**Программное обеспечение современных ЭВМ**

# Web-программирование

**Конспект лекций**

---

## Структура курса

### Цель курса

Приобретение студентами теоретических знаний и практических навыков в области Web-программирования, изучение технологий HTML, CSS, JavaScript, PHP, MySQL для разработки Web-приложений и Web-интерфейсов к базам данных.

### Задачи курса

1. Обзор современных технологий web-программирования.
2. Изучение web-технологий HTML, CSS, JavaScript программирования на стороне клиента.
3. Изучение web-технологий PHP, MySQL программирования на стороне сервера.

### Структура учебного плана

- Количество лекционных занятий: 36 академических часа.
- Количество лабораторных занятий: 18 академических часов.
- Индивидуальная работа: 72 академических часа.
- 3 контрольные работы.
- Экзамен.

### План лекций

Таблица 1. План лекций

| № | Тема | | Вопросы |
|---|---|---|---|
| 1 | Язык гипертекстовой разметки HTML | Предмет Web-программирования. Архитектура WWW. Обзор Web-технологий. | Предмет Web-программирования: основные понятия и определения; сдерживающие факторы развития web-технологий. Архитектура WWW: клиент/серверная архитектура Интернет. Обзор Web-технологий: клиентские Web-технологий; серверные Web-технологий; веб стандарты. |
| 2 | | Введение в HTML. Структура HTML документа. | Введение в HTML: основные понятия и определения; инструменты и технологии программирования. Структура HTML документа: структура документа; структура и параметры тегов. |
| 3 | | Форматирование текста. | Форматирование текста: представление текстовой информации; Escape-последовательности; комментарии; организация списков. |
| 4 | | Ссылки. Графика. | Ссылки. Графика: гиперссылки; представление графической информации; карты изображений. |
| 5 | | Таблицы в HTML. Табличная верстка. | Таблицы в HTML: организация таблиц, параметры таблиц. Табличная верстка: построение модульной сетки при помощи таблиц; типовые модульные сетки HTML документа. |
| 6 | | Интерактивные формы HTML. Фреймы. | Формы HTML: организация формы; основные параметры формы; стандартные элементы управления. Фреймы: представление HTML документа в виде фреймов; основные параметры фреймов; типовые |

| | | | |
|---|---|---|---|
| | | | структуры HTML документа с использованием фреймов. |
| 7 | | Каскадные таблицы стилей CSS. Форматирование блоков. Форматирование текста. | Каскадные таблицы стилей CSS: основные понятия и определения; методы подключения таблиц стилей к HTML документам. Форматирование блоков: свойства блоков. Форматирование текста: свойства текста. |
| 8 | | Слои. CSS верстка. Контрольная работа 1 | Слои: основные понятия; область применения; параметры слоев. CSS верстка: принципы верстки при помощи слоев; построение модульной сетки при помощи слоев; типовые модульные сетки HTML документа. |
| 9 | Язык клиентских сценариев JavaScript | Введение в JavaSript. Объектная модель. Синтаксис языка JavaScript. | Введение в JavaSript: основные понятия и определения;. Структура HTML документа: методы подключения JavaSript к HTML документам. Объектная модель: модель DOM. Синтаксис языка JavaScript. |
| 10 | | Типы данных. Операторы JavaScript | Типы данных, локальные и глобальные переменные, массивы, арифметические и логические операции, циклы и управляющие структуры. |
| 11 | | Обработка событий. Примеры эффективного программирования на JS. | Обработка событий. Примеры эффективного программирования на JS. |
| 12 | | Основы DHTML. Контрольная работа 2 | Основы DHTML: принципы технологии DHTML; примеры использования DHTML. |
| 13 | Язык серверных сценариев PHP | Введение в PHP. Основы программирования на стороне сервера. | Введение в PHP: основные понятия и определения; методы подключения php к HTML документам. Программирование на стороне сервера: протокол http; передача параметров серверу; запоминание состояния. |
| 14 | | Синтаксис языка php. Операторы PHP. | Синтаксис языка php: типы данных, локальные и глобальные переменные, массивы, арифметические и логические операции. Операторы PHP: циклы и управляющие структуры. |
| 15 | | Примеры эффективного программирования на PHP. | Примеры эффективного программирования на PHP: передача данных по HTTP протоколу; обработка форм. |
| 16 | | Введение в Интернет базы данных. Характеристики MySQL. | Принцип работы Интернет базы данных. Характеристики MySQL. Интерфейс базы данных MySQL с PHP. |
| 17 | | Типовые примеры работы с базами данных MeSQL. | Типовые примеры работы с базами данных MeSQL: организация доступа к данням; чтение, изменение, удаление, добавление даннях в базу. |
| 18 | | Специальные вопросы. Перспективы развития. Контрольная работа 3 | |

**Литература. Учебные материалы**

Обновленный список рекомендованной литературы для изучения курса доступен по адресу http://www.edu.cassiopeia.com.ua

**Лекция 1**

# 1.1. Предмет Web-программирования

### 1.1.1. Основные понятия и определения.

*World Wide Web (Web)* — это сеть информационных ресурсов. Важнейшими механизмами этой сети являются:

- Единая схема наименования для поиска ресурсов в Web. Например, *URL* (Uniform Resource Locator) – метод разметки ресурсов. Он состоит из названия протокола, двоеточия, двух косых черт «/», названия машины и пути к ресурсу. Например, http://www.edu.cassiopeia.com.ua/.

- Протоколы для доступа к именованным ресурсам через Web. Например, *HTTP* (Hyper Text Transfer Protocol) – протокол передачи гипертекста).

Гипертекст предложен для простого перемещения по ресурсам (например, HTML - Hyper Text Markup Language). *Гипертекст* - документ, содержащий гиперссылки (ссылки на другие документы).

*Web-программирование* - это отдельное направление в программировании, используемое для создания web-приложений. В связи с ускоренными темпами развития Интернет технологий постоянно появляются новые направления в этой области. Поэтому Web-программирование не сформировавшаяся наука, а скорее набор существующих программных технологий (клиентских и серверных), используемых для организации работы пользователя в сети Интернет.

*Сервер* – компьютер, на котором находятся документы и приложения.

*Клиент* – рабочая станция, используемая для просмотра документов и приложений, хранящихся на сервере.

### 1.1.2. Сдерживающие факторы развития web-технологий.

- ограничения, связанные с сетью Интернет (скорость);
- отсутствие централизованного управления Интернет (конкуренция браузеров);
- ограничения, связанные с технологиями программирования.

# 1.2. Архитектура WWW

### 1.2.1. Клиент/серверная архитектура Интернет.

В простейшем случае структуру Интернет можно представить следующей схемой:

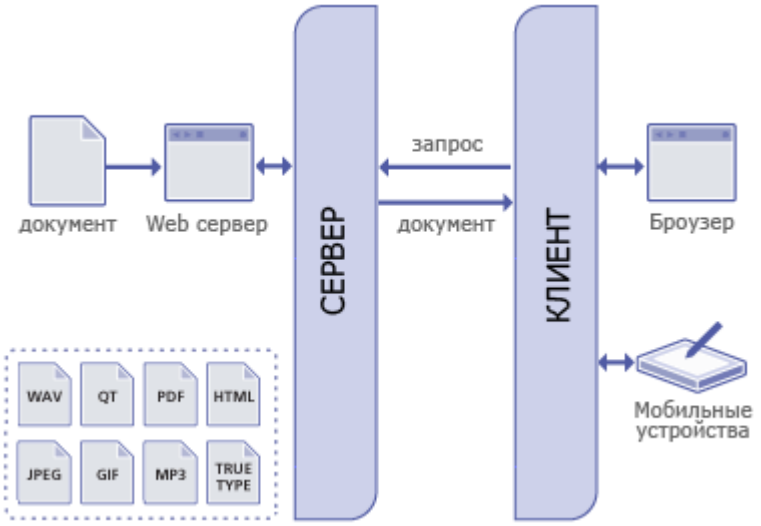


Рис.1. Клиент/серверная архитектура Интернет.

Работа такой системы осуществляется в следующей последовательности:

1. Браузер посылает запрос документа и служебную информацию на сервер.
2. Сервер находит и отправляет запрошенный документ обратно.
3. Браузер получает запрошенный документ.

Основными недостатками такой архитектуры являются:

1. Возможность просмотра только статических документами (отсутствие интерактивности).
2. Затруднения в работе с большими объемами данных.

### 1.2.2. Клиентские Web-технологии.

Частично преодолеть ограничения клиент/серверной архитектуры Интернет позволяют Клиентские Web-технологии.

