MINISTRY FOR DEVELOPMENT OF INFORMATION TECHNOLOGIES AND COMMUNICATIONS OF THE REPUBLIC OF UZBEKISTAN

FERGANA BRANCH OF THE TASHKENT UNIVERSITY OF INFORMATION TECHNOLOGIES NAMED AFTER MUHAMMAD AL-KHWARIZMI

Faculty of “Telecommunication technologies and professional education”

Department of “Program engineering”

METHODOLOGICAL GUIDELINESBY

to perform laboratory and practical work on the discipline

“WEB APPLICATION DEVELOPMENT”

Sphere knowledge:	*300 000 — Manufacturing and technique sphere;*
Educational sphere:	*330000 - Computer technologies and informatics*
Educational direction:	*5330600 - Program engineering*
	5330500 - Computer engineering (“Computer engineering”, “IT Service”, “Information Security”, “Multimedia Technologies”)

Fergana-2017 years

Methodical instructions for the course "Creating web applications" was approved by the council of the branch of TUIT named after Muhammad al-Khorazmiy and designed for students in the directions 5330500-Computer Engineering (Computer Engineering, IT Service) and 5330600-Program engineering.

Compiled by:

Abdukadirov B. - Senior Lecturer of the Department "Program Engineering", Fergana Branch of TUIT named after Muhammad al-Khorazmiy

Reviewers:

Abdullaev A. - prof. Department of "Software Engineering", Fergana branch of TUIT named after Muhammad al-Khorazmiy

Kholmurzaev A. - Head of the Department "Architecture", Ferghana Polytechnic Institute

Approved on the advice of the branch of TUIT
named after Muhammad al-Khorazmiy
Protocol No. ______ dated ___________201__

Dean of the Faculty of "Telecommunication Technologies
and Professional Education"
__________ O. Kuldashev

Session of the Department "Program Engineering" of the Fergana branch of the TUIT Protocol No. ___ of _______ 201__

The purpose of methodical instructions - methodical maintenance of process of carrying out of laboratory and practical works on discipline «Web application development». The methodical instructions are calculated on use in educational process by preparation of bachelors in a directions «Program engineering» and «Computer Engineering (Computer Engineering, IT Service)».

Author of course **B.Abdukadirov**

Reviewers:
docent of Information technologies department FerSU **A. Mirzakulov**

It is considered at session educational methodical council of Branch of TUIT (the report № __from 2017)

It is considered at the meeting of department of «Program Engineering» (the report № ____from 2017)

Head of the department of "Program Engineering": F.Mulaydinov

Approved by:
Vise–director of "Education and scientific works "
_______ F. Polvonov
___ ____ 2017

Advised by
Head of the department
"Languages and sport" ________
N. Qurbonov ___ _____ 2017

Approved by:
Head of "Telecommunication systems and professional education faculty"
O. Kuldashov
___ ______ 2017

Advised by:
Head of "Educational department"
SH. Umarov
___ _______ 2017

INTRODUCTION

The goal of the course is to learn the technology of web programming, create static and dynamic web pages, learn modern technologies, create web pages such as HTML5, CSS3, JavaScript, jQuery, LAMP / WAMP platforms, MVC frameworks, AJAX technologies, create a variety of web applications: from protozoa single-window applications to distributed database management programs, use a variety of utilities that provide data bases and other tasks.

As a result of the discipline, students should have an idea:

• about the technology of web programming;
• on the methods of developing structures and web design;
• on the management of databases;
• application of knowledge of web programming in practice.

know:

• Basic concepts and functions of web programming.
• The robustness and fault tolerance of web programming.
• Custom Websites.
• Work with databases in web programming.

Be able to:

• Creating static sites.
• Creation of dynamic sites.
• Work with CSS2 and CSS3 technology.
• Work with HTML5.
• Work with the JavaScript programming language.
• Work with the jQuery library.
• Work with LAMP and WAMP platforms.
• Work with MVC frameworks.
• Work with AJAX technology
• Work with databases
• Work with graphics on a web page.
• Create modern sites.

The course is based on the materials of the disciplines "Web Programming", "Computer Networks", "Web Application Development", "Internet Technologies", "Programming Technology", "Programming Methods".

Laboratory №1
Theme: Installing the Web application creation environment

1. Objective

In the course of this lab work, you need to master the basic structural tags and use the HTML language to create a web page layout. Exploring the installation environment for creating web applications on a personal computer

2. Required tools

A personal computer, among the creatures of web applications emmet-sublime-master or Notepad ++, the text editor MS Word

3. Brief information from theory

In computing, a web application or web app is a client–server computer program in which the client (including the user interface and client-side logic) runs in a web browser.[1] Common web applications include webmail, online retail sales, online auctions, wikis, instant messaging services and many other functions.

The general distinction between a dynamic web page of any kind and a "web application" is unclear. Web sites most likely to be referred to as "web applications" are those which have similar functionality to a desktop software application, or to a mobile app. HTML5 introduced explicit language support for making applications that are loaded as web pages, but can store data locally and continue to function while offline.

Single-page applications are more application-like because they reject the more typical web paradigm of moving between distinct pages with different URLs. Single-page frameworks like Sencha Touch and AngularJS might be used to speed development of such a web app for a mobile platform.

In earlier computing models like client–server, the processing load for the application was shared between code on the server and code installed on each client locally. In other words, an application had its own pre-compiled client program which served as its user interface and had to be separately installed on each user's personal computer. An upgrade to the server-side code of the application would typically also require an upgrade to the client-side code installed on each user workstation, adding to the support cost and decreasing productivity. In addition, both the client and server components of the application were usually tightly bound to a particular computer architecture and operating system and porting them to others was often prohibitively expensive for all but the largest applications. (Today, of course, native apps for mobile devices are also hobbled by some or all of the foregoing issues.)

In contrast, web applications use web documents written in a standard format such as HTML and JavaScript, which are supported by a variety of web browsers. Web applications can be considered as a specific variant of client–server software where the client software is downloaded to the client machine when visiting the relevant web page, using standard procedures such as HTTP. Client web software updates may happen each time the web page is visited. During the session, the web browser interprets and displays the pages, and acts as the universal client for any web application.

In the early days of the Web, each individual web page was delivered to the client as a static document, but the sequence of pages could still provide an interactive experience, as user input was returned through web form elements embedded in the page markup. However, every significant change to the web page required a round trip back to the server to refresh the entire page.

Applications are usually broken into logical chunks called "tiers", where every tier is assigned a role.[5] Traditional applications consist only of 1 tier, which resides on the client machine, but web applications lend themselves to an n-tiered approach by nature.[5] Though many variations are possible, the most common structure is the three-tiered application.[5] In its most common form, the three tiers are called presentation, application and storage, in this order. A web browser is the first tier (presentation), an engine using some dynamic Web content technology (such as ASP, CGI, ColdFusion, Dart, JSP/Java, Node.js, PHP, Python or Ruby on Rails) is the middle tier (application logic), and a database is the third tier (storage).[5] The web browser sends requests to the middle tier, which services them by making queries and updates against the database and generates a user interface.

For more complex applications, a 3-tier solution may fall short, and it may be beneficial to use an n-tiered approach, where the greatest benefit is breaking the business logic, which resides on the application tier, into a more fine-grained model.[5] Another benefit may be adding an integration tier that separates the data tier from the rest of tiers by providing an easy-to-use interface to access the data.[5] For example, the client data would be accessed by calling a "list_clients()" function instead of making an SQL query directly against the client table on the database. This allows the underlying database to be replaced without making any change to the other tiers.[5]

There are some who view a web application as a two-tier architecture. This can be a "smart" client that performs all the work and queries a "dumb" server, or a "dumb" client that relies on a "smart" server.[5] The client would handle the presentation tier, the server would have the database (storage tier), and the business logic (application tier) would be on one of them or on both.[5] While this increases the scalability of the applications and separates the display and the database, it still doesn't allow for true specialization of layers, so most applications will outgrow this model.[5]

An emerging strategy for application software companies is to provide web access to software previously distributed as local applications. Depending on the type of application, it may require the development of an entirely different browser-based interface, or merely adapting an existing application to use different presentation technology. These programs allow the user to pay a monthly or yearly fee for use of a software application without having to install it on a local hard drive. A company which follows this strategy is known as an application service provider (ASP), and ASPs are currently receiving much attention in the software industry.

Security breaches on these kinds of applications are a major concern because it can involve both enterprise information and private customer data. Protecting these assets is an important part of any web application and there are some key operational areas that must be included in the development process.[6] This includes processes for authentication, authorization, asset handling, input, and logging and auditing. Building security into the applications from the beginning can be more effective and less disruptive in the long run.

Cloud Computing model web applications are software as a service (SaaS). There are business applications provided as SaaS for enterprises for fixed or usage dependent fee. Other web applications are offered free of charge, often generating income from advertisements shown in web application interface.

Development

Writing a web application is often simplified by open source software such as Django, Ruby on Rails or Symfony called web application frameworks. These frameworks facilitate rapid application development by allowing a development team to focus on the parts of their application which are unique to their goals without having to resolve common development issues such as user management. While many of these frameworks are open source, this is by no means a requirement.

The use of web application frameworks can often reduce the number of errors in a program, both by making the code simpler, and by allowing one team to concentrate on the framework while another focuses on a specified use case. In applications which are exposed to constant hacking attempts on the Internet, security-related problems can be caused by errors in the program. Frameworks can also promote the use of best practices such as GET after POST.

In addition, there is potential for the development of applications on Internet operating systems, although currently there are not many viable platforms that fit this model.

Applications

Examples of browser applications are simple office software (word processors, online spreadsheets, and presentation tools), but can also include more advanced applications such as project management, computer-aided design, video editing and point-of-sale.

4. The order of performance of work

Install emmet-sublime-master or Notepad ++ among web application creators. Creating an HTML project on the environment of creating web applications emmet-sublijme-master or Notepad ++.

Contents of the report

1. The name of the laboratory work.
2. The purpose of laboratory work.
3. Required tools.
4. Brief information from theory.
5. The decision of an example on a variant.
6. Conclusion on the work.

Recommended literature

1. Burlakov MV CorelDRAW 12. - St. Petersburg; BHV-Petersburg, 2004. - 688 p.
2. Jamsa Chris. Effective tutorial on creative Web-design. HTML, XHTML, CSS, JavaScript, PHP, ASP, ActiveX. Text, graphics, sound and animation. Per with English / Chris Jamsa, Conrad King, Andy Anderson - M .: "DiSoftTUP", 2005.- 672 p.
3. Inkova NA, Zaitseva EA, Kuzmina NV, Tolstykh SG The creation of Web-sites: Educational-methodical manual. Part 5. Tambov: Publishing house of Tamb. state. tech. University, 2005. - 56 p.
4. Orlov L. V. Web-site without secrets. L.V. Orlov. - 2 nd ed. - Moscow: Buk-press, 2006. - 512 p.
5. Polonskaya E.L. The HTML language. Self-teacher: - M .: Publishing house "Williams", 2005. - 320 p.
6. Creation of Web-pages and Web-sites. Textbook: [учеб. allowance] / ed. VN Pechnikova. - Moscow: Publishing House Triumph, 2006.- 464 p.
7. Yakushev, L. V. We begin to work on the Internet. Quick Start Guide. - M .: Publishing house "Williams", 2006. -128 s

Laboratory №2
Theme: HTML Basics

1. Objective

Learn the basic HTML concepts necessary to understand the principles of developing web documents, study the markup language of hypertext and its main tags, learn how to create web pages using HTML tags

2. Required tools

A personal computer, among the creatures of web applications emmet-sublime-master or Notepad ++, the text editor MS Word

3. Brief information from theory

A static web page (sometimes called a flat page/stationary page) is a web page that is delivered to the user exactly as stored, in contrast to dynamic web pages, which are generated by a web application.

Consequently, a static web page displays the same information for all users, from all contexts, subject to modern capabilities of a web server to negotiate content-type or language of the document where such versions are available and the server is configured to do so.

Static web pages are often HTML documents stored as files in the file system and made available by the web server over HTTP (nevertheless URLs ending with ".html" are not always static). However, loose interpretations of the term could include web pages stored in a database, and could even include pages formatted using a template and served through an application server, as long as the page served is unchanging and presented essentially as stored.

Static web pages are suitable for the contents that never or rarely need to be updated, though modern static site generators are changing. Maintaining large numbers of static pages as files can be impractical without automated tools, such as Static site generators described in Web template system. Any personalization or interactivity has to run client-side, which is restricting.

Advantages of a static website

- Provide improved security over dynamic websites[1]
- Improved performance for end users compared to dynamic websites[2]
- Less or no dependencies on systems such as databases or other application servers

The most important new HTML tags

Tag	What it is	When to use it
<A>	Anchor (most commonly a link)	Vital. Use to create links in content. Use the title attribute whenever the contents of the <a>…</a> pair do not accurately describe what you'll get from selecting the link. Title attribute often displays as a tooltip in visual browsers, which may be a helpful usability aid.
<ABBR>	Defines an abbreviation	Works in a similar way to <dfn> and <acronym>, using a title attribute (displays a tooltip in standard visual browsers). e.g. <abbr title="Hypertext markup language">HTML</abbr>

<ADDRESS>	Used for marking up a physical (e.g. mailing) address	Not commonly used. Recommend looking into microformats, which allow for more detail and interoperability.
<APPLET>	Inserts a Java applet	The old way to insert a Java app. Use <object> instead today.
<AREA>	Hotspot in image map	Avoid image maps where possible. Occasionally necessary.
<BIG>	Larger text	Display info – never use it
<BODY>	Document body	Essential (unless you're using frames)
<BR>	Line break	This is arguably display information. Still in common use, but use with restraint.
<B>	Bold text	Display info – never use it
<BUTTON>	Used for a standard clickable button within a form	Often better than <input type="button" /> or <input type="submit" />, as it allows you to assign different styles based on the HTML element alone, whereas differentiating style based on the type of input is less well supported.
<CAPTION>	Caption for a table: describes the table's contents	The correct way to assign a title to a table
<CENTER>	Centred block	Display info – never use it. Use <div>or some other block-level tag with the style text-align:center instead
<CITE>	Defines a citation	Defines the source of a quotation (in conjunction with content in <q> or <blockquote> pairs).
<CODE>	Defines an extract of code	Not commonly used. Similar to <pre>tag, but collapses consecutive white spaces and line breaks in the source.
<DIV>	Division	Specifies a logical division within a document. Use it to separate or identify chunks of content that are not otherwise distinguished naturally using other tags. One of the most common HTML tags.
<DL>	Definition list	Contains one or more definition-term / definition-description pairs.
<DT>	Definition term	Used as part of a <dt></dt><dd></dd>pair within a definition list (<dl></dl>)
<DD>	Definition description	
<EM>	Emphasis	Commonly used in place of the old <i> (italics) tag to indicate emphasis (but less than <strong>)
<FONT>	Font settings	Display info – never use it
<FORM>	Input form	Essential for data input
<H1>	Level 1 header	Aim to have one H1 on each page, containing a description of what the page is about.
<H2>	Level 2 header	Defines a section of the page
<H3>	Level 3 header	Defines a sub-section of the page (should always follow an H2 in the logical hierarchy)
<H4>	Level 4 header	Etc. Less commonly used

