МИНОБРНАУКИ РОССИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ЧЕРЕПОВЕЦКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Институт информационных технологий

Кафедра математики и информатики

УЧЕБНО-МЕТОДИЧЕСКИЕ МАТЕРИАЛЫ ПО ДИСЦИПЛИНЕ

«ТЕЛЕКОММУНИКАЦИОННЫЕ ТЕХНОЛОГИИ»

Направление подготовки (специальность):
01.03.02 Прикладная математика и информатика

Образовательная программа:
ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ

Очная форма обучения

Составители:

Лавров В.В., старший
преподаватель кафедры МиИ

г. Череповец - 2022

Перечень основной и дополнительной учебной литературы, необходимой для освоения дисциплины (модуля)

Основная литература:

1. Телекоммуникационные сети и технологии : учебное пособие / Х. Ш. Кульбикаян, Б. Х. Кульбикаян, А. В. Дицков, А. В. Шандыбин ; под редакцией Х. Ш. Кульбикаяна. — Ростов-на-Дону : РГУПС, 2019. — 212 с. — ISBN 978-5-88814-869-3. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/134039
2. Пуговкин, А. В. Телекоммуникационные системы : учебное пособие / А. В. Пуговкин. — Москва : ТУСУР, 2007. — 202 с. — ISBN 5-86889-337-9. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/4939

Дополнительная литература:

1. Логинов, В. И. Спутниковые телекоммуникационные технологии : учебное пособие / В. И. Логинов. — Нижний Новгород : ВГУВТ, 2014. — 72 с. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/51564
2. Многоканальные телекоммуникационные системы : учебное пособие. — Санкт-Петербург : СПбГУТ им. М.А. Бонч-Бруевича, 2016 — Часть 1 — 2016. — 111 с. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/180121

Перечень ресурсов информационно-телекоммуникационной сети «Интернет», необходимых для освоения дисциплины (модуля), включая перечень информационных справочных систем (при необходимости)

1 Интерактивная доска.
2 http://www.ois.org.ua/spravka/mat/index.htm - электронная библиотека по математике.
3 http://eqworld.ipmnet.ru/ru/library.htm- учебно-образовательная физико-математическая библиотека.
4 http://www.exponenta.ru/- образовательный математический сайт.

Учебно-методические указания и рекомендации к изучению тем лекционных и практических занятий, самостоятельной работе студентов

Лекции

№ п/п	Тема лекции	Количество часов
1	Понятие «телекоммуникационная сеть». Аппаратное и программное обеспечение телекоммуникационной сети.	2
2	Многоуровневая сетевая модель. Сетевые протоколы. Методы доступа к сети	2
3	Адресация в интернете. Ресурсы и сервисы интернета.	2
4	Язык разметки гипертекста (HTML).	4
5	HTML5.	2
6	Каскадные таблицы стилей (CSS).	4
7	Основы веб-программирования.	4
	Итого	20

Лабораторные работы

№ п/п	Тема лабораторной работы	Количество часов
1	Язык разметки гипертекста HTML	8
2	Оформление веб-страниц с использованием CSS	10
3	JavaScript	10
4	HTML5	8
	Итого	36

Лабораторная работа №1

Тема: Язык разметки гипертекста HTML

Цель работы: научиться создавать веб-страницы с использованием HTML.

Задания

1. Используя редактор HTML кода Notepad++, создайте html-файл (кодировка utf-8) с заголовком «Задание 1», результат которого показан на рис.1. Выясните различие тегов <b> и , <i> и .

2.

Контрольные вопросы

1. Какова структура HTML документа?
2. Чем отличаются друг от друга теги <b> и ?
3. Чем отличаются друг от друга теги <i> и ?
4.

Тема: Оформление веб-страниц с использованием CSS

Цель работы: изучить основы формального языка CSS и применить его для оформления веб-страниц.

►Оформление текста

1. Изучите свойства, используемые для оформления текста при помощи CSS (http://www.wisdomweb.ru/CSS/text.php).
2. Используя селекторы идентификаторов и изученные свойства, оформите следующий текст:

```
В данном абзаце отступ между буквами равен 17 пикс., а отступ между словами 5 пикс. Данный абзац оранжевого цвета.

Текст данного элемента подчеркнут, отступ между буквами в нем равен 15 пикселей. Данный абзац серого цвета.

Текст данного элемента выровнен по центру, отступ между словами в нем равен 10 пикселей. Данный элемент имеет цвет #ff3366.

Текст данного элемента выравнен по правому краю, отступ между буквами в нем равен 6 пикселей. Текст написан маленькими буквами красного цвета.

Текст данного элемента выравнен по центру, подчеркнут, отступ между буквами в нем равен 7 пикселей. Текст написан большими буквами зеленого цвета.
```

3. Сохраните файл под именем 1.html в папке CSS_Практика.

►Установки шрифта

1. Изучите свойства, используемы для определения шрифта (http://www.wisdomweb.ru/CSS/font.php).
2. Используя селекторы идентификаторов и изученные свойства, оформите следующий текст:

```
Данный абзац написан шрифтом Arial Black и имеет размер 20 пикселей.

Данный абзац написан курсивным шрифтом Courier New и имеет размер 24 пикселя.

Данный абзац написан жирным шрифтом Verdana и имеет размер 10 пикселей.

Данный абзац написан шрифтом Georgia и имеет размер 2.5em.

Данный абзац написан курсивным шрифтом Comic Sans MS и имеет размер 1.3em.
```

3. Сохраните файл под именем 2.html в папке CSS_Практика.

►Оформление фона в CSS

1. Изучите свойства, используемы для оформления фона в CSS (http://www.wisdomweb.ru/CSS/background.php).
2. Используя селектор тега <body> и изученные свойства, выполните задание в окне on-line редактора http://www.wisdomweb.ru/editor/wweditor.php?fname=css_backgrex2:

```
Вставьте картинку из файла spider2.gif на фон страницы (картинка находится в папке редактора, так что обращайтесь к ней просто по имени). Изображение при этом не должно повторятся.

Перенесите паучка в рамку. При этом при прокрутке страницы изображение паука должно оставаться в рамке.
```

3. Сохраните файл под именем 3.html в папке CSS_Практика.

