

## Лабораторная работа №2

### Динамика на веб-страницах. Язык программирования JavaScript.

*Цель работы:* ознакомиться с основными возможностями языка JavaScript, синтаксисом, встроенными объектами, событиями DHTML, получить практические навыки программирования на языке JavaScript. Получить практические навыки создания интерактивных веб-страниц.

#### Основные понятия

JavaScript — это язык программирования, основанный на объектах: и языковые средства, и возможности среды, и html-элементы представляются объектами, а сценарий (программа) на JavaScript — это набор взаимодействующих объектов. Программа на языке JavaScript исполняется только в веб-браузере на компьютере клиента, и ей, соответственно, доступны только те объекты, которые в данный момент загружены в текущую вкладку браузера. Т.е. веб-браузер является средой для исполнения программного кода и предоставляет встроенные объекты для манипулирования ими. Разные браузеры обычно имеют некоторый набор стандартных объектов, однако имеются и уникальные свойства для каждой среды. Программный код сценариев реагирует только на определенные события (мыши, клавиатуры, другие действия пользователя) в виде их обработчиков и не требует главной программы.

#### Структура программ на языке JavaScript

Программа на языке JavaScript — это текст, состоящий из операторов, блоков операторов, и комментариев. Операторы могут содержать переменные, константы и выражения.

Скрипты могут существовать как в виде отдельных внешних файлов (\*.js) и подключаться к веб-странице при помощи тега <script>, так и записываться непосредственно между открывающим и закрывающим тегами <script>.

#### Комментарии

Комментарии в JavaScript могут быть однострочными и многострочными. Однострочные комментарии начинаются с символов // и продолжаются до конца текущей строки.

Многострочные комментарии заключаются в маркеры /\* и \*/.

## Переменные

Переменные используются в качестве символических имен, принимающих различные значения. Переменная создается в момент ее декларации. JavaScript позволяет декларировать переменную двумя способами:

- С помощью ключевого слова **var**, например, `var x;` или `var x = 21;`.
- Просто присваиванием переменной значения, например `x = 21;`.

Существуют три области видимости переменных:

- *Глобальный контекст*, т.е. исходный текст сценариев, не включая тела функций.
- *Локальный контекст*, т.е. исходный текст сценариев, являющийся телом функции.
- *Eval-контекст*, т.е. аргумент метода **eval**. Если параметром метода **eval** является строка текста, то она интерпретируется как программа на языке JavaScript, имеющая тот же контекст, в котором был вызван этот метод.

## Типы данных

1. Тип Undefined (неопределенный)
2. Тип Null (нулевой)
3. Тип Boolean (логический)
4. Тип String (строковый)
5. Тип Number (числовой)
6. Тип Object (объектный)

## Операции

### Операции сравнения

*Операции сравнения* сравнивают два операнда и возвращают логическое значение, означающее результат этого сравнения. Строки сравниваются в лексикографическом порядке в кодировке Unicode. Если типы операндов различны, то делается попытка преобразовать их к одному типу.

### Арифметические операции

*Арифметические операции* применяются к числовым операндам и возвращают числовое значение, означающее результат операции. Если типы операндов различны, то делается попытка преобразовать их к числовому типу.

### Битовые операции

*Битовые операции* применяются к числовым операндам, представленным как двоичные числа (т.е. как цепочки из 32 битов), и возвращают числовое значение, означающее результат

операции. Перед выполнением операции операнды преобразуются в целые числа, а результат операции преобразуется в **Number**.

### Логические операции

*Логические операции* применяются к логическим операндам и возвращают логическое значение, означающее результат операции. Если типы операндов различны, то делается попытка преобразовать их к логическому типу.

### Строковые операции

На сегодняшний день JavaScript поддерживает единственную строковую операцию, а именно *конкатенацию строк*, которая обозначается символом "+". Если хотя бы один операнд является строкой, то результатом операции является слияние строк-операндов.

