Лекция №1
Основы языка JavaScript

План:

1. Ввод и вывод данных
2. Комментарии
3. Переменная
4. Типы данных
5. События

JavaScript – Язык сценариев для придания интерактивности web-страницам.

Язык программирования JavaScript (JS) придает веб-страницам возможность реагировать на действия пользователя и превращать статичные страницы в динамические, так, чтобы страницы буквально "оживали" на глазах.

Программы на этом языке называются ***скриптами***. В браузере они подключаются напрямую к HTML и, как только загружается страничка — тут же выполняются.

Программы на JavaScript — обычный текст. Они не требуют какой-то специальной подготовки.

Есть как минимум *три* замечательных особенности JavaScript:

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

Этих трёх вещей одновременно нет больше ни в одной браузерной технологии.

Поэтому JavaScript и является самым распространённым средством создания браузерных интерфейсов. В этой лекции мы рассмотрим основы языка JavaScript: важнейшие понятия и синтаксис.

Для того, чтобы добавить сценарий на страницу нужно дописать следующий код между тегами

```
<script type="text/javascript" > </script>
```

Ниже приведен пример javascript программы:

```
<html>
<head><title>Примеры</title></head>
<script type="text/javascript">
var x = 5;
var y = x + 3;
alert(y);
</script>
<body>
```

Тело html

```
</body>
</html>
```

Для того, чтобы прикрепить внешний файл с расширением .js на страницу нужно в тег **script** добавить следующий атрибут **src** со значением, в котором прописываем нужный файл:

```
<script type="text/javascript" src="example.js"></script>
```

1.1 Ввод и вывод данных

Метод – фрагмент кода многократного использования, предназначенный для решения общих задач.

В JavaScript используются три стандартных метода для ввода и вывода данных: alert(), prompt() и confirm(). Рассмотрим эти методы браузера подробнее.

1.2.1. alert

Данный метод позволяет выводить диалоговое окно с заданным сообщением и кнопкой ОК. Синтаксис соответствующего выражения имеет следующий вид:

```
alert ("сообщение”)
```

Если ваше сообщение представляет собой определенный набор символов, то его необходимо заключить в двойные или одинарные кавычки. Например, **alert("Hello, World")**

Сообщение представляет собой данные любого типа: последовательность символов, заключенную в кавычки, число (в кавычках или без них), переменную или выражение.

Диалоговое окно, выведенное на экран методом alert(), можно убрать, щелкнув на кнопке ОК. До тех пор пока вы не сделаете этого, переход к ранее открытым окнам невозможен. Окна, обладающие свойством останавливать все последующие действия пользователя и программ, называются ***модальными***. Таким образом, окно, создаваемое посредством alert(), является модальным.

1.2.2. confirm

Метод confirm позволяет вывести диалоговое окно с сообщением и двумя кнопками — ОК и Отмена (Cancel). Этот метод возвращает логическую величину, значение которой зависит от того, на какой из двух кнопок щелкнул пользователь. Если он щелкнул на кнопке ОК, то возвращается значение true; если же он щелкнул на кнопке Отмена, то возвращается значение false. Возвращаемое значение можно обработать в программе. Синтаксис применения метода confirm имеет следующий вид:

```
confirm(сообщение)
```

Если ваше сообщение представляет собой определенный набор символов, то его необходимо заключить в кавычки, двойные или одинарные. Например, confirm("Вы действительно хотите завершить работу?")

Сообщение представляет собой данные любого типа: последовательность символов, заключенную в кавычки, число (в кавычках или без них), переменную или выражение.

Диалоговое окно, выведенное на экран методом confirm(), можно убрать щелчком на любой из двух кнопок — ОК или Отмена. До тех пор пока вы не сделаете этого, переход к ранее открытым окнам невозможен. Окно, создаваемое посредством confirm(), является модальным.

1.2.3. prompt

Метод prompt позволяет вывести на экран диалоговое окно с сообщением, а также с текстовым полем, в которое пользователь может ввести данные (рис. 1.7).

Кроме того, в этом окне предусмотрены две кнопки: ОК и Отмена **(Cancel). В** отличие от методов **alert()** и **confirm()** данный метод принимает два параметра: сообщение и значение, которое должно появиться в текстовом поле ввода данных по умолчанию. Если пользователь щелкнет на кнопке ОК, то метод вернет содержимое поля ввода данных, а если он щелкнет на кнопке Отмена, то возвращается логическое значение false. Возвращаемое значение можно затем обработать в программе. Синтаксис применения метода prompt имеет следующий вид:

```
prompt(сообщение, значение_поля_ввода_данных);
```

Параметры метода prompt() не являются обязательными. Если вы их не укажете, то будет выведено окно без сообщения, а в поле ввода данных подставлено значение по умолчанию — undefined (не определено). Если вы не хотите, чтобы в поле ввода данных появлялось значение по умолчанию, то подставьте в качестве значения второго параметра пустую строку "". Например,

```
prompt("Введите Ваше имя, пожалуйста", "");
```

1.2 Комментарии

Со временем программа становится большой и сложной. Появляется необходимость добавить комментарии, которые объясняют, что происходит и почему.

Комментарии могут находиться в любом месте программы и никак не влияют на ее выполнение. Интерпретатор JavaScript попросту игнорирует их.

Однострочные комментарии начинаются с двойного слэша //. Текст считается комментарием до конца строки:

Многострочные комментарии начинаются слешем-звездочкой "/*" и заканчиваются звездочкой-слэшем "*/".

1.3 Переменная

Переменная JS — это именованная область в памяти, которая хранит в себе данные (значение). К этим данным можно получить доступ, обратившись по имени переменной, в которой они хранятся.