<H5>	Level 5 header	Less commonly used. Only complex academic documents will break down to this level of detail.
<H6>	Level 6 header	Less commonly used
<HEAD>	Document head	Essential. Contains information about a page that does not constitute content to be communicated as part of the page.
<HR>	Horizontal rule	Display info with no semantic value – never use it. “Horizontal”, by definition, is a visual attribute.
<HTML>		Core element of every web page.
<IMG >	Show an image	Vital. Always use the alt or longdescattributes when the image has content value
<INPUT>	Input fields within forms	Vital. (I prefer to use <button> for buttons and submit buttons though)
<I>	Italicised text	Display info – never use it
<KBD>	Keyboard input	Display info – never use it
<LINK>	Defines a relationship to another document	Commonly used to reference external stylesheets, but has other minor uses
<LI>	List item	Specifies an item in an unordered or ordered list (<ul> or <ol>)
<MARQUEE>	Makes text scroll across the screen	See <blink>
<MENU>	Menu item list	Deprecated. Do not use. Use other standard list types instead.
<META>	Meta-information	Useful way to insert relevant information into the <head> section of the page that does not need to be displayed.
<OL>	Ordered list	Type of list where the order of elements has some meaning. Generally rendered with item numbers (best managed with CSS).
<PRE>	Preformatted text	Renders text in a pre-formatted style, preserving line breaks and all spaces present in the source. May be useful. (This one’s a paradox, as it is strictly display info that applies only to visual browsing, but it’s still so commonly used and useful that I’m hesitant to advise against using it.)
<P>	Paragraph	Only use to denote a paragraph of text. Never use for spacing alone.
<Q>	Short quotation	Use for inline quotations (whereas <blockquote> should be used for quotations of a paragraph or more). Often used in conjunction with <cite>to cite the quotation’s source.
<SAMP>	Denotes sample output text	Similar to the <code> tag. Rarely used. Avoid.
<SCRIPT>	Inline script (e.g. JavaScript)	It’s better to have all scripts as separate files than to write inline or in the <head> section, however still has its uses.
<SMALL>	Smaller text	Display info – never use it
<SPAN>	An inline span within text	Use to apply meaning (and style) to a span of text that goes with the flow of content (whereas a <div> tag is block-level and breaks the flow)

<STRONG>	Strong emphasis	Use this instead of the old <b> tag.
<STYLE>	CSS style settings	Normally used in <head> section of a page. Try to use external stylesheets, to enable you to apply different styles for different output media.
<SUB>	Subscript text	Arguably display info – recommend using alternative tags (e.g. <cite>). May be required in some academic uses, e.g. Chemical formulas.
<SUP>	Superscript text	
<TABLE>	Table	Use for repeated data that has a naturally tabular form. Never use for layout purposes.
<TD>	Table data cell	A cell containing actual data. If a cell actually contains a descriptor or identifier for a row or column, use a <th> (table header) tag, not a <td>.This usually applies to column headers (within a <thead>), column footers (within a <tfoot>), as well as row headers (usually the first cell in a row in the <tbody>).
<TH>	Table column or row header cell	May appear in a <thead> (to denote a column header cell), <tbody> (to denote a row header), and in <tfoot>(to denote a column foot cell, e.g. a total)
<TITLE>	Document title	Essential
<TR>	Table row	Essential with tables
<TT>	"Teletype" – simulates typewriter output	Similar to <pre>, except that it collapses white space like normal HTML (whereas <pre> leaves all consecutive white space intact). Avoid if possible
<UL>	Unordered list	Essential. Use for lists where the order or items has no particular importance.
<U>	Underline text	Display info – never use it
<VAR>	Variable in computer code	Obscure tag, may only be useful in academic documents. Avoid.

4. The order of performance of work

Contents of the report

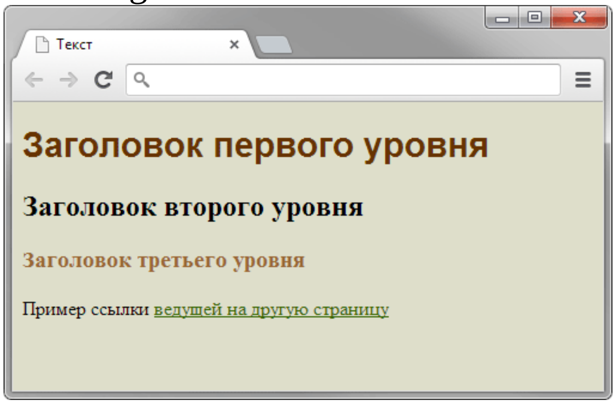
1. The name of the laboratory work.
2. The purpose of laboratory work.
3. Required tools.
4. Brief information from theory.
5. The decision of an example on a variant.
6. Conclusion on the work.

Recommended literature

1. Lomov A.Yu., HTML, CSS, Scripts - practice of creating websites. - St. Petersburg "BHV-Petersburg" 2006. - ISBN: 5-94157-698-6.
2. Peter Lubbers, Brian Olbers, Frank Salim. HTML5 for professionals: powerful tools for developing modern web applications = Pro HTML5 Programming: Powerful APIs for Richer Internet Application Development. - M .: Williams, 2011. - P. 272. - ISBN 978-5-8459-1715-7.
3. Stephen Holtzner. HTML5 for 10 minutes, 5th edition = Sams Teach Yourself HTML5 in 10 Minutes, 5th Edition. - M .: Williams, 2011. - ISBN 978-5-8459-1745-4.
4. Arseniy Mirny HTML5 vs. Flash-video // UP Special: magazine. - 2010. - No. 5. - P. 42-45.

4. Options for the task

1. Make the page shown in Fig. 1.



Заголовок первого уровня

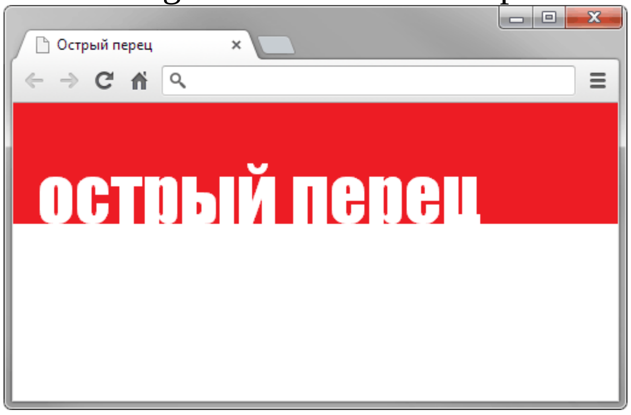
Заголовок второго уровня

Заголовок третьего уровня

Пример ссылки ведущей на другую страницу

Fig. 1

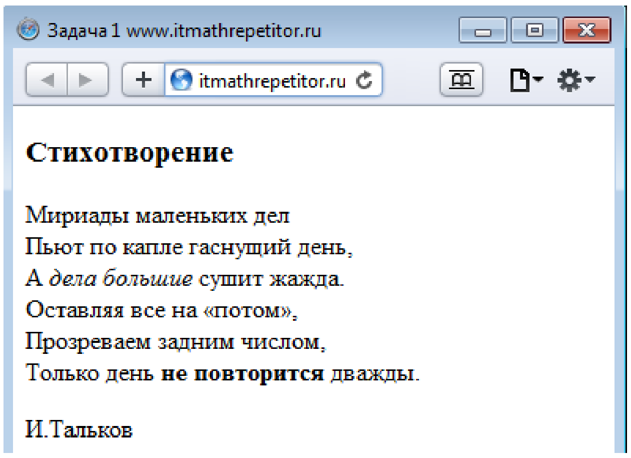
2. Make the text as shown in Fig. 2. Set the font as Impact.



острый перец

Fig. 2

3. Create an html-file (encoding utf-8) with the title "Task 3", the result of which is shown in Figure. Use a suitable title tag (<h1> - <h6>), tags <br>, <p>, <i>, <em>, <strong>, <b>. Find out the difference between the <b> and <strong> tags, <i> tags, and <em> tags. In the html-code, add comments: the date of the solution of this task and the full name.

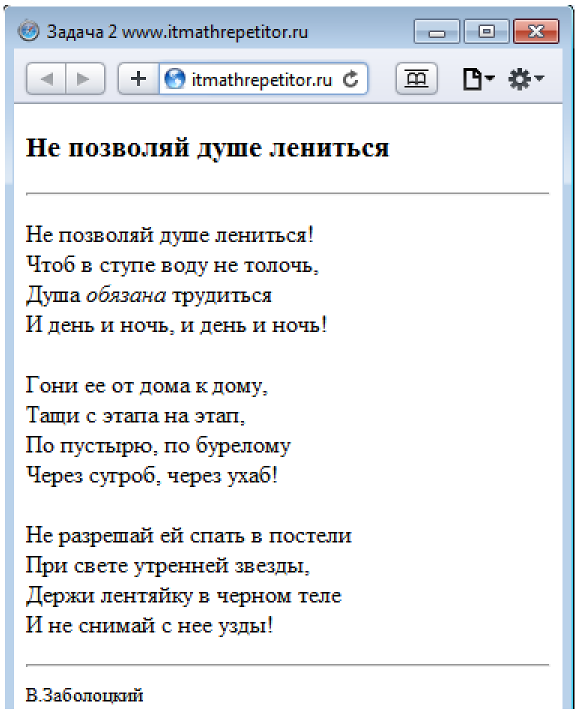


Стихотворение

Мириады маленьких дел
Пьют по капле гаснущий день,
А *дела большие* сушит жажда.
Оставляя все на «потом»,
Прозреваем задним числом,
Только день **не повторится** дважды.

И.Тальков

4. Create an html-file (encoding utf-8) with the title "Task 4", the result of which is shown in the Figure. Use a suitable title tag (h1-h6), tags <br>, <p>, <i>, <em>, <strong>, <b>, <hr>, <small>. Please note that the font of the author's surname is smaller. Note that the <hr> tag in different browsers can be displayed in different ways. In the html-code, add the condition of this task in the form of comments.



Не позволяй душе лениться

Не позволяй душе лениться!
Чтоб в ступе воду не толочь,
Душа *обязана* трудиться
И день и ночь, и день и ночь!

Гони ее от дома к дому,
Тащи с этапа на этап,
По пустырю, по бурелому
Через сугроб, через ухаб!

Не разрешай ей спать в постели
При свете утренней звезды,
Держи лентяйку в черном теле
И не снимай с нее узды!

В.Заболоцкий

5. Create a table in which several cells in a row and several cells in a column are combined.
6. Create text in which different selection methods will be used, incl. italics, bold, underlined text.
7. Display a formula containing variables with indices, exponentiation and Greek letters.
8. Display the formula with the second-order determinant without using tables.
9. Display the formula containing the second-order determinant using tables.
10. Display simple lists: a list of definitions, numbered and unnumbered lists.
11. Create an HTML document that has hyperlinks to the pages located: in the current directory, in the directory at the level above, in the subdirectory, on the specified domain. And also use the transition to the anchor (anchor).
12. Create an HTML document that has hyperlinks to the pages and describe the parameters of the BODY tag: MARGINHEIGHT, TOPMARGIN, MARGINWIDTH, LEFTMARGIN, BACKGROUND, BGCOLOR, TEXT, LINK.
13. Create an HTML document that has hyperlinks to the pages and describe the parameters of the font tag
14. Create an HTML document that has hyperlinks to the pages and describe the parameters of the table tag
15. Create an HTML document that has hyperlinks to the pages and describe the parameters of the frame tag

Laboratory №3
Theme: Using HTML5 features

1. Objective

Study HTML5 hypertext markup language and its additional tags, learn how to create web pages using HTML5 tags

2. Required tools

A personal computer, among the creatures of web applications emmet-sublime-master or Notepad ++, the text editor MS Word

3. Brief information from theory

HTML5 is the latest and most enhanced version of HTML.Technically, HTML is not a programming language, but rather a mark up language.

This tutorial has been designed for beginners in HTML5 providing the basic to advanced concepts of the subject.

- HTML5 is the newest version of HTML, only recently gaining partial support by the makers of web browsers.
- It incorporates all features from earlier versions of HTML, including the stricter XHTML.
- It adds a diverse set of new tools for the web developer to use.

It is still a work in progress. No browsers have full HTML5 support. It will be many years – perhaps not until 2018 or later - before being fully defined and supported.

Goals of HTML5

- Support all existing web pages. With HTML5, there is no requirement to go back and revise older websites.
- Reduce the need for external plugins and scripts to show website content.
- Improve the semantic definition (i.e. meaning and purpose) of page elements.
- Make the rendering of web content universal and independent of the device being used.
- Handle web documents errors in a better and more consistent fashion.

Other New Features in HTML5

- Built-in audio and video support (without plugins)
- Enhanced form controls and attributes
- The Canvas (a way to draw directly on a web page)
- Drag and Drop functionality
- Support for CSS3 (the newer and more powerful version of CSS)
- More advanced features for web developers, such as data storage and offline applications.

HTML5 Browser Support

- HTML5 is supported in all modern browsers.
- In addition, all browsers, old and new, automatically handle unrecognized elements as inline elements.

- Because of this, you can "teach" older browsers to handle "unknown" HTML elements.

Tag <audio>

- <audio src="2.mp3" >
- Controls – show control menu for audioplayer
- Autoplay – Playing audio on loading
- Loop – repeating count of audio

Tag <video>

- <video src="2.mp4" >
- Controls – show control menu for audioplayer
- Autoplay – Playing audio on loading
- Loop – repeating count of audio
- Width – width of video's window
- Height – height of video's window

Form elements

- Forms uses in HTML for creating data.
- HTML has many types of data.
- For forms elements used tag <input>

he HTML **<form>** element defines a form that is used to collect user input:
<input type="text">

- Attributes are
- maxlength
- placeholder
- disabled
- readonly
- size

Tag <form>

```
 First name:<br>
 <input type="text" name="firstname"><br>
 Last name:<br>
 <input type="text" name="lastname">
</form>
```

<input type="password"> For inputting password
<input type="email"

- <input type="date">
- <input type="file">
- <input type="radio">
- <input type="checkbox">

```
<!DOCTYPE html>
<html>

  <head>
    <meta charset="utf-8">
```

```
      <title>Tutorials Point</title>
   </head>
    <body>

      <header role="banner">
         <h1>HTML5 Document Structure Example</h1>
         <p>This page should be tried in safari, chrome or Mozila.</p>
      </header>

      <footer>
         <p>Created by <a href="http://tutorialspoint.com/">Tutorials Point</a></p>
      </footer>

   </body>
</html>
```

4. The order of performance of work

Contents of the report

1. The name of the laboratory work.
2. The purpose of laboratory work.
3. Required tools.
4. Brief information from theory.
5. The decision of an example on a variant.
6. Conclusion on the work.

Recommended literature

1. Lomov A.Yu., HTML, CSS, Scripts - practice of creating websites. - St. Petersburg "BHV-Petersburg" 2006. - ISBN: 5-94157-698-6.
2. Peter Lubbers, Brian Olbers, Frank Salim. HTML5 for professionals: powerful tools for developing modern web applications = Pro HTML5 Programming: Powerful APIs for Richer Internet Application Development. - M .: Williams, 2011. - P. 272. - ISBN 978-5-8459-1715-7.
3. Stephen Holtzner. HTML5 for 10 minutes, 5th edition = Sams Teach Yourself HTML5 in 10 Minutes, 5th Edition. - M .: Williams, 2011. - ISBN 978-5-8459-1745-4.
4. Arseniy Mirny HTML5 vs. Flash-video // UP Special: magazine. - 2010. - No. 5. - P. 42-45.