### Условная операция

*Условная операция* — это единственная тернарная операция в JavaScript. Она имеет вид:  
test ? value1 : value2

где test — любое логическое выражение, а value1 и value2 — любые выражения. Если test истинно, то операция возвращает значение value1, в противном случае она возвращает значение value2. Пример:

```
var status = (age >= 18) ? "взрослый" : "подросток";
```

## Основные функциональные блоки

- условный оператор **if...else**;
- оператор выбора **switch**;
- операторы цикла **for**, **while**, **do...while**, **break** и **continue**;
- оператор итерации **for...in**;
- оператор указания объекта **with**;
- операторы обработки исключений **try...catch** и **throw**;

## Функции

Функция в JavaScript — это набор операторов, выполняющих определенную задачу.

Для того, чтобы пользоваться функцией, мы должны сначала ее определить. *Декларация функции* имеет вид:

```
function имя (аргументы) {  
    операторы  
}
```

Здесь *имя* — идентификатор, задающий имя функции, *аргументы* — необязательный список идентификаторов, разделенных запятыми, который содержит имена формальных аргументов функции, а *операторы* — любой набор операторов, который называется телом функции и выполняется при ее вызове.

Допускается и другой способ задания функций:

```
имя = function (аргументы) {  
    операторы  
}
```

Вызов и исполнение такой функции ничем не отличается от предыдущей.

## Объекты

*Объект* JavaScript — это неупорядоченный набор *свойств*. Свойство, являющееся функцией, называется *методом*.

Все объекты, доступные сценарию на языке JavaScript, подразделяются на три группы:

- *встроенные объекты* исполняющей системы (т.е. встроенные в JavaScript);
- *объекты среды*, в которой исполняется сценарий (т. е. обычно это объекты браузера);
- *пользовательские объекты*, создаваемые сценарием в процессе его выполнения.

Примеры создания объектов:

```
var myBrowser = {name: "Microsoft Internet Explorer", version: "5.5"};  
var myBrowser = {name: "Microsoft Internet Explorer", version: "5.5",  
    options: {enableJava: true, enableCookies: false}};
```

```
function Browser(name, version) {  
    this.name = name;  
    this.version = version;  
}
```

## Примитивные встроенные объекты

JavaScript содержит глобальный объект, который является средой его исполняющей системы, а также следующие *встроенные объекты*:

Объект	Описание	Объект	Описание
<b>Array</b>	Массивы	<b>Math</b>	Математические функции и константы
<b>Boolean</b>	Логические объекты	<b>Number</b>	Числовые объекты
<b>Date</b>	Дата и время	<b>Object</b>	Прототип остальных объектов
<b>Error</b>	Исключения	<b>RegExp</b>	Регулярные выражения
<b>Function</b>	Функции	<b>String</b>	Строковые объекты

Глобальный объект (**Global**) создается исполняющей системой JavaScript перед началом исполнения сценария. Это единственный объект, который не имеет имени, и потому доступ к его свойствам и методам осуществляется без имени объекта. По этой причине их иногда называют свойствами и методами верхнего уровня.

Свойства глобального объекта

Свойство	Описание
Infinity	Специальное значение "бесконечность".
NaN	Специальное значение "не число".
undefined	Неопределенное значение.

Методы глобального объекта

Метод	Описание
escape	Преобразует строку в шестнадцатеричную кодировку Unicode.
eval	Исполняет строку кода JavaScript.
isFinite	Возвращает <b>true</b> , если аргумент является конечным числом.
isNaN	Возвращает <b>true</b> , если аргумент равен <b>NaN</b> .
parseFloat	Преобразует строку в плавающее число.
parseInt	Преобразует строку в целое число.
unescape	Преобразует шестнадцатеричную кодировку Unicode в строку.

## Строки

Свойства объекта String

Свойство	Описание
length	Количество символов в строке.

Стандартные методы объекта String

Метод	Описание
charAt	Возвращает символ, находящийся в данной позиции строки.
charCodeAt	Возвращает код символа, находящегося в данной позиции строки.
concat	Возвращает конкатенацию строк.
indexOf	Возвращает позицию первого вхождения заданной подстроки.
lastIndexOf	Возвращает позицию последнего вхождения заданной подстроки.
localeCompare	Сравнивает две строки с учетом языка операционной системы.
match	Сопоставляет строку с регулярным выражением.
replace	Сопоставляет строку с регулярным выражением и заменяет найденную подстроку новой подстрокой.
search	Ищет сопоставление строки с регулярным выражением.
slice	Извлекает часть строки и возвращает новую строку.
split	Разбивает строку на массив подстрок.
substr	Возвращает подстроку, заданную позицией и длиной.
substring	Возвращает подстроку, заданную начальной и конечной позициями.
toLocaleLowerCase	Преобразует все буквы строки в строчные с учетом языка операционной системы.

