

LABORATORY PRACTICUM

WITH HTML AND JAVASCRIPT
FOR STUDENTS OF ALL MAJORS

CONTENT

INTRODUCTION.....	4
PRINCIPLE OF HYPERTEXT TEXT MARKUP.....	4
GROUPS OF TAG IN HTML.....	5
Lab No. 1. A simple HTML page	6
Laboratory work No. 2. HTML-lists, pictures, HTML-tables.....	9
Laboratory work No. 3. Creation of new windows, frames.....	14
Laboratory work No. 4. Cascading style tables	18
Laboratory work No. 5. Basics of JavaScript.....	22
Laboratory work No. 6. Using functions in JavaScript.....	23
Laboratory work No. 7. Creating a clock using Java Script.....	26
Laboratory work No. 8. Events in Java Script.....	27
Laboratory work No. 9. Conditional expressions, assignment operators and comparison, logical operations, comment in Java Script	29
Laboratory work No. 10. Loop operators in Java Script.....	33
Laboratory work No. 11. Creation of dynamic interfaces by means of Java Script.....	36
Laboratory work No. 12. Working with databases using Java Script.....	39
LIST OF REFERENCES.....	41

INTRODUCTION

Methodical guidelines contain information about modern web technologies. The principles of hypertext markup, construction of simple pages, formatting of text, lists, tables, frames are consistently considered. Controlling page display using cascading style sheets is also briefly discussed.

The object-oriented scripting language JavaScript is described separately. Methods of code placement, loops, logical and mathematical operators, events, working with databases are consistently considered, examples of practical use of JavaScript code in web pages are given and analyzed.

Methodical instructions are intended for students and postgraduates who are engaged in web programming and use internet technologies in the educational process and scientific work.

THE PRINCIPLE OF HYPERTEXT TEXT MARKUP

HTML is a descriptive document markup language, it uses markup pointers (tags). The tag model describes a document as a collection of containers, each of which begins and ends with tags, that is, an HTML document is nothing more than an ordinary ASCII file with control HTML codes (tags) added to it.

The tags of HTML documents are mostly simple and clear, because they are formed with the help of commonly used words of the English language, understandable abbreviations and designations. An HTML tag consists of a name, which may be followed by an optional list of tag attributes. The tag text is enclosed in angle brackets ("`<`" and "`>`"). The simplest version of a tag is a name enclosed in angle brackets, for example, `<HEAD>` or `<I>`. A number of tags are characterized by the presence of attributes that can have specific values set by the author to change the function of the tag.

Tag attributes follow the name and are separated from each other by one or more paragraphs, spaces, or carriage returns. The order of writing attributes in the thesis does not matter. The value of the attribute, if any, follows the equal sign that follows the attribute name. If the value of the attribute is one word or number, then it can simply be specified after the equal sign, without highlighting it additionally. All other values must be enclosed in single or double quotes, especially if they contain multiple words separated by spaces. The length of the attribute value is limited to 1024 characters. The case of characters in tag and attribute names is not taken into account, which cannot be said about attribute values. For example, it is especially important to use the correct case when entering the URL (Uniform Resource Locator) of other documents as the value of the HREF attribute.

Most often, HTML markup elements, or HTML containers, consist of initial and final components, between which text and other document elements are placed. The name of the final tag is identical to the name of the initial one, but the name of the final tag is preceded by a slash (/) (for example, for the font style tag - italic `<I>`

closes the pair represents `</I>`, for the title tag `<TITLE>` closes will be a pair `</TITLE>`). End tags never contain attributes. By their meaning, tags are close to the concept of brackets "begin / end" in universal programming languages, which specify the scope of names of local variables, etc. Tags define the scope of rules for interpreting text documents.

When using nested markup elements in the document, special care should be taken. Nested tags must be closed, starting with the last one. Some markup elements do not have a final component because they are standalone elements. For example, the image tag ``, which serves to insert a graphic image into the document, does not require a final component. Standalone markup elements also include a line break (`
`), a horizontal ruler (`<HR>`), and tags that contain information about the document that does not affect its display, content, such as the `<META>` and `<BASE>` tags .

In some cases, the end tags in the document can be omitted. Most browsers are arranged in such a way that when processing the text of the document, the beginning tag is perceived as the end tag of the previous one. The most common tag of this type is the `<p>` paragraph tag. Since it is used very often in the document, it is usually placed only at the beginning of each paragraph. When one paragraph ends, the next `<P>` tag signals the browser to end that paragraph and start the next one.

The general scheme of building a container in HTML format can be written in the following form:

```
"Container" =  
<"Tag name" "attribute list">  
the contents of the container  
</ "Tag Name">
```

In addition to tags, HTML elements are CER (Character Entity Reference), they are designed to represent special characters in an HTML document that may not be correctly processed by the browser. Suppose an HTML document is created, which talks about the elements of this language. If you specify the name of the `<BODY>` tag simply in the document, the browser can perceive it as a direct start tag. CER is used to output such symbols.

For example, to represent the "<" character in an HTML document, you need to replace it with `<` and the ">" character with `>`. That is, if you specify the line `<BODY>` in the HTML text , it will appear on the screen as the text `<BODY>`.

TAG GROUPS IN HTML

All HTML tags can be divided into the following main groups according to their purpose and scope:

- determine the structure of the document;
- design of hypertext blocks (paragraphs, lists, tables, pictures);
- hypertext links and bookmarks;
- forms for organizing dialogue;
- calling programs.

The structure of a hypertext network is defined by hypertext links. A hypertext

link is the address of another HTML document or Internet information resource that is thematically, logically, or in any other way related to the document in which the link is defined.

Under such conditions, the addressing scheme of all available information resources is very important.

The actual mechanism for interpreting a resource identifier based on a URI (Uniform Resource Identifier) is called a URL, and that's what WWW users deal with.

The following example can be considered a typical example of using such a record:

This text contains:

```
<A HREF="http://www.udhtu.dp.ua/help/index.html">
```

```
hypertext link </ A>
```

In the above example, the "A" tag, which is called an anchor in HTML, uses the HREF attribute, which denotes a Hypertext Reference, to write this link in the form of a URL. This link points to a document named "index.html" in the "help" directory on the "www.udhtu.com.ua" server, which is accessed via the HTTP protocol.

Hypertext links in HTML are divided into two classes: contextual hypertext links and general. Contextual links are embedded in the body of the document, as demonstrated in the previous example, while general links are linked to the entire document as a whole and can be used when viewing any part of the document.

The structure of the HTML document allows nested containers to be used. Actually, the document itself is one big container that starts with the <HTML> tag and ends with the </HTML> tag.