4. Options for the task

Make the same layout using tables.

Option №1.

1.	*2.*		*3.*	*4.*
5.	*6.*	*7.*		
	8.	*9.*	*10.*	*11.*
12.				

Option № 2.

1.			*2.*	*3.*
4.	*5.*	*6.*		
	7.	*8.*	*9.*	
	10.	*11.*		

Option № 3.

1.	*2.*	*3.*	*4.*	
5.		*6.*		
7.	*8.*	*9.*	*10.*	*11.*
12.		*13.*		

Option № 4.

1.	*2.*	*3.*		*4.*
5.		*6.*	*7.*	
8.	*9.*		*10.*	
	11.			*12.*

Option № 5.

1.	*2.*		*3.*	*4.*
5.	*6.*	*7.*		
	8.	*9.*	*10.*	*11.*
12.				

Option № 6.

1.			*2.*	*3.*
4.	*5.*	*6.*		
	7.	*8.*	*9.*	
	10.	*11.*		

Option № 7.

1.	*2.*	*3.*		*4.*
	5.		*6.*	*7.*
	8.	*9.*		*10.*
	11.	*12.*		

Option № 8.

1.				
2.	*3.*	*4.*	*5.*	*6.*
7.			*8.*	
9.	*10.*			*11.*

Option № 9.

1	2	3	4	5
	6		7	
8	9			10

Option № 10.

1.	2.	3.	4.	
	5.		6.	7.
	8.		9.	

Option № 11.

1.		2.	3.	4.
5.	6.	7.		
8.		9.	10.	11.

Option № 12.

1.	2.	3.	4.	5.
6.		7.		8.
	9.		10.	

Option № 13.

1.			2.	3.
4.	5.	6.		
7.		8.	9.	10.

Option № 14.

1.	2.	3.	4.	5.
	6.		7.	
	8.		9.	10.

Option № 15.

1.	2.		3.	
4.	5.	6.	7.	8.
9.		10.		11.

Option № 16.

1.	2.	3.	4.	5.
	6.	7.		8.
9.			10.	

Option № 17.

1.	2.	3.	4.	5.
	6.		7.	
8.	9.			10.

Option № 18.

1.	2.	3.	4.	5.
6.	7.	8.		9.
10.		11.		

Option № 19.

1.	2.			3.
	4.	5.	6.	
	7.			

Option № 20.

1.		2.	3.	
4.	5.		6.	7.
8.			9.	

Option № 21.

1.	2.	3.	4.	5.
		6.		
		7.	8.	9.

Option № 22.

1.	2.	3.	4.	5.
6.	7.			
8.	9.	10.		

Option № 23.

1.	2.	3.	4.	
5.			6.	7.
8.			9.	

Option № 24.

1.	2.	3.	4.
		5.	6.
	7.	8.	9.

Option № 25.

1.	2.	3.	4.	5.	6.
	7.	8.			
	9.	10.	11.	12.	

Laboratory №4
Theme: Using the CSS feature

1. Objective

Research and study of cascading style sheets, creation of styles and work off.

2. Required tools

A personal computer, among the creatures of web applications emmet-sublime-master or Notepad ++, the text editor MS Word

3. Brief information from theory

Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language. Although most often used to set the visual style of web pages and user interfaces written in HTML and XHTML, the language can be applied to any XML document, including plain XML, SVG and XUL, and is applicable to rendering in speech, or on other media. Along with HTML and JavaScript, CSS is a cornerstone technology used by most websites to create visually engaging webpages, user interfaces for web applications, and user interfaces for many mobile applications.

CSS is designed primarily to enable the separation of presentation and content, including aspects such as the layout, colors, and fonts. This separation can improve content accessibility, provide more flexibility and control in the specification of presentation characteristics, enable multiple HTML pages to share formatting by specifying the relevant CSS in a separate .css file, and reduce complexity and repetition in the structural content.

CSS is a language that describes the style of an HTML document. CSS describes how HTML elements should be displayed.

CSS has a simple syntax and uses a number of English keywords to specify the names of various style properties.

A style sheet consists of a list of *rules*. Each rule or rule-set consists of one or more *selectors*, and a *declaration block*.

In CSS, *selectors* declare which part of the markup a style applies to by matching tags and attributes in the markup itself.

Selectors may apply to:

- all elements of a specific type, e.g. the second-level headers h2
- elements specified by attribute, in particular:
  - *id*: an identifier unique within the document
  - *class*: an identifier that can annotate multiple elements in a document
- elements depending on how they are placed relative to others in the document tree.

Classes and IDs are case-sensitive, start with letters, and can include alphanumeric characters and underscores. A class may apply to any number of instances of any elements. An ID may only be applied to a single element.

Pseudo-classes are used in CSS selectors to permit formatting based on information that is not contained in the document tree. One example of a widely used pseudo-class is :hover, which identifies content only when the user "points to" the visible element, usually by holding the mouse cursor over it. It is appended to a selector as in **a**:hover or #**elementid**:hover. A pseudo-class classifies document elements, such as :link or :visited, whereas a *pseudo-element* makes a selection that may consist of partial elements, such as ::first-line or ::first-letter.

CSS Example

```
body { background-color: lightblue; }
h1 {  color: white;  text-align: center; }
p {   font-family: verdana; font-size: 20px; }
```

The 3 ways to insert CSS into your web pages

The wise people who created CSS came up with 3 basic ways for you to use CSS in your web pages:

1. With an external file that you link to in your web page:

```
<link href="myCSSfile.css" rel="stylesheet" type="text/css">
```

or using the import method:

```
<style type="text/css" media="all">
  @import "myCSSfile.css";
</style>
```

Why use the import method versus the link method when using external style sheets? Use the import method if you want to support really old browsers like Netscape 4.0.

Let me explain: Netscape can’t handle most CSS beyond font settings and colors, if it finds any other types of CSS it could cause good old Netscape 4 to crash in some occasions or at the very least mangle your page.

Netscape 4 does not understand the @import method of linking to a style sheet like the newer browsers can, so Netscape just ignores it. You can use this to hide the fancy CCS code in the external style sheet by using the @import method so all the good browsers can reference it while keeping Netscape 4 out of the picture.

Netscape 4.0 is pretty much dead (that is really a good thing) so I personally don’t worry much about it. But for some of you, it may be of concern so I thought it should be mentioned.

2. By creating a CSS block in the web page itself; typically inserted at the top of the web page in between the <head> and </head> tags:

```
<head>
  <style type="text/css">
    p { padding-bottom: 12px; }
  </style>
</head>
```

3. By inserting the CSS code right on the tag itself:

```
<p style="padding-bottom:12px;">Your Text</p>
```

So some of you may be asking why have the 3 methods of including the CSS in a web page? The answer is: flexibility and laziness! Ok I'm kidding about the laziness, replace it with 'precision'. So what the heck does that mean?

I think the easiest way to explain to you what's going on, is by giving you real examples that demonstrate the differences. Wait a second, don't fall asleep … the examples are short and I think that once you finish, you will see how easy it really is!

Another reason that you want to continue reading this article is that you will gain a good understanding of some fundamental (and practical) CSS principles – remember that the difference between people who are really good at what they do, and those who are not so good, is in the mastery of the basics. Lets get down on it!

Method 1: Create a separate CSS file and link it to your web page(s)

The heading for this part of the article hints at why you would want to go through the trouble of creating a separate CSS file instead of just typing in the CSS code in the web page itself. Can you guess? Keep trying … times up! Did you get it? I could quote you some nerd centric description that describes the advantage; the problem is that only nerds who already know would understand!

In a nutshell: by keeping the CSS code in its own file, you can link that CSS file to as many web pages as you want. This has two major advantages:

1. You will have much less code in all your HTML pages – makes the pages neater and easier to manage and makes the web pages a little faster on the download. (Although this point is really minor in most cases, and is really over blown in my opinion by some people)
2. It can potentially reduce the amount of work you have to do in a big way. Why you ask? Simple; lets say you have 50 web pages where you've set the text to be black and the headline text (text in between the <h3> tags for example) to blue. Then one day you decide you want to change the color of the text. Since the CSS that controls the text color for the 50 pages is in one CSS file, you can easily change the color of your text in all 50 pages by changing one line in the CSS file!

If on the other hand you had decided to include all your font color information in each page, you would have had to change all 50 pages. This would have been even worse if you had been using font tags, or CSS right on each tag, you would have to change the color settings/code on all the <p> and <h3> tags in all 50 pages! I can tell you from experience that that sucks big time!

The rule: If you are going to have more than one web page with the same stylistic properties (that look the same in some way) you should create a separate CSS file and link your web pages to it.

Method 2: Create a CSS block in the web page itself

The Rule: Use this method if you want to override the CSS you have in a linked CSS file or if you only have a one-page web site.

Now that we covered the first method of putting all your CSS code in a separate file and linking to it, the other methods are easy to describe.

CSS stands for (is the acronym for): 'Cascading Style Sheets.' I think the words 'style sheets' in CSS are self-describing … we know what 'style' in style sheets mean. But what is the meaning of 'cascading' in CSS?

The cascading effect in CSS

The word ‘cascading’ in CSS describes a cascading mechanism; that is to say that the CSS code on the page itself will override the CSS code in a separate linked file. And subsequently, CSS declared ‘inline’ on the tag itself would override all other CSS.

So let’s look a practical example; let’s say you have a web site with 50 pages where the layout and fonts are the same on all 50 pages. Wisely you put the CSS information that sets the layout and font choices in a separate style sheet, but for a particular page you need to change the color of some of the text and add a border around a paragraph. This is a perfect example where you might want to place a little CSS in the page itself since the color and border will be unique to that page. Is this all sinking in?

Method 3: Embed the CSS right on the tags themselves (called inline CSS)

The Rule: Use this method on a unique element/tag that you want to affect with CSS.

An example can be with a special heading on the page where you want to have a little more padding than you typically do for a heading. Instead of creating a class elsewhere that will only be used on this one occasion, it makes sense to me to just include the CSS inline. I have to stress that inline CSS is something you should rarely if ever use because it can get messy quick.

4. The order of performance of work

Contents of the report

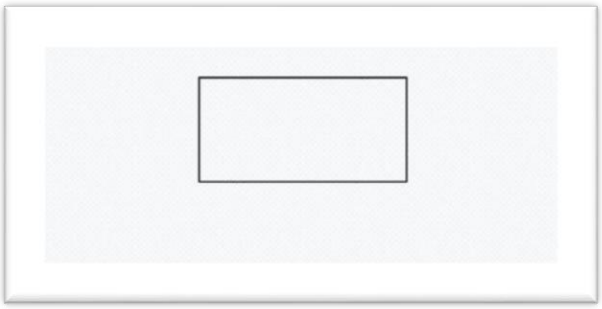
1. The name of the laboratory work.
2. The purpose of laboratory work.
3. Required tools.
4. Brief information from theory.
5. The decision of an example on a variant.
6. Conclusion on the work.

Recommended literature

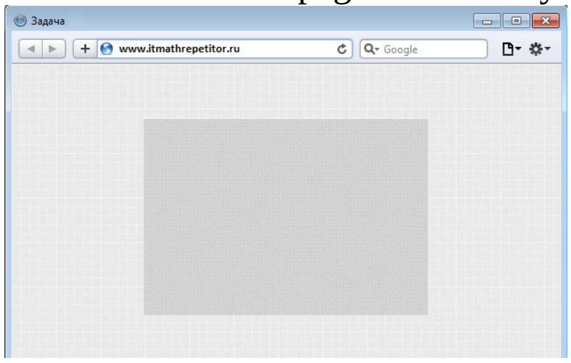
1. Lomov A.Yu., HTML, CSS, Scripts - practice of creating websites. - St. Petersburg "BHV-Petersburg" 2006. - ISBN: 5-94157-698-6.
2. Ed Tittel, Jeff Noble. HTML, XHTML and CSS for Dummies, 7th Edition = HTML, XHTML & CSS For Dummies, 7th Edition. - Moscow: Dialectics, 2011. - 400 p. - ISBN 978-5-8459-1752-2.
3. Stephen Schafer. HTML, XHTML and CSS. The user's Bible, 5th edition = HTML, XHTML, and CSS Bible, 5th Edition. - Moscow: Dialectics, 2011. - 656 p. - ISBN 978-5-8459-1676-1.
4. Andy Budd, Cameron Mall, Simon Collison. CSS: Professional application of Web standards = CSS Mastery: Advanced Web Standards Solutions. - Moscow: Williams, 2008. - 272 p. - ISBN 978-5-8459-1199-5.
5. Christopher Schmitt. CSS. Programming recipes = CSS. Cookbook. - St. Petersburg: BHV-Petersburg, 2007. - 592 p. - ISBN 978-5-9775-0075-3.
6. Eric A. Meyer. CSS-cascading style sheets: detailed manual = Cascading Style Sheets: The definitive Guide. - Moscow: Symbol, 2006. - 576 p. - ISBN 5-93286-075-8.

4. Options for the task

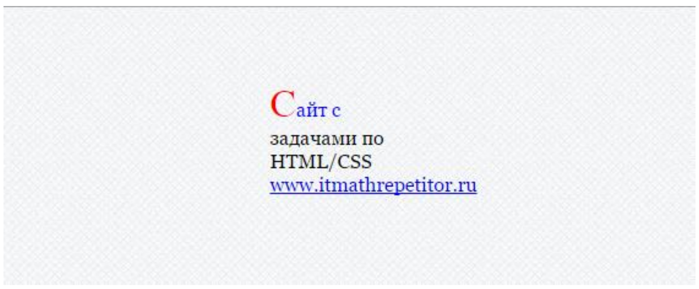
Option №1. Create an html-file, the result of which is shown in the figure. A block with a black border is centered on the page horizontally.



Option №2. Create an html-file, the result of which is shown in the figure. Block with a dark background is centered on the page horizontally and vertically.

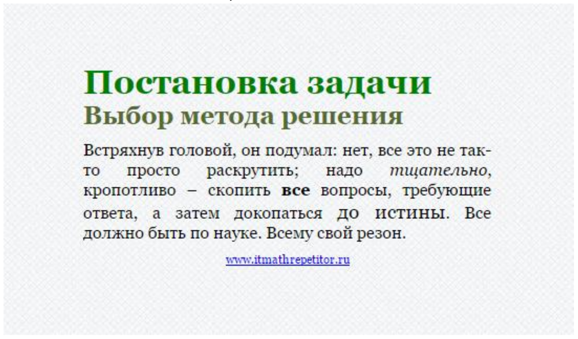


Option №3. Create html and css files, the result of which is shown in the figure.



Сайт с
задачами по
HTML/CSS
www.itmathrepetitor.ru

Option №4. Create html and css files, the result of which is shown in the figure.