toLocaleUpperCase	Преобразует все буквы строки в прописные с учетом языка операционной системы.
toLowerCase	Преобразует все буквы строки в строчные.
toString	Преобразует объект в строку.
toUpperCase	Преобразует все буквы строки в прописные.
valueOf	Возвращает примитивное значение объекта.

### Массивы: встроенный объект Array

Объект **Array** используется для создания *массивов*, т. е. упорядоченных наборов элементов любого типа. Доступ к элементу массива производится по его номеру в массиве, называемому *индексом элемента*; обозначается *i*-й элемент массива *a* как *a[i]*. Элементы массива нумеруются с нуля, т. е. массив *a*, состоящий из *N* элементов, содержит элементы *a[0]*, *a[1]*, ..., *a[N-1]*.

Для создания массивов используются следующие *конструкторы массивов*:

```
new Array()
new Array(размер)
new Array(элемент0, элемент1, ..., элементN)
```

Здесь *размер* — любое числовое выражение, задающее количество элементов в массиве; *элемент0*, *элемент1*, ..., *элементN* — любые выражения.

#### Примеры:

```
var a = new Array(5);           // массив из 5 элементов
var b = new Array("строка");   // массив из 1 элемента "строка"
var c = new Array(1, 2, 3);    // массив из 3 элементов: 1, 2 и 3
var d = ["1", "2", "3"];       // то же самое
```

Мы можем неявно увеличить размер массива, присвоив значение элементу *c* несуществующим индексом, например:

```
var colors = new Array();      // пустой массив
colors[99] = "пурпурный";     // размер массива стал равен 100
```

#### Методы объекта Array

Метод	Описание
concat	Объединяет два массива в один новый и возвращает его.
join	Объединяет все элементы массива в текстовую строку.
pop	Удаляет последний элемент массива.
push	Добавляет элементы в конец массива.
reverse	Изменяет порядок элементов массива на противоположный.
shift	Удаляет первый элемент массива и возвращает его.
slice	Извлекает часть массива и возвращает новый массив.
sort	Сортирует элементы массива.
splice	Заменяет часть массива.

toLocaleString	Преобразует массив в строку с учетом формата операционной системы.
toString	Преобразует массив в строку.
unshift	Добавляет элементы в начало массива.
valueOf	Возвращает примитивное значение массива.

## Дата и время: встроенный объект Date

Объект **Date** предназначен для манипуляций с датами и временами. Его примитивным значением является число, равное количеству миллисекунд относительно *базового времени*, равного полуночи 1 января 1970 г. по Гринвичскому меридиану (UTC, Universal Coordinated Time). Если это значение равно **NaN**, то оно считается неопределенным.

Для создания объектов Date используются следующие конструкторы:

```
new Date()
new Date(число)
new Date(строка)
new Date(год, месяц, день [, часы [, минуты [, секунды [, мс] ] ] ] ] ] )
```

Здесь:

- *число* — числовое выражение, задающее примитивное значение объекта в миллисекундах;
- *строка* — строковое выражение, задающее дату и время в формате, описанном в методе **parse**;
- *год* — числовое выражение, задающее полный номер года (например, 1988, а не 88);
- *месяц* — числовое выражение, задающее номер месяца (0 = январь, 1 = февраль, ..., 11 = декабрь);
- *день* — числовое выражение, задающее номер дня в месяце от 1 до 31;
- *часы* — необязательное числовое выражение, задающее номер часа от 0 до 23;
- *минуты* — необязательное числовое выражение, задающее номер минуты от 0 до 59;
- *секунды* — необязательное числовое выражение, задающее номер секунды от 0 до 59;
- *мс* — необязательное числовое выражение, задающее номер миллисекунды от 0 до 999.

