Для этого мы можем использовать оператор `if` и условный оператор `?`, который также называют «оператор вопросительный знак».

Оператор «if»

Оператор `if(...)` вычисляет условие в скобках и, если результат `true`, то выполняет блок кода.

```
if (year == 2015) {

  alert( "Правильно!" );

  alert( "Вы такой умный!" );

    }
```

```
if (0) { // 0 is falsy
    ...
}
```

Блок «else»

Оператор `if` может содержать необязательный блок «else» («иначе»). Выполняется, когда условие ложно.

```
let year = prompt('В каком году появилась спецификация ECMAScript-2015?', '');

if (year == 2015) {

  alert( 'Да вы знаток!' );

} else {

  alert( 'А вот и неправильно!' ); // любое значение, кроме 2015

}
```

Условный оператор „?“

Так называемый «условный» оператор «вопросительный знак» позволяет нам сделать это более коротким и простым способом.

Оператор представлен знаком вопроса `?`. Его также называют «тернарный», так как этот оператор, единственный в своём роде, имеет три аргумента.

Синтаксис:

```
let result = условие ? значение1 : значение2;
```

Логические операторы

Оператор «ИЛИ» выглядит как двойной символ вертикальной черты:

```
result = a || b;
```

Оператор И пишется как два амперсанда `&&`:

```
result = a && b;
```

Оператор НЕ представлен восклицательным знаком `!`.

Синтаксис довольно прост:

```
result = !value;
```

Цикл «while»

Код из тела цикла выполняется, пока условие истинно.

```
let i = 0;

while (i < 3) { // выводит 0, затем 1, затем 2

  alert( i );

  i++;

    }
```

Цикл «for»

Давайте разберёмся, что означает каждая часть, на примере. Цикл ниже выполняет `alert(i)` для `i` от `0`до (но не включая) `3`:

```
for (let i = 0; i < 3; i++) { // выведет 0, затем 1, затем 2

  alert(i);


}
```

```
// for (let i = 0; i < 3; i++) alert(i)

// Выполнить начало

let i = 0;

// Если условие == true → Выполнить тело, Выполнить шаг

if (i < 3) { alert(i); i++ }

// Если условие == true → Выполнить тело, Выполнить шаг

if (i < 3) { alert(i); i++ }

// Если условие == true → Выполнить тело, Выполнить шаг

if (i < 3) { alert(i); i++ }

    // ...конец, потому что теперь i == 3
```

Функции

Для создания функций мы можем использовать *объявление функции*.

Пример объявления функции:

```
function showMessage() {

  alert( 'Всем привет!' );


}
```

```
function имя(параметры) {
  ...тело...
}
```

Наша новая функция может быть вызвана по её имени: `showMessage()`.

```
function showMessage() {
  alert( 'Всем привет!' );
}

showMessage();
showMessage();
```

Параметры

Мы можем передать внутрь функции любую информацию, используя параметры (также называемые *аргументы функции*).

```
function showMessage(from, text) { // аргументы: from, text

  alert(from + ': ' + text);

}

showMessage('Аня', 'Привет!'); // Аня: Привет! (*)

showMessage('Аня', "Как дела?"); // Аня: Как дела? (**)
```

Возврат значения

Функция может вернуть результат, который будет передан в вызвавший её код.

Простейшим примером может служить функция сложения двух чисел

```
function sum(a, b) {

  return a + b;

}

let result = sum(1, 2);


alert( result ); // 3
```