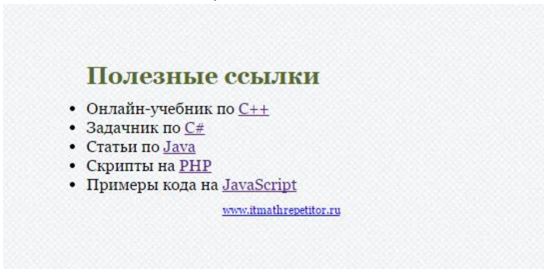
Постановка задачи

Выбор метода решения

Встряхнув головой, он подумал: нет, все это не так-то просто раскрутить; надо *тщательно*, кропотливо – скопить **все** вопросы, требующие ответа, а затем докопаться до истины. Все должно быть по науке. Всему свой резон.

www.itmathrepetitor.ru

Option №5. Create html and css files, the result of which is shown in the figure.

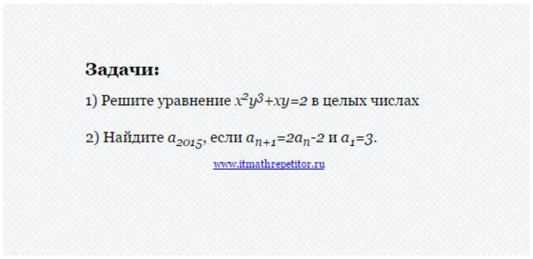


Полезные ссылки

- Онлайн-учебник по C++
- Задачник по C#
- Статьи по Java
- Скрипты на PHP
- Примеры кода на JavaScript

www.itmathrepetitor.ru

Option №6. Create html and css files, the result of which is shown in the figure.



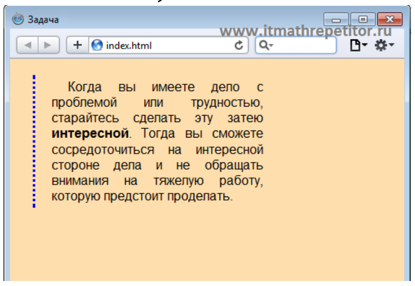
Задачи:

1) Решите уравнение $x^2y^3+xy=2$ в целых числах

2) Найдите a_{2015}, если $a_{n+1}=2a_n-2$ и $a_1=3$.

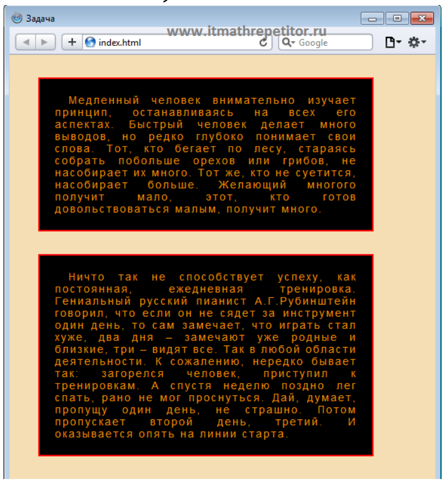
www.itmathrepetitor.ru

Option №7. Create html and css files, the result of which is shown in the figure.



Когда вы имеете дело с проблемой или трудностью, старайтесь сделать эту затею **интересной**. Тогда вы сможете сосредоточиться на интересной стороне дела и не обращать внимания на тяжелую работу, которую предстоит проделать.

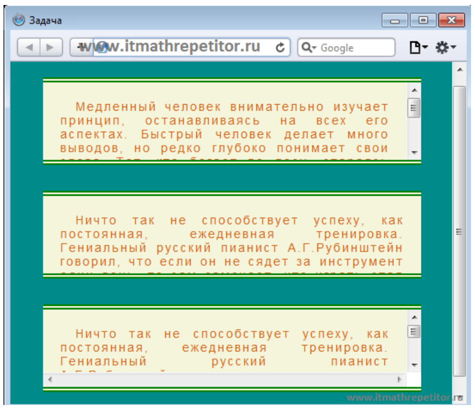
Option №8. Create html and css files, the result of which is shown in the figure.



Медленный человек внимательно изучает принцип, останавливаясь на всех его аспектах. Быстрый человек делает много выводов, но редко глубоко понимает свои слова. Тот, кто бегает по лесу, стараясь собрать побольше орехов или грибов, не насобирает их много. Тот же, кто не суетится, насобирает больше. Желающий многого получит мало, этот, кто готов довольствоваться малым, получит много.

Ничто так не способствует успеху, как постоянная, ежедневная тренировка. Гениальный русский пианист А.Г.Рубинштейн говорил, что если он не сядет за инструмент один день, то сам замечает, что играть стал хуже, два дня – замечают уже родные и близкие, три – видят все. Так в любой области деятельности. К сожалению, нередко бывает так: загорелся человек, приступил к тренировкам. А спустя неделю поздно лег спать, рано не мог проснуться. Дай, думает, пропущу один день, не страшно. Потом пропускает второй день, третий. И оказывается опять на линии старта.

Option №9. Create html and css files, the result of which is shown in the figure.

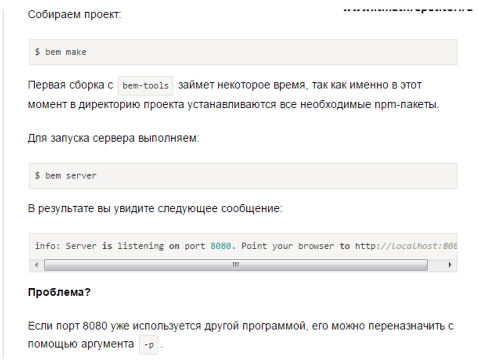


Медленный человек внимательно изучает принцип, останавливаясь на всех его аспектах. Быстрый человек делает много выводов, но редко глубоко понимает свои

Ничто так не способствует успеху, как постоянная, ежедневная тренировка. Гениальный русский пианист А.Г.Рубинштейн говорил, что если он не сядет за инструмент

Ничто так не способствует успеху, как постоянная, ежедневная тренировка. Гениальный русский пианист

Option №10. Create html and css files, the result of which is shown in the figure.



Собираем проект:

```
$ bem make
```

Первая сборка с `bem-tools` займет некоторое время, так как именно в этот момент в директорию проекта устанавливаются все необходимые npm-пакеты.

Для запуска сервера выполняем:

```
$ bem server
```

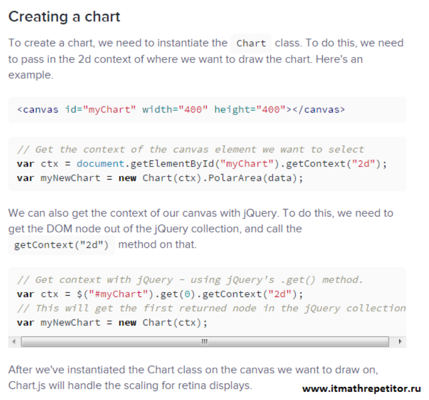
В результате вы увидите следующее сообщение:

```
info: Server is listening on port 8080. Point your browser to http://localhost:808
```

Проблема?

Если порт 8080 уже используется другой программой, его можно переназначить с помощью аргумента `-p`.

Option №11. Create html and css files, the result of which is shown in the figure.



Creating a chart

To create a chart, we need to instantiate the `Chart` class. To do this, we need to pass in the 2d context of where we want to draw the chart. Here's an example.

```
<canvas id="myChart" width="400" height="400"></canvas>
```

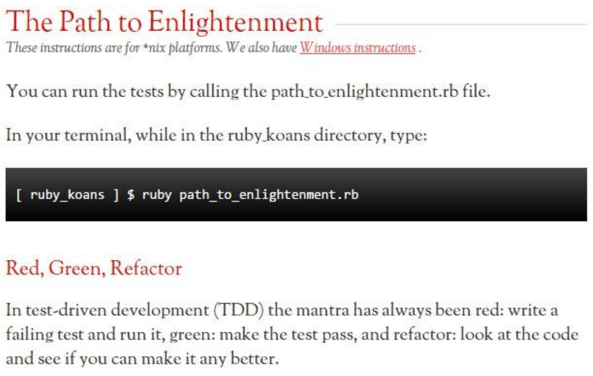
```
// Get the context of the canvas element we want to select
var ctx = document.getElementById("myChart").getContext("2d");
var myNewChart = new Chart(ctx).PolarArea(data);
```

We can also get the context of our canvas with jQuery. To do this, we need to get the DOM node out of the jQuery collection, and call the `getContext("2d")` method on that.

```
// Get context with jQuery - using jQuery's .get() method.
var ctx = $("#myChart").get(0).getContext("2d");
// This will get the first returned node in the jQuery collection
var myNewChart = new Chart(ctx);
```

After we've instantiated the Chart class on the canvas we want to draw on, Chart.js will handle the scaling for retina displays.

Option №12. Create html and css files, the result of which is shown in the figure.



The Path to Enlightenment

*These instructions are for *nix platforms. We also have Windows instructions.*

You can run the tests by calling the path_to_enlightenment.rb file.

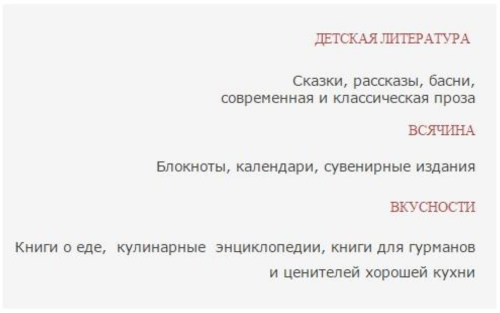
In your terminal, while in the ruby_koans directory, type:

```
[ ruby_koans ] $ ruby path_to_enlightenment.rb
```

Red, Green, Refactor

In test-driven development (TDD) the mantra has always been red: write a failing test and run it, green: make the test pass, and refactor: look at the code and see if you can make it any better.

Option №13. Create html and css files, the result of which is shown in the figure.



ДЕТСКАЯ ЛИТЕРАТУРА

Сказки, рассказы, басни,
современная и классическая проза

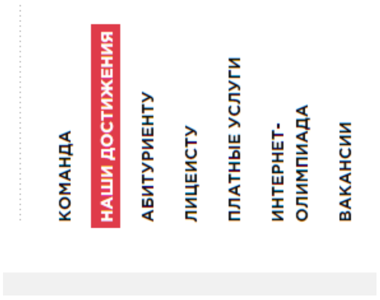
ВСЯЧИНА

Блокноты, календари, сувенирные издания

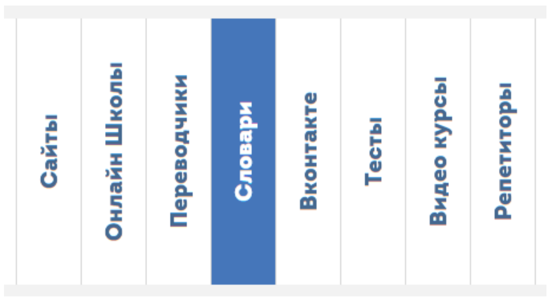
ВКУСНОСТИ

Книги о еде, кулинарные энциклопедии, книги для гурманов
и ценителей хорошей кухни

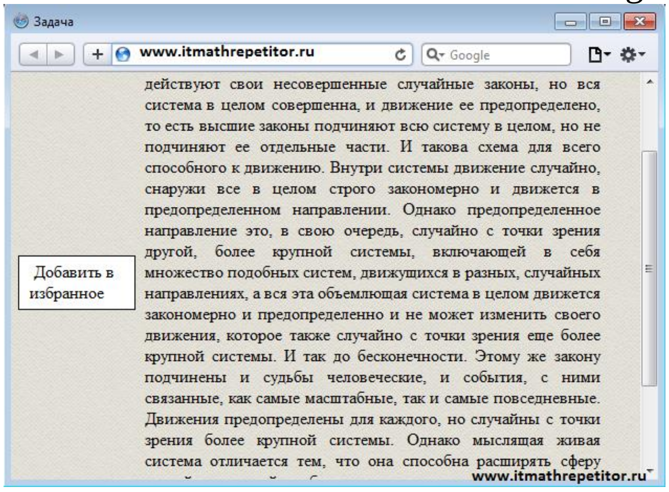
Option №14. Create html and css files, the result of which is shown in the figure.



Option №15. Create html and css files, the result of which is shown in the figure.



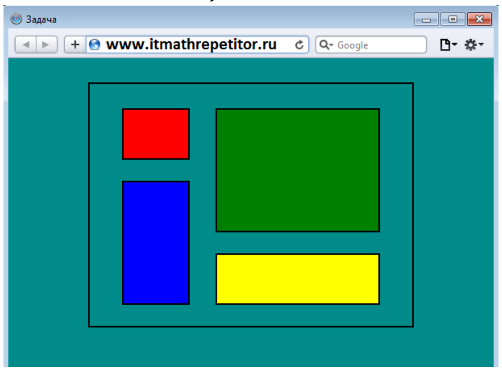
Option №16. Create html and css files, the result of which is shown in the figure. The "Add to Favorites" element is fixed when viewing the page.



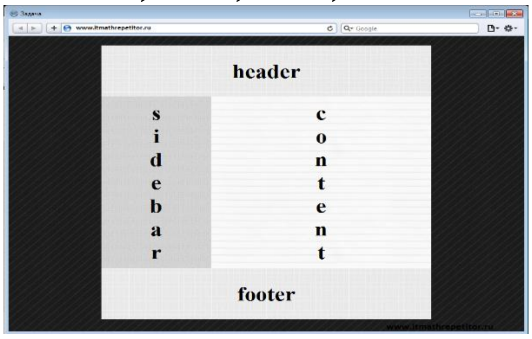
действуют свои несовершенные случайные законы, но вся система в целом совершенна, и движение ее предопределено, то есть высшие законы подчиняют всю систему в целом, но не подчиняют ее отдельные части. И такова схема для всего способного к движению. Внутри системы движение случайно, снаружи все в целом строго закономерно и движется в предопределенном направлении. Однако предопределенное направление это, в свою очередь, случайно с точки зрения другой, более крупной системы, включающей в себя множество подобных систем, движущихся в разных, случайных направлениях, а вся эта объемлющая система в целом движется закономерно и предопределенно и не может изменить своего движения, которое также случайно с точки зрения еще более крупной системы. И так до бесконечности. Этому же закону подчинены и судьбы человеческие, и события, с ними связанные, как самые масштабные, так и самые повседневные. Движения предопределены для каждого, но случайны с точки зрения более крупной системы. Однако мыслящая живая система отличается тем, что она способна расширять сферу

www.itmathrepetitor.ru

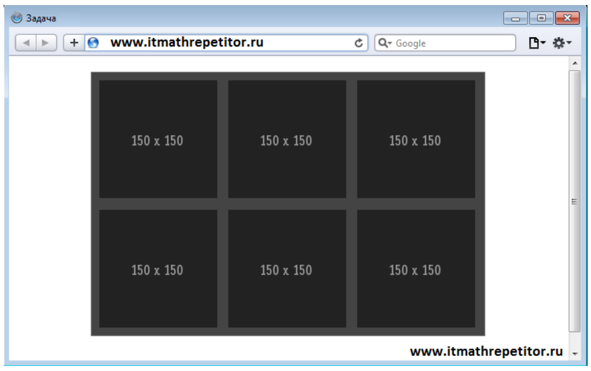
Option №17. Create html and css files, the result of which is shown in the figure.