Примеры:

```
var a = new Date(); // текущие дата и время
var b = new Date("May 21, 1958 10:15 AM"); // заданные дата и время
var c = new Date(1958, 4, 21, 10, 15); // то же самое в другом формате
```

Методы объекта Date

Метод	Описание
getDate	Возвращает день месяца по местному времени.
getDay	Возвращает день недели по местному времени.
getFullYear	Возвращает полный номер года по местному времени.
getHours	Возвращает часы по местному времени.
getMilliseconds	Возвращает миллисекунды по местному времени.
getMinutes	Возвращает минуты по местному времени.
getMonth	Возвращает месяц по местному времени.

getSeconds	Возвращает секунды по местному времени.
getTime	Возвращает примитивное значение объекта.
getTimezoneOffset	Возвращает разницу в минутах между местным временем и UTC.
getUTCDate	Возвращает день месяца по времени UTC.
getUTCDay	Возвращает день недели по времени UTC.
getUTCFullYear	Возвращает полный номер года по времени UTC.
getUTCHours	Возвращает часы по времени UTC.
getUTCMilliseconds	Возвращает миллисекунды по времени UTC.
getUTCMinutes	Возвращает минуты по времени UTC.
getUTCMonth	Возвращает месяц по времени UTC.
getUTCSeconds	Возвращает секунды по времени UTC.
getVarDate	Возвращает примитивное значение объекта в формате VT_DATE.
getYear	Возвращает номер года по местному времени.
setDate	Устанавливает день месяца по местному времени.
setFullYear	Устанавливает полный номер года по местному времени.
setHours	Устанавливает часы по местному времени.
setMilliseconds	Устанавливает миллисекунды по местному времени.
setMinutes	Устанавливает минуты по местному времени.
setMonth	Устанавливает месяц по местному времени.
setSeconds	Устанавливает секунды по местному времени.
setTime	Устанавливает примитивное значение объекта.
setUTCDate	Устанавливает день месяца по времени UTC.
setUTCFullYear	Устанавливает полный номер года по времени UTC.
setUTCHours	Устанавливает часы по времени UTC.
setUTCMilliseconds	Устанавливает миллисекунды по времени UTC.
setUTCMinutes	Устанавливает минуты по времени UTC.
setUTCMonth	Устанавливает месяц по времени UTC.
setUTCSeconds	Устанавливает секунды по времени UTC.
setYear	Устанавливает номер года по местному времени.
toDateString	Преобразует примитивное значение объекта в строку даты.
toGMTString	Преобразует примитивное значение объекта в строку даты и времени по Гринвичскому меридиану.
toLocaleDateString	Преобразует примитивное значение объекта в строку даты в формате операционной системы.
toLocaleString	Преобразует примитивное значение объекта в строку даты и времени в формате операционной системы.
toLocaleTimeString	Преобразует примитивное значение объекта в строку времени в формате операционной системы.
toString	Преобразует примитивное значение объекта в строку.
toTimeString	Преобразует примитивное значение объекта в строку времени.
toUTCString	Преобразует примитивное значение объекта в строку даты и времени по UTC.
UTC	Возвращает примитивное значение объекта по времени UTC.
valueOf	Возвращает примитивное значение объекта.

## Математические функции и константы: объект Math

Объект **Math** обеспечивает доступ к различным математическим константам и функциям. Он существует в единственном экземпляре и потому не имеет конструктора. Соответственно все его свойства и методы являются статическими и должны вызываться обращением к объекту **Math**, а не его реализациям. Прототипа объект **Math** не имеет.

Свойства объекта Math

Свойство	Описание
E	Основание натуральных логарифмов e.
LN10	Число $\ln 10$ .
LN2	Число $\ln 2$ .
LOG10E	Число $\lg e$ .
LOG2E	Число $\log_2 e$ .
PI	Число $\pi$ .
SQRT1_2	Квадратный корень из $1/2$ .
SQRT2	Квадратный корень из 2.