Для объявления или, другими словами, создания переменной используется ключевое слово var:

```
var message;
```

После объявления, можно записать в переменную данные:

```
var message;
message = 'Привет'; // сохраним в переменной строку
```

Всегда определяйте переменные через var. *Это хороший тон в программировании и помогает избежать ошибок.*

Константа — это постоянная величина, которая никогда не меняется. Как правило, их называют большими буквами, через подчёркивание. Например:

```
var COLOR_RED = "#F00";
var COLOR_GREEN = "#0F0";
var COLOR_BLUE = "#00F";
var COLOR_ORANGE = "#FF7F00";
```

Во-первых, константа — это понятное имя, в отличие от строки "#FF7F00".

Технически, константа является обычной переменной, то есть её можно изменить. Но мы договариваемся этого не делать.

Зачем нужны константы?

Во-вторых, опечатка в строке может быть не замечена, а в имени константы её упустить невозможно — будет ошибка при выполнении.

Константы используют вместо строк и цифр, чтобы сделать программу понятнее и избежать ошибок.

Имена переменных

На имя переменной в JavaScript наложено несколько ограничений.

1. Имя может состоять из: букв, цифр, символов $ и _
2. Первый символ не должен быть цифрой.
3. В качестве имён переменных нельзя использовать ключевые слова и зарезервированные слова.

Ключевые слова в JavaScript — это слова, которые существуют в ядре языка JavaScript и встроены в его синтаксис, например слово var.

Зарезервированные слова в JavaScript — это слова, которые пока еще не существуют в ядре языка JavaScript и не встроены в его синтаксис, но в будущем, эти слова могут быть внедрены в ядро JavaScript, например слово abstract.

Например:

```
var myName;
var test123;
```

1.4 Типы данных

Что здесь особенно интересно - доллар '$' и знак подчеркивания '_' являются такими же обычными символами, как буквы.

Прежде чем рассмотреть **типы данных JavaScript**, познакомимся сначала с оператором typeof, он позволяет узнать какой тип данных присвоен переменной, делается это следующим образом:

```
alert(typeof имяПеременной);
```

После этого скрипт должен выдать какое-либо сообщение: number, string, boolean, undefined, object.

1.Число number:

```
var n = 123;
n = 12.345;
```

Единый тип число используется как для целых, так и для дробных чисел.

Существуют специальные числовые значения **Infinity** (бесконечность) и **NaN** (ошибка вычислений). Они также принадлежат типу «число».

Например, бесконечность Infinity получается при делении на ноль:

```
alert( 1 / 0 ); // Infinity
```

Ошибка вычислений NaN будет результатом некорректной математической операции, например:

```
alert( "нечисло" * 2 ); // NaN, ошибка
```

2. Строка string: Ошибка вычислений NaN будет результатом некорректной математической операции, например:

```
var str = "Мама мыла раму";
str = 'Одинарные кавычки тоже подойдут';
```

В JavaScript одинарные и двойные кавычки равноправны. Можно использовать или те или другие.

3. Булевый (логический) тип boolean. У него всего два значения - true (истина) и false (ложь).

Как правило, такой тип используется для хранения значения типа да/нет, например:

```
var checked = true; // поле формы помечено галочкой
checked = false;    // поле формы не содержит галочки
```

4. Null — специальное значение. Оно имеет смысл «ничего». Значение null не относится ни к одному из типов выше, а образует свой отдельный тип, состоящий из единственного значения null:

```
var age = null;
```

В JavaScript **null** – просто специальное значение, которое имеет смысл «ничего» или «значение неизвестно».

В частности, код выше говорит о том, что возраст **age** неизвестен.

5. Undefined — специальное значение, которое, как и null, образует свой собственный тип. Оно имеет смысл «значение не присвоено».

Если переменная объявлена, но в неё ничего не записано, то ее значение как раз и есть undefined:

```
var u;
alert(u); // выведет "undefined"
```

Можно присвоить undefined и в явном виде, хотя это делается редко:

```
var x = 123;
x = undefined;
```

В явном виде undefined обычно не присваивают, так как это противоречит его смыслу. Для записи в переменную «пустого значения» используется null.

6. Объекты object. Первые 5 типов называют *«примитивными»*.

Особняком стоит шестой тип: *«объекты»*. Объект (т. е. член объектного типа данных) представляет собой коллекцию значений (либо элементарных, таких как числа и

строки, либо сложных, например других объектов). К нему относятся, например, даты, он используется для коллекций данных и для многого другого.

Тип данных определяется после того, как переменной или константе было присвоено значение.

1.5 События

События и обработчики событий являются очень важной частью для программирования на языке JavaScript. События, инициируются теми или иными действиями пользователя. Если он щелкает по некоторой кнопке, происходит событие "Click". Если указатель мыши пересекает какую-либо ссылку гипертекста - происходит событие MouseOver. Существует несколько различных типов событий.

Обработчики Событий JavaScript

Событие	Применяется к	Возникает, когда	Обработчик
Blur	окнам и всем элементам формы	пользователь убирает фокус ввода с окна или элемента формы	onBlur
Click	кнопкам, radio-кнопкам, переключателям/checkboxes, кнопкам submit и reset, гиперссылкам	пользователь щёлкает по элементу формы или кнопке	onClick
Focus	окнам и всем элементам формы	пользователь передаёт фокус окну или элементу формы	onFocus
Load	телу документа	пользователь загружает страницу в Navigator	onLoad
MouseOut	областям, гиперссылкам	пользователь перемещает курсор за пределы клиентской карты изображений или гиперссылки	onMouseOut
MouseOver	гиперссылкам	пользователь перемещает курсор над гиперссылкой	onMouseOver
Resize	окнам	пользователь или скрипт изменяет размер окна	onResize
Unload	телу документа	пользователь покидает страницу	onUnload