►Оформление ссылок

1. Изучите свойства, используемы для оформления ссылок в CSS (http://www.wisdomweb.ru/CSS/link.php).
2. Откройте файл 6.html, созданный при выполнении практики №1. Используя изученные свойства и селектор тега <a>, оформите ссылки следующим образом:

```
a:link – определяет оформление обычной не посещенной ссылки (ссылка
подчеркнута, цвет зеленый).

a:visited - определяет оформление посещенной пользователем ссылки (ссылка не
подчеркнута, цвет синий).

a:hover - определяет оформление ссылки, на которую наведен курсор мыши
(ссылка подчеркнута, цвет красный, размер шрифта 1.1em).

a:active - определяет оформление ссылки, на которую щелкнули мышкой (ссылка
не подчеркнута, цвет синий, размер шрифта 1.1em).
```

3. Сохраните файл под именем 4.html в папке CSS_Практика.

►Оформление списка

1. Изучите свойства, используемы для оформления списков в CSS (http://www.wisdomweb.ru/CSS/list.php).
2. Откройте файл 4.html, созданный при выполнении практики №1. Используя изученные свойства и селекторы классов, оформите маркированный список квадратиками, а нумерованный – римскими цифрами.

```
…
<style type='text/css'>
        ul.list1
                { /* Определяйте стили здесь */}
        ol.list2
                { /* Определяйте стили здесь */}
</style>
…
<ul class='list1'>
…
        <ol class='list2'>
…
        </ol>
</ul>
…
```

3. Сохраните файл под именем 4.html в папке CSS_Практика.

►Оформление таблицы

1. Изучите свойства, используемы для оформления таблицы в CSS (http://www.wisdomweb.ru/CSS/table.php).
2. Оформите таблицу, представленную ниже, используя изученные свойства. Сохраните файл под именем 5.html в папке CSS_Практика.

```
Ширина таблицы равна 700 пикселей. Высота первой строки равна 70 пикселей.
Границы таблицы и ячеек соединены.
```

Текст ячейки выравнивается по верхнему краю	Текст ячейки выравнивается по центру	Текст ячейки выравнивается по нижнему краю	Текст ячейки выравнивается по центру
Толщина границы ячейки равна 2 пикселя	Толщина границы ячейки равна 3 пикселя	Толщина границы ячейки равна 4 пикселя	Толщина границы ячейки равна 1 пиксель
Текст ячейки выравнивается по левому краю	Текст ячейки выравнивается по правому краю	Текст ячейки выравнивается по центру	

Тема: JavaScript

Цель работы: изучить основы JavaScript и применить JavaScript для разработки интерактивной веб-страницы.

►Добавление JavaScript на веб-страницы

- Изучите способы подключения JavaScript к веб-странице (http://www.wisdomweb.ru/JS/start.php).
- Выведите следующие строчки текста на страницу с помощью JavaScript:

1. **Лес** — часть поверхности Земного шара, покрытая древесными растениями.
2. В настоящее время леса занимают около трети площади суши.
3. Площадь леса в России составляет 8 млн кв.км.

- Сохраните файл под именем 1.html в папке JS_Практика.

►Размещение скрипта в <head>, <body>, подключение внешних скриптов

- Изучите способы размещения скрипта в <head>, <body> и подключение внешних скриптов (http://www.wisdomweb.ru/JS/include.php)
- Выполните подключение к веб-странице внешнего скрипта из файла `secred.js`:

```
<html>
<body>
//Пишите код подключения здесь
</body>
</html>
```

- Сохраните файл под именем 2.html в папке JS_Практика.

►Команды JavaScript

- Изучите структуру кода JavaScript, описание и вызов функций JavaScript (http://www.wisdomweb.ru/JS/comand.php).
- Разместите на веб-странице кнопку (см. http://www.wisdomweb.ru/HTML/forms.php), при щелчке по которой на данной странице должен появиться текст, выводимый с использованием функции JavaScript (см. рис.1):

Рис.1

- Сохраните файл под именем 3.html в папке JS_Практика.

►Переменные в JavaScript

- Изучите способы объявления переменных в JavaScript (http://www.wisdomweb.ru/JS/var.php).
- Исправьте ошибки при описании и выводе значений переменных:

```
<html>
<body>
<script type=”text/javascript”>
var 33var = 33;
document.write(33var+”<br>”);
var str1 = Привет всем!;
document.write(str1+”<br>”);
var vaR = 288;
```

```
document.write(var+"<br>");
</script>
</body>
</html>
```

- Сохраните файл под именем 4.html в папке JS_Практика.
- Во внешнем файле `secred1.js` содержится переменные `sec1`, `sec2`, `sec3` и `sec4`. Подключите внешний файл и узнайте значение переменных путем вывода их значений на страницу.
- Сохраните изменения в файле 4.html.

►Арифметические операции в JavaScript

- Рассмотрите арифметические операции в JavaScript (http://www.wisdomweb.ru/JS/oper.php).
- Используя JavaScript, найдите значение выражения (35*y+25*x)/5+232 при x=3, y=20 и значение выражения (188*y/8+25*x/5 - 435)*6 при x=16, y=20. Выведите результат на веб-страницу.
- Сохраните файл под именем 5.html в папке JS_Практика.

► Условные конструкции JavaScript

- Ознакомьтесь с условными конструкциями в JavaScript (http://www.wisdomweb.ru/JS/ifelsw.php).
- Решите задачу, используя конструкцию `if`:

  На плоскости XOY своими координатами задана точка A. Укажите, где она расположена: на какой оси или в каком координатном угле.

  Примечание: для запроса значений переменных используйте функцию `prompt`:
  `ex1=prompt("Введите Ваше имя:", "Дмитрий");`

- Сохраните файл под именем 5.html в папке JS_Практика.
- Решите задачу, используя конструкцию `switch`:

  Имеется пронумерованный список деталей: 1 – шуруп; 2 – гайка; 3 – винт; 4 – гвоздь; 5 – болт. По номеру детали выведите ее название.

- Сохраните файл под именем 6.html в папке JS_Практика.

► Циклы JavaScript

- Ознакомьтесь с описанием циклов в JavaScript (http://www.wisdomweb.ru/JS/forlp.php).
- Решите задачу, используя цикл `for`:

  Выведите решение ребуса `ДРУГ-ГУРД=2727` (здесь различные буквы – это различные цифры, а первая цифра – не 0).