LABORATORY WORKS

Laboratory work No. 1

A simple html page

The purpose of the work: - to learn how to make the simplest HTML pages, to get acquainted in practice with the concepts of container, header, page body, hyperlink, text formatting, use of CER.

Basics of designing HTML pages

In general, an HTML document is divided into a <HEAD> header and a <BODY> body

The header is intended for placing official information about the page, the content of this tag is not visible on the screen.

media content , etc. in it .

All visible information is contained within the <BODY> block

Block attributes:

BGCOLOR	Defines the page background color;
TITLE	This attribute allows you to display a pop-up tooltip;
TEXT	Defines the basic color of the text in the document;

LINK	Defines the color of the hyperlink in the document;
ALINK	Determines the color of the hyperlink highlighting when clicked;
VLINK	Defines the color of the visited link;
BGPROPERTIES	This attribute sets the properties of the background image;
BACKGROUND	Defines an image and fills the background of the page with this image.

Headings define the beginning of a document section, there are 6 levels from <H1> to <H6>

The <P> tag is used to highlight paragraphs, its ALIGN attribute is designed to align the text in the center, right or left edge, and justify

Tags controlling the display form:

Tag	Value;
<I> ...</ I>	Italic;
 ...</ B>	Amplification (Vold);
<TT> ...</ TT>	Teletype;
<U> ...</ U>	Underline;
<S> ...</ S>	Strikethrough text;
<BIG> ...</ BIG>	Increased font size;
<SMALL> ...</ SMALL>	Reduced font size ;
<SUB> ...</ SUB>	Substring characters;
<SUP> ...</ SUR>	Superstring characters.

These tags allow nesting, so they all have a start tag and an end tag. When using such tags, it should be remembered that their display depends on the settings of the user interface program, which may not coincide with the settings of the hypertext developer program.

Tags characterizing the type of information

 ...</ EM>	typographical enhancement ;
<CITE> ...</ CITE>	Citation ;
 ...</ STRONG>	Gain ;
<CODE> ...</ CODE>	Displays code examples;
<SAMP> ...</ SAMP>	Sequence of literals;
<KBD> ...</ KBD>	An example of entering characters from the keyboard
;	
<VAR> ...</ VAR>	Variable ;
<DFN> ...</ DFN>	Definition ;
<Q> ...</ Q>	Text enclosed in double quotes .

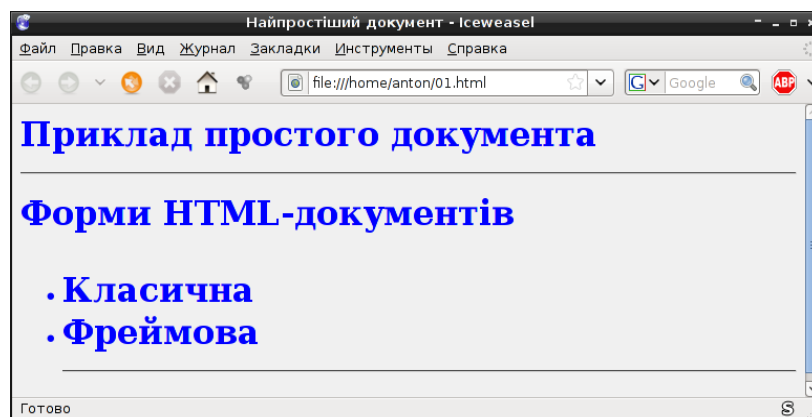
An example of the simplest HTML page is the code of the following type

```

<HTML>
<HEAD>
<TITLE> The simplest document </ TITLE>
</HEAD>
<BODY TEXT=#0000ff BGCOLOR=#f0f0f0>
<H1> Example of a simple document </ H1>
<HR>
Forms of HTML documents
<UL>
<LI> Classic
<LI> Freymov
</UL>
<HR>
</BODY>
</ HTML>

```

Such a page looks like this



References are specified using type anchors

```
<A HREF="http://www.google.com"> site link </ A>
```

when writing addresses to local sources (pictures, links to neighboring pages), you can use both a backslash "\" and a simple slash "/", but it is better to use just "/".

For example, a link to a page in the same folder as the one being called would look like this:

```
<A HREF="/page.html"> text </ A>
```

Clicking on the text will bring up page.html

In addition to tags, HTML elements are CER (Character Entity Reference), they are designed to represent special characters in an HTML document that may not be correctly processed by the browser.

CERs begin with an "&", CER names are case-sensitive. Also, CER names can be specified not in the form of a name, but with the help of three-digit character codes in the form of & # nnn;. The following table lists the most commonly used CERs and their corresponding numeric codes.

Numeric code	Nominal	Symbol	Description
--------------	---------	--------	-------------

	replacement		
"	"	"	Foot
&	&	&	ampersand
<	<	<	Less
>	>	>	More
 	 		Non-breaking space
¡	&ie excl;	¡	Inverted exclamation mark
¢	& cent;	¢	Cent
£	£	£	Pound
¤	¤	¤	Currency
¥	¥	¥	Yen
¨	¨	¨	Umlaut
©	©	©	Copyright
«	"	"	Left square foot
®	®	®	Registered trademark
±	±	±	Plus or minus
»	"	»	Right corner foot

Task: create a mini-site on any topic with information about yourself. The site should contain 3 pages with the use of cross-references, formatted text (italics, underlined and reinforced texts, headings of three levels). If necessary, use CER.

Laboratory work No. 2

Html-lists, pictures, html-tables

The purpose of the work: learn how to design HTML pages using lists, tables, and insert pictures.

Lists

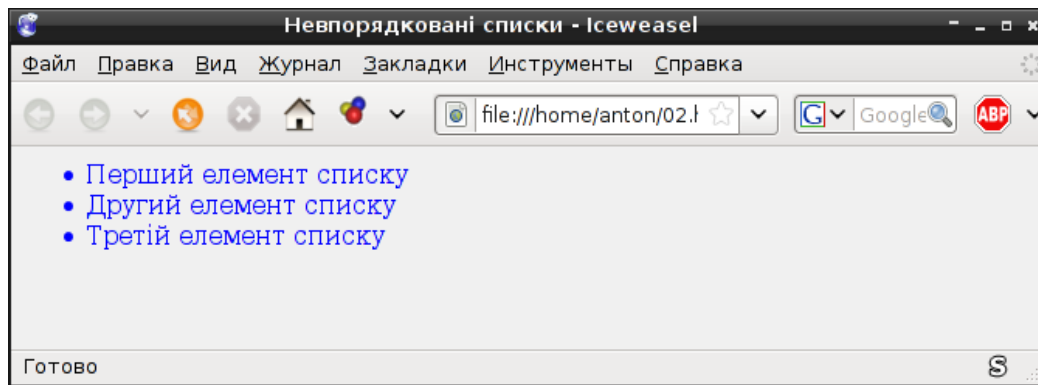
Unordered lists - tag

 The first element of the list

 The second element of the list

 The third element of the list

Such lists have the following form:



The `` and `` tags are the beginning and end tags of an unnumbered list, the `` (List Item) tag specifies the list item tag. In addition to these tags, there is a tag that allows you to name lists - `<LH>` (List Header).