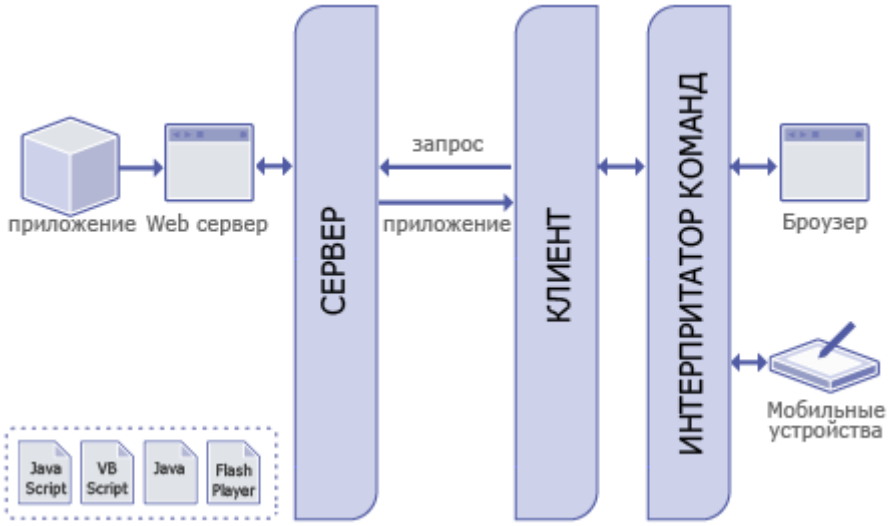


Рис.2. Архитектура Интернет с интерпретатором команд на стороне клиента.

Работа такой системы осуществляется в следующей последовательности:

Браузер посылает запрос приложения и служебную информацию на сервер.

Сервер находит и отправляет запрошенное приложение обратно.

Интерпретатор команд обрабатывает код приложения и выдает браузеру сформированный документ.

Преимуществом такой архитектуры являются возможность работы с динамическими документами (появляется интерактивность), а также возможность однажды загрузив приложение работать в нем, не обращаясь к серверу.

Однако недостатком остаются затруднения в работе с большими объемами данных.

### 1.2.3. Серверные Web-технологий.

Другой вариант развития Web-программирования связан с использованием серверных Web-технологий.

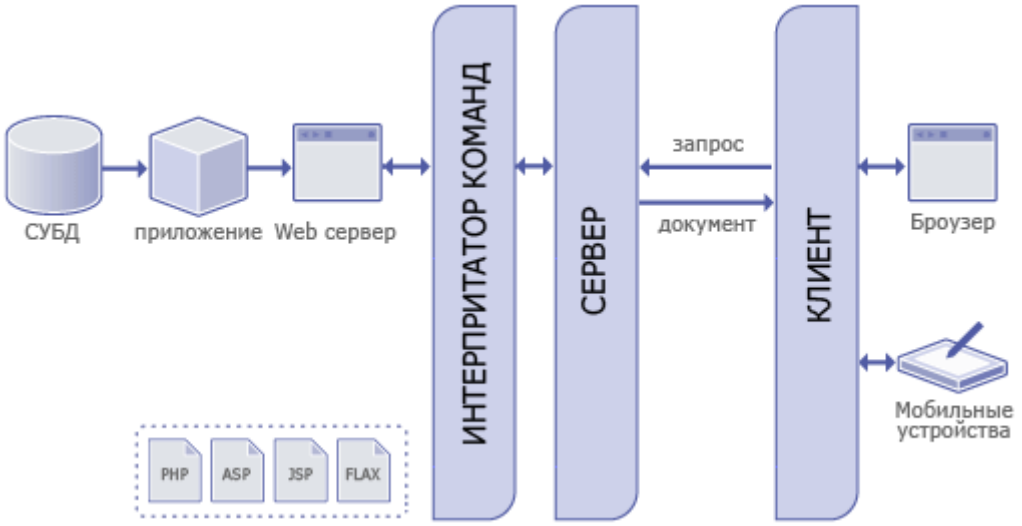


Рис.3. Архитектура Интернет с интерпретатором команд на стороне сервера.

Работа такой системы осуществляется в следующей последовательности:

1. Браузер посылает запрос приложения и служебную информацию на сервер.

2. Сервер находит и интерпретирует запрошенное приложение в документ и посылает его обратно.

3. Браузер получает сформированный документ и отображает его.

Преимуществом такой архитектуры являются:

1. Возможность работы с большими объемами данных, базами данных, не передавая их клиенту.

2. Защищенность данных.

3. Интерактивность документов.

Однако существенным недостатком такой архитектуры является то, что для совершения каждого действия над документом необходимо обращаться к серверу.

На основе рассмотренных выше архитектур строятся все современные технологии web-программирования.

### 1.3. Веб стандарты.

Основными организациями, занимающимися разработкой новых стандартов и рекомендаций в области web-программирования являются:

- Internet Engineering Task Force (IETF). Основана в 1986 г. http://www.ietf.org
- World Wide Web Consortium (W3C). Основан в 1994 г. http://www.w3.org

### 1.3.4. Вопросы для обсуждения.

1. Какие из web-технологий получат большее развитие в ближайшем будущем в связи с постоянным увеличением скорости Интернет?

2. Какие web-технологии являются потенциальными источниками вирусных атак для компьютера, подключенного к Интернет?

В чем заключается концепция «Семантической Сети»?

**Выдающиеся личности Интернет:** Тим Бернерс-Ли.

Тим Бернерс-Ли родился в Лондоне (Англия). После окончания Оксфордского университета в 1976 году сменил несколько компаний, где работал на должностях программиста, консультанта по программному обеспечению, системного архитектора.

В 1989 году Бернерс-Ли предложил глобальный гипертекстовый проект, ныне известный как Всемирная паутина. С 1991 по 1993 год Тим Бернерс-Ли предложил спецификации URI, HTTP и HTML. В 1994 году он основал и возглавил Консорциум Всемирной паутины, который занимается разработкой и внедрением стандартов для Интернета.

В качестве основных принципов сети Тим Бернерс-Ли отмечает:

1. Возможность редактировать информацию Паутины не менее важна, чем возможность просто лазать по ней. В этом смысле Бернерс-Ли очень рассчитывает на концепцию WYSIWYG, хотя Wiki - это тоже шаг в нужном направлении.

2. Компьютеры могут быть использованы для «фоновых процессов», помогающих людям работать сообща.

3. Каждый аспект Интернета должен работать как паутина, а не как иерархия. В этом смысле очень неприятным исключением является система имён доменов (англ. Domain Name System, DNS).

4. Учёные-компьютерщики несут не только техническую ответственность, но и моральную.

**Лекция 2**

# 2. Введение в HTML

## 2.1. Основные понятия и определения.

*HTML (HyperText Markup Language)* – язык создания гипертестовых документов. HTML не является языком программирования, т.к. предназначен только для представления данных (команды языка указывают клиенту, в каком виде должна быть отображена та или иная часть документа). В связи с этим язык HTML имеет следующие ограничения:

- не содержит средств обработки информации.
- не предоставляет никакой возможности сохранять данные в процессе работы приложения.
- имеет ограниченные возможности для взаимодействия с пользователем

Любая HTML страница состоит из набора объектов: текста, графики, ссылок, списков и т.д. Для представления этих объектов в языке HTML используются специальные операторы - тэги (от англ **tag** - ярлык).

Любой тег состоит из 3 элементов: начала тега, (открывающий тег) обозначаемого скобками <>, тела тега (например, текстовой информации) и конца тега (закрывающего тега) обозначаемого скобками </>), например:

```
<TITLE>Мой первый сайт</TITLE>
```

Таким образом, HTML документ состоит из последовательности тегов.

При написании HTML-документов нужно помнить следующие особенности:

- HTML не чувствителен к регистру. Команда <title> эквивалентна <TITLE> или <TiTlE>.
- HTML заменяет произвольные последовательности пробелов, табуляций и символов перехода на новую строку одним пробелом.
- Не все теги поддерживаются всеми Web-браузерами. Если браузер не поддерживает тег, он его просто игнорирует.

## 2.2. Инструменты и технологии программирования.

HTML документы представляют собой обычные текстовые документы и могут быть созданы в любом текстовом редакторе, например в Блокноте. Существуют специализированные WYSIWYG (What You See Is What You Get) HTML-редакторы, например, Adobe Dream Waver или MS Frontpage.

HTML документы принято сохранять в файлы с расширением *.htm или *.html.

Просмотр HTML документов осуществляется при помощи специальных программ-просмоторщиков (web-браузеров), которые интерпретируют HTML код для отображения на экране.

## 2.3. Структура HTML-документа.

HTML со времени своего рождения (1989 г.) постоянно развивался, непрерывно претерпевая изменения и дополнения стандартов. В данном курсе рассматривается последняя версия HTML 4.0.

Согласно спецификации HTML 4.0 структура HTML-документа выглядит следующим образом:

```
<HTML>
    <HEAD>
        <TITLE>Заголовок документа</TITLE>
    </HEAD>

    <BODY>
        Тело документа
    </BODY>
```

```
</HTML>
```

Как видно из приведенного кода, HTML документ всегда помещается в тег <HTML> и состоит из двух частей:
заголовка документа, определяемого тегом <HEAD>,
тела документа, определяемого элементом <BODY>.
В заголовке HTML-документа приводится информация о документе, которая не отображается в окне браузера. Исключением является тег <TITLE>, содержимое которого выводится в заголовке окна браузера и используется для идентификации документа пользователем или поисковой машиной.
Каждый HTML документ должен иметь название, например:

```
<TITLE>Введение в HTML</TITLE>
```

В заголовок также включаются метатеги <META> для указания кодировки, ключевых слов, описания документа и т.д., например:

```
<meta name ="Content-Type" content="text/html; charset=windows-1251">
<meta name="keywords" content="Web, программирование, HTML">
<meta name="description" content="Курс лекций по Web-программированию">
```

Тело HTML-документа обозначенное тегом <BODY></BODY> определяет видимую часть документа.