- Сохраните файл под именем 7.html в папке JS_Практика.
- Решите задачу, используя цикл while:

  Начав тренировки, спортсмен в первый день пробежал 10 км. Каждый день он увеличивал дневную норму на 10% от нормы предыдущего дня. Определите, на какой день спортсмен пробежал больше s км.
- Сохраните файл под именем 8.html в папке JS_Практика.

Тема: HTML5

Введение

HTML5 - это новая версия HTML.

Предыдущая версия HTML была выпущена в 1999 году, за это время всемирная паутина сильно изменилась и поэтому изменения требовались и от HTML.

В HTML5 было добавлено много нового, поэтому некоторые возможности, которые были доступны в HTML4 только с использованием внешних плагинов (*таких как Adobe Flash*) или клиентских скриптов теперь доступны с помощью обычных разметочных тэгов.

HTML5 все еще в разработке, но его частичную поддержку имеют все современные браузеры, поэтому использовать новые возможности Вы можете уже сейчас.

Пример

```
<video width='300' height='200' controls='controls'>
<source src='mountvideo.ogv' type='video/ogg' />
<source src='mountvideo.mp4' type='video/mp4' />
<source src='mountvideo.webm' type='video/webm' />
</video>
<audio src='ghost_k-stop.ogv' controls='controls'></audio>
```

Попробовать в редакторе

Возможности HTML5

Наиболее интересные нововведения HTML5:

- Поддержка видео и аудио (*элементы video и audio*);
- Возможности рисования на веб-страницах произвольных объектов (*элемент canvas*);
- Улучшение форм (*новые значения type для <input> и множество новых элементов и атрибутов*);
- Добавление семантических тэгов, позволяющих сделать веб-страницы более понятными для поисковых систем, браузеров и других программ и устройств читающих веб-страницы (*элементы footer, header, nav, article*);
- DOM хранилища - более функциональная альтернатива cookie.

HTML5 DOCTYPE

Все HTML5 документы должны начинаться с объявления DOCTYPE.

Предыдущие версии HTML имели несколько типов DOCTYPE. HTML5 имеет только один:

```
<!DOCTYPE html>
```

Данное объявление переводит все браузеры в нормальный режим. Браузеры не поддерживающие HTML5 в данном режиме будут интерпретировать старые теги и игнорировать новые, которые они не поддерживают.

HTML5 Видео

Одна минута видео иногда может рассказать пользователям больше, чем десятки страниц текста.

HTML5 вводит специальный тэг **<video>** позволяющий встраивать в веб-страницы видео файлы.

Обратите внимание: поддержку видео имеют браузеры: Internet Explorer 9+, Opera 10.50+, Firefox 3.5+, Chrome 3.0+, Safari 3.1+.

Атрибут src тэга <video> позволяет указать путь к видео файлу, который будет воспроизведен.

Атрибут controls отображает в плеере кнопки управления видео, а атрибуты height (*высота*) и width (*ширина*) задают размеры плеера.

Пример

```
<video src="mountvideo.ogv" width='300' height='200'
controls='controls'></video>
```

Попробовать в редакторе

Видео из предыдущего примера будет нормально проигрываться только в браузерах Opera, Firefox и Chrome. Это происходит из-за того, что в данный момент разные браузеры поддерживают разные форматы.

Opera и Firefox поддерживают формат Ogg (*с видео кодеком Theora и аудио кодеком Vorbis*) и WebM (*с видео кодеком VP8 и аудио кодеком Vorbis*), а Internet Explorer и Safari поддерживают MPEG4 (*с видео кодеком H.264 и аудио кодеком AAC*). Браузер Chrome имеет поддержку всех перечисленных форматов.

Таким образом для того, чтобы видео нормально воспроизводилось во всех браузерах необходимо добавить источники в формате Ogg и MPEG4 (*или WebM и MPEG4*).

С помощью тэга **<source>** Вы можете предоставить несколько источников видео в разных форматах для воспроизведения. Браузер будет использовать первый поддерживаемый им формат.

Пример

```
<video width="300" height="200" controls="controls">
<source src="mountvideo.ogv" type="video/ogg" />
<source src="mountvideo.mp4" type="video/mp4" />
<source src="mountvideo.webm" type="video/webm" />
</video>
```

Попробовать в редакторе

Обратите внимание: с помощью атрибута type Вы должны указать mime-тип видео файла. Видео в формате ogg должно иметь тип 'video/ogg', mp4 - 'video/mp4', а webm - 'video/webm'.

Поддержка старых браузеров

Текст помещенный в содержимое элемента video будет отображен в случае если браузер пользователя не поддерживает тэг видео.

Пример

```
<video src='mountvideo.ogv' width="300" height="200" controls="controls">Ваш
браузер
не поддерживает элемент video.</video>
```

Попробовать в редакторе

Конвертация видео в форматы поддерживаемые HTML5

Для быстрой конвертации видео в необходимые форматы (*Ogg, MPEG4 и WebM*) мы можем порекомендовать бесплатную программу Miro Video Converter.

HTML5 Аудио

С помощью нового HTML5 элемента **<audio>** Вы можете добавить на Ваш веб-сайт музыкальный трек или подкаст с помощью обычного разметочного тэга.

Обратите внимание: поддержку аудио имеют следующие браузеры: Internet Explorer 9+, Opera 10.50+, Firefox 3.5+, Chrome 3.0+, Safari 3.0+.

Атрибут src тэга <audio> позволяет указать путь к аудио файлу, а атрибут controls добавляет кнопки управления.

Пример

```
<audio src="ghost_k-stop.ogv" controls='controls'></audio>
```

Попробовать в редакторе

В предыдущем примере используется только формат Ogg, поэтому пример будет работать только в Firefox, Opera и Chrome. Для того, чтобы файл нормально проигрывался в Safari и Internet Explorer необходимо добавить также файл в формате MP3.

С помощью элемента **<source>** Вы можете предоставить несколько источников в разных форматах для воспроизведения. Браузер будет использовать первый поддерживаемый им формат.

Текст находящийся в содержимом элемента будет отображен в случае если браузер пользователя не поддерживает элемент audio.

Пример

```
<audio controls="controls">
<source src="ghost_k-stop.ogv" type="audio/ogg" />
```

```
<source src="ghost_k-stop.mp3" type="audio/mpeg" />
Данный текст будет выведен если браузер пользователя не поддерживает элемент
audio.
</audio>
```
Попробовать в редакторе

С помощью атрибута **autoplay** Вы можете заставить трек автоматически проигрываться после загрузки страницы.