Marker attributes in an unnumbered list

The `<UL TYPE=DISC>` tag creates solid tokens of the type found in default first-level lists.

`<UL TYPE=CIRCLE>` The tag creates markers in the form of circles.

The `<UL TYPE=SQUARE>` tag creates solid square markers.

Ordered lists - `` tag

Numbered lists. The `` tag together with the `TYPE =` attribute in HTML 3.2 allows you to create numbered lists using not only ordinary numbers, but also lowercase and uppercase letters, as well as lowercase and uppercase Roman numerals. If necessary, you can even mix these types of numbering in one list:

`<OL TYPE = 1>` The tag creates a list with numbering in the format 1., 2., 3., 4., etc.

`<OL TYPE = A>` The tag creates a list with numbering in the format A., B., S., D., etc.

`<OL TYPE=a>` The tag creates a numbered list in the format a., B., c., D., etc.

`<OL TYPE = I>` The tag creates a list with numbering in the format I., II., III., IV. etc.

Drawings

An example of inserting an image:

```
<IMG SRC="image.gif" ALT="IMAGE">
```

Attributes and their arguments

An image tag has one mandatory SRC attribute and the following optional attributes: ALT, ALIGN, USEMAP, HSPACE, VSPACE, BORDER, WIDTH, HEIGHT.

The SRC attribute indicates the image file and the path to it; the image must be added to the browser and placed in the place of the document where the image tag is located.

The ALT attribute allows you to specify text that will be displayed instead of an image by browsers that cannot display graphics. In some cases, when the bandwidth of communication lines is insufficient, users disable the display of graphics. The presence of names instead of pictures makes it easier to perceive Web pages in this mode.

The ALIGN attribute determines the position of the image in relation to its surrounding text. Possible values of the argument are ["top" | "middle" | "bottom"] (respectively, "on top", "in the middle", "below").

ALIGN = "top" aligns the top of the image to the top edge of the tallest element in the surrounding text line.

ALIGN = "middle" aligns the center of the image to the baseline of the surrounding text line.

ALIGN = "bottom" aligns the bottom edge of the image to the baseline of the surrounding text line.

In addition to the main values of the ALIGN = "keyword" attribute, there are a number of other arguments that expand the possibilities of mutual placement of graphics and text. Let's consider them in more detail.

Additional possible values of the argument are ["left" | "right" | "top" | "texttop" | "middle" | "absmiddle" | "baseline" | "bottom" | "absbottom"].

ALIGN = "left" defines the text to surround the image. The image is located along the left border of the document, and the following lines of text wrap around it on the right.

ALIGN = "right" defines the text to surround the image. The image is placed along the right border of the document, and the following lines of text wrap around its pages.

ALIGN = "top" aligns the top of the image to the top edge of the tallest element in the surrounding text line exactly as if using the standard attribute set.

ALIGN = "texttop" aligns the top of the image to the top edge of the tallest text character in the surrounding text line. The action of this argument is in most cases, but not always, similar to the action of the ALIGN = "top" argument.

ALIGN = "middle" aligns the center of the image to the baseline of the surrounding text line exactly as when using the standard set of attributes.

ALIGN = "absmiddle" aligns the center of the image to the center of the surrounding text line.

ALIGN = "baseline" aligns the bottom edge of the image with the baseline of the surrounding text line, i.e. it has the same effect as ALIGN = "bottom".

ALIGN = "bottom" aligns the bottom edge of the image to the baseline of the surrounding text line exactly as when using the standard set of attributes.

ALIGN = "absbottom" aligns the bottom edge of the image to the bottom edge of the surrounding text line.

If the USEMAP attribute and <MAP> tags are present, the image becomes a responsive map, or "graphical menu". If you click the mouse button on the active area of the image for which the USEMAP attribute is defined, a hypertext transition will occur to the information resource set for that area.

The BORDER attribute. The integer value of the argument specifies the thickness of the border around the image. If the value is zero, there is no frame. In order not to confuse users, you should not use BORDER = 0 in images that are part of an anchor element, because images that are used as hyperlinks are usually highlighted with a colored border.

The HSPACE attribute. An integer value of this attribute specifies the horizontal

distance between the vertical border of the page and the image, and between the image and the text.

The VSPACE attribute. The integer value of this attribute specifies the vertical distance between the lines of text and the image.

WIDTH and HEIGHT attributes. Both attributes set integer values for the horizontal and vertical dimensions of the image, respectively. This allows you to reduce the time of loading a page with graphics. The browser immediately removes the frame for the image and continues to load the text onto the page. While the graphics are loading, the user can start reading the text. Determining the size of the image is not difficult, for this it is enough to use any program for viewing graphic files, for example ACDSee or the graphics editor Corel PhotoPaint or Adobe Photoshop. Open the file in a graphic editor and define the size of the image in pixels. The width and height of the image should be specified in the title of the image.

```
<IMG SRC = "image.gif" ALT = "image" WIDTH = "100" HEIGHT = "200"
HSPACE = "10" VSPACE = "10"
BORDER = "2" ALIGN = "left">
```

Tables

The <TABLE> tag is used to describe the tables. The <TABLE> tag, like many others, automatically translates the line before and after the table.

Creating a row of a table - <TR> tag

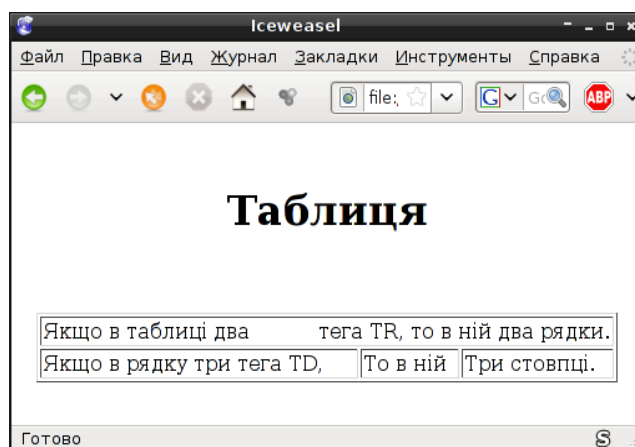
The <TR> (Table Row) tag creates a table row. All text, other tags, and attributes that need to fit on a single line must be placed between <TR> </TR> tags.