### 2.4. Параметры тегов.

Открывающие теги могут содержать дополнительную информацию - параметры, которые существенно расширяют возможности представления информации. Параметры в открывающем теге записываются после названия тега в виде *параметр=”значение*” или *параметр=’значение’ и* разделяются пробелами. Порядок следования параметров в теге не произвольный. Если параметр отсутствует, его значение принимается по умолчанию согласно спецификации. Пример использования параметров в теге <BODY> приведен ниже:

```
<BODY text="black" bgcolor="white">
```

Основные параметры тега <BODY> приведены в табл. 2:

Таблица 2.

Основные параметры тега <BODY>

| Параметр | Значение |
|---|---|
| text | Устанавливает цвет текста документа, используя значение цвета в виде RRGGBB. Например: text=”000000” - черный цвет. |
| link | Устанавливает цвет гиперссылок, используя значение цвета в виде RRGGBB. Например: text=”FF0000” – красный цвет. |
| vlink | Устанавливает цвет гиперссылок на котох пользовтель уже побывал. Например: text=”00FF00” – зеленый цвет. |
| alink | Устанавливает цвет гиперссылок при нажатии. Например: text=”0000FF” – синий цвет. |
| bgcolor | Устанавливает цвет фона документа, используя значение цвета в виде RRGGBB. Например: text=”FFFFFF” – белый цвет. |
| background | Устанавливает изображение фона документа. Например: Background=“bg.gif” |
| Topmargin (marginheight*) | Устанавливает величину верхнего поля документа. Пример: Topmargin=”0” |
| Leftmargin | Устанавливает величину левого поля документа. Пример: |

| | |
|---|---|
| (marginwidth*) | Leftmargin=”10” |

Праметры marginheight и marginwidth используются для установки величины верхнего и левого поля в браузерах Netscape.

**2.5. Вопросы для обсуждения.**

1. С какой целью браузеры в HTML документах заменяют произвольные последовательности пробелов, табуляций и символов перехода на новую строку одним пробелом?
2. Как правильно указать величины верхнего и левого полей документа, чтобы web-страница корректно просматривалась различными браузерами?
3. Является ли свойство пропускать незнакомые теги Web-браузерами положительным или отрицательным?

**Выдающиеся личности Интернет:** Марк Андриссен

Еще в 1992 году "паутина" была ориентирована на текст и была мало известна вне академических кругов. Ею пользовались в основном инженеры, ученые и те, для кого компьютер было хобби. Весной 1993 года Марк Андриссен в составе группы студентов Университета штата Иллинойс осуществил революцию в области интернет-навигации, значительно ее упростив за счет создания первого интернет-браузера Mosaic, способного просматривать как текстовые, так и графические документы. Переход к визуальному интерфейсу вызвал взрыв интереса к "паутине", а Марка Андрисен стал позднее одним из основателей Netscape, одного из двух главных браузеров в "паутине".

Лекция 3

# 3. Форматирование текста

## 3.1. Представление текстовой информации.

Основные теги для представления текстовой информации представлены в табл.3.

Таблица 3.

Основные теги для представления текстовой информации

| Тег | Описание |
|---|---|
| <H*i*><br><br><br>align | 6 уровней заголовков, пронумерованных от H1 до H6, где H1 имеет наибольшее выделение. Заголовки отображаются жирными шрифтами большего размера шрифты обычного текста:<br>устанавливает выравнивание заголовка относительно одной из сторон документа (align=left\|center\|right\|justify) |
| <P><br>align | Новый параграф.<br>устанавливает выравнивание параграфа относительно одной из сторон документа (align=left\|center\|right\|justify) |
| <BR> | Перевод строки |
| <HR><br>align<br><br>size<br>width<br>color<br>noshade | Горизонтальная линия:<br>устанавливает выравнивание линии относительно одной из сторон документа (align=left\|center\|right\|justify)<br>высота линии<br>ширина линии<br>цвет линии<br>не отбрасывать тень |
| <B> | Жирный текст |
| <I> | Наклонный текст |
| <U> | Подчеркнутый текст |
| <TT> | Моноширинный текст |
| <PRE> | Заранее отформатированный текст. Используется для того, чтобы текст с пробелами, символами перехода на новые строки и символами табуляции отображались браузером. В секциях <PRE> могут быть использованы гипертекстовые ссылки, однако, нельзя использовать другие HTML теги. |
| <FONT><br>size<br>face<br>color | Задает свойства шрифта:<br>размер шрифта от 1 до 7<br>начертание шрифта<br>цвет шрифта |

## 3.2. Escape-последовательности.

В случае, если кодировка или конфигурация оборудования и программного обеспечения не позволяет вводить определенные символы, например, ©, ®, £ или §, можно использовать ссылки на символы, называемые Escape-последовательностями. Escape-последовательности - это независимый от кодировки механизм ввода любых символов. Escape-последовательности в HTML могут принимать две формы:

- Числовые ссылки на символы, напимер, © (символ авторского права ©).
- Буквенные обозначения символов, напимер, © (символ авторского права ©).

Список наиболее часто используемых Escape-последовательностей приведен в табл. 4.

Таблица 4.

Основные Escape-последовательности

| Символ | Название символа | Escape-последовательность |
|---|---|---|
| < | меньше | < |
| > | больше | > |
| & | амперсанд | & |
| " | кавычка | " |
| | неразрывный пробел |   |
| © | символ авторского права | © |
| « | левая кавычка | « |
| » | правая кавычка | » |

### 3.3. Комментарии.

Комментарии в HTML имеют следующий синтаксис:

```
<!-- это комментарий -->
<!-- это тоже комментарий,
    он занимает несколько строк -->
```

Информация в комментариях не имеет специального значения и никак не влияет на отображение документа.

### 3.4. Организация списков.

HTML поддерживает 3 вида списков: ненумерованные, пронумерованные и списки-определения.

Ненумерованные списки создаются при помощи тегов <UL> и <LI>. Пример ненумерованного списка из двух элементов:

```
<UL>
    <LI>Глава1
    <LI>Глава2
</UL>
```

Результат выглядит следующим образом:

- Глава1
- Глава1

Пронумерованные списки создаются при помощи тегов <OL> и <LI>. Пример пронумерованного списка из двух элементов:

```
<OL>
    <LI>Документ1
    <LI>Документ2
    <LI>Документ3
</OL>
```

Результат выглядит следующим образом:

1. Документ1
2. Документ2
3. Документ3

Список-определение состоит из термина <DT> и его определения <DD>. Пример пронумерованного списка из двух элементов:

```
<DL>
    <DT> WWW
    <DD> World Wide Web

    <DT> HTML
    <DD> HyperText Markup Language
</DL>
```

Результат выглядит следующим образом:

HTML
    Hypertext Markup Language
WWW
    World Wide Web

Теги <LI>, <DT> и <DD> могут содержать вложенные элементы, например, параграфы, списки или другую информацию.
Пример вложенного списка:

```
<UL>
    <LI> Введение:
        <UL>
            <LI>Сравнительные характеристики
            <LI>Достоинства и недостатки
        </UL>
    <LI> Основной раздел:
        <UL>
            <LI>Пояснительная записка
            <LI>Список литературы
        </UL>
</UL>
```

Результат выглядит следующим образом:

Введение:
    Сравнительные характеристики
    Достоинства и недостатки
Основной раздел:
    Пояснительная записка
    Список литературы

**3.5. Вопросы для обсуждения.**

1. Почему в спецификации HTML 4.0 тег <FONT> отмечен как не рекомендуемый для использования?
2. Для чего введены escape-последовательности символов “<”, “>”, “&”, если эти символы присутствуют на любой клавиатуре?

**Выдающиеся личности Интернет:** Сергей Брин, Ларри Пейдж
Сергей Брин родился в Москве, в семье советских математиков. В 1993 он поступил в Стэнфордский университет в Калифорнии, где получил диплом магистра и начал работать над диссертацией. В 1995 г. в Стэнфордском университете Брин встретился с другим аспирантом-математиком - Лэрри Пейджем, вместе с которым позже (в 1998) они основали компанию Google.

Лекция 4

# 4. Ссылки. Графика

## 4.1. Гиперссылки.

Основная мощь HTML происходит из его возможности связывать документы между собой при помощи гипертекстовых ссылок. Пример гипертекстовой ссылки:

```
<A HREF="news.html">Новости</A>
```

Это выражение делает слово "Новости" гипертекстовой ссылкой на документ news.html, который находится в той же папке, что и основной документ. Чтобы сделать ссылку на документ, расположенный в другой папке, необходимо указать *относительный путь* от текущего документа к документу, на который делается ссылка. Например, ссылка на файл arhiv.html расположенная в папке NewsArhiv будет выглядеть так:

```
<A HREF="NewsArhiv/news.html">Архив новостей</A>
```

Для ссылки на документ, расположенный на другом сайте необходимо указывать адрес этого сайта и полный путь к документу относительно папки сайта. Например:

```
<A  HREF="http://www.transmarket.net/education/lib/books.txt">список  книг
по дизайну и программированию</A>
```

Ссылки также могут быть использованы для перехода к определенным частям документа. Предположим, вы хотите сделать ссылку в документе news.html на раздел «Анонс выпуска», расположенный ниже. Для этого необходимо:
1. Отметить место в документе, на которое будет вести ссылка, следующим образом:

```
<A NAME = "point1">Анонс выпуска</a>
```

2. Создать ссылку на помеченную область:

```
<A HREF = "#point1">Просмотреть анонс</A>
```

Здесь point1 – имя помеченной области, которое задается разработчиком.
Создание ссылки внутрь другого документа выполняется аналогично, за исключением того, что в ссылке помимо имени помеченной области указывается имя другого документа. Например:

```
<A href="anonce.html#point1">Просмотреть анонс</A>
```

По умолчанию загружаемый по ссылке документ отображается в том же окне, что и предыдущий. Для того, чтобы новых документ загружался в новом окне, используется атрибут target="_blank". Например:

```
<A href="anonce.html#point1" target="_blank">Просмотреть анонс</A>
```

Для вставки графической ссылки вместо текстовой информации в тег <A> помещается графический объект.