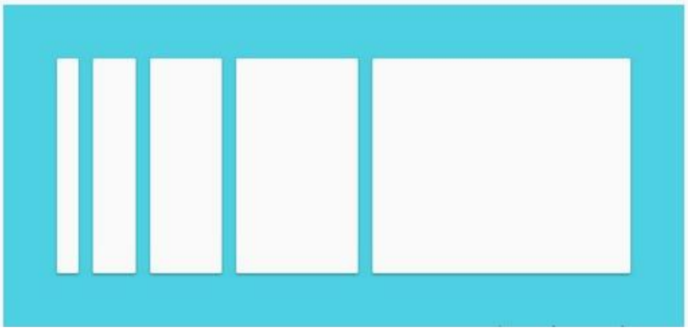
Option №18. Create html and css files, the result of which is shown in the figure. Images for the background: link1, link2, link3, link4. Fixed width.



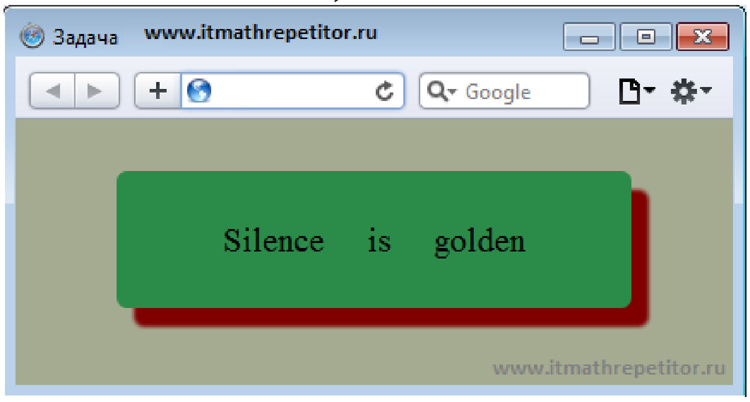
Option №19. Condition: Create html and css files, the result of which is shown in the figure.



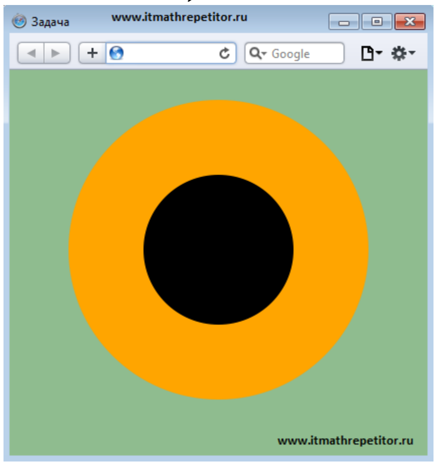
Option №20. Create html and css files, the result of which is shown in the figure. The width of the blocks dynamically varies depending on the width of the page.



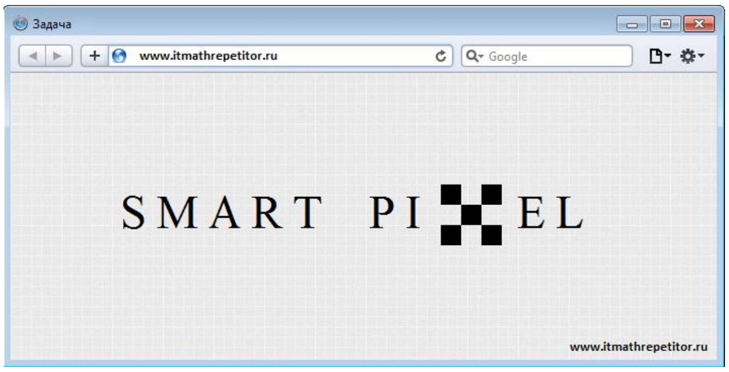
Option №21. Create html and css files, the result of which is shown in the figure.



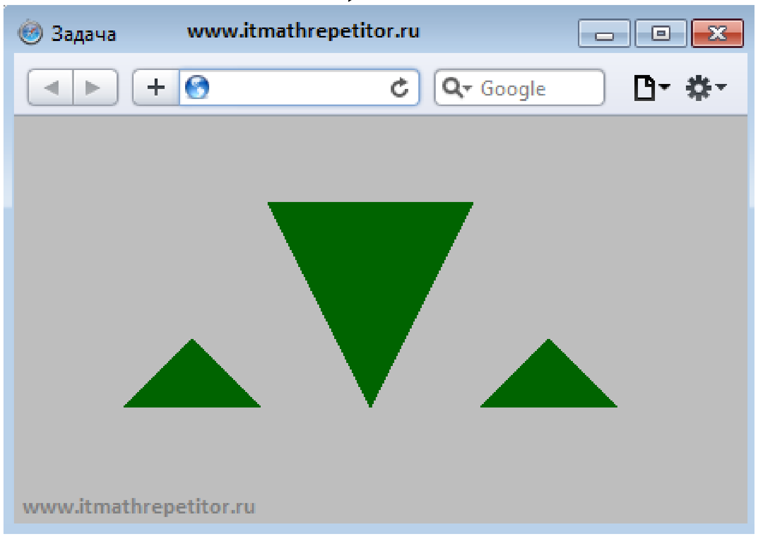
Option №22. Create html and css files, the result of which is shown in the figure.



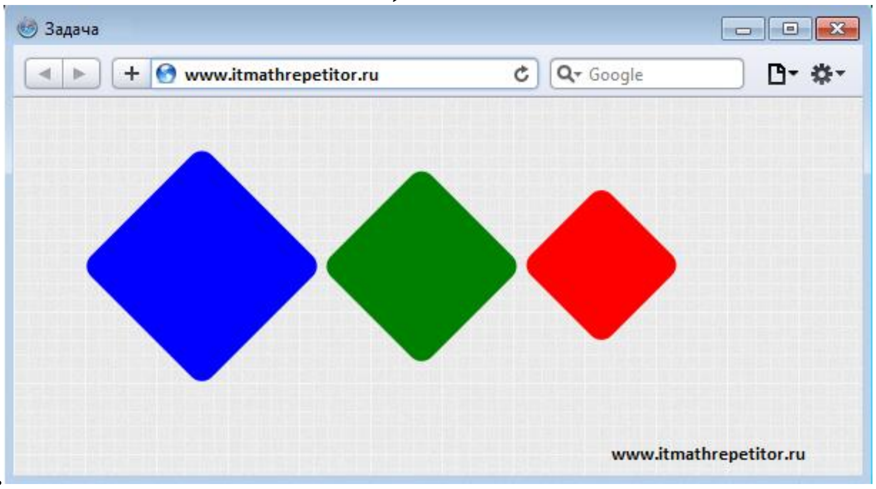
Option №23. Create html and css files, the result of which is shown in the figure.



Option №24. Create html and css files, the result of which is shown in the figure.



Option №25. Create html and css files, the result of which is shown in the figure



.

Laboratory №5
Theme: Creating web pages using blocks using CSS

1. Objective

Study and study cascading style sheets, create web page structures using blocks and CSS classes

2. Required tools

A personal computer, among the creatures of web applications emmet-sublime-master or Notepad ++, the text editor MS Word

3. Brief information from theory

There are many ways to create a two column layout using Cascading Style Sheets (CSS).

Using Float to Create a Two-Column Effect

While there are many methods of using CSS to create a two-column site, this tutorial uses the float property to move one column to the side of another. In many ways, this method is arguably easier to work with and more flexible than the absolute positioning method currently used on thesitewizard.com. It also allows you to add optional header and footer bars that span both columns if you wish.

The HTML Code for a Basic 2 Column Website

The HTML part of the code is fairly simple. You basically need 2 div blocks, one for each column.

```
<div id="container">
  <div id="content">Content here</div>
  <div id="navbar">Navigation</div>
</div>
```

The words "Navigation" and "Content here" are merely placeholders for the navigation side bar and main content. You should remove those words when you put your real content. The "container" div is merely a block enclosing both the two columns, and is useful if you want to apply certain styles that affect both columns as a unit.

The CSS Code for a 2 Column Website

The CSS below uses percent ("%") for the widths of both columns. Since they are measured in relative units, they will expand or contract to fill the entire browser window no matter how large it may be. At the time I write this, thesitewizard.com uses such a fluid layout for its right column as well. For example, if you change the size of your browser window, the software will reformat the article column as far as possible to fit within the window (unless you resize it too small).

The CSS code for this is simple.

```
#content {float: right;   width: 80%;  }
#navbar {float: right;   width: 20%;   }
```

The CSS code has to go either into the style section of your web page or an external style sheet.

How It Works: Explanation of the CSS Code

The "float: right" rule causes the DIV block to be taken out of the document flow (the default way in which the elements on your page are arranged), and placed on the right with all other text and images flowing around it on the left. The first DIV block to be encountered on the HTML page is floated first.

In the case of the page given above, "#content" is first shifted to the right and given a width of 80% of the browser width. Our next rule also floats "#navbar" and shifts it to the right (yes, to the right, not left). Since there is already a floated element in that position, this second block is placed to the left of that existing element if there is sufficent space, otherwise it is placed below it. In view of this, the total of the width for both blocks must equal to 100% or less, else there will not be enough space for both blocks to be positioned side-by-side.

How to Put the Navigation (Side) Column on the Right

The earlier code puts the navigation menu in the left column, just like what you see on all the pages of thesitewizard.com, including this very page you are reading. If you prefer that the navigation menu be on the right, as is commonly found in blogs, change the code so that it looks like the following:

```
#content {  float: left ;   width: 80% ; }
#navbar {   float: left ;   width: 20% ; }
```

How to Change the Width Correctly

The above style sets the side column to 20% of the width of the browser window and the content column to 80%, giving a total of 100% (ie, 20% + 80%).

If you are planning to change the values to some other number, make sure that their total is equal to 100% or less, otherwise the browser will place one column below the other.

If One of Your Columns Drops Below The Other: How to Debug and Fix It

If you find that one of your columns is placed below the other, either in a staggered fashion or some other way, instead of being placed side-by-side, it means that the total width of both columns is more than 100% of the browser width.

This can happen even if you use my numbers "20%" and "80%" above. For example, if you have added margins, borders and padding to either or both your columns, the widths of those columns will increase accordingly, leading to a total exceeding 100%.

There are at least 2 ways to solve this:

- Decrease the percentage used for the width of your columns until the browser renders it the way you want it to.

  However, remember that percentage is a relative measurement. It is tied to your visitor's browser window width. So just because you tested it on your system and found that if you added (say) a "padding-left: 10px ;" to one of your columns and decreased its width by 1%, and everything still fits perfectly, you cannot conclude that 1% equals 10 pixels. To put it another way, 1% of 1024 pixels is different from 1% of 1920 pixels, and so on. Make sure that you make some allowance for the differences so that your columns will still appear beside each other on a system with a different window width.

- The solution I prefer is to create a nested DIV block within your "#navbar" and "#content" blocks and put your padding, margin, border and real content there. That way, you can leave your percentages at 20% and 80% for your outer blocks without the nuisance of including paddings, margins and borders in your calculations.

For example, in the Two Column Layout Demo, the following HTML code is used to create a nested DIV block.

```
<div id="container">
<div id="content">
  <div id="innercontent">
  Content here
  </div>
</div>
<div id="navbar">
  <div id="innernavbar">
  Navigation
  </div>
</div>
</div>
```

Then, in addition to the CSS given earlier, the rules for "#innercontent" and "#innernavbar" are specified as follows:

```
#innercontent {
  padding-left: 10px ;
  padding-right: 10px ;
}
#innernavbar {
  padding-left: 5px ;
  padding-right: 5px ;
}
```

(The rules for "#content" and "#navbar" remain the same as described in the first half of this tutorial.)

Since the padding is applied to the inner DIV block, the measurements for the outer ones remain unchanged, and the two column layout is preserved.

Some sites have a top header spanning both columns. They may either place the site's logo or name here, or, perhaps even banner advertisements, or both. Some sites also include a footer that span both columns. Among other things, the footer may be used for things like copyright notices.

To use a header and footer using this 2-column layout, modify your HTML code to include two additional DIV blocks.

```
<div id="container">
  <div id="header">Top Header</div>
  <div id="content">Content here</div>
  <div id="navbar">Navigation</div>
  <div id="footer">Bottom Footer</div>
</div>
```

Add the following CSS code to your existing style sheet. Simply place it after the styles you created earlier.

```
#footer {
clear: both ;  }
```

If you want the text in your header to be centred, add the following. Otherwise, there's no need to define a header style.

```
#header {
text-align: center ;  }
```

The same text-align property can be added to the footer to centre the text there, if you wish.

Conclusion

With the CSS code given above, you're now on your way to creating your 2-column website. You may also wish to check out the CSS navigation menu bar wizard as well, if you wish to add a CSS-driven navigation menu bar to your side panel that has mouse-over hover effects.

4. The order of performance of work

Contents of the report

1. The name of the laboratory work.
2. The purpose of laboratory work.
3. Required tools.
4. Brief information from theory.
5. The decision of an example on a variant.
6. Conclusion on the work.

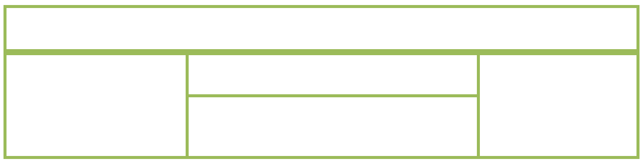
Recommended literature

1. Lomov A.Yu., HTML, CSS, Scripts - practice of creating websites. - St. Petersburg "BHV-Petersburg" 2006. - ISBN: 5-94157-698-6.
2. Ed Tittel, Jeff Noble. HTML, XHTML and CSS for Dummies, 7th Edition = HTML, XHTML & CSS For Dummies, 7th Edition. - Moscow: Dialectics, 2011. - 400 p. - ISBN 978-5-8459-1752-2.
3. Stephen Schafer. HTML, XHTML and CSS. The user's Bible, 5th edition = HTML, XHTML, and CSS Bible, 5th Edition. - Moscow: Dialectics, 2011. - 656 p. - ISBN 978-5-8459-1676-1.
4. Andy Budd, Cameron Mall, Simon Collison. CSS: Professional application of Web standards = CSS Mastery: Advanced Web Standards Solutions. - Moscow: Williams, 2008. - 272 p. - ISBN 978-5-8459-1199-5.
5. Christopher Schmitt. CSS. Programming recipes = CSS. Cookbook. - St. Petersburg: BHV-Petersburg, 2007. - 592 p. - ISBN 978-5-9775-0075-3.
6. Eric A. Meyer. CSS-cascading style sheets: detailed manual = Cascading Style Sheets: The definitive Guide. - Moscow: Symbol, 2006. - 576 p. - ISBN 5-93286-075-8.

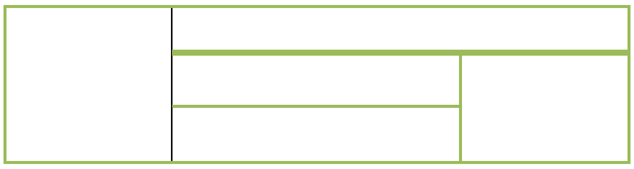
4. Options for the task

Split the web page using blocks

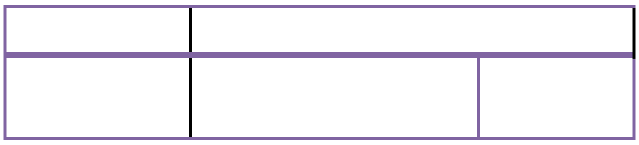
Option №1.