The definition of table elements is the <TD> tag

Cells with data are usually placed inside a table row. Each cell that contains text or an image must be surrounded by <TD> </TD> tags. The number of <TD> </TD> tags in a row determines the number of cells.

```
<HTML>
<BODY>
<H1 ALIGN=center> Table </H1>
<CENTER>
<TABLE BORDER>
  <TR>
    <TD COLSPAN=3> If the table has two
TR tag, then it contains two lines. </TD>
  </TR>
  <TR>
    <TD> If there are three TD tags in a line, </TD>
    <TD> It's in her </TD>
    <TD> Three columns. </TD>
  </TR>
</TABLE>
</CENTER>
</BODY>
</HTML>
```

The document obtained in this way will look like this



Task: create a mini-site with information about yourself. When creating a site, use tables, lists, and pictures. When organizing site content, use page formatting using tables.

No	Numbered lists	Unnumbered lists	Tables	No	Numbered lists	Unnumbered lists	Tables
1	3	6	5x3	16	3	6	5x4
2	4	5	5x4	17	7	7	6x3
3	5	4	6x3	18	6	3	6x4
4	6	3	6x4	19	5	4	5x3
5	7	7	7x3	20	4	6	5x4
6	3	6	7x4	21	3	7	6x3
7	4	5	5x3	22	6	3	6x4
8	5	4	5x4	23	7	4	7x3
9	6	3	6x3	24	3	3	7x4
10	7	7	6x4	25	4	7	5x3
11	3	6	7x3	26	5	6	5x4
12	4	5	7x4	27	6	5	6x3
13	5	4	5x4	28	5	4	6x4
14	6	3	6x3	29	4	3	7x3
15	7	7	6x4	30	3	4	7x4

Laboratory work No. 3

Creation of new windows, frames

The purpose of the work: to get acquainted with the concept of frames, to learn how to format sites using frames.

A frame is a separate, finished HTML document that can be displayed in a web browser window together with other HTML documents.

Frames are essentially similar to table cells, but more versatile. Frames divide a web page into separate mini-frames located on the same screen, which are independent of each other. Each window can have its own address. When you click on any of the links located in one frame, you can view the pages shown in another window.

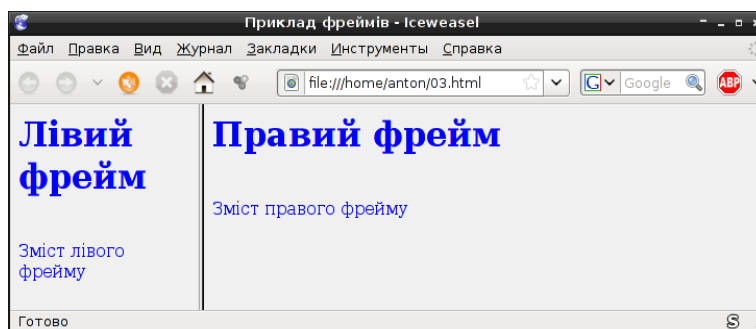
Frames were quite often used for website navigation. At the same time, the navigation page is located in one window, and the text pages in another.

Formation of the frame structure

To begin with, we need to imagine the general appearance of the page - where to place the frames and what size they will be. Then you can think about their content. Below is the code for a simple frame structure using the `<FRAMESET>` tag. Please note: the page with the frame structure does not contain the `<BODY>` tag.

```
<HTML>
<HEAD>
<TITLE> Example frames </ TITLE>
</HEAD>
<FRAMESET COLS="25%, 75%">
<FRAME SRC="menu.html">
<FRAME SRC="main.html" NAME="main">
</FRAMESET>
</ HTML>
```

The resulting HTML document will look like this



Preparation of frame content

Now let's load frames with content. Let's set the menu.html page in the left frame where we are going to click the mouse, switching between the two pages in the right frame. menu.html is a normal HTML page built as a table of contents. In fact, we can take a ready-made content page and use it. Keep in mind that this frame is narrow and tall, so the page that will load into it must be designed accordingly. Now we need to define where the other pages will appear when the mouse is clicked on the link. Since we want them to appear in the right frame, we'll add the TARGET attribute (TARGET = "main") to the link tag. This means that when the user clicks on the link, the page that is called appears in the main frame. We're rendering all pages

in the main frame, so let's add the TARGET = "main" attribute to all link tags in the content. If we do not define the TARGET attribute, the page will appear where we clicked with the mouse - in the left frame.

Preparation of the main frame

The right main frame will contain the HTML pages themselves. Your task is to design them so that they look good in a smaller window than usual, because part of the screen will be taken up by the left frame of the content. But these pages are nothing more remarkable.

Using the <NOFRAMES> tag

Some users still have browsers that do not know how to handle frames. For this reason, it's smart to provide access to a frameless version of your main pages. If a reader with an older browser lands on your framed page, anything between the <NOFRAMES> and </NOFRAMES> tags will look fine - the browser will simply ignore the frames. That is why you must use <BODY> </BODY> tags. Perhaps the screen without frames will have to be organized differently.

An example of a page with a frame structure with a <NOFRAMES> section added at the end.

```
<HTML>
<HEAD>
<TITLE> Example frames </TITLE>
</HEAD>
<FRAMESET COLS="25%, 75%">
<FRAME SRC="menu.html">
<FRAME SRC="main.html" NAME="main">
<NOFRAMES>
You are viewing this page using a browser that does not support frames.
</NOFRAMES>
</FRAMESET>
</HTML>
```

Note that a browser that supports frames will ignore anything between the <NOFRAMES> and </NOFRAMES> tags. Conversely, a browser that doesn't support frames will ignore anything between the <FRAMESET> and </FRAMESET> tags. The code without frames can be placed both at the beginning and at the end of the page.

Layout of frames - <FRAMESET> tag

<FRAMESET> tags enclose text describing the layout of the frames. Information about the number of frames, their sizes and orientation (horizontal or vertical) is placed here. The <FRAMESET> tag has only two possible attributes: ROWS, which specifies the number of rows, and COLS, which specifies the number of columns. The <WHERE> tag is not required between the <FRAMESET> tags, but can be placed between the <NOFRAMES> tags at the end of the frame structure. There should not be any tags or attributes between <FRAMESET> tags that are normally used between <BODY> tags. The only tags that can come between the

<FRAMESET> and </FRAMESET> tags are the <FRAME>, <FRAMESET>, and <NOFRAMES> tags. This simplifies the task. Basically everything is related to <FRAME> tags and their attributes. If you want to experiment, you can create nested <FRAMESET> tags similar to <TABLE> tags.