Методы объекта Math

Метод	Описание
abs	Возвращает абсолютную величину аргумента.
acos	Возвращает арккосинус аргумента.
asin	Возвращает арксинус аргумента.
atan	Возвращает арктангенс аргумента.
atan2	Возвращает арктангенс частного от деления аргументов.
ceil	Возвращает наименьшее целое число, большее или равное аргументу.
cos	Возвращает косинус аргумента.
exp	Возвращает экспоненту аргумента.
floor	Возвращает наибольшее целое число, меньшее или равное аргументу.
log	Возвращает натуральный логарифм аргумента.
max	Возвращает наибольший из аргументов.
min	Возвращает наименьший из аргументов.
pow	Возводит первый аргумент в степень, заданную вторым.
random	Генерирует случайное число в диапазоне от 0 до 1.
round	Округляет аргумент до ближайшего целого числа.
sin	Возвращает синус аргумента.
sqrt	Возвращает квадратный корень из аргумента.
tan	Возвращает тангенс аргумента.

## DHTML

Dynamic HTML — это *набор технологий*, работающих на стороне клиента и призванных преодолеть статичность традиционных Веб-страниц. Точнее говоря, это технологии, которые обеспечивают:

- динамическое формирование Веб-страницы в процессе ее загрузки, и
- динамическое изменение Веб-страницы в ответ на действия пользователя.

Для достижения перечисленных целей используются следующие методы:

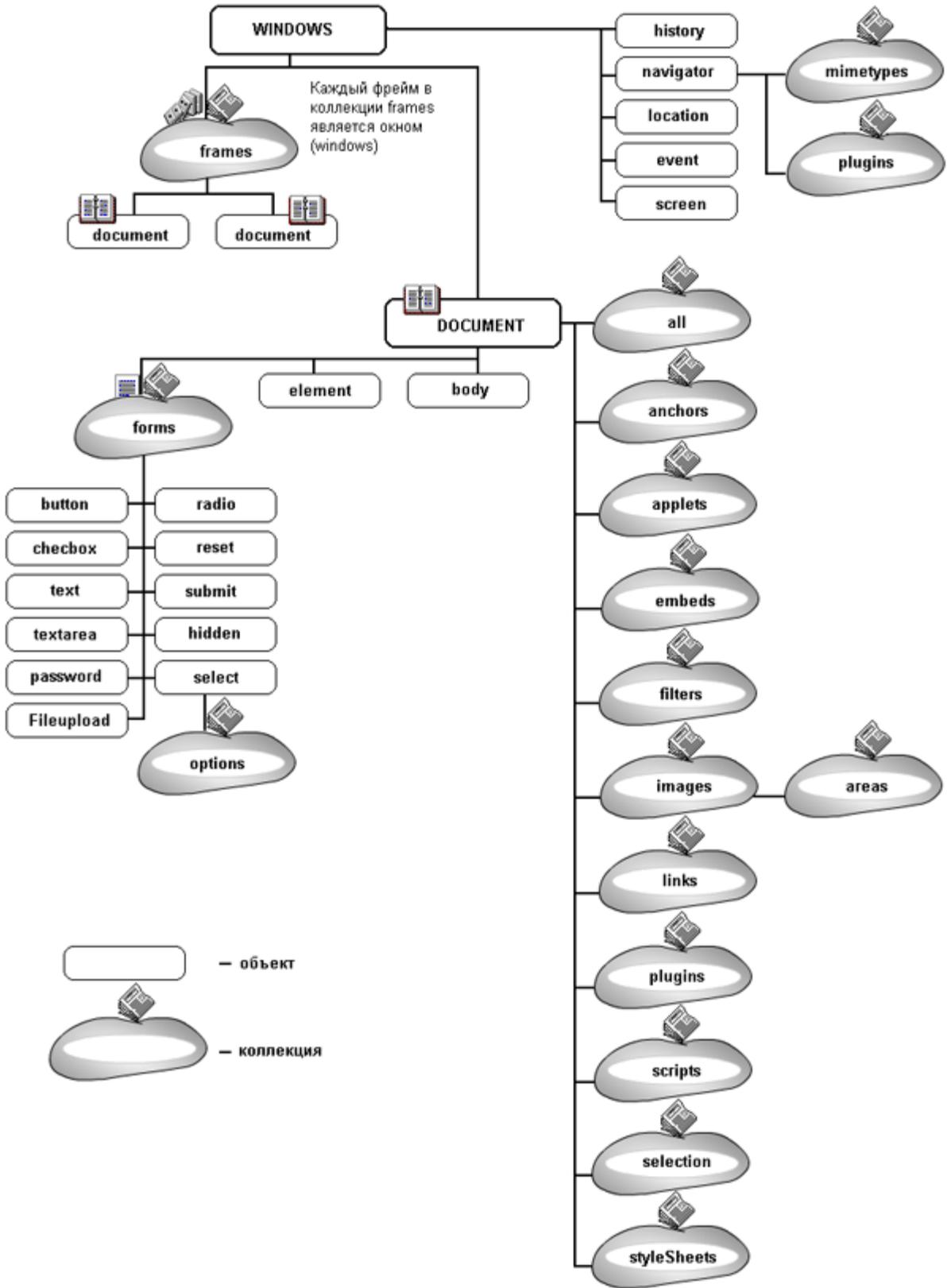
- динамическое изменение атрибутов и стилей элементов, составляющих HTML-документ;
- динамическое извлечение данных из внешних источников и включение их в Веб-страницу;
- использование динамически загружаемых шрифтов;
- поддержка визуальных и мультимедийных эффектов при отображении страниц;
- механизмы сохранения информации на компьютере-клиенте между сессиями работы.