Пример

```
<audio autoplay='autoplay'>
<source src="ghost_k-stop.ogv" type="audio/ogg" />
<source src="ghost_k-stop.mp3" type="audio/mpeg" />
Данный текст будет выведен если браузер пользователя не поддерживает элемент
audio.
</audio>
```
Попробовать в редакторе

Конвертация аудио в форматы поддерживаемые HTML5

С помощью представленной ранее программы Miro Video Converter Вы можете также конвертировать в необходимые форматы (*MP3 и Ogg*) аудио файлы.

Семантические тэги

В HTML4 благодаря использованию CSS Вы могли создавать страницы с хорошо понятной для пользователей визуальной структурой, но были ли эти страницы также понятны для поисковых систем или браузеров?

Например, как поисковый робот может отличить содержимое документа от навигационного меню если они размечены с помощью одинаковых div элементов?

Для того, чтобы разрешить эту проблему в HTML5 были введены **семантические тэги**. С помощью семантических тэгов Вы можете сделать страницы сайтов более понятными для поисковых систем и браузеров.

Тэг	Описание
<footer>	Определяет футер.
<header>	Определяет заголовочный блок сайта.
<nav>	Определяет навигационное меню.

Структура документа

Так выглядела общая структура документа в HTML4.

Так будет выглядеть общая структура документа в HTML5 с использованием новых тэгов разметки.

Группировка содержимого

С помощью тэга **<section>** Вы можете группировать логически связанное содержимого в документе.

Если логически связанное содержимое является автономным (*может использоваться в других документах независимо от остального содержимого на странице*) необходимо использовать вместо <section> тэг **<article>**.

```
<article>
<h1>Титаник</h1>
<p>«Титаник» — британский пароход компании «Уайт Стар Лайн». Во время первого
рейса 14
```

```
апреля 1912 года столкнулся с айсбергом и через 2 часа 40 минут затонул. На
борту
находилось 1316 пассажиров и 908 членов экипажа, всего 2224 человека. </p>
<section>
<h2>Постройка</h2>
<p>Заложен 31 марта 1909 года на верфях судостроительной компании «Харланд
энд Вольф»
в Куинс-Айленд (Белфаст, Северная Ирландия), спущен на воду 31 мая 1911 года,
прошёл
ходовые испытания 2 апреля 1912 года.</p>
</section>
<section>
<h2>Местонахождение</h2>
<p>1 сентября 1985 года экспедиция под руководством директора Института
океанологии
города Вудс-Холл, штат Массачусетс, доктора Роберта Балларда (Robert D.
Ballard)
обнаружила место залегания «Титаника» на дне Атлантического океана на глубине
3750 метров.</p>
</section>
</article>
```

Обратите внимание: предположим сайт, на котором была опубликована данная статья содержит также несколько других статей. Так как текст этой статьи является автономным т.е. мы можем понять его смысл не читая остальные статьи опубликованные на этом сайте и даже без проблем опубликовать его в любом другом месте, мы выделяем статью о Титанике целиком тэгом <article>. Подразделы данной статьи, однако, не могут является автономными т.к. они логически связанны между собой и в данном случае мы уже должны использовать тэг <section>.

Тэг aside

Тэг aside используется для выделения элементов, которые не являются частью содержимого, но косвенно с ним связаны.

Данным тэгом могут выделяться: цитаты, дополнительная информация к статье, словарь с терминами, список ссылок и т.д.

```
<article>
<h1>Титаник</h1>
<p>«Титаник» — британский пароход компании «Уайт Стар Лайн». Во время первого
рейса 14
апреля 1912 года столкнулся с айсбергом и через 2 часа 40 минут затонул. На
борту
находилось 1316 пассажиров и 908 членов экипажа, всего 2224 человека. </p>
<aside>Гибель Титаника - это одно из самых крупных кораблекрушений в истории
человечества.</aside>
</article>
```

Подсветка текста

С помощью тэга **<mark>** Вы можете выделить "важную" часть в тексте.

Пример

```
<p>Я считаю, что <mark>HTML5</mark> облегчает жизнь веб-разработчиков.</p>
```

Попробовать в редакторе

Подписи для иллюстраций

В журналах и газетах иллюстрации часто сопровождаются подписями. В HTML4 невозможно было создавать подписи не прибегая к использованию CSS.

В HTML5 это проблема решена добавлением новых тэгов: **<figure>** и **<figcaption>**.

Пример

```
<figure>
<img src='mountimg3.jpg' width='300' height='230' />
<figcaption>Скала "Братья", Западный Саян </figcaption>
</figure>
```

Попробовать в редакторе

Новые типы input в HTML5 формах

Обратите внимание: на данный момент (*Ноябрь 2011*) новые типы input поддерживаются полностью только в браузерах Opera 11+ и Chrome 12+.

- **input type=email** определяет поле, которые должно содержать email адрес. Значение введенное в поле автоматически проверяется перед отправкой на сервер.
- **input type=url** определяет поле, которое должно содержать url адрес. Значение введенное в поле автоматически проверяется перед отправкой на сервер.
- **input type=tel** определяет поле для ввода телефонного номера. С помощью атрибута pattern Вы может установить формат принимаемого телефонного номера. Формат задается с помощью регулярных выражений.
- **input type=number** определяет поле, которое должно содержать числа. Вы можете ограничивать диапазон принимаемых чисел с помощью атрибутов min (*минимальное допустимое число*) и max (*максимальное допустимое число*). С помощью атрибута step Вы можете задать шаг допустимых чисел (*к примеру если шаг равен 2, то в поле могут вводиться числа 0,2,4,6 и т.д.*)
- **input type=range** определяет поле, которые может содержать значения в определенном интервале. Отображается как ползунок, который можно перетаскивать мышкой. Вы можете ограничивать диапазон принимаемых чисел с помощью атрибутов min (*минимальное допустимое число*) и max (*максимальное допустимое число*). С помощью атрибута step Вы можете задать шаг допустимых чисел (*к примеру если шаг равен 2, то в поле могут вводиться числа 0,2,4,6 и т.д.*)
- **input type=search** определяет поле поиска (*может использоваться например для создания поиска по сайту*).