ROWS and SOLS attributes. Each row and column mentioned in the <FRAMESET> thesis requires its own set of <FRAME> tags.

The ROWS attribute of the <FRAMESET> tag specifies the number and size of rows on the page. The number of <FRAME> tags should correspond to the specified number of lines. To the right of the "=" sign, you can specify the size of each line in pixels, as a percentage of the screen height, or in relative values (usually this is an indication to take up the remaining space). You should use quotation marks and commas, and leave spaces between attribute values. For example, the following entry forms a screen consisting of three lines: the height of the top is 20 pixels, the middle is 80 pixels, and the bottom is 20 pixels:

```
<FRAMESET ROWS="20, 80, 20">
```

The following tag - <FRAMESET> - creates a screen where the top row occupies 10% of the screen height, the middle row occupies 60%, and the bottom row occupies the remaining 30%:

```
<FRAMESET ROWS="10%, 60%, 30%">
```

Relative values can be specified in combination with fixed values expressed in percentages or pixels. For example, the following tag creates a screen where the top row is 20 pixels high, the middle row is 80 pixels high, and the bottom row takes up all the remaining space:

```
<FRAMESET ROWS="20, 80, *">
```

SOLS attribute. Columns are specified in the same way as rows. The same attributes apply to them.

The <FRAME> tag defines the appearance and behavior of the frame. This tag has no closing tag because it contains nothing. The whole point of the <FRAME> tag is its attributes. There are six of them: NAME, MARGINWIDTH, MARGINHEIGHT, SCROLLING, NORESIZE and SRC.

NAME attribute. If you want the corresponding page to be displayed in a certain frame when you click the mouse on the link below, you need to specify this frame so that the page "knows" what to download. In the previous examples, we called the large right frame main, and it was in it that the pages selected from the content in the left frame appeared. The frame in which the pages are displayed is called the target. Frames that are not targeted do not need to be named. For example, you can write the following line:

```
NAME SRC = "my.html" NAME = "main">
```

Target frame names must start with a letter or number. The same names are allowed to be used in several frame structures. By clicking the mouse, the corresponding pages will be displayed in the named frame.

The MARGINWIDTH attribute works similarly to the CELLPADDING table attribute. It sets the horizontal margin between the content of the frame and its borders. The smallest value of this attribute is 1. You cannot specify 0. You can not assign anything - the attribute is 6 by default.

The MARGINHEIGHT attribute works in the same way as MARGINWIDTH. It sets the margins at the top and bottom of the frame.

The SCROLLING attribute enables scrolling in the frame. Possible options: SCROLLING = yes, SCROLLING = no, SCROLLING = auto. SCROLLING = yes means that the frame will always have scroll bars, even if it is not needed. If you set SCROLLING = no, there will be no scroll bars even when needed. If the document is too large, and you set the mode without scrolling, the document will simply be cropped. The SCROLLING = auto attribute allows the browser to decide whether scroll bars are needed or not. If the SCROLLING attribute is missing, the result will be the same as using SCROLLING = auto.

As a rule, the user can change its size by moving the border of the frame with the mouse. It is convenient, but not always. Sometimes the NORESIZE attribute is required. Remember: all borders of the frame for which you set NORESIZE become fixed - accordingly, it may turn out that the sizes of neighboring frames will also become fixed. Use this attribute with caution.

The SRC attribute is used in the FRAME tag when designing a frame structure in order to determine which page will appear in a particular frame. If you do not set the SRC attribute for all frames, you will have problems. Even if the pages displayed in a frame are selected in a neighboring frame, you should at least specify a start page for each frame. If you do not specify the starting page and URL, the frame will be empty, and the results can be most unexpected.

To understand the TARGET attribute, it is necessary to return to a simple example with a content frame. When the user clicks the mouse on one of the links in the left frame, the corresponding page should appear in the right frame, while the content remains unchanged. To achieve this, you need to define the target frame TARGET, in which the page for each item of content will be displayed. Target frames are specified in left frame references. That's why all frames in the frame structure were assigned names. The right frame is called main, so you need to add the attribute TARGET = "main" in each link, as a result of which the corresponding page will appear in the main frame. Please note: each link contains the TARGET = "main" attribute, which, when clicked, displays the page in the main frame.

The TARGET attribute can be specified for several different tags. When used in the <YACE> thesis, it directs all links to a specific target frame, unless otherwise specified. You can set the TARGET attribute in the <AREA> tag in the active image or in the <FORM> tag. Frames are useful for organizing forms. Users will see both the form and the result of their selection at the same time. Usually, when the mouse is clicked on the Submit button, the form disappears, and a page with the selection results appears. A combination of forms and frames can be a convenient way to navigate.

Task: create a page with frames, such that one frame contains the navigation menu, the second contains general information, and the third contains photos.

Ensure that pages are always opened in the same frame, return to the main page - on each page.

Laboratory robot No. 4

Cascading style tables

The purpose of the work: to get acquainted with cascading style tables, to learn how to work with them.

CSS syntax

Formally, the display style of markup elements is specified by a reference in the markup element to the style selector. The style description syntax is generally distributed as follows:

selector [, selector [, ...]]

(Attribute: value;

[Attribute: value ;...])

or

selector selector [selector ...]

(Attribute: value;

[Attribute: value ;...])

The first option lists the selectors for which this style description applies. The second option sets the nesting hierarchy of the selector, for the set of which the style is defined. We remind you that in this case we are talking about descriptions of styles in text / css notation. Style descriptions are placed either inside the STYLE element or in an external file.

As a selector, you can use the name of the markup element, the name of the class, and the identifier of the object on the HTML page.

The attribute defines the property of the displayed element, for example, the left margin of the paragraph (margin-left), and the value (value) is the value of this attribute, for example, 10 typographic points () 10 pt.

Selector – the name of the markup element

When a Web site author wants to define a common style for all pages, he simply prescribes styles for all HTML markup elements that will be used on the pages. This makes it possible to compose pages from logical elements, and to describe the display style of elements in an external file.

This way of creating a site allows the author to change the appearance of all pages by making changes to the stylesheet file, rather than to the HTML page files.

The external file can look like this:

I, EM (color: # 003366; font-style: normal;)

AI (font-style: normal; font-weight: bold;

text-decoration: line-through;)

The first line of this description lists the element selectors that will be displayed the same way:

<I> It is italic </ I> and it is also italic </ EM>

The last line defines the display style of the italics embedded in the hypertext link:

 <I> intuit </ I> </ A>

In this case, the override is that the text is displayed inside the hypertext link

crossed out, and in bold.