### **Объектная модель документа**

**DOM** (от англ. *Document Object Model* — «объектная модель документа») — это независимый от платформы и языка программный интерфейс, позволяющий программам и скриптам получить доступ к содержимому документов, а также изменять содержимое, структуру и оформление документов.

Модель DOM не накладывает ограничений на структуру документа. Любой документ известной структуры с помощью DOM может быть представлен в виде дерева узлов, каждый узел которого представляет собой элемент, атрибут, текстовый, графический или любой другой объект. Узлы связаны между собой отношениями родительский-дочерний.

Изначально различные браузеры имели собственные модели документов (DOM), не совместимые с остальными. Для того, чтобы обеспечить взаимную и обратную совместимость, специалисты международного консорциума W3C классифицировали эту модель по уровням, для каждого из которых была создана своя спецификация. Все эти спецификации объединены в общую группу, носящую название W3C DOM.

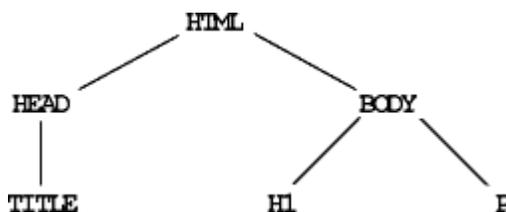


## Базовая объектная модель документа

Каждый элемент дерева соответствует элементу HTML и, следовательно, имеет тег(и), содержимое и набор атрибутов. Для перехода к объектной модели документа остается сделать единственный шаг: назвать все элементы дерева объектами, а их атрибуты сделать доступными для чтения и для изменения из сценариев. В результате дерево элементов HTML-документа становится динамически управляемым; более того, теперь мы можем легко добавлять к каждому элементу новые свойства, помимо стандартных атрибутов HTML.

Именно такой подход был положен в основу динамической модели HTML, а затем принят за основу стандартов W3C, получивших название *объектная модель документа* (Document Object Model или DOM). При этом W3C расширил понятие DOM на любые XML-документы, рассматривая HTML DOM как специализированный частный случай с дополнительными возможностями. Таким образом, DOM — это модель HTML- и XML-документов, независимая от платформы и языка программирования, которая определяет:

- интерфейсы и объекты, которые используются для представления документа и манипулирования им;
- семантику этих интерфейсов и объектов, включая их атрибуты и реакцию на события;
- взаимосвязи между этими интерфейсами и объектами.



Каждый элемент представляет собой узел.

Свойства узлов

Свойство	Изменяемое	Описание
attributes	Нет	Атрибуты узла.
childNodes	Нет	Список детей.
firstChild	Нет	Первый ребенок.
lastChild	Нет	Последний ребенок.
localName	Нет	Локальное имя в пространстве имен.
namespaceURI	Нет	URI пространства имен.
nextSibling	Нет	Следующий узел дерева.
nodeName	Нет	Имя узла.
nodeType	Нет	Тип узла.
nodeValue	Да	Значение узла.
ownerDocument	Нет	Документ — владелец узла.
parentNode	Нет	Отец данного узла.

prefix	Да	Префикс пространства имен.
previousSibling	Нет	Предыдущий узел дерева.