Пример

```
<input name='email' type='email' value='Не email' />
<input name='url' type='url' value='Не url' />
<input name='tel1' type='tel' pattern='8[0-9]{10}' />
<input name='tel2' type='tel' pattern='[0-9]{2,3}-[0-9]{2}-[0-9]{2}' />
<input name='number' type='number' value='10' />
<input name='range' type='range' min='1' max='5' />
<input name='search' type='search' value='Поиск по сайту' />
<input name='color' type='color' />
```

Попробовать в редакторе

Новые элементы в HTML5 формах

Обратите внимание: на данный момент (*Ноябрь 2011*) новые элементы форм поддерживаются полностью только в браузерах Firefox 4+ и Opera 10.50+.

- **datalist** позволяет привязать список к полям формы. Значения списка будут выводится как поисковые подсказки во время ввода информации в поле связанное с ним.
- **keygen** позволяет генерировать открытые и закрытые ключи, которые используются для безопасной связи с сервером.
- **output** может использоваться для вывода различной информации. С помощью атрибута for можно указать связанные поля.

Пример

```
<input name='city' list="city" />
<datalist id="city">
<option label="Москва" value="Москва, Россия" />
<option label="Санкт-Петербург" value="Санкт-Петербург, Россия" />
<option label="Новосибирск" value="Новосибирск, Новосибирская область,
Россия" />
</datalist>
<keygen name='keygen' />
<output name='result' for='first second'>100</output>
```

Попробовать в редакторе

Новые атрибуты в HTML5 формах

Обратите внимание: на данный момент (*Ноябрь 2011*) новые атрибуты элементов формы поддерживаются только в браузерах Firefox 4+, Opera 11+ и Chrome 10+.

- **autofocus** делает поле активным после загрузки страницы (*может использоваться со всеми типами input*).
- **form** указывает форму, которой принадлежит данное поле (*может использоваться со всеми типами input*).
- **multiple** указывает, что данное поле может принимать несколько значений одновременно (*может использоваться с input типов email и file*).
- **novalidate** указывает, что данное поле не должно проверяться перед отправкой (*может использоваться с form и input*).
- **placeholder** отображает текст-подсказку в поле (*может использоваться с input следующих типов: text, search, url, tel, email и password.*).
- **required** указывает, что данное поле должно быть обязательно заполнено перед отправкой.

Пример

```
<input type='text' autofocus="autofocus" /><br /><br />
<input type='file' multiple='multiple' /><br /><br />
<input type='text' form='form1' /><br /><br />
<form action='html5.php' novalidate='novalidate'>
<input type='email' placeholder='Введите Ваш email' /><br /><br />
<input type='text' required="required" />
```

Попробовать в редакторе

Выбор даты

В HTML5 были добавлены новые элементы ввода позволяющие удобно выбирать дату и время.

Обратите внимание: на данный момент (*Ноябрь 2011*) поля для выбора даты поддерживаются только в браузерах Opera 9+, Chrome 10+ и Safari 5.1+.

- **date** позволяет выбрать дату в формате год-месяц-день_месяца.
- **time** позволяет выбрать время.
- **datetime** позволяет выбрать дату в формате год-месяц-день_месяцаTвремяZ (*отчет ведется по глобальному времени*).
- **datetime-local** позволяет выбрать дату в формате год-месяц-день_месяцаTвремя (*отчет ведется по местному времени*).
- **month** позволяет выбрать дату в формате год-месяц.
- **week** позволяет выбрать дату в формате год-Wнеделя.

Выберите дату:
Выберите время:
Выберите глобальное время и дату:
Выберите местное время и дату:
Выберите месяц:
Выберите неделю:

```
<input type='date' /><br />
<input type='time' /><br />
<input type='datetime' /><br />
<input type='datetime-local' /><br />
<input type='month' /><br />
<input type='week' /><br />
```

Попробовать в редакторе

Веб Хранилища

Обратите внимание: все HTML5 технологии, которые будут разобраны в данной и последующих главах этого учебника требуют знания JavaScript для их использования. Вы можете ознакомится с JavaScript в нашем JavaScript учебнике.

Веб хранилища представляют собой более функциональную альтернативу cookie.

Преимущества веб хранилищ:

- Можно хранить неограниченные объемы информации;
- Информация, сохраненная в хранилищах, доступна даже без подключения к интернету;
- Данные, находящиеся в хранилище, не отсылаются при каждом запросе страниц;
- Информацию более удобно сохранять и извлекать;
- Хранилища более безопасны чем cookie.

Веб хранилища поддерживаются в следующих браузерах: IE8+, Safari 4+, Chrome 4+, Firefox 3.5+, Opera 10.50+.

Сохранение данных

Вы можете сохранять данные используя два специальных объекта: **sessionStorage** и **localStorage**.

Обращаться к данным объектам можно с помощью JavaScript и других клиентских скриптов.

Обратите внимание: каждый веб-сайт имеет доступ только к своим данным в хранилище и поэтому не может обращаться к данным, которые были сохранены другими сайтами.

Использование sessionStorage

Объект sessionStorage сохраняет данные в течении пользовательской сессии. Когда браузер пользователя будет закрыт данные сохраненные в объекте будут удалены.

Пример

```
//Сохраняем данные
sessionStorage.html='HTML5 предоставляет множество интересных возможностей.';
sessionStorage.wisdomweb='Вы изучаете HTML5 на wisdomweb.ru.';
/* В течении пользовательской сессии мы можем извлечь сохраненные ранее
данные
следующим образом: */
document.write(sessionStorage.html+'<br />');
document.write(sessionStorage.wisdomweb);
```

Попробовать в редакторе

Быстрый просмотр

Данные в хранилище доступны со всех страниц сайта, а не только с той с которой они были сохранены.

Пример

```
<script>
<script type='text/javascript'>
function saveData(){
   sessionStorage.data='Я изучаю HTML5 на wisdomweb.ru';
   alert('Строка "Я изучаю HTML5 на wisdomweb.ru" была сохранена в
хранилище.');
}
</script>
<p>Данные в хранилище доступны со всех страниц сайта, а не только с той с
которой они были сохранены.</p>
<input type='button' value='Сохранить данные из хранилища'
onclick='saveData()' />
<br /><br />
<iframe src='storage1.html' />
```

Попробовать в редакторе

Использование localStorage

Объект localStorage сохраняет данные на неограниченный период времени.