The selector is the name of the class

The class name is not any standard HTML markup element name. It defines the description of the class of markup elements that will be displayed the same. In order to assign a markup element to one or another class, you need to use its CLASS attribute:

```
<STYLE>
. Test (color: white; background-color: black;)
</STYLE>

...
<P CLASS="test">
We will display this paragraph in white on a black background
</ P>

...
<P>
This is a <A CLASS="test"> hypertext link </ A>
we will display in white on a black background.
</ P>
```

In this way, any markup element can refer to the description of the display class. At the same time, it is not necessary for the marking elements to be of the same type. In the example, both a paragraph and a hypertext link in another paragraph are assigned to the same class.

The first dot in the class name can be omitted. It is set for reasons of preserving the unity of the description. For example, you can define display classes of the same type of markup elements:

```
a.menu (color: red; background-color: white;
text-decoration: none;)
a.paragraph (color: navy;
text-decoration: underline;)
```

In this example, the menu hyperlink class has one style description, while the paragraph hyperlink class has a completely different style description. At the same time, each of these classes cannot be applied to other markup elements, for example, a paragraph or a list. If the name of the markup element is not specified, it means that the class can be assigned to any markup element - the root class of the stylesheet. This is very similar to the designation of the root domain name in the domain name system. Actually, there is nothing strange here, because the system of object classes on the HTML page is a tree. Markup elements are tree nodes.

The selector is the identifier of the object

The Document Object Model describes a document as a tree of objects. The objects are: the document itself, its sections (DIV element), pictures, paragraphs, attachments and others. You can give each object a name and refer to it by name. This feature is used when programming pages on the client side.

The use of the object identifier is also justified in the case of modification of the

style description attribute for this object in its CSS description. Instead of two class descriptions that differ in only one parameter, you can create one class description and one object identifier description. The style description for an object is given by a string in which the selector is the name of this object with a leading "#":

```
a.mainlink (color: darkred;
text-decoration: underline;
font-style: italic;)
# Blue (color: # 003366)
...
<A CLASS=mainlink> main hypertext
link </ A>
<A CLASS=mainlink ID=blue> modified
hypertext link </ A>
```

Task: use cascading style sheets to change the display of the specified markup elements of your mini-site.

At the same time, apply all three described methods of changing the display style.

No	Title font	The font of the main text	Link font
1	14 pt, Verdana,	10 ptTimes New Roman	11 ptTimes New Roman
2	15 ptTahoma	11 ptVerdana	12 ptVerdana
3	16 ptArial	12 ptLucida Sans	13 ptTimes New Roman
4	17 ptTrebuchet MS	13 ptLucida Sans	14 ptImpact
5	14 ptLucida Sans	14 ptTimes New Roman	10 ptTahoma
6	15 ptImpact	10 ptTimes New Roman	11 ptTimes New Roman
7	16 ptArial	11 ptVerdana	12 ptVerdana
8	17 ptTimes New Roman	12 ptArial	13 ptLucida Sans
9	14 ptArial	13 ptTimes New Roman	14 ptLucida Sans
10	15 ptImpact	14 ptLucida Sans	10 ptArial
11	16 ptArial	10 ptTimes New Roman	11 ptTimes New Roman
12	17 ptTimes New Roman	11 ptVerdana	12 ptVerdana
13	14 ptTimes New Roman	12 ptLucida Sans	13 ptArial
14	15 ptImpact	13 ptTimes New Roman	14 ptLucida Sans
15	16 ptImpact	14 ptImpact	10 pt
16	17 ptLucida Sans	10 ptTimes New Roman	11 ptTimes New Roman
17	14 ptImpact	11 ptVerdana	12 ptVerdana

18	15 ptLucida Sans	12 ptLucida Sans	13 ptTrebuchet MS
19	16 ptImpact	13 ptTimes New Roman	14 ptLucida Sans
20	17 ptTrebuchet MS	14 ptLucida Sans	10 ptTrebuchet MS
21	14 ptLucida Sans	10 ptTimes New Roman	11 ptTimes New Roman
22	15 ptTahoma	11 ptTahoma	12 ptVerdana
23	16 ptImpact	12 ptLucida Sans	13 ptTimes New Roman
24	17 ptTahoma	13 ptImpact	14 ptTahoma
25	14 ptTrebuchet MS	14 ptLucida Sans	10 ptImpact
26	15 ptLucida Sans	10 ptTimes New Roman	11 ptTimes New Roman
27	16 ptTrebuchet MS	11 ptVerdana	12 ptVerdana
28	17 ptTimes New Roman	12 ptImpact	13 ptLucida Sans
29	14 ptImpact	13 ptLucida Sans	14 ptTahoma
30	15 ptTahoma	14 ptImpact	10 ptTimes New Roman

Laboratory work No. 5

Basics of JavaScript

Purpose of work : to acquire skills in writing JS scripts.

Theoretical information

JavaScript is a new scripting language developed by Netscape. Using JavaScript, you can create interactive Web pages. To run scripts written using the JavaScript language, a browser capable of working with JavaScript is required - for example, Netscape Navigator (from version 3.0) or Microsoft Internet Explorer (MSIE - from version 3.0). JavaScript script code is placed directly on an HTML page using the `<script>` tag. Events and their handlers are one of the most important issues in JavaScript programming. Events are mainly triggered by user actions. For example, a click on a button or a text field provokes the occurrence of the "Click" event. If the mouse cursor crosses any hypertext link, the MouseOver event, etc., occurs. There are several different types of events, in this lab we will only look at the onClick event. The following code is an example of a simple onClick event handler:

```
<form>
<input type="button" value="Click me" onClick="alert('Hello!!!')">
</form>
```

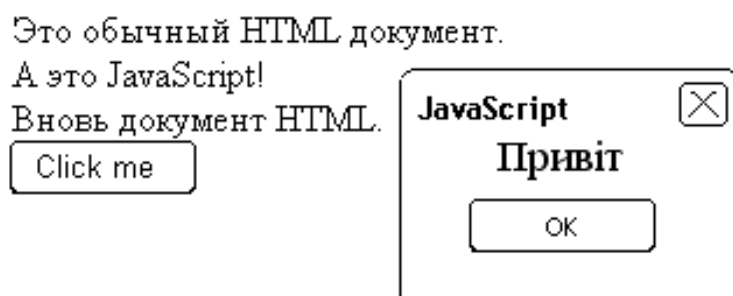
One of the most famous functions is alert(), which allows you to issue information or warnings to the user in a pop-up window. When it is called, some string is passed as a parameter, in the case presented below it is "Hello!!!". Thus, when the user clicks on a button or other element of the form, the script creates a window containing the text "Hello!!!".