## 4.2. Представление графической информации.

Для размещения грфических элементов в HTML документах используются грфические файлы 3 типов: *.JPG, *.GIF или *.PNG.
Размещение графического элемента в HTML документе осуществляется при помощи тега <IMG>. Например:

```
<IMG src=”file_name”>
```

где file_name это имя графического файла. В табл. 5 перечислены дополнительные параметры тега <IMG>.

Таблица 5.

Основные параметры тега <IMG>

| Параметр | Значение |
|---|---|
| align | Определяет положение объекта относительно окружающего текста. align=”top\|middle\|bottom\|left\|right” |
| alt | Задает альтернативный текст, выводимый вместо графического объекта, если тот не может быть отображен. |
| border | Задает ширину рамки вокруг элемента в пикселях |
| height | Задает высоту элемента в пикселях |
| width | Задает ширину элемента в пикселях |
| usemap | Задает расположение и имя карты изображений |

### 4.3. Карты изображений.

Для распределения ссылок по картинке, например, для создания графического меню из одной большой картинки, используются карты изображений. Для применения карты изображений к графическому элементу необходимо:

- Указать имя карты изображений для графического элемента. Например:

```
<IMG src="menu.gif" usemap="file_name#map_name">
```

- Создать карту изображений с именем map_name и поместить ее в файл с именем file_name. Если URL не указан, то поиск карты изображений map_name ведется в текущем документе.

Код карты изображений записывается в следующем виде:

```
<MAP NAME="map_name">
     <AREA  [shape="default|rect|circle|poly"]  coords="x,y,..."  [href="
     reference"] [nohref]>
</MAP>
```

Здесь тэг <AREA> определяет область на картинке. Дополнительные параметры тега <AREA> приведены в табл. 6

Таблица 6.

Основные параметры тега <AREA>

| Параметр | Значение |
|---|---|
| shape | Определяет форму области. Можно задать одну из следующих областей:<br>• default - стандартная форма<br>• rect – прямоугольник<br>• circle –круг<br>• poly -многоугольник произвольной формы |
| coords | Задает координаты области в пикселях. Круг имеет три координаты, прямоугольник - четыре, для многоугольника задаются координаты каждого его угла. |
| href | Задает ссылку для выделенной области |
| noref | Указывает, что в данной области картинки отсутствует ссылка |

Например карта изображений, задающая 2 активных области на картинке будет иметь следующий вид:

```
<MAP NAME="mymap">
     <AREA shape="rect" coords="0,0,50, 20" href="news.htm">
     <AREA shape="circle" coords="100,25,25" href="contacts.htm">
</MAP>
```

**4.4. Вопросы для обсуждения.**

1. Куда ссылается следующая конструкция:

```
<A HREF = "#">ссылка</A>
```

Лекция 5

# 5. Таблицы в HTML. Табличная верстка.

## 5.1. Организация таблиц, параметры таблиц.

Таблицы в HTML документах используются в двух случаях:

Представление табличных данных на странице.

Верстка HTML документа (создание модульной сетки документа).

Таблица создается при помощи тега <TABLE>. Дополнительные параметры тега <TABLE> приведены в табл. 7.

Таблица 7.

Основные параметры тега <TABLE>

| Параметр | Значение |
|---|---|
| align | Задает выравнивание таблицы. align=left\|center\|right |
| border | задает в пикселях толщину линий таблицы |
| bgcolor | Устанавливает цвет фона документа |
| background* | Устанавливает изображение фона документа |
| cellpadding | задает в пикселях ширину промежутков между содержимым ячейки и ее границами, то есть задает поля внутри ячейки |
| cellspacing | задает в пикселях ширину промежутков между ячейками |
| width | задает ширину таблицы в пикселях, процентах или частях |

*При просмотре таблиц в браузерах семейства Netscape картинка, заданная параметром background автоматически становится фоном каждой ячейки, а не всей таблицы.

Для создания заголовка таблицы используется тег <CAPTION>. По умолчанию заголовки центрируются и размещаются либо над (<CAPTION align=”top”>), либо под таблицей (<CAPTION align=”bottom”>). Заголовок может состоять из любого текста и изображений. Текст будет разбит на строки, соответствующие ширине таблицы.
Каждая новая строка таблицы создается тегом <TR> (Table Row). Дополнительные параметры тега <TR> приведены в табл. 8.

Таблица 8.

Основные параметры тега <TR>

| Параметр | Значение |
|---|---|
| align | Задает горизонтальное выравнивание информации в ячейках align=left\|center\|right\|justify |
| valign | Задает вертикальное выравнивание информации в ячейках align=top\|middle\|bottom |

Внутри строки таблицы размещаются ячейки с данными, создаваемые тагами <TD>. Число тагов <TD> в строке определяет число ячеек (столбцов) таблицы. Дополнительные параметры тега <TR> приведены в табл. 9.
Для задания заголовков столбцов и строк таблицы используется тег заголовка <TH> (Table Header). Этот тег аналогичен <TD> с разницей лиш в том, что текст в теге <TH> по умолчанию выделяется жирным шрифтом и располагается по центру ячейки.

Таблица 9.

Основные параметры тега <TD>

| Параметр | Значение |
|---|---|
| align | Задает горизонтальное выравнивание информации в ячейке align=left\|center\|right\|justify |
| valign | Задает вертикальное выравнивание информации в ячейке |

| | |
|---|---|
| | align=top\|middle\|bottom |
| colspan | Объединение столбцов в строке. Например, colspan=2 |
| rowspan | Объединение строк в столбце. Например, rowspan=3 |
| width | Задает ширину ячейки в пикселях или процентах |
| nowarp | Запрещение перехода текста в ячейке на новую стоку |

Пример HTML кода для создания простой таблицы приведен ниже:

```
<TABLE width="100%" cellpadding="0" cellspacing="0" border="1">
     <CAPTION align=”top”>Статистика посещений сайтов online
     тестирования</CAPTION>
     <TR>
          <TH>URL</TH>
          <TH>Хосты</TH>
          <TH>Хиты</TH>
     </TR >
     <TR>
          <TD>www.brainbench.com</TD>
          <TD>12136</TD>
          <TD>22178</TD>
     </TR>
     <TR>
          <TD>www.transmarket.net/education</TD>
          <TD>12</TD>
          <TD>27</TD>
     </TR>
</TABLE>
```

### 5.2. Табличная верстка: построение модульной сетки при помощи таблиц.

Благодаря наличию большого числа параметров, а также возможности создания границ нулевой толщины, таблица может выступает в роли невидимой модульной сетки, относительно которой добавляется текст, изображения и другие элементы, организуя информацию на странице.

Возможности табличной верстки:

- Создание колонок
- «Резиновый» макет
- «Склейка» изображений
- Фоновые рисунки
- Выравнивание элементов
- Особенности браузеров

Недостатки табличной верстки:

- Долгая загрузка
- Громоздкий код
- Плохая индексация поисковиками
- Нет разделения содержимого и оформления
- Несоответствие стандартам

### 5.3. Типовые модульные сетки HTML документа

**Двухколонная модульная сетка** часто применяется на небольших информационных сайтах. Как правило, в первой колонке располагается логотип и меню сайта, а во второй — основной материал. Пример такой модульной сетки приведен на рис. 4.


Рис.4. Двухколонная модульная сетка.

Ниже приведен HTML код, реализующий данную структуру.

```
<TABLE width="760" cellpadding="10" cellspacing="10" border="1">
     <TR>
          <TD align=”center”>Логотип</TD>
          <TD rowspan=”2”>Основной материал</TD>
     </TR>
     <TR>
          <TD>Меню</TD>
     </TR>
</TABLE>
```

**Трехколонная модульная сетка «резинового» макета** применяется на крупных порталах с множеством информационных сервисов. Как правило, в первой колонке располагается меню сайта и дополнительная информация, во второй — основной материал, в третьей – дополнительные функции. Часто в модульную сетку сайта добавляют блоки «Заголовок сайта» и «Окончание сайта». Пример такой модульной сетки приведен на рис. 5.