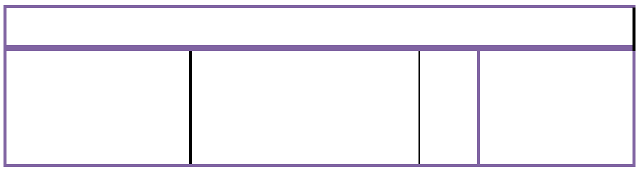
Option №2.



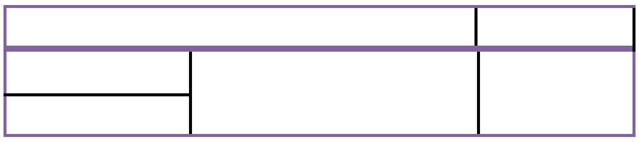
Option №3.



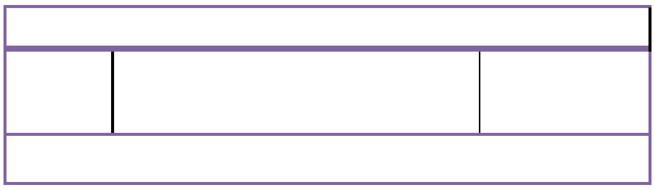
Option №4.



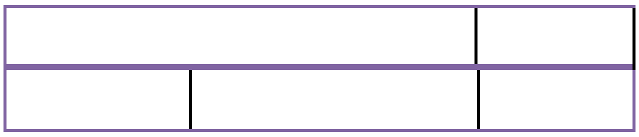
Option №5.



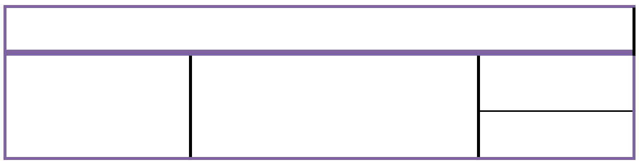
Option №6.



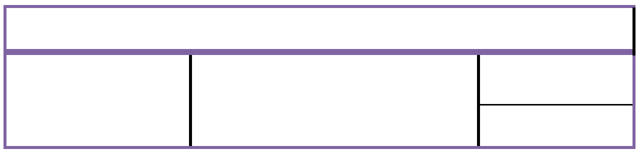
Option №7.



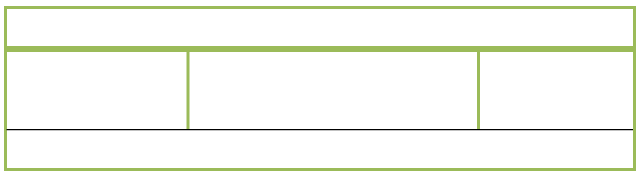
Option №8.



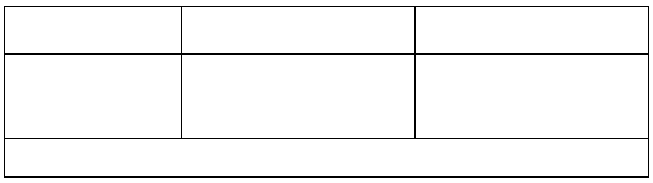
Option №9.



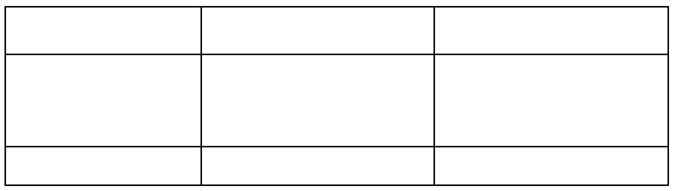
Option №10.



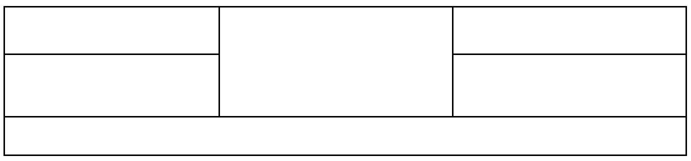
Option №11.



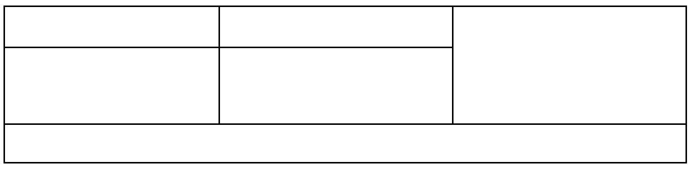
Option №12.



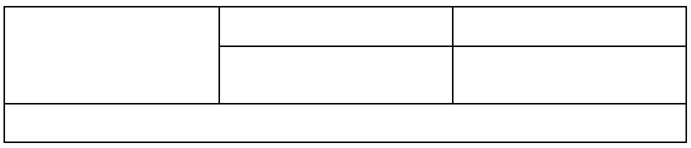
Option №13.



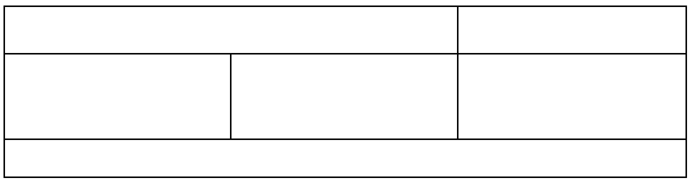
Option №14



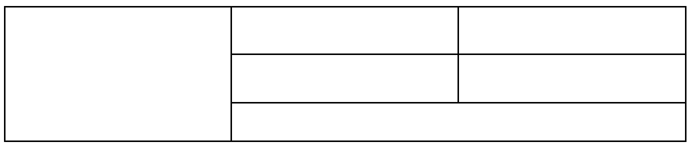
Option №15.



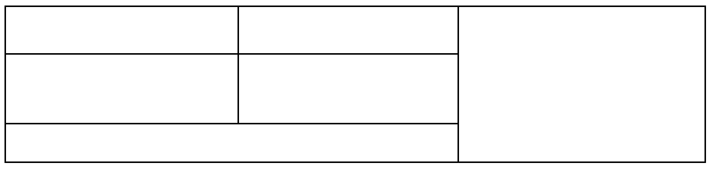
Option №16.



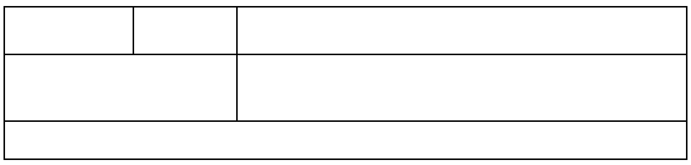
Option №17.



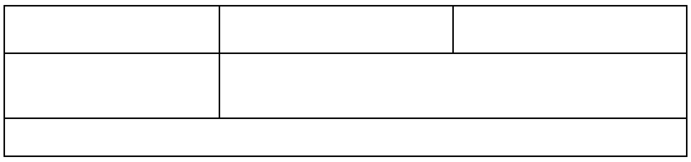
Option №18.



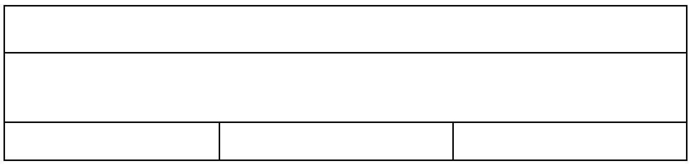
Option №19.



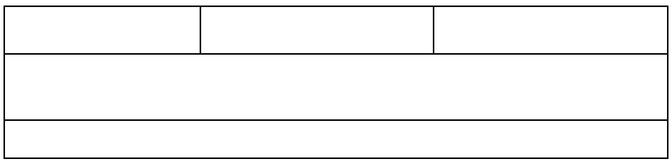
Option №20.



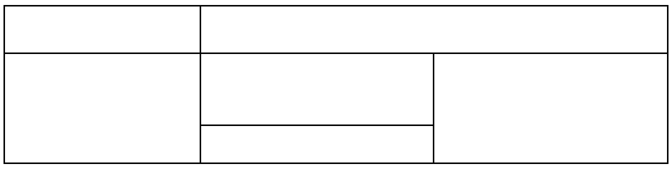
Option №21.



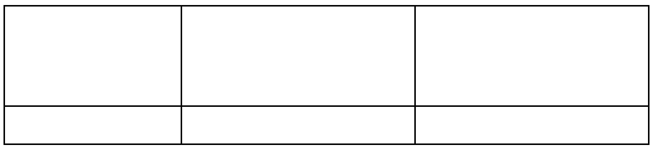
Option №22.



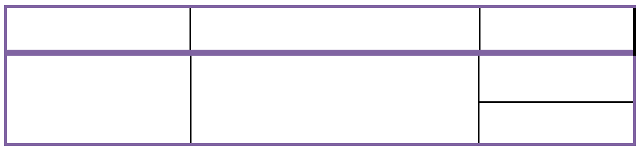
Option №23.



Option №24.



Option №25.



Laboratory №6
Theme: Additional CSS3 features

1. Objective

Learning cascading style sheets version 3, creating additional animation and special effects on CSS3

2. Required tools

A personal computer, among the creatures of web applications emmet-sublime-master or Notepad ++, the text editor MS Word

3. Brief information from theory

CSS3 is the latest standard for CSS. CSS3 is completely backwards-compatible with earlier versions of CSS.

Unlike CSS 2, which is a large single specification defining various features, CSS 3 is divided into several separate documents called "modules". Each module adds new capabilities or extends features defined in CSS 2, preserving backward compatibility. Work on CSS level 3 started around the time of publication of the original CSS 2 recommendation. The earliest CSS 3 drafts were published in June 1999.

Due to the modularization, different modules have different stability and statuses. As of June 2012, there are over fifty CSS modules published from the CSS Working Group, and four of these have been published as formal recommendations:

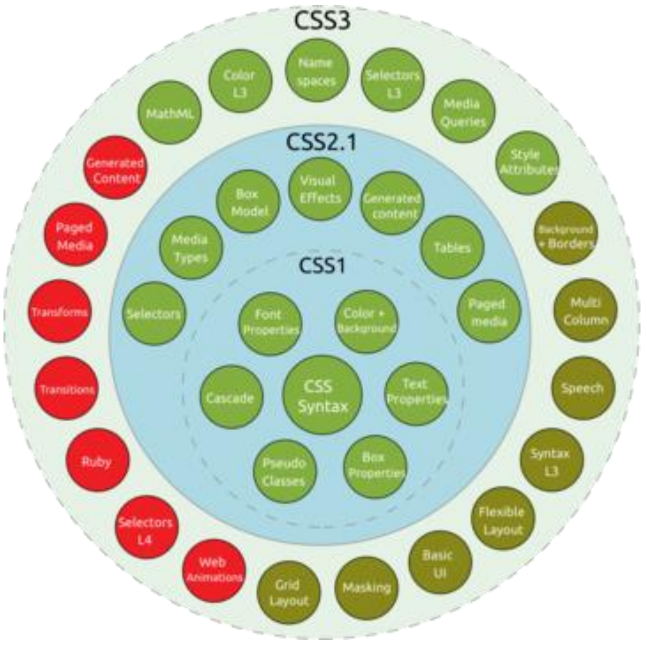
- 2012-06-19: Media Queries
- 2011-09-29: Namespaces
- 2011-09-29: Selectors Level 3
- 2011-06-07: Color

CSS3 has been split into "modules". It contains the "old CSS specification" (which has been split into smaller pieces). In addition, new modules are added.
Some of the most important CSS3 modules are:

- Selectors
- Box Model
- Backgrounds and Borders
- Image Values and Replaced Content
- Text Effects
- 2D/3D Transformations
- Animations
- Multiple Column Layout
- User Interface

Most of the new CSS3 properties are implemented in modern browsers.

Some modules have Candidate Recommendation (CR) status and are considered moderately stable. At CR stage, implementations are advised to drop vendor prefixes.



Pic. 6.1. Taxonomy and status of CSS3 **modules**

● Recommendation ● Candidate Recommendation ● Last Call ● Working Draft.

Tab.6.1. Summary of main module-specifications

Module	Specification title	Status	Date
css3-background	CSS Backgrounds and Borders Module Level 3	*Candidate* Rec.	Sep 2014
css3-box	CSS basic box model	Working *Draft*,	Aug 2007
css-cascade-3	CSS Cascading and Inheritance Level 3	*Candidate* Rec.	May 2016
css3-color	CSS Color Module Level 3	*Recommendation*	Jun 2011
css3-content	CSS3 Generated and Replaced Content Module	Working *Draft*	Jun 2016
css-fonts-3	CSS Fonts Module Level 3	*Candidate* Rec.	Oct 2013
css3-gcpm	CSS Generated Content for Paged Media Module	Working *Draft*	May 2014
css3-layout	CSS Template Layout Module	*Note*	Mar 2015
css3-mediaqueries	Media Queries	*Recommendation*	Jun 2012

Tab.6.1. Summary of main module-specifications

Module	Specification title	Status	Date
css3-multicol	Multi-column Layout	*Candidate* Rec.	Apr 2011
css3-page	CSS Paged Media Module Level 3	Working *Draft*	Mar 2013
css3-selectors	Selectors Level 3	*Recommendation*	Sep 2011
css3-ui	CSS Basic User Interface Module Level 3 (CSS3 UI)	Working *Draft*	Jul 2015

Each web browser uses a layout engine to render web pages, and support for CSS functionality is not consistent between them. Because browsers do not parse CSS perfectly, multiple coding techniques have been developed to target specific browsers with workarounds (commonly known as CSS hacks or CSS filters). Adoption of new functionality in CSS can be hindered by lack of support in major browsers. For example, Internet Explorer was slow to add support for many CSS 3 features, which slowed adoption of those features and damaged the browser's reputation among developers. In order to ensure a consistent experience for their users, web developers often test their sites across multiple operating systems, browsers, and browser versions, increasing development time and complexity. Tools such as BrowserStack have been built to reduce the complexity of maintaining these environments.

In addition to these testing tools, many sites maintain lists of browser support for specific CSS properties, including CanIUse and the Mozilla Developer Network. Additionally, the CSS 3 defines feature queries, which provide an @supports directive that will allow developers to target browsers with support for certain functionality directly within their CSS. CSS that is not supported by older browsers can also sometimes be patched in using Javascript polyfills, which are pieces of Javascript code designed to make browsers behave consistently. These workarounds—and the need to support fallback functionality—can add complexity to development projects, and consequently, companies frequently define a list of browser versions that they will and will not support.

As websites adopt newer code standards that are incompatible with older browsers, these browsers can be cut off from accessing many of the resources on the web (sometimes intentionally). Many of the most popular sites on the internet are not just visually degraded on older browsers due to poor CSS support, but do not work at all, in large part due to the evolution of JavaScript and other web technologies.

4. The order of performance of work

Contents of the report

1. The name of the laboratory work.
2. The purpose of laboratory work.
3. Required tools.
4. Brief information from theory.
5. The decision of an example on a variant.
6. Conclusion on the work.