Task : To develop an Internet page that, along with the usual html text, will contain the student's last name, first name, patronymic, printed using Java Script. Place a button on the page, when the user presses it, a message about the date and place of birth of the student will appear on the screen.

Example :

```
<html>
<body>
<br>
This is a normal HTML document.
<br>
<script language="JavaScript">
document.write("And this is JavaScript!")
</script>
<br>
HTML document again.
<br>
<form>
<input type="button" value="Click me" onClick="alert('Hello')">
</form>
</body>
</html>
```

Performance results:



Laboratory work #6

Using functions in JavaScript

Purpose of work : To acquire skills in using functions in the JavaScript language.

Theoretical information

Most JavaScript programs, like other programming languages, make extensive use of functions. In most cases, functions are just a combination of several commands. For example, consider a script that prints some text three times in a row:

```
<html>
<script language="JavaScript">
document.write("Welcome to my page!<br>");
document.write("This is JavaScript!<br>");
```

```
document.write("Welcome to my page!<br>");
document.write("This is JavaScript!<br>");
```

```
document.write("Welcome to my page!<br>");
document.write("This is JavaScript!<br>");
```

```
</script>
```

```
</html>
```

Such a script will write

Welcome to my page!

It's JavaScript!

three times. As we can see, the task was solved with the help of the fact that the corresponding part of the code was duplicated three times. This approach is not effective. The following script accomplishes the same task using a function.

```
<html>
```

```
<script language="JavaScript">
```

```
function myFunction() {
```

```
document.write("Welcome to my page!<br>");
```

```
document.write("This is JavaScript!<br>");}
```

```
myFunction();
```

```
myFunction();
```

```
myFunction();
```

```
</script>
```

```
</html>
```

And although the number of terms in this case did not change, the structure of the program became more understandable, readable, and suitable for further modification. In this program code, we used a function containing the following terms:

```
function myFunction() {
```

```
document.write("Welcome to my page!<br>");
```

```
document.write("This is JavaScript!<br>");}
```

The function keyword is used to define a function. myFunction in this case is the name of the function. A number of parameters can be specified in round brackets. All script commands between curly braces - {} - belong to the myFunction() function and are executed when it is called. This means that both document.write() commands are related and can be executed when this function is called.

Task : Develop an Internet page on which there is only one button. Assign the function of calculating the arithmetic operation selected by the number of the student in the list to the click event:

No	F	No	F
1	$F = x + 2y$	15	$F = \frac{1}{x + y + z}$

2	$F = 2x - 3y^2$	16	$F = \frac{x-y}{2xz}$
3	$F = \frac{1-x^2}{y^3}$	17	$F = 21xy - xz + xyz$
4	$F = 2x + 3y - z$	18	$F = x^3 + x^2 + x + xy^2$
5	$F = 3xy - y + 2x$	19	$F = x + y^2 + z^{0.5}$
6	$F = \frac{x}{y} + \frac{y^2}{1-xy}$	20	$F = x^{-0.5}$
7	$F = x - xy + y^2$	21	$F = x^6 + x^5y + 0.5z + x^2$
8	$F = x + \sqrt{y}$	22	$F = \frac{xz}{y}$
9	$F = xy + xz - yz$	23	$F = \frac{\sqrt{x}}{y+z}$
10	$F = 1 - xy + \frac{x}{y}$	24	$F = \frac{x-y^{0.5}}{xz}$
11	$F = \sqrt{x + \sqrt{y + \sqrt{2z}}}$	25	$F = -3xz + \sqrt{x^3}$
12	$F = \left(1 - \frac{x}{y}\right)^2 + z$	26	$F = x^{y^z} + 2xz$
13	$F = x^y + y^z + z^x$	27	$F = \frac{\sqrt{y}}{\sqrt[3]{xz}}$
14	$F = 1 - x - y - z$	28	$F = 2x + y - \frac{z}{xy}$

Example:

```

<html>
<head>
<script language="JavaScript">
function calculation() {
var x= 12;
var y= 5;
var result = x + y;
alert(result);
}
</script>
</head>
<body>
<form>
<input type="button" value="Calculate" onClick="calculation()">

```

```

</form>
</body>
</html>

```

Performance results:



Laboratory work No. 7

Creating a clock using Java Script

Theoretical information

In JavaScript, it is allowed to use some predefined objects. Examples of such objects can be Date, Array or Math. The Date object allows you to work with both the current time and the date. This is necessary, for example, in order to display the time and date on the current html document. To create an object, you need to use the new operator:

```
today = new Date()
```

In this case, a new Date object named today is created. If you did not specify the time and date when creating this object, it will be set to the current value. After executing the `today = new Date()` command, the today object will contain the date and time when this command was executed. The Date() object has several methods that can be applied to the today object. For example, these methods are `getHours()`, `setHours()`, `getMinutes()`, `setMinutes()`, `getMonth()`, `setMonth()`, etc. To fix any other date and time, we can use the modified constructor:

```
today = new Date(1997, 0, 1, 17, 35, 23)
```

At the same time, an object will be created, in which January 1, 1997, 17:35 and 23 seconds will be recorded. That is, you set the date and time according to the following pattern:

```
Date(year, month, day, hours, minutes, seconds)
```

Task : Use JavaScript to create a clock or calendar (see individual version) in the format according to the number in the list:

No	Format	No	Format
1	day:month:year	15	year:month:day:hour:min:sec
2	hour/min/sec	16	year-moun-day
3	year#moun#day-hour#min#sec	17	hour/min/sec+year*moun*day
4	year:month:day	18	min:sec:hour

5	hour\$min\$sec-year\$moun\$day	19	year%moun%day
6	day#moun#year	20	hour:min:sec-year:moun:day
7	sec/min/hour/min/sec	21	day/month/year
8	hour%min%sec	22	year-moun-day-hour-min-sec
9	hour:min:sec	23	year#moun#day
10	min/sec/hour	24	year-month-day\$hour-min-sec
11	min%sec%hour	25	hour-min-sec
12	day-moun-year	26	sec*min*hour*min*sec
13	day%moun%year	27	year/month/day
14	min-sec-hour	28	year+moun+day-hour+min+sec

