Лекция 3. Функции и объекты

Ст. преподаватель Еремеев А.А.
YeremeevAA@mpei.ru

Функции

Язык программирования не может обойтись без механизма многократного использования *кода программы*.

1. Функция как тип данных;
2. функция как объект;
3. функция как конструктор объектов.

Функция как тип данных

Функцию определяют при помощи ключевого слова function:

```
function f(arg1,arg2,...)
{
/* тело функции */
}
```

Обратите внимание

1. function определяет *переменную* с именем f. Эта переменная имеет тип function.
2. Эта переменная, как и любая другая, имеет значение - свой исходный текст.

«Синоним» функции

Метод *valueOf()* применим как к числовой переменной i, так и к переменной f, и возвращает их значение. Более того, значение переменной f можно присвоить другой переменной, тем самым создав "синоним" функции f.

Применение «синонима»

Этим приемом удобно пользоваться для сокращения длины кода. Например, если нужно много раз вызвать метод *document.write()*, то можно ввести переменную:

var W = document.write (обратите внимание - без скобок!), а затем вызывать:

W('<H1>Лекция</H1>').

Функция как объект

У любого *типа данных* JavaScript существует *объектовая* "обертка", которая позволяет применять методы типов данных к переменным и литералам, а также получать значения их свойств. Например, длина *строки символов* определяется свойством *length*. Аналогичная "обертка" есть и у функций - это *класс объектов* Function.

Например, увидеть *значение функции* можно не только при помощи метода valueOf(), но и используя метод toString().

Свойства функции как объекта

Свойства функции как объекта доступны программисту только тогда, когда они вызываются внутри этой функции. Наиболее часто используемыми свойствами являются: массив (коллекция) *аргументов функции* (*arguments*[]), его длина (*length*), *имя функции*, вызвавшей данную функцию (*caller*), и *прототип*(*prototype*).

Объекты

Объект - это главный *тип данных* JavaScript. Любой другой *тип данных* имеет объектовую "обертку".

Это означает, что прежде чем можно будет получить *доступ* к значению переменной того или иного типа, происходит конвертирование переменной в *объект*, и только после этого выполняются действия над значением.

Виды объектов

- **клиентские объекты**, входящие в модель DOM.
- **серверные объекты**, отвечающие за взаимодействие клиент-сервер.
  Примеры: Server, Project, Client, File и т.п.
- **встроенные объекты.** Они представляют собой различные типы данных, свойства, методы, присущие самому языку JavaScript, независимо от содержимого HTML-страницы.
- **пользовательские объекты.** Они создаются программистом в процессе написания сценария с использованием конструкторов типа объектов (класса).

Операторы работы с объектами

- **for ... in ...**

Оператор for(переменная in объект) позволяет "пробежаться" по свойствам *объекта*.

- **with**

Оператор with задает объект по умолчанию для *блока операторов*, определенных в его теле.

Клиентские объекты

Для создания механизма управления страницами на клиентской стороне используется *объектная модель документа* (*DOM - Document Object Model*). Суть модели в том, что каждому HTML-контейнеру соответствует ***объект***, который характеризуется тройкой:

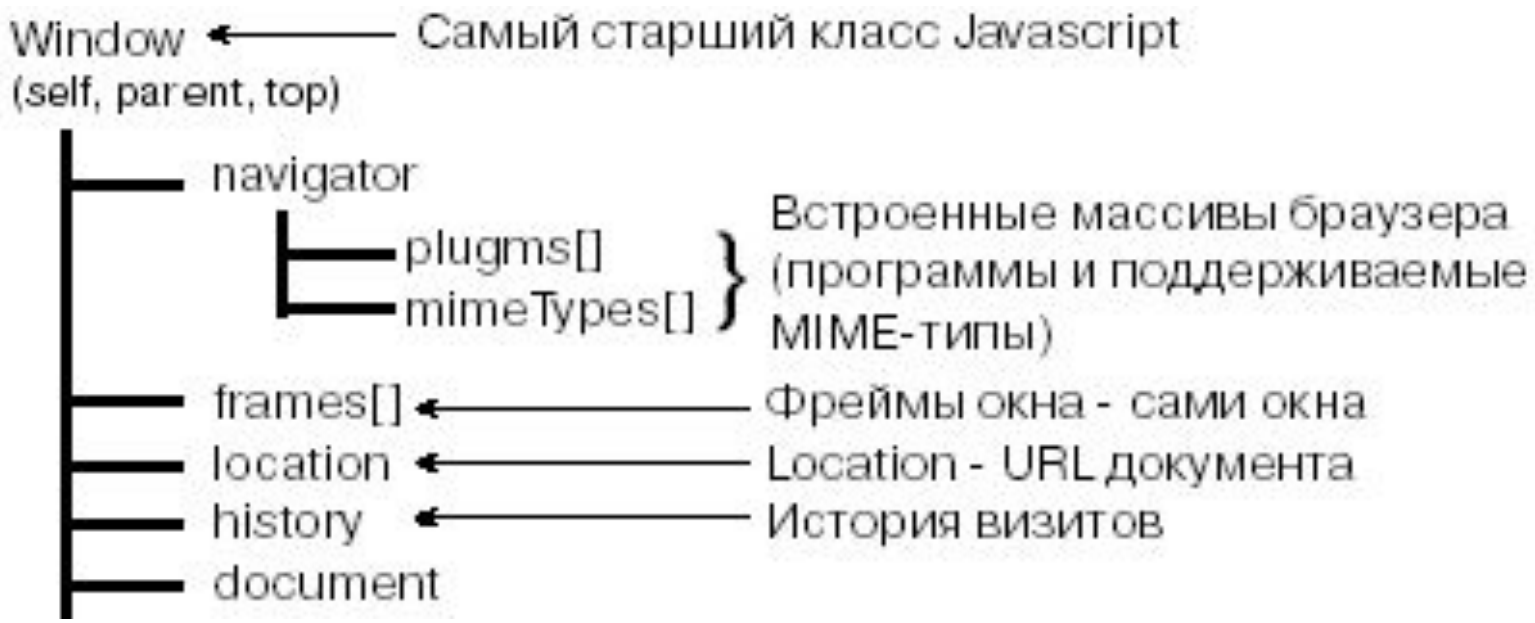
- *свойства;*
- *методы;*
- *события.*

Объектную модель можно представить как способ связи между страницами и браузером.