Данные сохраненные в данном объекте будут доступны даже без подключения к интернету.

Пример

```
<script>
function save(){
localStorage.data='Веб хранилища - это более функциональная альтернатива
cookie.';
}
function load(){
alert(localStorage.data);
}
</script>
<input type='button' value='Сохранить данные' onclick='save()' />
<input type='button' value='Извлечь данные' onclick='load()' />
<p><b>Обратите внимание:</b> Вы сможете извлечь сохраненные данные даже после
перезагрузки браузера.</p>
```

Попробовать в редакторе

Drag and Drop

Обратите внимание: браузеры Internet Explorer 9+, Chrome, Firefox, Opera, и Safari имеют поддержку drag and drop.

Технология **drag and drop** (иногда может упоминаться как drag'n'drop) позволяет сделать элементы на Ваших веб-страницах перетаскиваемыми (совсем как окна с программами в операционной системе).

Ранее это было возможным только с помощью реализаций сложных JavaScript функций или подключением jQuery. Сейчас технология drag and drop является частью спецификации HTML5 и ее поддержка встроена во все современные веб-браузеры.

Как сделать элемент перетаскиваемым

Практически любые элементы на веб-страницах можно сделать перетаскиваемыми включая картинки, ссылки и обычные текстовые элементы. Для того, чтобы сделать элемент перетаскиваемым необходимо добавить к нему HTML5 атрибут **draggable** со значением **true**.

Пример

```
<img draggable='true'>
```

Попробовать в редакторе

Мало просто сделать элемент перетаскиваемым (Вы можете в этом убедится запустив предыдущий пример), чтобы извлечь из этого какую-либо пользу мы должны дополнительно обработать перетаскиваемый элемент с помощью специальных **событий перетаскивания**, которые были добавлены в HTML5:

Событие	Описание
dragstart	Выполнится в начале операции перетаскивания

ondrag	Выполнится во время перетаскивания элемента
ondragenter	Выполнится когда перетаскиваемый элемент будет наведен на элемент, который может его принять
ondragleave	Выполнится когда перетаскиваемый элемент будет выведен за пределы границ элемента, который может его принять
ondragover	Выполнится когда перетаскиваемый элемент будет перемещаться в пределах границ элемента, который может его принять
ondrop	Выполнится когда перетаскиваемый элемент будет перемещен в принимающий его элемент
ondropend	Выполнится в конце операции перетаскивания

Обработка элемента во время перетаскивания

Всю обработку перетаскивания элемента можно условно разбить на два шага:

1. Сохранение данных перетаскиваемого элемента в начале операции перетаскивания.

2. Извлечение ранее сохраненных данных, после того, как перетаскиваемый элемент будет перемещен в принимающий его элемент.

Для того, чтобы сохранить данные мы должны воспользоваться методом **setData("тип_данных","сохраняемые_данные")** HTML5 объекта **DataTransfer**. Данные перетаскиваемого элемента необходимо сохранять в самом начале перетаскивания, то есть во время события **ondragstart**.

Пример

```
<script>
function dragImg(e) {
alert(e.target.id);
e.dataTransfer.setData("text/plain",e.target.id);
}
function dragEl(e) {
alert(e.target.innerHTML);
e.dataTransfer.setData("text/html",e.target.innerHTML);
}
</script>
<img src="gimg3.gif" draggable="true" ondragstart="dragImg(event)"
id="draggable-element">
<div id="dragel" draggable="true" ondragstart="dragEl(event)">Меня можно
перетаскивать!</div>
```

Попробовать в редакторе

Для того, чтобы извлечь ранее сохраненные данные мы должны воспользоваться методом **getData("тип_данных")** объекта **DataTransfer**. Данные перетаскиваемого элемента необходимо извлекать после того, как перетаскиваемый элемент будет перемещен в область принимающего его элемента, то есть во время события **ondrop**.

Обратите внимание: тип данных во время сохранения и извлечения должен совпадать, то есть если Вы сохраняли данные с типом "text/plain" Вы должны использовать "text/plain" и для их извлечения.

По умолчанию элементы не могут принимать другие элементы, поэтому если мы хотим создать принимающий элемент мы должны привязать к нему обработчик события **ondragover** и передать ему **event.preventDefault()**.

Пример

```
<script>
function makeDroppable(e) {
e.preventDefault();
}
function dragImg(e) {
e.dataTransfer.setData("text/plain",e.target.id);
}
function dropImg(e) {
var rdata = e.dataTransfer.getData("text/plain");
e.target.innerHTML="";
e.target.appendChild(document.getElementById(rdata))
}
</script>
<img src="gimg3.gif" draggable="true" ondragstart="dragImg(event)"
id="draggable-element">
<div id="dropel" ondrop="dropImg(event)"
ondragover="makeDroppable(event)">Перетащите элемент сюда!</div>
```

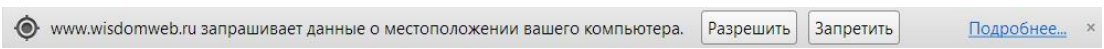
Попробовать в редакторе

О геолокации в HTML5

Обратите внимание: IE9+, Chrome, Firefox, Opera и Safari имеют поддержку данной технологии.

С помощью HTML5 геолокации Вы можете определить местоположение пользователя. Так как информация о местоположении может нарушать конфиденциальность, прежде чем сервер сможет получить данную информацию пользователь должен явным образом подтвердить, что не имеет возражений на этот счет.

Например вот так выглядит окно, которое появляется в браузере Chrome при попытке считать информацию о местоположении:



Обратите внимание: координаты местоположения определяются более точно на мобильных устройствах, потому что они имеют свой собственный встроенный GPS приемник. Местоположение пользователей на стационарных компьютерах определяется путем комбинированного анализа IP адреса самого компьютера и IP адресов окружающих WI-FI роутеров в радиусе покрытия которых компьютер находится в данный момент.

Использование геолокации в HTML5

Текущая позиция пользователя может быть определена с помощью метода **getCurrentPosition()** объекта **navigator.geolocation**.

```
navigator.geolocation.getCurrentPosition(success_function, error_function,
options);
```

success_function - имя функции, которая будет вызвана в случае если координаты пользователя будут считаны удачно.

error_function - имя функции, которая будет вызвана если при считывании координат произойдет ошибка.

options - позволяет задать настройки, которые будут использованы при считывании координат. Возможные значения: *enableHighAccuracy* - если имеет значение true браузер будет пытаться определить местоположение как можно точнее; *timeout* - устанавливает максимально допустимое время, которое браузер может использовать для считывания данных (по умолчанию время считывания не ограничено); *maximumAge* - как долго браузер будет хранить к кэше предыдущее сохраненное значение.