Recommended literature

1. Ed Tittel, Jeff Noble. HTML, XHTML and CSS for Dummies, 7th Edition = HTML, XHTML & CSS For Dummies, 7th Edition. - Moscow: Dialectics, 2011. - 400 p. - ISBN 978-5-8459-1752-2.
2. Stephen Schafer. HTML, XHTML and CSS. The user's Bible, 5th edition = HTML, XHTML, and CSS Bible, 5th Edition. - Moscow: Dialectics, 2011. - 656 p. - ISBN 978-5-8459-1676-1.
3. Andy Budd, Cameron Mall, Simon Collison. CSS: Professional application of Web standards = CSS Mastery: Advanced Web Standards Solutions. - Moscow: Williams, 2008. - 272 p. - ISBN 978-5-8459-1199-5.
4. Christopher Schmitt. CSS. Programming recipes = CSS. Cookbook. - St. Petersburg: BHV-Petersburg, 2007. - 592 p. - ISBN 978-5-9775-0075-3.
5. Eric A. Meyer. CSS-cascading style sheets: detailed manual = Cascading Style Sheets: The definitive Guide. - Moscow: Symbol, 2006. - 576 p. - ISBN 5-93286-075-8.

4. Options for the task

1. Using CSS3 technology, show the capabilities and meaning of the Background attribute.
2. Using the CSS3 technology, show the capabilities and meaning of the Border attribute.
3. Using the CSS3 technology, show the possibilities and the value of the Font attribute.
4. Using the CSS3 technology, show the possibilities and meaning of BOX MODEL.
5. Using the CSS3 technology, show the possibilities and meaning of TEXT.
6. Using the CSS3 technology, show the possibilities and meaning of COLUMN.
7. Using the CSS3 technology, show the possibilities and meaning of the COLOR attribute.
8. Using the CSS3 technology, show the possibilities and meaning of TEMPLATE LAYOUT.
9. Using the CSS3 technology, show the possibilities and meaning of the TABLE tag.
10. Using the technology of CSS3, show the possibilities and meaning of SPEECH.
11. Using the CSS3 technology, show the possibilities and meaning of LIST & MARKERS.
12. Using the CSS3 technology, show the possibilities and meaning of ANIMATION.
13. Using the CSS3 technology, show the possibilities and meaning of TRANSITIONS.
14. Using the technology of CSS3, show the possibilities and meaning of GRID POSITIONING.
15. Using CSS3 technology, show the possibilities and meaning of OUTLINE
16. Using the CSS3 technology, show the possibilities and meaning of the LINE BOX.
17. Using CSS3 technology, show the possibilities and meaning of HYPERLINK.
18. Using CSS3 technology, show the possibilities and meaning of POSITIONING.
19. Using CSS3 technology, show the possibilities and meaning of RUBY.
20. Using CSS3 technology, show the possibilities and meaning of 2D TRANSFORM.
21. Using CSS3 technology, show the possibilities and significance of 3D TRANSFORM.
22. Using CSS3 technology, show the possibilities and meaning of GENERATED CONTENT.
23. Using CSS3 technology, show the capabilities and meaning of the transform
24. Using CSS3 technology, show the possibilities and the value of linear / radial-gradient.
25. Using CSS3 technology, show the possibilities and meaning of the transition.

Laboratory №7
Theme: Language programming JavaScript

1. Objective

Research and study of multi-paradigm programming language JavaScript

2. Required tools

A personal computer, among the creatures of web applications emmet-sublime-master or Notepad ++, the text editor MS Word

3. Brief information from theory

JavaScript (ˈdʒɑːvəˌskrɪpt), often abbreviated as JS, is a high-level, dynamic, weakly typed, prototype-based, multi-paradigm, and interpreted programming language. Alongside HTML and CSS, JavaScript is one of the three core technologies of World Wide Web content production. It is used to make webpages interactive and provide online programs, including video games. The majority of websites employ it, and all modern web browsers support it without the need for plug-ins by means of a built-in JavaScript engine. Each of the many JavaScript engines represent a different implementation of JavaScript, all based on the ECMAScript specification, with some engines not supporting the spec fully, and with many engines supporting additional features beyond ECMA.

As a multi-paradigm language, JavaScript supports event-driven, functional, and imperative (including object-oriented and prototype-based) programming styles. It has an API for working with text, arrays, dates, regular expressions, and basic manipulation of the DOM, but the language itself does not include any I/O, such as networking, storage, or graphics facilities, relying for these upon the host environment in which it is embedded.

Initially only implemented client-side in web browsers, JavaScript engines are now embedded in many other types of host software, including server-side in web servers and databases, and in non-web programs such as word processors and PDF software, and in runtime environments that make JavaScript available for writing mobile and desktop applications, including desktop widgets.

Although there are strong outward similarities between JavaScript and Java, including language name, syntax, and respective standard libraries, the two languages are distinct and differ greatly in design; JavaScript was influenced by programming languages such as Self and Scheme.

Although it was developed under the name Mocha, the language was officially called LiveScript when it first shipped in beta releases of Netscape Navigator 2.0 in September 1995, but it was renamed JavaScript when it was deployed in the Netscape Navigator 2.0 beta 3 in December. The final choice of name caused confusion, giving the impression that the language was a spin-off of the Java programming language, and the choice has been characterized[by whom?] as a marketing ploy by Netscape to give JavaScript the cachet of what was then the hot new Web programming language.

There is a common misconception that JavaScript was influenced by an earlier Web page scripting language developed by Nombas named C-- (not to be confused with the later C-- created in 1997). Brendan Eich, however, had never heard of C-- before he created LiveScript. Nombas did pitch their embedded Web page scripting to Netscape, though Web page scripting was not a new concept, as shown by the ViolaWWW Web browser. Nombas later switched to offering JavaScript instead of C-- in their ScriptEase product and was part of the TC39 group that standardized ECMAScript.

In December 1995, soon after releasing JavaScript for browsers, Netscape introduced an implementation of the language for server-side scripting with Netscape Enterprise Server.

Since the mid-2000s, additional server-side JavaScript implementations have been introduced, such as Node.js in 2009.

Simple examples

Variables in JavaScript can be defined using the var keyword:[

var x; *// defines the variable x and assigns to it the special value "undefined" (not to be confused with an undefined value)*

var y = 2; *// defines the variable y and assigns to it the value 2*

var z = "Hello, World!"; *// defines the variable z and assigns to it a string entitled "Hello, World!"*

Note the comments in the example above, both of which were preceded with two forward slashes.

There is no built-in I/O functionality in JavaScript; the run-time environment provides that. The ECMAScript specification in edition 5.1 mentions: D:\DI\2017-2018\Ð’ÐµÐ± Ð¸Ð»Ð¾Ð²Ð°Ð»Ð°Ñ□Ð½Ð¸ Ð¸ÑˆÐ»Ð°Ð± Ñ‡Ð¸ÐºÐ¸Ñˆ 652-15\Ð£ÐœÐš 2017\6\JavaScript - Wikipedia.htm - cite_note-46

… indeed, there are no provisions in this specification for input of external data or output of computed results.

However, most runtime environments have a console objectD:\DI\2017-2018\Ð’ÐµÐ± Ð¸Ð»Ð¾Ð²Ð°Ð»Ð°Ñ□Ð½Ð¸ Ð¸ÑˆÐ»Ð°Ð± Ñ‡Ð¸ÐºÐ¸Ñˆ 652-15\Ð£ÐœÐš 2017\6\JavaScript - Wikipedia.htm - cite_note-47 that can be used to print output. Here is a minimalist Hello World program in **JavaScript**:

```
console.log("Hello World!");
```

A simple recursive function:

```
function factorial(n) {
   if (n === 0 || n === 1) {
      return 1;  // 0! = 1! = 1
   }
   return n * factorial(n - 1);
}
factorial(3); // returns 6
```

An anonymous function (or lambda):

```
function counter() {
   var count = 0;
```

```
    return function() {
        return ++count;
    };
}
var closure = counter();
closure(); // returns 1
closure(); // returns 2
closure(); // returns 3
```

In JavaScript, objects are created in the same way as functions, this is known as a function object.

Object example:

```
function Ball(r) {
    this.radius = r; //the radius variable is local to the ball object
    this.area = pi*r**2;
    this.show = function(){ //objects can contain functions
        drawCircle(r); //references a circle drawing function
    }
}
myBall = new Ball(5); //creates a new instance of the ball object with radius 5
myBall.show(); //this instance of the ball object has the show function performed on it
```

This example shows that, in JavaScript, function closures capture their non-local variables *by reference*.

Variadic function demonstration (arguments is a special variable):

```
function sum() {
    var x = 0;
    for (var i = 0; i < arguments.length; ++i) {
        x += arguments[i];
    }
    return x;
}
sum(1, 2); // returns 3
sum(1, 2, 3); // returns 6
```

Immediately-invoked function expressions are often used to create modules, as before ECMAScript 2015 there was not built-in construct in the language. Modules allow gathering properties and methods in a namespace and making some of them private:

```
var counter = (function () {
    var i = 0; // private property

    return {   // public methods
        get: function () {
            alert(i);
            },
            set: function (value) {
               i = value;
            },
            increment: function () {
               alert(++i);
            }
         };
      })(); // module

      counter.get();       // shows 0
      counter.set(6);
      counter.increment(); // shows 7
      counter.increment(); // shows 8
```

4. The order of performance of work

Contents of the report

1. The name of the laboratory work.
2. The purpose of laboratory work.
3. Required tools.
4. Brief information from theory.
5. The decision of an example on a variant.
6. Conclusion on the work.

Recommended literature

1. Chuck Musciano and Bill Kennedy, "HTML: The Definitive Guide", O "Reily & Associates, Inc (1996).
2. html.doc, "Microsoft Time", Auramedia magazine, special issue. "Solutions of Microsoft", Issue 5 (1996).
3. Michael J. Hannah, "HTML Reference Manual" (1996), http://www.sandia.gov/ sci_compute / html_ref.html
4. HTML 3.2. Features at a Glance, http://www.w3.org/pub/WWW/MarkUp/Wilbur/features.html
5. Netscape extensions to HTML 3.0, http://home.netscape.com/assist/net_sites/html_extensions_3.html
6. HTML 2.0 Standart, http://www.w3.org/pub/www/markUp/html-spec
7. Using JavaScript in HTML, http://home.netscape.com/eng/mozilla/2.0/handbook/javascript/index.html
8. Stefan Koch, "Introduction to JavaScript" (1996), http://www.webcom.com/java/java-script/intro/index.htm

4. Options for the task

1. The date variable is in the date variable in the format '2025-12-31'. Convert this date to '31 / 12/2025 'format.
2. Given the line 'I'm teaching javascript!'. Find the number of characters in this line.
3. The date variable is in the date variable in the format '2025-12-31'. Convert this date to '31 .12.2025 'format.
4. The line 'I'm teaching javascript!' Is given. Find the position of the substring 'teach'.
5. The line 'I'm teaching javascript!' Is given. Using the split method, write each word of this line into a separate element of the array.
6. Given the array ['I', 'teach', 'javascript', '!']. Using the join method, convert the array to the string 'i + teach + javascript +!'.
7. Convert the first letter of the string to uppercase.
8. Convert the string 'var_test_text' to 'varTestText'. The script, of course, should work with any similar lines.
9. Convert the first letter of each word of the string to uppercase.
10. The variables a and b are given. Find the module of the difference a and b. Check the script for yourself for different a and b.
11. The variables a and b are given. Take the variable b from a and assign the result to the variable c. Make sure that in any case a positive value is written into the variable c. Check the script for a and b, equal to 3 and 5, 6 and 1, respectively.
12. The array arr is given. Find the arithmetic mean of its elements. Check the task on the array with elements 12, 15, 20, 25, 59, 79.
13. Write a script that will find the factorial of a number. The factorial (denoted!) Is the product (multiplication) of all integers, less than a given number, and itself. For example, 4! = 1 * 2 * 3 * 4.
14. Given the variable str, which stores any text. Implement the truncation of long text according to the following principle: if the number of characters of this text is greater than the specified in the variable n, then in the result variable we write the first n characters of the str string and add the ellipsis '...' to the end. Otherwise, we write the contents of variable str into the variable result
15. Create an associative array (object) of wages obj. Display the salary of Petit and Koli.
16. Create an array arr with elements 1, 2, 3, 4, 5 in two different ways.
17. A multidimensional array arr is given:
var arr = {
'ru': ['blue', 'red', 'green'],
'en': ['blue', 'red', 'green'],
};}; Output with its help the word 'blue'.
18. Create the array arr = ['a', 'b', 'c']. Display it using the alert function.
19. Using the array arr from the previous number, display the contents of the first, second and third elements.

20. Create the array arr = ['a', 'b', 'c', 'd'] and use it to display the string 'a + b, c + d'.
21. Create an array arr with elements 2, 5, 3, 9. Multiply the first element of the array by the second, and the third element by the fourth. Summarize the results, assign the variable result. Display the value of this variable.
22. Create an object with the days of the week. Keys in it should be numbers of days from the beginning of the week (Monday is the first, etc.). Display the current day of the week.
23. Now suppose that the day of the week number is stored in the variable day, for example, there is a number of 3. Output the day of the week corresponding to the value of day.
24. An array of [[1, 2, 3], [4, 5, 6], [7,8,9]] is given. Display the number 4 from this array.

Laboratory №8

Theme: Using the jQuery library

1. Objective

Exploring the installation on the web page of the jQuery library and taking advantage of the capabilities of this library

2. Required tools

A personal computer, among the creatures of web applications emmet-sublime-master or Notepad ++, the text editor MS Word

3. Brief information from theory

jQuery is a cross-platform JavaScript library designed to simplify the client-side scripting of HTML. It is free, open-source software using the permissive MIT License. Web analysis indicates that it is the most widely deployed JavaScript library by a large margin.

jQuery's syntax is designed to make it easier to navigate a document, select DOM elements, create animations, handle events, and develop Ajax applications. jQuery also provides capabilities for developers to create plug-ins on top of the JavaScript library. This enables developers to create abstractions for low-level interaction and animation, advanced effects and high-level, themeable widgets. The modular approach to the jQuery library allows the creation of powerful dynamic web pages and Web applications.

The set of jQuery core features—DOM element selections, traversal and manipulation—enabled by its selector engine (named "Sizzle" from v1.3), created a new "programming style", fusing algorithms and DOM data structures. This style influenced the architecture of other JavaScript frameworks like YUI v3 and Dojo, later stimulating the creation of the standard Selectors API.

Microsoft and Nokia bundle jQuery on their platforms. Microsoft includes it with Visual Studio for use within Microsoft's ASP.NET AJAX and ASP.NET MVC frameworks while Nokia has integrated it into the Web Run-Time widget development platform.

jQuery, at its core, is a Document Object Model (DOM) manipulation library. The DOM is a tree-structure representation of all the elements of a Web page. jQuery simplifies the syntax for finding, selecting, and manipulating these DOM elements. For example, jQuery can be used for finding an element in the document with a certain property (e.g. all elements with an h1 tag), changing one or more of its attributes (e.g. color, visibility), or making it respond to an event (e.g. a mouse click).

jQuery also provides a paradigm for event handling that goes beyond basic DOM element selection and manipulation. The event assignment and the event callback function definition are done in a single step in a single location in the code. jQuery also aims to incorporate other highly used JavaScript functionality (e.g. fade ins and fade outs when hiding elements, animations by manipulating CSS properties).