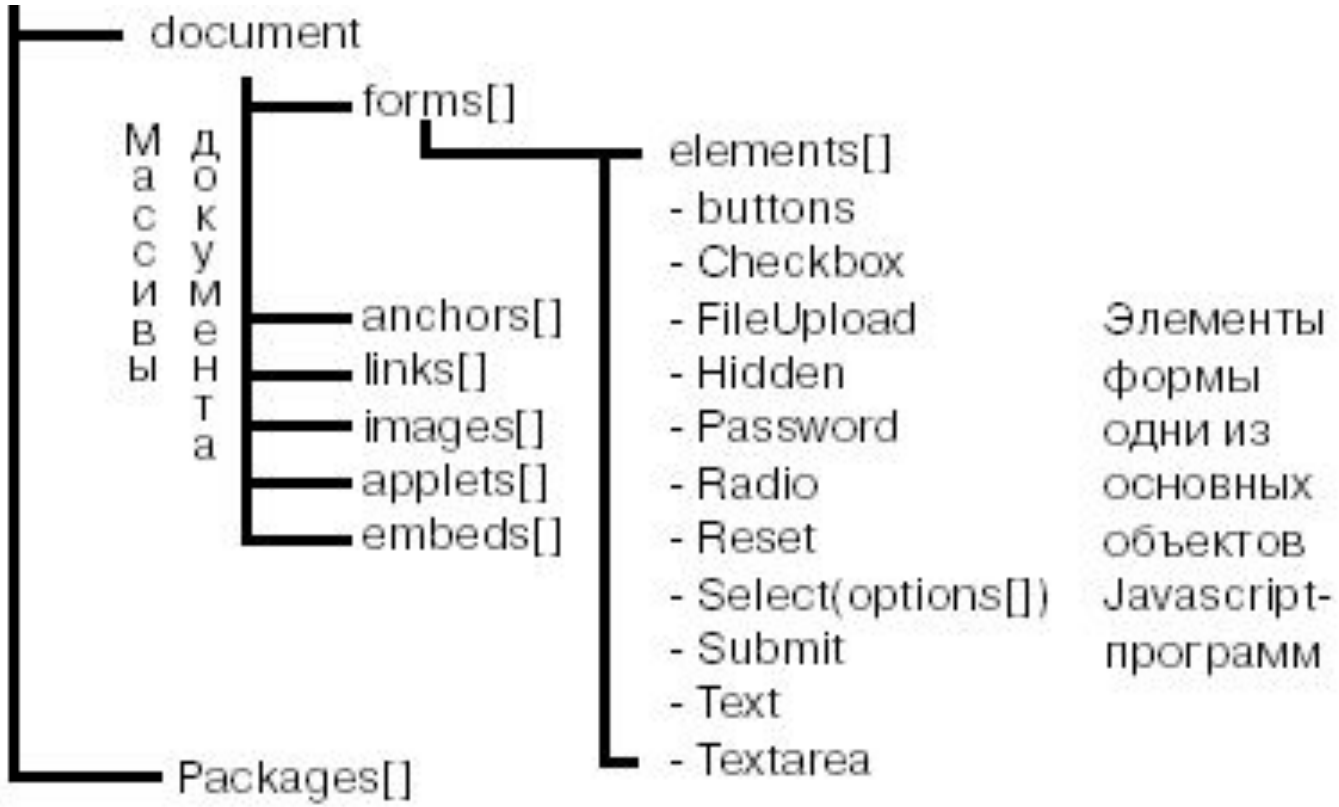
Иерархия классов DOM

Объектно-ориентированный язык программирования предполагает наличие *иерархии классов объектов*. В JavaScript такая иерархия начинается с *класса объектов* window, т.е. каждый объект приписан к тому или иному окну. Для обращения к любому объекту или его свойству указывают полное или частичное имя этого объекта или свойства объекта, начиная с имени объекта, старшего в иерархии, в который входит данный объект.

Фрагмент DOM



Фрагмент DOM (продолжение)



Коллекции

Коллекция - это *структура данных* JavaScript, похожая на массив. Отличие коллекции от массивов заключается в том, что массивы программист создает сам в *коде программы* и заполняет их данными; коллекции же создаются браузером и "населяются" объектами, связанными с элементами Web-страницы.

Основные коллекции в DOM

Коллекция	Описание
window.frames[ ]	Все фреймы - т.е. объекты, отвечающие контейнерам <FRAME>
document.all[ ]	Все объекты, отвечающие контейнерам внутри контейнера <BODY>
document.anchors[ ]	Все якоря - т.е. объекты, отвечающие контейнерам <A>

Основные коллекции в DOM

document.forms[ ]	**Все формы - т.е. объекты, отвечающие контейнерам <FORM>**
document.images[ ]	Все картинки - т.е. объекты, отвечающие контейнерам <IMG>
document.f.elements[ ]	Все элементы формы с именем f - т.е. объекты, отвечающие контейнерам <INPUT> и <SELECT>
document.f.s.options[ ]	Все опции (контейнеры <OPTION>) в контейнере <SELECT NAME=s> в форме <FORM NAME=f>

Свойства

Контейнер якоря <A ...>...</A> имеет атрибут *HREF*, который превращает его в гипертекстовую ссылку:

<A HREF="http://mpei.ru/">МЭИ</A>

Данной *гиперссылке* соответствует *объект* (класса URL) - document.links[0]. Тогда атрибуту *HREF* будет соответствовать свойство *href* этого объекта. К свойству объекта можно обращаться с помощью точечной нотации: *объект.свойство* .

Например:

document.links[0].href='http://ya.ru/';

Свойства

К свойствам можно также обращаться с помощью ***скобочной нотации***: объект['свойство'] .
В нашем примере:

document.links[0]['href']='http://ya.ru/';

У объектов, отвечающих гиперссылкам, есть также свойства, не имеющие аналогов среди атрибутов.

Например, свойство document.links[0].*protocol* в нашем примере будет равно "http:" и т.д.

Методы

В терминологии JavaScript ***методы*** объекта определяют функции, с помощью которых выполняются действия с этим объектом, например, изменение его *свойств*, отображения их на web-странице, *отправка данных* на сервер, перезагрузка страницы и т.п.

Методы. Пример.

Например, если есть ссылка <A *HREF*="http://*mpei*.ru/">*МЭИ*</A>, то у соответствующего ей объекта document.links[0] есть метод *click*(). Его вызов в любом месте JavaScript-программы равносилен тому, как если бы пользователь кликнул по ссылке:

<A HREF="http://mpei.ru/">МЭИ</A>

<SCRIPT> document.links[0].click();
</SCRIPT>

Метод toString()

Для всех объектов определен метод преобразования в *строку символов*: toString(). Например, при сложении числа и строки число будет преобразовано в строку:

“25"+5 = "25"+(5).toString()=“25"+"5"="255"

События

Кроме методов и свойств, объекты характеризуются *событиями*.

Например, с объектом типа *button* может происходить событие *Click*. В качестве значения атрибута OnClick указывается программа *обработки события*, которую должен написать на JavaScript автор HTML-документа:

<INPUT TYPE=button VALUE="Нажать« onClick="alert('Пожалуйста, нажмите еще раз')">

Примеры событий

- нажатие пользователем кнопки в форме;
- установка фокуса в поле формы или увод фокуса из нее;
- изменение введенного в поле значения;
- нажатие кнопки мыши;
- двойной щелчок кнопкой мыши на объекте;
- перемещение указателя мыши;
- выделение текста в *поле ввода* или на странице;
- и другие.