В случае если пользователь разрешил использовать данные о его местоположении и они были удачно считаны браузером. В функцию **success_function** в качестве параметра будет передан объект, который содержит свойство **timestamp** (содержит время считывание координат) и объект **coords**, который имеет следующие свойства:

Событие	Описание
latitude и longitude	Широта и долгота местоположения пользователя
altitude	Высота от уровня моря в метрах
accuracy	Точность определения широты и долготы (чем больше число тем меньше точность)
altitudeAccuracy	Точность определения высоты (чем больше число тем меньше точность)
heading	Показывает направление пользователя в градусах (т.е. 0 градусов значит, что пользователь направляется на север, 180 на юг и т.д.).
speed	Скорость перемещения в метрах в секунду

Пример

```
<script>
function getCoordinates() {
   navigator.geolocation.getCurrentPosition(showCoordinates);
}

function showCoordinates(position) {
   document.write('Широта: ' + position.coords.latitude + "<br />");
   document.write('Долгота: ' + position.coords.longitude);
}
</script>
```

Попробовать в редакторе

В случае если при считывании координат пользователя произошла какая-либо ошибка будет вызвана функция **error_function**. Возможные ошибки при считывании координат:

Ошибка	Описание
PERMISSION_DENIED	Пользователь запретил считывать информацию о его местоположении.
POSITION_UNAVAILABLE	Браузер не может определить местоположение пользователя.
TIMEOUT	Браузер не успел определить местоположение пользователя в выделенное ему время.
UNKNOWN_ERROR	Во время определения местоположения произошла неизвестная ошибка.

Пример

```
<script>
function showError(error) {
   switch(error.code) {
      case error.PERMISSION_DENIED:
        alert("Пользователь запретил считывать информацию о его
местоположении.");
        break;
      case error.POSITION_UNAVAILABLE:
        alert("Браузер не смог определить местоположение пользователя.");
        break;
      case error.TIMEOUT:
        alert("Браузер не успел определить местоположение за выделенное ему
время.");
        break;
      case error.UNKNOWN_ERROR:
        alert("Произошла неопределенная ошибка.");
        break;
   }
}
</script>
```

Попробовать в редакторе

Отслеживание изменения местоположения

С помощью метода **watchPosition()** Вы можете считывать данные о месте положения пользователя через определенные интервалы времени. После запуска данный метод возвращает специальный индикатор, который может использоваться позднее для завершения считывания с помощью метода **clearWatch()**.

Обратите внимание: чтобы увидеть как изменяются координаты во время перемещения мы рекомендуем тестировать данный пример на мобильных устройствах со встроенным GPS.

Пример

```
<script>
function getCoordinates() {
   watchID = navigator.geolocation.watchPosition(showCoordinates);
}

function stop() {
```

```
   if (watchID) {
      navigator.geolocation.clearWatch(watchID);
      watchID = null;
   }
}
</script>
```
Попробовать в редакторе

Поддержка старых браузеров

Проверить имеет ли браузер пользователя поддержку данной технологии или нет Вы можете с помощью следующей конструкции:

```
<script>
if (navigator.geolocation) {
   //Код для браузеров поддерживающих геолокацию
} else {
   //Код для браузеров неподдерживающих геолокацию
}
</script>
```

Знакомство с web worker

Обратите внимание: IE10+, Chrome, Firefox, Opera и Safari имеют поддержку данной технологии.

В HTML все JavaScript скрипты на странице могли выполнятся только поочередно. В некоторых случаях пользователи должны были ожидать завершения выполнения скрипта пока они смогут продолжить взаимодействие с веб-страницами.

Благодаря технологии **web worker** в HTML5 стало возможным создание скриптов, которые будут выполняться в фоновом режиме и не будут влиять на скорость загрузки основной страницы.

Создание web worker

Так как web worker работают параллельно с основной страницей их код всегда должен быть вынесен в отдельный .js файл. После этого на нашей основной странице необходимо создать новый объект Worker с ссылкой на внешний файл скрипта:

```
var worker = new Worker(ссылка_на_файл_скрипта);
```

Если указанный файл существует браузер начнет загрузку скрипта новым потоком асинхронно. После того, как скрипт будет загружен мы можем запустить наш worker с помощью:

```
worker.postMessage();
```

Взаимодействие с web worker

Взаимодействие web worker с основной веб-страницей происходит с помощью специальных событий и метода postMessage(). Например событие message срабатывает, когда worker получает какое-либо сообщение.

Пример

```
<script>
//Создадим новый объект worker
var worker=new Worker("worker1.js");
//Отправим сообщение "Я изучаю HTML5 на" в наш worker
worker.postMessage("Я изучаю HTML5 на ");
//Выведем ответ полученный от worker на страницу
worker.addEventListener('message', function(event) {
  document.getElementById('reply').innerHTML='Наш worker ответил: <b>' +
event.data + "</b>";
});
</script>
```

Попробовать в редакторе

Всегда имейте ввиду что web worker не имеют доступа к:

- DOM структуре основной страницы
- Объекту window
- Объекту document
- Объекту parent

Обратите внимание: помимо текстовых строк мы можем передавать в worker JSON объекты. Узнать о JSON объектах больше Вы можете здесь.

Обратите внимание: не следует применять web worker для таких простых задач как в примере. Обычно использование web worker оправдано только для задач, которые требуют значительных ресурсов компьютера для их выполнения, например кэширование больших массивов с данными для дальнейшего использование, проверка текста на ошибки, подсветка синтаксиса или другие операции с форматированием текста в реальном времени, обновление больших объемов информации в базах данных, анализирование видио или аудио файлов и др.

Выключение web worker

После того, как worker был создан он будет ожидать сообщения от основной страницы даже когда выполнение его кода завершится, поэтому Вы всегда должны явно завершать работу worker когда он больше не нужен.

Чтобы завершить работу worker и освободить ресурсы компьютера Вы должны использовать метод termitane():

```
worker.terminate();
```

Использование web worker в старых браузерах

Прежде чем использовать web worker необходимо проверить поддерживает ли эту технологию браузер пользователя.