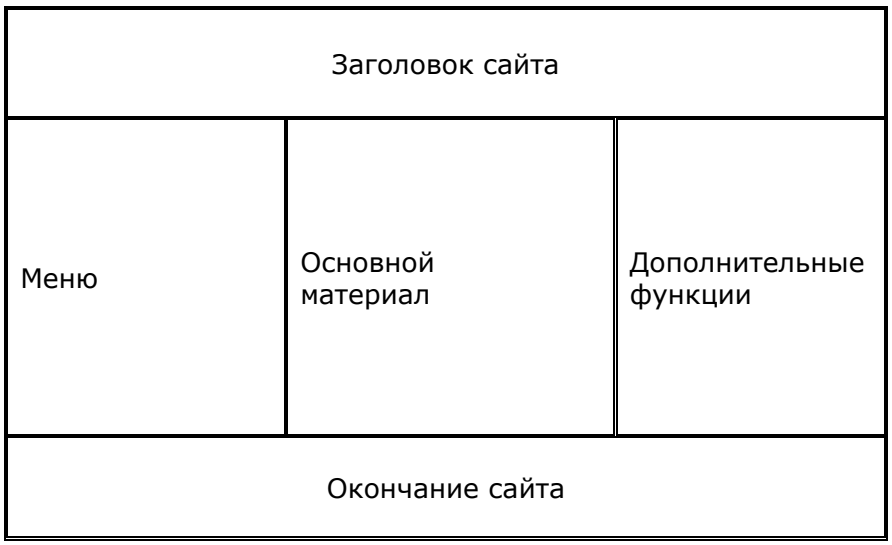


Рис.4. Трехколонная модульная сетка «резинового» макета.

Ниже приведен HTML код, реализующий данную структуру.

```
<TABLE width="100%" cellpadding="10" cellspacing="10" border="0">
     <TR align=”center”>
          <TD colspan=”3”>Заголовок сайта</TD>
     </TR>
     <TR>
          <TD width=”200px”>Меню</TD>
          <TD width=”*”>Основной материал</TD>
          <TD width=”200px”>Дополнительные функции</TD>
     </TR>
     <TR align=”center”>
          <TD colspan=”3”>Окончание сайта</TD>
     </TR>
</TABLE>
```

**5.3. Вопросы для обсуждения.**

1. Как создать эффект информационного блока со скругленными краями или с отбрасываемой тенью?
2. Как для браузеров семейства Netscape задать фоновое изображение для всей таблицы, а не для каждой ячейки в отдельности.

# 6. Фреймы. Формы

### 6.1. Фреймы.

Фреймы делят окно браузера на несколько областей, каждая из которых функционирует независимо от других и может отображать отдельный HTML-документ, как показано на рис.3.

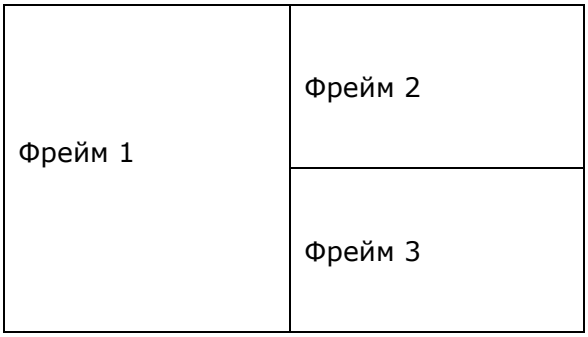


Рис.5. Пример разделения окна браузера на 3 фрейма.

Для того, чтобы страница могла содержать фреймы, тег <BODY> заменяется на тег <FRAMESET>. Параметры тега <FRAMESET> приведены в табл. 10.

Таблица 10.

Параметры тега <FRAMESET>

| Параметр | Значение |
|---|---|
| rows | определяет расположение горизонтальных фреймов. Это разделенный запятыми список пикселов, процентов и относительных длин. По умолчанию используется 100%, что означат одну строку. Например, rows=”20%, 80%” |
| cols | определяет расположение вертикальных фреймов. Это разделенный запятыми список пикселов, процентов и относительных длин. По умолчанию используется 100%, что означат одну строку. Например, cols=”*, *” |

Допускается использование вложенных друг в друга тегов <FRAMESET> для создания сложной структуры фреймов.
После задания структуры фреймов, каждый фрейм описывается при помощи тега <FRAME>. Параметры тега <FRAME> приведены в табл. 11.

Таблица 11.

Параметры тега <FRAME>

| Параметр | Значение |
|---|---|
| name | Имя фрейма |
| marginwidth | Горизонтальный отступ справа и слева (от 1 до 6) между фреймом и границей |
| marginheight | Вертикальный отступ сверху и снизу (от 1 до 6) между фреймом и его границей |
| src | Путь к гипертекстовому документу для фрейма.<br>src= “filename"\|_blank\|_self\|_parent\|_top |

| | |
|---|---|
| scrolling | Прокрутка фрейма. scrolling=”YES\|NO\|AUTO” |
| noresize | Фиксированный размер фрейма |
| frameborder | Задает ширину границы фрейма |
| bordercolor | Задает цвет границы фрейма |

Для браузеров, не поддерживающих фреймы или сконфигурированных так, чтобы не отображать их, можно указывать альтернативное содержимое при помощи тега <NOFRAMES>. Содержимое тега <NOFRAMES> отображается, только если не отображаются фреймы.
Пример HTML-кода, реализующего фреймы:

```
<html>
<head><title>Страница с фреймами</title></head>
<frameset cols=”50%, 50%”>
     <frame name=”left” src=”left.htm”>
     <frameset rows=”100px, *”>
          <frame name=”top” src=”top.htm”>
          <frame name=”bottom” src=”bottom.htm”>
     </frameset>
     <NOFRAMES>
          <P>Это <A href="noframes.html">версия документа без фреймов.</A>
     </NOFRAMES>
</frameset>
</html>
```

### 6.1. Создание HTML форм

HTML форма представляет собой раздел HTML документа, содержащего элементы управления: текстовые поля, флажки, кнопки с зависимой фиксацией, меню и т.д. HTML форма задается тегом <FORM>. Параметры тега <FORM> приведены в табл. 12.

Таблица 12.

Параметры тега <FORM>

| Параметр | Значение |
|---|---|
| action | Задает URL программы-обработчика формы или адрес электронной почты для отправки полей формы |
| method | Задает режим передачи данных из формы в программу-обработчик формы. Method=”GET\|POST” |

### 6.3. Элементы управления

Пользователи взаимодействуют с формами с помощью управляющих элементов. Для задания большинства управляющих элементов используется тег <INPUT>. Параметры тега <INPUT> приведены в табл. 12.

Таблица 13.

Параметры тега <INPUT>

| Параметр | Значение |
|---|---|
| type | Задает тип єлемента управления.<br>type=”text\|password\|checkbox\|radio\|submit\|reset\|file\|hidden”<br>text – текстовое поле<br>password – текстовое поле для ввода секретной информации<br>checkbox – флажок<br>radio – кнопки с зависимой фиксацией<br>submit – кнопка отправки формы |

| | |
|---|---|
| | reset – кнопка очистки формы<br>file – поле для выбора файла<br>hidden – скрітое поле |
| name | Имя элемента управления |
| value | Начальное значение управляющего элемента |
| checked | Задает активный элемент для кнопок checkbox и radio. |
| size | Ширина для текстовых полей |
| maxlength | Максимальное число символов, воспринимаемое текстовым полем |

Для создания многострочного строчного текстового поля используется тег <TEXTAREA>. Параметры тега <TEXTAREA> приведены в табл. 13.

Таблица 13.

Параметры тега <TEXTAREA>

| **Параметр** | **Значение** |
|---|---|
| name | имя элемента управления |
| Cols | ширина строки, задается в символах |
| Rows | количество строк |
| value | Начальное значение управляющего элемента |

Для создания выпадающего списка используется тег <SELECT>. Параметры тега <SELECT> приведены в табл. 14.

Таблица 14.

Параметры тега <SELECT>

| **Параметр** | **Значение** |
|---|---|
| name | имя элемента управления |
| multiple | позволяет выбирать более одной альтернативы списка при помощи удержания клавиши Ctrl |
| size | устанавливает количество пунктов меню, которое будет показано на экране, остальные будут доступны при использовании прокрутки |

Для указания каждого элемента списка используется тег <OPTION>. Параметры тега <OPTION> приведены в табл. 15.

Таблица 15.

Параметры тега <OPTION>

| **Параметр** | **Значение** |
|---|---|
| value | индекс текущего элемент списка. Именно это значение передается обработчику формы |
| selected | указывает активный элемент списка |

Пример HTML-кода, реализующего форму:

```
<html>
<head><title>Страница регистрации</title></head>
<body>

<form method=get action='form.php' name="form">
<table>
     <tr>
```

```
            <td>Фамилия, имя</td>
            <td>
                <input type="text" name="name" size="50" maxlength =”150”>
            </td>
        </tr>
        <tr>
            <td>Образование</td>
            <td>
                <SELECT name="request">
                    <OPTION value=0>...</OPTION>
                    <OPTION value=1>среднее</OPTION>
                    <OPTION value=2>среднее техническое</OPTION>
                    <OPTION value=3>высшее</OPTION>
                </SELECT>
            </td>
        </tr>
        <tr>
            <td>Пол</td>
            <td>
                <input type="radio" name="human_sex"> М
                <input type="radio" name="human_sex"> Ж
            </td>
        </tr>
        <tr>
            <td>Дополнительная информация</td>
            <td>
                <input type="file" name="human_file">
            </td>
        </tr>
        <tr>
            <td>Фото</td>
            <td>
                <TEXTAREA cols=”50” rows=”5”></TEXTAREA>
            </td>
        </tr>
        <tr>
            <td>Желаете получать новости сайта по электронной почте?</td>
            <td>
                <input type="checkbox" name="human_news" checked>
            </td>
        </tr>
        <tr>
            <td></td>
            <td>
                <input name="submit" type=”submit” value=”Отправить”>
                <input name=”reset” type=”reset” value=”Очистить”>
            </td>
        </tr>
    </form>
    </table>
    </body>
    </html>
```

**6.3. Вопросы для обсуждения.**

1. Назовите альтернативы фреймам в web-программировании.