Методы узлов для работы с узлами

Свойство	Описание
appendChild	Добавляет сына в конец списка детей.
cloneNode	Создает копию данного узла.
hasAttributes	Проверяет наличие у узла атрибутов.
childNodes	Проверяет наличие у узла детей.
insertBefore	Вставляет новый узел перед заданным сыном.
isSupported	Проверяет свойство реализации DOM.
normalize	Нормализует текстовое содержимое узла.
removeChild	Удаляет заданного сына.
replaceChild	Заменяет заданного сына новым узлом.

Методы узлов для работы с атрибутами

Метод	Описание
getAttribute	Возвращает значение заданного атрибута.
getAttributeNS	Возвращает значение заданного атрибута с учетом пространства имен.
getAttributeNode	Возвращает заданный узел Attr.
getAttributeNodeNS	Возвращает заданный узел Attr с учетом пространства имен.
getElementsByTagName	Возвращает список потомков, имеющих заданный тег.
getElementsByTagNameNS	Возвращает список потомков, имеющих заданный тег, с учетом пространства имен.
hasAttribute	Проверяет наличие заданного атрибута.
hasAttributeNS	Проверяет наличие заданного атрибута с учетом пространства имен.
removeAttribute	Удаляет заданный атрибут.
removeAttributeNS	Удаляет заданный атрибут с учетом пространства имен.
removeAttributeNode	Удаляет заданный узел Attr.
setAttribute	Добавляет новый атрибут.
setAttributeNS	Добавляет новый атрибут с учетом пространства имен.
setAttributeNode	Добавляет новый узел Attr.
setAttributeNodeNS	Добавляет новый узел Attr с учетом пространства имен.

## События DHTML

### onBeforeCopy

Наступает перед копированием данных из текущего элемента страницы в буфер обмена Windows. Событие onBeforeCopy() можно использовать, чтобы разрешить или запретить пункт контекстного меню Скопировать. Для этого достаточно присвоить свойству returnValue объекта event значение false. Значение false разрешает, т.к. в этом случае мы отменяем поведение меню по умолчанию.

### **onBeforePaste**

Наступает непосредственно перед вставкой данных из буфера обмена в текущий элемент страницы.

Событие `onBeforePaste()` можно использовать, чтобы разрешить или запретить пункт контекстного меню Вставить.

Для этого достаточно присвоить свойству `returnValue` объекта `event` значение `false`. Значение `false` разрешает, т.к. в этом случае мы отменяем поведение меню по умолчанию.

### **onBlur**

Событие происходит при переходе фокуса с этого элемента с помощью курсора мышки или последовательности перехода.

### **onChange**

Событие происходит при потере управляющим элементом фокуса ввода, если его значение было изменено с момента получения фокуса. Элемент `<input>`, `<select>`, `<textarea>`.

### **onClick**

Событие происходит при однократном щелчке левой кнопки мышки на элементе, при нажатии клавиши `<Enter>` на форме, использовании клавиши-ускорителя или выборе пункта в списке.

### **onContextMenu**

Наступает при щелчке правой кнопкой мыши по элементу страницы, перед выводом контекстного меню.

### **onCopy**

Наступает при копировании данных из текущего элемента страницы в буфер обмена Windows.

### **onDoubleClick**

Событие происходит при двойном щелчке левой кнопкой мышки на элементе.

### **onDrag**

Наступает во время операции `drag-n-drop` в элементе-источнике.

### **onDragDrop**

Событие наступает, когда пользователь "бросает" в окно Web-обозревателя ссылки на интернет-адреса.

**onDragEnd**

Событие наступает, когда пользователь отпускает кнопку мыши, завершая операцию drag-n-drop, в элементе-источнике.

**onDragEnter**

Событие наступает, когда пользователь перетаскивает данные в допустимое место, в элемент-цели.

**onDragLeave**

Событие наступает, когда пользователь перетаскивает данные из допустимого места, в элементе-цели.

**onDragOver**

Событие наступает во время операции drag-n-drop в элементе-цели, когда пользователь перетаскивает данные над допустимым местом.