Это можно сделать с помощью следующей конструкции:

```
if( typeof(Worker) !== "undefined" ) {
  //Браузер пользователя имеет поддержку web worker
}
else {
  //Браузер пользователя не поддерживает web worker
}
```

Усовершенствованное кэширование страниц

Обратите внимание: IE10+, Chrome, Firefox, Opera и Safari имеют поддержку данной технологии.

HTML5 расширяет возможности браузерного кэширования. Теперь веб-страницы могут быть полностью доступны для пользователей даже без подключения к интернету.

Использование HTML5 кэширования дает следующие преимущества:

- Возможность просмотра веб-страниц пользователями даже без подключения к интернету.
- Увеличение скорости загрузки страниц - страницы хранятся локально на компьютере пользователя и поэтому будут загружаться намного быстрее.
- Уменьшение нагрузки на сервер - серверу не придется обрабатывать некоторые из запросов пользователей.

Пример использования HTML5 кэширования

Пример

```
<!DOCTYPE HTML>
<html manifest="example.appcache">
...Содержимое документа...
</html>
```

Попробовать в редакторе

Декларирование HTML5 кэширования

Для того, чтобы использовать механизм кэширования на Ваших веб-страницах необходимо добавить к тэгу <html> атрибут manifest и в качестве его значения указать путь к специальному файлу, в котором декларируются параметры кэширования.

```
<html manifest="example.appcache">
```

Если данный атрибут не был задан на веб-странице и ссылка на нее отсутствует в файле кэширования, страница не будет кэширована.

Файл кэширования может иметь любое расширение (например .appcache или .mf), но обязан иметь специальный MIME тип: "text/cache-manifest".

В некоторых случаях веб-серверу может потребоваться дополнительная настройка, чтобы обслуживать данный MIME тип. Например, чтобы настроить веб-сервер Apache необходимо добавить следующий код в .htaccess файл:

```
AddType text/cache-manifest .appcache
```

Содержимое файла кэширования

Файл кэширования является обычным текстовым файлом, который указывает браузеру какие файлы необходимо кэшировать.

Файл может содержать три секции:

- CACHE MANIFEST - в данной секции указываются ссылки на файлы, которые необходимо кэшировать. Браузер будет автоматически кэшировать все перечисленные файлы сразу после первой загрузки.
- NETWORK - в данной секции указываются файлы, которые требуют постоянного подключения к интернету. Браузер не будет кэшировать файлы перечисленные в данной секции.
- FALLBACK - если файлы указанные в этой секции будут по какой-либо причине будут недоступны пользователи автоматически будут перенаправляться на другой указанный файл.

Секция CACHE MANIFEST обязательно должна присутствовать во всех файлах кэширования. Секции NETWORK и FALLBACK могут отсутствовать.

Пример файла кэширования:

```
CACHE MANIFEST
#В данной секции перечислены файлы, которые будут кэшированы
index.html
flower.png

NETWORK
#Здесь перечислены файлы, которые требуют подключение к интернету
login-page.php

FALLBACK
#Если mob.html недоступен пользователь будет перенаправлен на offline.html
/mob.html /offline.html
#Если какой-либо из HTML файлов недоступен пользователь будет перенаправлен
на offline.html
*.html /offline.html
```

Обратите внимание: веб-страница которая ссылается на файл кэширования будет автоматически прокэширован, поэтому включать ее в сам файл кэширования необязательно, но тем не менее рекомендуется.

Обратите внимание: некоторые браузеры могут иметь ограничение на размер кэшируемого содержимого на одном сайте.

Обновление кэшированных файлов

После того, как файлы будет кэшированы браузер продолжит показывать их кэшированную версию снова и снова даже если Вы измените содержимое этих файлов на сервере.

Для того, чтобы обновить кэшированое содержимое Вы должны сделать одно из следующих действий:

- Очистить кэш в браузере пользователя
- Обновить содержимое файла кэширования
- Обновить кэш браузера программно (с помощью JavaScript)

Для того, чтобы обновить содержимое файла Вы можете использовать следующий трюк: вместо того, чтобы обновлять непосредственно содержимое файла Вы можете добавить к нему комментарий с указанием даты изменения и/или версией файла и в будущем изменять только содержимое этого комментария всякий раз, когда Вы хотите обновить кэшированное содержимое:

```
CACHE MANIFEST
# 29.09.2013 v1.2
index.html
flower.png
somescript.js
```

Средства контроля качества обучения

Вопросы к зачету:

1. Понятие «телекоммуникационная сеть».Аппаратное обеспечение телекоммуникационной сети.
2. Программное обеспечение телекоммуникационной сети. Способы подключения к сети.
3. Сетевые протоколы. Многоуровневая сетевая модель.
4. Адресация в интернете. Ресурсы и сервисы интернета.
5. Поисковые системы интернета. Создание HTML документа, элементы, атрибуты, заголовки.
6. Форматирование в HTML. Каскадные таблицы стилей.
7. Добавление изображений. Таблицы HTML.
8. Списки в HTML. Формы.
9. Фреймы. Объявление типа документа, секция head, скрипты, спецсимволы.
10. HTML5: назначение; видео; аудио; семантика.HTML5: формы; хранилища; технология Draganddrop.
11. HTML5: Webworker; геолокация; кэширование.HTML5: canvas; стили.
12. CSS3: введение; фон; цвет.CSS3: границы; шрифт; текст.
13. CSS3: прозрачность; трансформации; градиент.CSS3: переходы; анимация; столбцы.
14. Основные понятия и термины web-программирования. Основы JavaScript.
15. Включения, команды, комментарии JavaScript. Переменные, операторы, выражения JavaScript.
16. Конструкции, окна, функции JavaScript. Циклы, события, ошибки JavaScript.
17. Расписание JavaScript. Специальные операторы JavaScript.
18. Проверка, символы JavaScript. Объекты JavaScript, свойства и методы объектов.
19. Массивы JavaScript. Строки, дата, математические операции JavaScript.
20. Регулярные выражения JavaScript. Объекты пользователя JavaScript.
21. Структура JavaScript. Объектная модель браузера (BOM), свойства и методы объектов.
22. ОбъектNavigator. ОбъектScreen.
23. ОбъектHistory. ОбъектLocation.