# 7. Каскадные таблицы стилей CSS

## 7.1. Концепция „Документ-представление”

Широко распространенная в ООП концепция „Документ-представление” заключается в представлении документов в виде 2 независимых объектов, хранящих отдельно содержание и оформление документа.

Технология CSS реализует концепцию „Документ-представление” и позволяет отделять оформление HTML документа от его структуры. Кроме того, CSS существенно расширяет возможности представления (оформления) документов, внося множество новых возможностей.

## 7.2. Подключение каскадных таблиц стилей.

Для того, чтобы таблица стилей влияла на вид документа она должна быть подключена к HTML-документу. Подключение каскадных таблиц стилей может быть выполнено одним из трех способов.

1. Внешние таблицы стилей, оформленные в виде отдельных файлов подключаются к HTML-документу при помощи тега <LINK> в заголовке документа. Например:

```
<LINK rel="stylesheet" href="style.css" type="text/css">
```

2. Внутренние таблицы стилей в составе HTML-документа помещаются в заголовок страницы при помощи тега <STYLE>. Например:

```
<HEAD>
      <STYLE>
              P {color: #FF0000}
      </STYLE>
</HEAD>
```

3. Локальные таблицы стилей объявляются непосредственно внутри тега, к которому они относятся при помощи параметра style. Например:

```
<P style="color: #FF0000">Каскадные таблицы стилей</p>
```

## 7.3. Структура каскадных таблиц стилей.

Таблицы стилей записываются в виде последовательности тегов, для каждого из которых в фигурных скобках указывается его свойства в виде **параметр: свойство**. Параметры внутри фигурных скобок разделяются точкой с запятой. Например:

```
P {color: #CCCCCC; font-size: 10px}
H1{ color: #FF0000}
```

## 7.4. Основные свойства каскадных таблиц стилей.

Таблица 16.

Свойства фона и цвета

| Параметр | Значение |
|---|---|
| background-color | Цвет заднего плана |
| background-image | Изображение заднего плана |
| background-repeat | Дублирование изображения заднего плана. Значения: repeat \| repeat-x \| repeat-y \| no-repeat |
| background-attachment | Фиксация изображения заднего плана. Значения: scroll \| |

| | |
|---|---|
| | fixed |
| background-position | Начальное положение изображения заднего плана |

Таблица 17.

Свойства шрифта

| Параметр | Значение |
|---|---|
| font-style | Стиль шрифта. Значения: normal \| italic |
| font-weight | Начертание шрифта. Значения: normal \| bold |
| font-size | Размер |
| font-family | список имен шрифтов в порядке их приоритета |

Таблица 18.

Свойства текста

| Параметр | Значение |
|---|---|
| word-spacing | Установка промежутка между словами |
| letter-spacing | Установка высоты строки |
| text-indent | Установка абзацного отступа |
| text-align | Выравнивание текста |
| vertical-align | Установка вертикального выравнивания текста |
| text-decoration | Преобразование текста. Значение: none \| [ underline \|\| overline \|\| line-through \|\| blink |

Таблица 19.

Свойства полей и отступов

| Параметр | Значение |
|---|---|
| margin-top | Установка верхнего поля |
| margin-right | Установка правого поля |
| margin-bottom | Установка нижнего поля |
| margin-left | Установка левого поля |
| margin | Установка всех полей. Например: margin: 10px 10px 10px 0px |
| padding -top | Установка верхнего отступа |
| padding -right | Установка правого отступа |
| padding -bottom | Установка нижнего отступа |
| padding -left | Установка левого отступа |
| padding | Установка всех отступов. Например: padding: 10px 10px 10px 0px |

Таблица 20.

Свойства границ

| Параметр | Значение |
|---|---|
| border-width | Установка ширины границы |
| border-color | Установка цвета границы |
| border-style | Установка стиля границы. Значения: none \| dotted \| dashed \| solid \| double |

### 7.5. Классы каскадных таблиц стилей.

Для увеличения гибкости контроля над элементами HTML используются классы, которые позволяют задавать различные стили для одного и того же тега. В таблицах стилей имя класса записывается после тега и отделяется от него точкой. Например:

```
P.green {color: #00FF00}
P.blue {color: #0000FF}
```

В HTML коде соответствие тега определенному классу указывается при помощи параметра class. Например:

```
<p class="green">Зеленый текст</p>
<p class="blue">Синий текст</p>
```

### 7.6. Псевдоклассы каскадных таблиц стилей.

Для задания свойств стандартных состояний элементов HTML используются предопределенные классы – псевдоклассы. Перечень основных псевдоклассов приведен в табл. 21.

Таблица 21.

Свойства границ

| Параметр | Значение |
|---|---|
| Псевдоклассы ссылок<br>:link<br>:visited | <br>свойства ссылки по умолчанию<br>свойства посещенной ссылки |
| Динамические псевдоклассы<br>:hover<br>:active<br>:focus | <br>свойство объекта при наведении мыши<br>свойства активного объекта<br>свойства объекта, находящегося в фокусе |

Например:

```
A:link    { color: red }
A:visited { color: blue }
A:active  { color: white }
A:hover   { color: yellow }
```

### 7.7. Вопросы для обсуждения.

1. Назовите преимущества концепции „Документ-представление".

Лекция 8

# 8. Язык программирования JavaScript

## 8.1. Объектная модель браузера

JavaScript является языком клиентским скриптов. Коды программ, написанные на JavaScript, передаются в клиентский браузер и исполняются им. Поэтому важным вопросом изучения языка JavaScript является понимание объектной модели браузера.

Объектная модель браузера представляет собой строгую иерархическую структуру, позволяющую обращаться к любой части браузера или загруженных страниц с помощью языка JavaScript. Обобщенная объектная модель представлена на рис. 4.

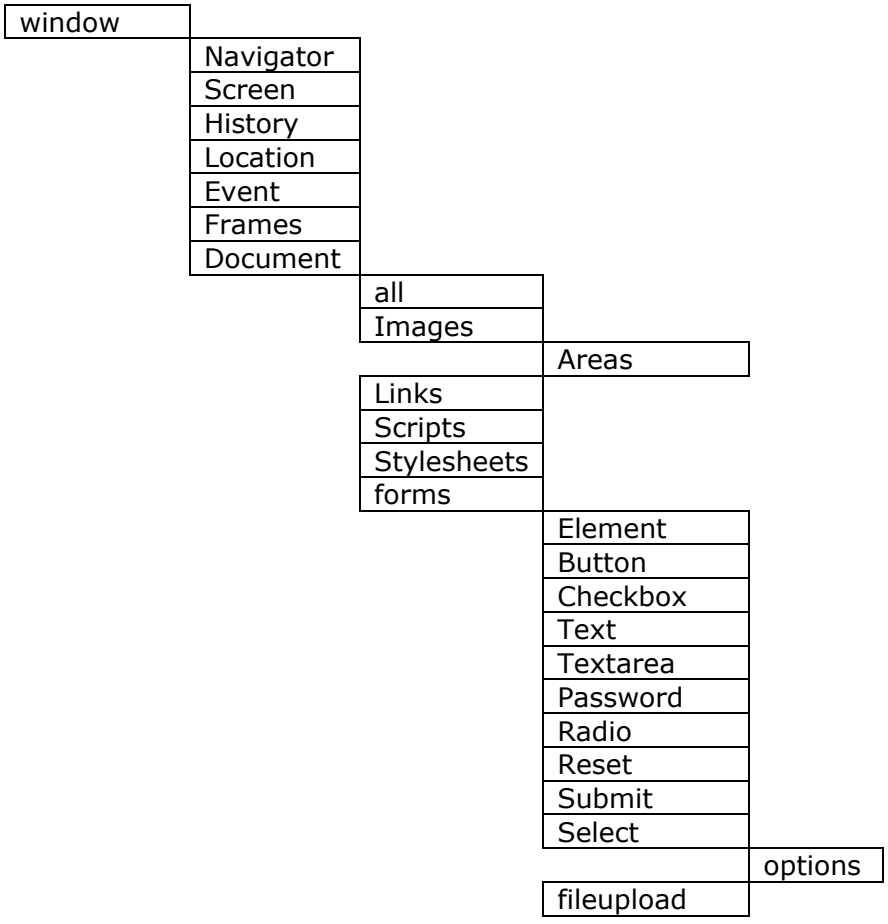


Рис.4. Объектная модель браузера.