**onDragStart**

Событие наступает, когда пользователь начинает перетаскивать данные, в элементе-источнике.

**onDrop**

Событие наступает, когда пользователь отпускает кнопку мыши, завершая операцию drag-n-drop, в элементе-цели.

**onFocus**

Событие происходит при получении элементом фокуса с помощью мышки или последовательного перехода.

**onHelp**

Возникает при нажатии клавиши F1.

**onKeyDown**

Событие происходит при нажатии клавиши на клавиатуре.

**onKeyPress**

Событие происходит при нажатии и отпускании клавиши на клавиатуре.

**onKeyUp**

Событие происходит при отпускании клавиши на клавиатуре.

**onMouseDown**

Событие происходит при нажатии кнопки мышки на элементе.

#### **onMouseEnter**

Событие наступает, когда пользователь помещает курсор мыши на элемент страницы.

#### **onMouseLeave**

Событие наступает, когда пользователь убирает курсор мыши с элемента страницы.

#### **onMouseMove**

Событие наступает, когда пользователь перемещает курсор мыши над элементом страницы.

#### **onMouseOut**

Событие происходит при перемещении курсора мышки за пределы элемента.

#### **onMouseOver**

Событие происходит при наведении курсора мыши на элемент.

#### **onMouseUp**

Событие происходит при отпускании кнопки мышки на элементе.

#### **onMove**

Событие наступает, когда пользователь перемещает окно Web-обозревателя или изменяет размера фрейма.

#### **onPaste**

Событие наступает при вставке данных из буфера обмена Windows в текущий элемент страницы.

#### **onReset**

Событие происходит при сбросе формы.

#### **onResize**

Возникает при изменении размеров окна, фрейма или другого элемента страницы.

#### **onResizeEnd**

Событие наступает по окончании изменения пользователем размеров элемента страницы, имеющего собственный пользовательский интерфейс.

#### **onResizeStart**

Событие наступает, когда пользователь начинает изменять размеры элемента страницы, имеющего собственный пользовательский интерфейс.

### **onScroll**

Возникает при скроллинге содержимого окна.

### **onSelect**

Событие происходит при выделении некоторого текста в текстовом поле.

### **onSelection**

Событие наступает, когда пользователь выделяет фрагмент содержимого страницы или поля ввода.

### **onSelectionChange**

Событие наступает, когда меняется тип выделения.

### **onSelectStart**

Событие наступает, когда пользователь начинает выделять фрагмент содержимого элемента страницы.

### **onSubmit**

Событие происходит при отправке формы.

### **onUnload**

Событие происходит, когда "браузер" удаляет документ из окна или фрейма.

### **Задание к лабораторной работе:**

1. Изменить web-сайт, разработанный в лабораторной работе 1, добавив динамические эффекты на страницах при помощи средств JavaScript и DHTML.
2. Украсить свой сайт различными эффектами (параметры текста, замена изображения, динамическое изменение картинки, изменения прозрачности и других css-свойств, всплывающие подсказки и т.д.) по наступлению различных событий.
3. Сделать выпадающее меню. Допускается использование готовых решений.
4. *Произвести проверку данных, введенных в поля формы (на заполнение, на правильность email, на ввод только числового значения и т.д.) средствами JavaScript, не используя новые атрибуты html5 .Применить регулярные выражения.*
5. *В задании обязательно использовать таймер (осуществлять загрузку страницы через некоторый промежуток времени, часы в тексте и т.д.), встроенные объекты Date, Math, String и др.*
6. Для реализации динамических эффектов создать пользовательские функции во внешних файлах скриптов.
7. В задании использовать 5-10 различных событий, сделав на каждое из них отдельный обработчик.
8. Использовать как внутренние (инлайновые) скрипты, так и внешние.
9. Страницы должны быть работоспособны в любом браузере.

### **Каждый отчет должен содержать:**

1. Заголовок лабораторной работы (название и цель работы).
2. Фамилия, инициалы и группа студента.
3. Задание к лабораторной работе.
4. Краткие теоретические сведения.
5. Описание алгоритмов, функций, примененных решений.
6. Экранные формы разработанных страниц.
7. Основные тексты страниц.
8. Выводы по сделанной работе.