The principles of developing with jQuery are:

- Separation of JavaScript and HTML: The jQuery library provides simple syntax for adding event handlers to the DOM using JavaScript, rather than adding HTML event attributes to call JavaScript functions. Thus, it encourages developers to completely separate JavaScript code from HTML markup.
- Brevity and clarity: jQuery promotes brevity and clarity with features like chainable functions and shorthand function names.
- Elimination of cross-browser incompatibilities: The JavaScript engines of different browsers differ slightly so JavaScript code that works for one browser may not work for another. Like other JavaScript toolkits, jQuery handles all these cross-browser inconsistencies and provides a consistent interface that works across different browsers.
- Extensibility: New events, elements, and methods can be easily added and then reused as a plugin.

jQuery was originally released in January 2006 at BarCamp NYC by John Resig and was influenced by Dean Edwards' earlier cssQuery library. It is currently maintained by a team of developers led by Timmy Willison (with the jQuery selector engine, Sizzle, being led by Richard Gibson).

jQuery also has an interesting software license history. Originally licensed under the CC BY-SA 2.5, it was relicensed to the MIT license in 2006. At the end of 2006, it was dual-licensed under GPL and MIT licenses. As this led to some confusion, in 2012 the GPL was dropped and is now only licensed under the MIT license.

jQuery includes the following features:

- DOM element selections using the multi-browser open source selector engine Sizzle, a spin-off of the jQuery project
- DOM manipulation based on CSS selectors that uses elements' names and attributes, such as id and class, as criteria to select nodes in the DOM
- Events
- Effects and animations
- Ajax
- Deferred and Promise objects to control asynchronous processing
- JSON parsing
- Extensibility through plug-ins
- Utilities, such as feature detection
- Compatibility methods that are natively available in modern browsers, but need fall backs for older ones, such as inArray( ) and each( )
- Multi-browser (not to be confused with cross-browser) support

Including the library

The jQuery library is a single JavaScript file containing all of its common DOM, event, effects, and Ajax functions. It can be included within a Web page by linking to a local copy or to one of the many copies available from public servers. jQuery has a content delivery network (CDN) hosted by MaxCDN. Google and Microsoft host it as well.

```
<script src="jquery.js">      </script>
```

It is also possible to include jQuery directly from a CDN:

```
<script
  src="https://code.jquery.com/jquery-3.2.1.min.js"
  integrity="sha256-hwg4gsxgFZhOsEEamdOYGBf13FyQuiTwlAQgxVSNgt4="
  crossorigin="anonymous"></script>
```

Usage styles

jQuery has two usage styles:

- Via the $ function, which is a factory method for the jQuery object. These functions, often called *commands*, are *chainable* as they all return jQuery objects.
- Via $.-prefixed functions. These are *utility functions*, which do not act upon the jQuery object directly.

Access to and manipulation of multiple DOM nodes in jQuery typically begins with calling the $ function with a CSS selector string. This returns a jQuery object referencing all the matching elements in the HTML page. $("div.test"), for example, returns a jQuery object with all the div elements of class test. This node set can be manipulated by calling methods on the returned jQuery object or on the nodes themselves.

jQuery also includes .noConflict() mode, which relinquishes control of $. This is helpful if jQuery is used with other libraries that also use $ as an identifier. In no-conflict mode, developers can use jQuery as a replacement for $ without losing functionality.

It is possible to perform cross-browser Ajax requests using $.ajax(). Its associated methods can be used to load and manipulate remote data.

```
$.ajax({
  type: 'POST',
  url: '/process/submit.php',
  data: {
    name : 'John',
    location : 'Boston',
  },
}).done(function(msg) {
  alert('Data Saved: ' + msg);
}).fail(function(xmlHttpRequest, statusText, errorThrown) {
  alert(
    'Your form submission failed.\n\n'
      + 'XML Http Request: ' + JSON.stringify(xmlHttpRequest)
      + ',\nStatus Text: ' + statusText
      + ',\nError Thrown: ' + errorThrown);
});
```

This example posts the
data name=John and location=Boston to /process/submit.php on the server. When this request finishes the success function is called to alert the user. If the request fails it will alert the user to the failure, the status of the request, and the specific error.

4. The order of performance of work

Contents of the report

1. The name of the laboratory work.
2. The purpose of laboratory work.
3. Required tools.
4. Brief information from theory.
5. The decision of an example on a variant.
6. Conclusion on the work.

Recommended literature

1. Adam Freeman. jQuery for professionals = Pro jQuery. - M .: Williams, 2012. - 960 p. - ISBN 978-5-8459-1799-7.
2. Jason Lengstorf. PHP and jQuery for professionals = Pro PHP and jQuery. - M .: Williams, 2010. - P. 352. - ISBN 978-5-8459-1693-8.
3. The females of G. jQuery. Collection of recipes. - St. Petersburg: BHV-Petersburg, 2010. - P. 416. - ISBN 978-5-9775-0495-9.
4. John Resig (speaker) (2007-04-13). Advancing JavaScript with Libraries (Part 1) (Yahoo! Video). YUI Theater. Retrieved 2009-05-04.
5. John Resig (speaker) (2007-04-13). Advancing JavaScript with Libraries (Part 2) (Yahoo! Video). YUI Theater. Retrieved 2009-05-04.
6. Krill, Paul (2006-08-31). "JavaScript, .Net developers aided in separate project". InfoWorld. Retrieved 2009-05-04.
7. Taft, Darryl K. (2006-08-30). "jQuery Eases JavaScript, AJAX Development". eWeek. Retrieved 2009-05-04.
8. Jamsa Chris. Effective tutorial on creative Web-design. HTML, XHTML, CSS, JavaScript, PHP, ASP, ActiveX. Text, graphics, sound and animation. Per with English / Chris Jamsa, Conrad King, Andy Anderson - M .: "DiSoftTUP", 2005.- 672 p.

Options for the task

1. Given the object {js: ['jQuery', 'Angular'], php: 'hello', css: 'world'}. Print with it the word 'jQuery'.
2. Create a two-dimensional array. The first two keys are 'ru' and 'en'. Let the first key contain an element, which is an array of names of the days of the week in Russian, and the second - in English. Output with this array Monday in Russian and Wednesday in English (let Monday is a zero day).
3. The lang variable stores the language (it takes one of the values, either 'ru', or 'en' - either or both), and in the day variable - the day number. Print the day of the week corresponding to the variables lang and day. That is: if, for example, lang = 'ru' and day = 3 - then output 'environment'.
4. The variable min is a number from 0 to 59. Determine in what quarter of the hour this number falls (in the first, second, third or fourth). If the variable a is 10, print 'True', otherwise print 'False'.
5. If the variable a is zero, print 'True', otherwise print 'False'. Check the script for a, equal to 1, 0, -3.
6. The variable num can take 4 values: 1, 2, 3 or 4. If it has a value of '1', we'll write 'winter' in the variable result, if it has the value '2' - 'spring' and so on. Solve the problem through a switch-case.
7. In the day variable there is some number from the interval from 1 to 31. Determine in which decade of the month this number falls (in the first, second or third).
8. In the month variable there is some number from the interval from 1 to 12. Determine at what time of the year this month falls (winter, summer, spring, autumn).
9. Given a string consisting of characters, for example, 'abcde'. Check that the first character of this line is the letter 'a'. If this is the case, print 'yes', otherwise print 'no'.
10. A string with numbers is given, for example, '12345'. Check that the first character of this line is a digit 1, 2 or 3. If this is the case, print 'yes', otherwise print 'no'.
11. Given a string of 3 digits. Find the sum of these digits. That is, add as numbers the first character of the line, the second and third.
12. Given a string of 6 digits. Check that the sum of the first three digits is equal to the sum of the second three digits. If this is the case, print 'yes', otherwise print 'no'.
13. Display a column of numbers from 1 to 50.
14. Display a column of numbers from 11 to 33.
15. Output the column of even numbers in the range from 0 to 100
16. Use the loop to find the sum of numbers from 1 to 100.
17. An array with elements [1, 2, 3, 4, 5] is given. Use the for loop to display all of these elements on the screen.
18. An array with elements [2, 3, 4, 5] is given. Use the for loop to find the product of the elements in this array.
19. An array with elements [1, 2, 3, 4, 5] is given. Use the for loop to display all of these elements on the screen.

Laboratory №9

Theme: Event in jQuery

1. Objective

Exploring and research the jQuery library, an event in JavaScript and jQuery

2. Required tools

A personal computer, among the creatures of web applications emmet-sublime-master or Notepad ++, the text editor MS Word

3. Brief information from theory

jQuery is tailor-made to respond to events in an HTML page. All the different visitor's actions that a web page can respond to are called events. An event represents the precise moment when something happens.

Examples:

- moving a mouse over an element
- selecting a radio button
- clicking on an element

The term **"fires/fired"** is often used with events. Example: "The keypress event is fired, the moment you press a key".

Here are some common DOM events:

Mouse Events	Keyboard Events	Form Events	Document/Window Events
click	keypress	submit	load
dblclick	keydown	change	resize
mouseenter	keyup	Focus	scroll
mouseleave		Blur	unload

jQuery Syntax For Event Methods

In jQuery, most DOM events have an equivalent jQuery method.

To assign a click event to all paragraphs on a page, you can do this:

```
$("p").click( );
```

The next step is to define what should happen when the event fires. You must pass a function to the event:

```
$("p").click(function( ){
  // action goes here!!
});
```

Commonly Used jQuery Event Methods

$(document).ready( ) The $(document).ready( ) method allows us to execute a function when the document is fully loaded.

click( ) The click( ) method attaches an event handler function to an HTML element. The function is executed when the user clicks on the HTML element.

The following example says: When a click event fires on a <p> element; hide the current <p> element:

Example

```
 $("p").click(function( ){
   $(this).hide( );
});
```

dblclick( ) The dblclick( ) method attaches an event handler function to an HTML element. The function is executed when the user double-clicks on the HTML element:

Example

```
$("p").dblclick(function( ){
   $(this).hide( );
});
```

mouseenter( ) The mouseenter( ) method attaches an event handler function to an HTML element. The function is executed when the mouse pointer enters the HTML element:

Example

```
$("#p1").mouseenter(function( ){
   alert("You entered p1!");
});
```

mouseleave( ) The mouseleave( ) method attaches an event handler function to an HTML element. The function is executed when the mouse pointer leaves the HTML element:

Example

```
$("#p1").mouseleave(function( ){
   alert("Bye! You now leave p1!");
});
```

mousedown( ) The mousedown( ) method attaches an event handler function to an HTML element. The function is executed, when the left, middle or right mouse button is pressed down, while the mouse is over the HTML element:

Example

```
$("#p1").mousedown(function( ){
   alert("Mouse down over p1!");
});
```

mouseup( ) The mouseup() method attaches an event handler function to an HTML element. The function is executed, when the left, middle or right mouse button is released, while the mouse is over the HTML element:

Example

```
$("#p1").mouseup(function(){
   alert("Mouse up over p1!");
});
```
hover( ) The hover( ) method takes two functions and is a combination of the mouseenter( ) and mouseleave( ) methods. The first function is executed when the mouse enters the HTML element, and the second function is executed when the mouse leaves the HTML element:

Example
```
$("#p1").hover(function( ){
   alert("You entered p1!");
},
function( ){
   alert("Bye! You now leave p1!");
});
```

focus( ) The focus() method attaches an event handler function to an HTML form field. The function is executed when the form field gets focus:

Example
```
$("input").focus(function( ){
   $(this).css("background-color", "#cccccc");
});
```

blur( ) The blur( ) method attaches an event handler function to an HTML form field. The function is executed when the form field loses focus:

Example
```
$("input").blur(function( ){
   $(this).css("background-color", "#ffffff");
});
```

The on( ) Method

The on( ) method attaches one or more event handlers for the selected elements. Attach a click event to a <p> element:

Example
```
$("p").on("click", function( ){
   $(this).hide( );
});
```

Attach multiple event handlers to a <p> element:

Example
```
$("p").on({
   mouseenter: function( ){
      $(this).css("background-color", "lightgray");
   },
   mouseleave: function( ){
        $(this).css("background-color", "lightblue");
    },
    click: function( ){
        $(this).css("background-color", "yellow");
    }
});
```

4. The order of performance of work

Contents of the report

1. The name of the laboratory work.
2. The purpose of laboratory work.
3. Required tools.
4. Brief information from theory.
5. The decision of an example on a variant.
6. Conclusion on the work.

Recommended literature

1. Adam Freeman. jQuery for professionals = Pro jQuery. - M .: Williams, 2012. - 960 p. - ISBN 978-5-8459-1799-7.
2. Jason Lengstorf. PHP and jQuery for professionals = Pro PHP and jQuery. - M .: Williams, 2010. - P. 352. - ISBN 978-5-8459-1693-8.
3. The females of G. jQuery. Collection of recipes. - St. Petersburg: BHV-Petersburg, 2010. - P. 416. - ISBN 978-5-9775-0495-9.
4. John Resig (speaker) (2007-04-13). Advancing JavaScript with Libraries (Part 1) (Yahoo! Video). YUI Theater. Retrieved 2009-05-04.
5. John Resig (speaker) (2007-04-13). Advancing JavaScript with Libraries (Part 2) (Yahoo! Video). YUI Theater. Retrieved 2009-05-04.
6. Krill, Paul (2006-08-31). "JavaScript, .Net developers aided in separate project". InfoWorld. Retrieved 2009-05-04.
7. Taft, Darryl K. (2006-08-30). "jQuery Eases JavaScript, AJAX Development". eWeek. Retrieved 2009-05-04.
8. Robin Nixon. Create dynamic websites using PHP, MySQL, javascript and CSS (2nd edition). - O'Reilly Media, Inc.: Peter, 2013. - P. 560.
9. Sam Ruby, Dave Thomas, David Hanson. Rails 4. Flexible development of web applications. - St. Petersburg: Peter, 2014. - 448 p.

Options for the task

1. Create an object event using the Mouse Events (click)
2. Create an event using the Mouse Events (dblclick) method.
3. Create an event using the Mouse Events (mouseenter) method.
4. Create an event using the Mouse Events (mouseleave) method.
5. Create an event using the Keyboard Events method (keypress)
6. Create an event using the Events mousedown () method.
7. Create an event using the Events mouseup () method.
8. Create an event using the Events hover () method.
9. Create an event using the Events on () method.
10. Create an event using the Keyboard Events (keyup) method.
11. Create an event using the Form Events (submit) method.
12. Create an event using the Form Events (change) method.
13. Create an event using the Form Events (focus) method.
14. Create an event using the Form Events (blur) method.
15. Create an object event using the Document / Window Events (load) method.
16. Create an object event using the Document / Window Events (resize) method.
17. Create an object event using the Document / Window Events (scroll)
18. Create an object event using the Document / Window Events (unload) method.
19. Create object effects using the jQuery hide () method.
20. Create object effects using the jQuery show () method.
21. Create object effects using the jQuery toggle () method.
22. Create object effects using the jQuery fadeIn () method.
23. Create object effects using the jQuery method fadeOut ()
24. Create object effects using the jQuery method fadeToggle ()
25. Create object effects using the jQuery method fadeTo ()
26. Create object effects using the jQuery slideDown () method.
27. Create object effects using the jQuery slideUp () method.
28. Create object effects using the jQuery slideToggle () method.
29. Create object effects using jQuery Animations - The animate () Method
30. Create object effects using the jQuery animate () method - Manipulate Multiple Properties