В качестве примера на рис.5 представлена HTML-страница с указанием соответствия каждого ее элемента объектной модели.
Используя синтаксис JavaScript можно обратиться к любому элементу HTML-документа и изменить его поведение при помощи свойств и методов этого элемента, например:

```
document.forms[0].elements[0].value = "Milosh";
```

Если мы имеем дело с большими страницами, то процедура адресации к различным объектам по номеру может стать весьма неудобной. Во избежание подобной проблемы можно присваивать различным тегам уникальные имена, например:

```
<form name="myForm">
     Name: <input type="text" name="name" value=""><br>
     E-mail: <input type="text" name="email" value=""><br>
     <input type="submit" name="submit" value="Push me">
</form>
```

В этом случае обращение к элементам HTML-документа выглядит следующим образом:

```
document.myForm.name.value = "Milosh";
```

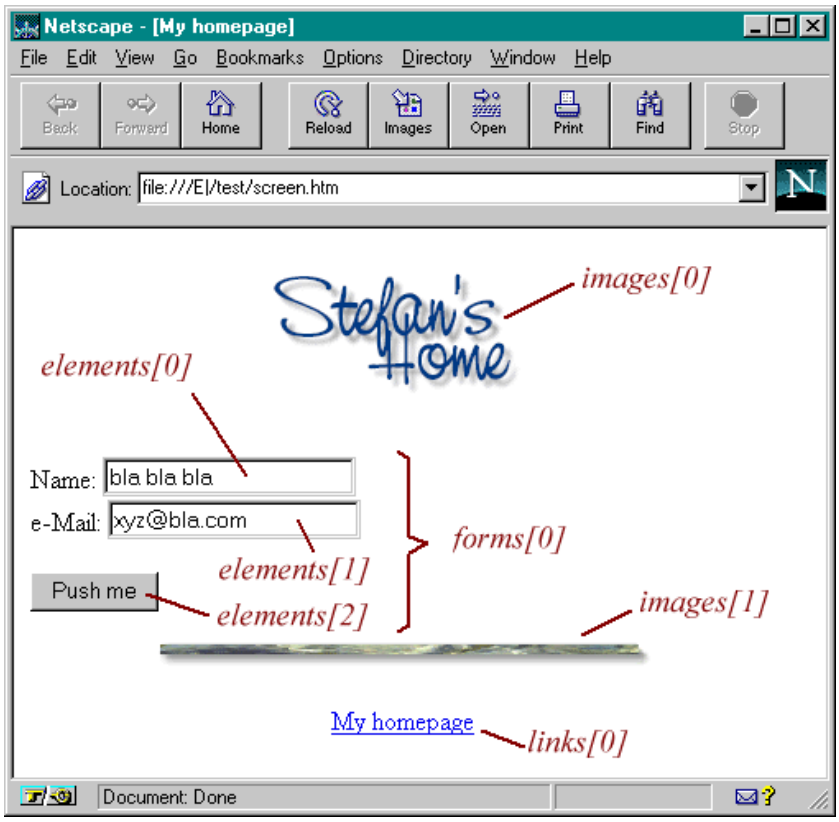


Рис.5. Соответствие HTML-элементов объектной модели браузера.

## 8.2. Объект window

Объект window – главный объект в иерархии объектной модели браузера. Объект window обращается к активному окну. Перечень основных свойств, методов и событий объекта window приведен в табл. 22, 23 и 24 соответственно.

Таблица 22.

Свойства объекта window

| Свойство | Значение |
|---|---|
| parent | Возвращает родительское окно |
| name | Название окна |
| status | Текст, отображаемый в строке состояния |
| document | Ссылка «только для чтения» на объект окна document |

| | |
|---|---|
| event | Ссылка «только для чтения» на объект окна event |
| history | Ссылка «только для чтения» на объект окна history |
| location | Ссылка «только для чтения» на объект окна location |
| navigator | Ссылка «только для чтения» на объект окна location |
| screen | Ссылка «только для чтения» на объект окна screen |

Таблица 23.

Методы объекта window

| Метод | Значение |
|---|---|
| alert | Вызов системного окна с простым сообщением. Например, alert(“Заполните пустые поля формы”) |
| Confirm(text) | Вызов системного окна с кнопками OK/CANCEL. Например, alert(“Продолжить поиск?”) |
| Prompt(text) | Вызов системного окна с полем для ввода текста с клавиатуры. Например, alert(“Введите Ваше имя”) |
| Open(url, name, settings) | Открывает новое окно. Например, window.open(“help.htm”, “helpWindow”, “*width=400,height=300,status=no,toolbar=no,menubar=no*”) |
| Close() | |

Таблица 24.

События объекта window

| Событие | Значение |
|---|---|
| onblur | Потеря фокуса |
| onfocus | Окно в фокусе |
| onhelp | Нажатие F1 или Help |
| onresize | Изменение размеров окна |
| onscroll | Прокрутка окна |
| onload | Страница полностью загружена |
| onunload | Перед выгрузкой страницы |

### 8.3. Дочерние объекты window

Объект window имеет несколько дочерних объектов, которые доступны с его помощью:

- Объект history содержит информацию о посещенных адресах.
- Объект navigator содержит информацию о браузере.
- Объект location содержит информацию о URL текущей страницы.
- Объект event содержит информацию о событиях, происходящих в браузере.
- Объект screen содержит информацию о возможностях экрана клиента.
- Объект document содержит отображаемый документ.

Перечень основных свойств, методов и событий дочерних объектов window приведен в табл. 25.

Таблица 25.

Свойства, методы и события дочерних объектов window

| Объект | Значение |
|---|---|
| **History** | |
| length | Длина списка посещенных адресов |
| back() | Загружает предыдущий адрес |

| | |
|---|---|
| forward() | Загружает следующий адрес |
| go(n) | Загружает адрес с номером n |
| **Navigator** | |
| appName | Название браузера |
| appVersion | Версия браузера |
| cookieEnabled | Возможность работы с cookie |
| mimeTypes | Список поддерживаемых браузером типов файлов |
| plugins | Список поддерживаемых браузером внедряемых объектов |
| javaEnabled() | Возможность запуска JavaScript |
| **Location** | |
| href | Указывет URL страницы |
| reload() | Обновляет страницу |
| **Event** | |
| X | Горизонтальная позиция курсора мыши |
| Y | Вертикальная позиция курсора мыши |
| button | Кнопка мыши, если она нажата |
| altkey | Состояние клавиши Alt |
| Ctrlkey | Состояние клавиши Ctrl |
| Shiftkey | Состояние клавиши Shift |
| Keycode | ASCII код нажатой клавиши |
| **Screen** | |
| Width | Разрешение по ширине экрана |
| Height | Разрешение по высоте экрана |
| colorDepth | Глубина цвета палитры экрана |

### 8.4. Объект document

Перечень основных свойств, методов и событий дочерних объектов document приведен в табл. 26, 27 и 28.

Таблица 26.

Свойства объекта document

| Объект | Значение |
|---|---|
| Title | Название документа, определяемое в теге TITLE |
| Bgcolor | Цвет фона |
| Fgcolor | Цвет текста |
| linkColor | Цвет ссылок |
| alinkColor | Цвет активных ссылок |
| vlinkColor | Цвет посещенных ссылок |
| activeElement | Элемент в фокусе |

Таблица 27.

Методы объекта document

| Объект | Значение |
|---|---|
| open() | Открывает объект document для редактирования |
| write(text) | Запись текста в объект document |
| close() | Закрывает объект document для редактирования |
| clear | Очищает содержимое объекта document |

| | |
|---|---|
| alinkColor | Цвет активных ссылок |
| vlinkColor | Цвет посещенных ссылок |
| activeElement | Элемент в фокусе |

Таблица 28.

События объекта document

| Объект | Значение |
|---|---|
| Onclick | Щелчок мыши |
| Ondblclick | Двойной щелчок мыши |
| Onmouseout | Мышь уходит с элемента |
| Onmouseover | Мышь появляется над элементом |
| Onkeydown | Нажатие клавиши |
| onhelp | Нажатие F1 или Help |
| onload | Полная загрузка элемента |

### 8.5. Размещение JavaScript на HTML-странице

Код скрипта JavaScript может объявляться в HTML-коде следующими способами:

1. Непосредственное размещение в любой части HTML-кода при помощи тега SCRIPT, например:

```
<script language="JavaScript">
      alert("Hello, world!")
</script>
```

2. Размещение в отдельном файле с обїявлением в заголовке HTML-документа при помощи тега SCRIPT, например:

```
<script language="JavaScript" src="script.js">
</script>
```

3. Контекстное размещение в обработчике событий. например:

```
<input type="button" onClick=" alert('Hello, world!')">
```

Лекция 9

# 9. Синтаксис языка JavaScript

### 9.1. Правила написания программ JavaScript.

Текст программы на JavaScript состоит из последовательности операторов. Именно в такой последовательности операторы и исполняются интерпретатором (броузером).
Особенности записи программ на языке JavaScript:

- Синтаксис языка JavaScript близка очень близок языку C++
- Один оператор в JavaScript может быть разбит на несколько строк, или, наоборот, в одной строке может быть несколько операторов
- В программе отдельные операторы записываются через точку с запятой (допускается не ставить ; если оператор является в строке последним)
- JavaScript не имеет жестких требований к форматированию текста программы, таким образом возможно использование символов перевода строки и отступов для придания тексту программы лучшей читабельности
- Однострочные комментарии в JavaScript записываются после симовлов //, многострочные – между символами /* */.

### 9.2. Арифметические и логические операции

Допустимые синтаксисом языка JavaScript операции представлены в табл. 29.

Таблица 29

Арифметические и логические операции

| **Арифметические операторы** | |
|---|---|
| +,-,*,/,% | Сложение, вычетание, умножение, деления, остаток от деления |
| ++, -- | Приращение на единицу, Уменьшение на единицу |
| - | Изменение знака |
| **Операторы присваивания** | |
| = | Присваивание значения |
| +=, -=, *=, /=, %= | Увеличение, уменьшение, умножение, деление на заданную величину, взятие модуля заданной величины |
| **Операторы сравнения** | |
| == | Эквивалентность сравниваемых объектов |
| != | Не равно |
| >, >=, <, <= | Больше, больше или равно, меньше, меньше или равно |
| **Логические операторы** | |
| &&, \|\|, ! | логическое И, ИЛИ, НЕ |

### 9.3. Типы данных

Переменные JavaScript могут иметь один из следующих типов:

- Числовой (целые числа или числа с плавающей точкой)
- Булевый
- Строковый
- Нулевой тип (определяется ключевым словом null)

Тип данных при объявлении переменной не указывается. Тип присваивается переменной только тогда, когда ей присваивается какое-либо значение. Переменные в программе объявляются при помощи ключевого слова var (допускается опускать), например:

```
var x=1;
y=”string”;
```

## 9.4. Массивы JavaScript

Тип массив введен в JavaScript для возможности манипулирования разными объектами: это список всех гипертекстовых ссылок страницы, список всех картинок на странице, список, список всех элементов формы и т.п. Пользователь может создать и свой собственный массив, используя конструктор Array(). Например:

```
new_array = new Array();
new_array5 = new Array(5);
colors = new Array ("red","white","blue")
```

Размерность массива может динамически изменяться. Можно сначала определить массив, а потом присвоить одному из его элементов значение. Как только это значение будет присвоено, изменится и размерность массива:

```
colors = new Array();
colors[5] = "red"
```

В данном случае массив будет состоять из 6 элементов, т.к. первым элементом массива считается элемент с индексом 0.

Перечень основных свойств и методов массивов приведен в табл. 30.

Таблица 30.

Свойства и методы массивов

| Объект | Значение |
|---|---|
| length | число элементов массива |
| join | объединяет элементы массива в строку символов, в качестве аргумента в этом методе задается разделитель, например: string = acolors.join("+") |
| reverse | изменяет порядок элементов массива на обратный |
| sort | сортирует элементов массива в порядке возрастания |

## 9.5. Функции в языке JavaScript

Часто используемые фрагменты программы можно оформить в виде функций, вызывая их по мере необходимости из различных мест программы. Функции должны быть определена перед вызовом, и размещение всех определений функций в разделе заголовка HEAD документа гарантирует доступность этих функций при обработке документа.

Формат определения функции представлен ниже:

```
function имя([параметр 1] [,параметр 2] [...,параметр N])
{
     тело функции
     [return значение]
}
```

С помощью ключевого слова return функция может вернуть значение.

## 9.6. Управление потоком вычислений

### 9.6.1. Условный оператор

```
if(i>0)
     {выражение}
else
     {выражение}
```

### 9.6.2. Оператор выбора

```
switch (значение)
{
     case label :
          выражение;
          break;
     case label :
          выражение;
          break;
     ...
     default : выражение;
}
```

### 9.6.3. Операторы цикла

```
for(i=0;i<9;i++)
     {выражение}

for(i in obj)
     {выражение}

wile(условие)
     {выражение}

do {выражение}
while (условие)
```

### 9.6.4. Операторы прерывания потока вычислений

Оператор break используется для прерывания выполнения цикла, либо оператора switch, например:

```
while(i==0)
     {if(j==3) break;}
```

Оператор continue используется для рестарта операторов цикла.

```
while(i==0)
     {if(j==3) continue;}
```

Ниже приведен пример использование языка JavaScript для проведения экспресс опроса посетителей сайта:

# 10. Язык серверных скриптов PHP

## 10.1. Основные понятия

PHP – это язык серверных скриптов (server scripting language), встраиваемый в HTML, который интерпретируется и выполняется на сервере.
Файлы, которые подвергаются обработке препроцессором, должны иметь определенное расширение (обычно это .php) и содержать код для препроцессора. Перед отправкой страницы PHP-код проигрывается на сервере и брозеру выдается результат в виде HTML-страницы. Обычные же страницы, имеющие расширение .html/.htm Web-сервер будет отправлять броузеру без какой-либо обработки.

## 10.2. Размещение php скриптов на HTML-странице

Код скрипта PHP может объявляться в HTML-коде следующими способами:
1. Непосредственное размещение в любой части HTML-кода при помощи тега SCRIPT, например:

```
<script language="PHP">
      echo "Hello, world!";
</script>
```

2. Краткая форма размещения в любой части HTML-кода при помощи конструкции <? … ?>, например:

```
<?
      echo "Hello, world!";
?>
```

## 10.3. Правила написания программ PHP

Текст программы на PHP состоит из последовательности операторов. Именно в такой последовательности операторы и исполняются интерпретатором (броузером).
Особенности записи программ на языке PHP:

- Синтаксис языка PHP близка очень близок языку C++
- Один оператор в PHP может быть разбит на несколько строк, или, наоборот, в одной строке может быть несколько операторов
- В программе отдельные операторы записываются через точку с запятой (допускается не ставить ; если оператор является в строке последним)
- PHP не имеет жестких требований к форматированию текста программы, таким образом возможно использование символов перевода строки и отступов для придания тексту программы лучшей читабельности
- Однострочные комментарии в JavaScript записываются после симовлов //, многострочные – между символами /* */.

## 10.4. Арифметические и логические операции

Допустимые синтаксисом языка PHP операции представлены в табл. 29.

Таблица 31

Арифметические и логические операции

| **Арифметические операторы** | |
|---|---|
| +,-,*,/,% | Сложение, вычетание, умножение, деления, остаток от деления |
| ++, -- | Приращение на единицу, Уменьшение на единицу |
| - | Изменение знака |
| **Операторы присваивания** | |

| | |
|---|---|
| = | Присваивание значения |
| +=, -=, *=, /=, %= | Увеличение, уменьшение, умножение, деление на заданную величину, взятие модуля заданной величины |
| **Операторы сравнения** | |
| == | Эквивалентность сравниваемых объектов |
| != | Не равно |
| >, >=, <, <= | Больше, больше или равно, меньше, меньше или равно |
| **Логические операторы** | |
| &&, \|\|, ! | логическое И, ИЛИ, НЕ |

### 8.4. Типы данных

Переменные PHP могут иметь один из следующих типов:

- Числовой (целые числа или числа с плавающей точкой)
- Булевый
- Строковый
- Нулевой тип (определяется ключевым словом null)

Тип данных при объявлении переменной не указывается. Тип присваивается переменной только тогда, когда ей присваивается какое-либо значение. Имена переменных в программе всегда должны начинаться с символа $, например:

```
$x=1;
$y=”string”;
```

### 8.5. Массивы PHP

Тип массив введен в JavaScript для возможности манипулирования разными объектами: это список всех гипертекстовых ссылок страницы, список всех картинок на странице, список, список всех элементов формы и т.п. Пользователь может создать и свой собственный массив, используя конструктор Array(). Например:

```
new_array = new Array();
new_array5 = new Array(5);
colors = new Array ("red","white","blue")
```

Размерность массива может динамически изменяться. Можно сначала определить массив, а потом присвоить одному из его элементов значение. Как только это значение будет присвоено, изменится и размерность массива:

```
colors = new Array();
colors[5] = "red"
```

В данном случае массив будет состоять из 6 элементов, т.к. первым элементом массива считается элемент с индексом 0.

Перечень основных свойств и методов массивов приведен в табл. 30.

Таблица 30.

Свойства и методы массивов

| Объект | Значение |
|---|---|
| length | число элементов массива |
| join | объединяет элементы массива в строку символов, в качестве аргумента в этом методе задается разделитель, например: string = acolors.join("+") |

| | |
|---|---|
| reverse | изменяет порядок элементов массива на обратный |
| sort | сортирует элементов массива в порядке возрастания |

## 8.6. Функции в языке PHP

Часто используемые фрагменты программы можно оформить в виде функций, вызывая их по мере необходимости из различных мест программы. Функции должны быть определена перед вызовом, и размещение всех определений функций в разделе заголовка HEAD документа гарантирует доступность этих функций при обработке документа.

Формат определения функции представлен ниже:

```
function имя([параметр 1] [,параметр 2] [...,параметр N])
{
     тело функции
     [return значение]
}
```

С помощью ключевого слова return функция может вернуть значение.

## 8.7. Управление потоком вычислений

### 8.6.1. Условный оператор

```
if(i>0)
     {выражение}
else
     {выражение}
```

### 8.6.2. Оператор выбора

```
switch (значение)
{
     case label :
          выражение;
          break;
     case label :
          выражение;
          break;
     ...
     default : выражение;
}
```

### 8.6.3. Операторы цикла

```
for(i=0;i<9;i++)
     {выражение}

for(i in obj)
     {выражение}

wile(условие)
     {выражение}
```

```
do {выражение}
while (условие)
```

### 8.6.4. Операторы прерывания потока вычислений

Оператор break используется для прерывания выполнения цикла, либо оператора switch, например:

```
while(i==0)
     {if(j==3) break;}
```

Оператор continue используется для рестарта операторов цикла.

```
while(i==0)
     {if(j==3) continue;}
```