Example:

```

<html>
<head>
<script Language="JavaScript">
var timeStr, dateStr;
function clock() {
now = new Date();
hours = now.getHours();
minutes = now.getMinutes();
seconds= now.getSeconds();
timeStr = "" + hours;
timeStr+= ((minutes < 10) ? ":0" : ":") + minutes;
timeStr+= ((seconds < 10) ? ":0" : ":") + seconds;
document.clock.time.value = timeStr;
date= now.getDate();
month = now.getMonth()+1;
year = now.getYear();
dateStr = "" + month;
dateStr+= ((date < 10) ? "/0" : "/" ) + date;
dateStr+= "/" + year;
document.clock.date.value = dateStr;
Timer= setTimeout("clock()",1000);
}
</script>
</head>
<body onLoad="clock()">
<form name="clock">
Time:
<input type="text" name="time" size="8" value=""><br>
Date:
<input type="text" name="date" size="8" value="">

```



```

</form>
</body>
</html>

```

Performance results:

Время: 14:09:44

Дата: 2/08/2010

Laboratory work No. 8

Events in Java Script

Example : using the data given in the table, create an Internet page with two text elements of the form. Assign to the events according to the variant of the tasks of the first element the issue of corresponding messages in the second text element.

HTML attribute	The condition of occurrence of the event
onBlur	Loss of focus by a form element
onChange	Changing the offset of an input field or text area, or selecting a new list item
onClick	A mouse click on a form element or hyperlink'
onFocus	Getting input focus from a form element
onLoad	Document download complete
onMouseOver	Moving the mouse pointer over the hyperlink
onMouseOut	Mouseover not on hyperlink'
onSelect	Highlight text in an input field or text area
onSubmit	Transferring form data
onUnload	Downloading the current document and starting to upload a new one

Task : create an html document in which to process events according to the option number.

No	Events	No	Events
1	onChange, onClick	15	onFocus, onMouseOut
2	onClick, onLoad	16	onBlur, onClick
3	onBlur, onMouseOver	17	onClick, onMouseOver
4	onFocus, onClick	18	onChange, onSelect
5	onLoad, onMouseOut	19	onFocus, onChange
6	onBlur, onChange	20	onLoad, onMouseOut
7	onChange, onMouseOver	21	onClick, onFocus
8	onFocus, onSelect	22	onChange, onLoad
9	onClick, onChange	23	onBlur, onSelect
10	onFocus, onSelect	24	onFocus, onMouseOver
11	onClick, onMouseOut	25	onChange, onFocus
12	onLoad, onMouseOver	26	onLoad, onMouseOut
13	onBlur, onSelect	27	onClick, onSelect
14	onChange, onMouseOut	28	onBlur, onMouseOver

Example:

```

<html>
<head>
<script>
function fun(a)
{
main_form.t_res.value = a;
}
</script>
</head>
<body onLoad = "fun('The page has been loaded');">
<form id = "main_form">
<input type="text" size = "100" value="Calculate" onClick="fun('The text
element of the form was clicked');" onBlur = "fun('Text form element has lost
focus');" onChange = "fun('The content of the text field of the form has changed');"
onFocus = "fun('The text element of the form received focus');" onSelect = "fun('Text
is selected in the text field of the form');" ><br>

<input type = "text" size = "100" id = "t_res" onMouseOver = "fun('The cursor
is on a hyperlink');" onMouseOut = "fun('The cursor is not on a hyperlink');"><br>

<a href = "d:\"" >Link</a>
</form>
</body>
</html>

```

Performance results:

Calculate
Відбувся клік на текстовому елементі форми
<u>Посилання</u>

Laboratory work No. 9**Conditional expressions, assignment and comparison operators, logical operations, comment in Java Script**

Example : using the Java Script language to implement the branching process of calculations according to the following formula:

$$d = \begin{cases} ab - c, abc > 0 \\ a - bc, abc < 0 \\ ac - \sqrt{b}, abc = 0 \end{cases}$$

Theoretical information : when developing a functional database of Internet pages using the JavaScript language, conditional operators are one of the necessary elements. Performing complex mathematical operations requires parallel checking of some conditions. The structure of the conditional operator in JavaScript has the following form:

if(<logical_expression>) <operator_1> else <operator_2>

In this case, <logical_expression> is a full or abbreviated version of a certain expression. Under the condition when <logical_expression> takes the value true, <operator_1> is executed, otherwise, that is, when its value is false <operator_2>. When constructing logical expressions, the following comparison operations are mostly used:

No	Marking	Content
1	==	is equal to
2	!=	is not equal to
3	>	more
4	>=	greater than or equal to
5	<	Less
6	<=	less than or equal to

It should be noted that in most cases, the following simplifications are used when writing logical expressions to shorten the program code and increase its readability:

<i>a</i> ==0	! <i>a</i>
<i>a</i> !=0	<i>a</i>
<i>str</i> ==""	! <i>str</i>
<i>str</i> !=""	<i>str</i>

There may be cases when, when constructing logical expressions, it is necessary to take into account not one but several conditions. The Java Script language provides for the use of special symbols for constructing complex conditional operators:

Marking	Operator
& &	and
	or

The structural element else of the conditional statement if is used in most cases to optimize the program code and improve its readability. It should be noted that in cases where the conditional if statement options are mutually exclusive, the use of the else statement allows you to reduce the amount of checks and, as a result, speed up the program. In the case when the selection options do not complement each other, the use of the else structural element can cause an error. Below are two code fragments that demonstrate not only the optimization properties of conditional operators, but also the options for their use:

Mutually exclusive conditions	Complementary conditions
-------------------------------	--------------------------

if(a==1) b=a; else if(a==2) b=a*a; else if(a==3) b=a*a*a; else if(a==4) b=1/a; else if(a==5) b=1/a*a; else	if(str.length) flag=1; if(((str[2]=="a") (str[2]=="k"))&&(str[0]=="")) flag=0; if(str.length>=5) flag=str.length if(str=="twenty") {number= 20; flag=0;}
------------------------------------------------------------------------------------------------------------------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------

Task : create an html document to implement a branched computing process according to the task option:

No	Task option	No	Task option
1	$F = \begin{cases} x + y, xy > 0 \\ x, x^2 > 100 \\ x^3 + y, \text{інакше} \end{cases}$	15	$F = \begin{cases} x + y, xy \neq 0 \\ x, x^3 > 100 \\ x^3 + y, \text{інакше} \end{cases}$
2	$F = \begin{cases} x + y, (x > 0) \text{та} (y > 0) \\ x^2 + y^2, (x < 3) \text{та} (y < 4) \\ x^3, \text{інакше} \end{cases}$	16	$F = \begin{cases} 2x + y, (x > 0) \text{та} (y > 0) \\ x^2 + 2y^2, (x < 3) \text{та} (y > 4) \\ x^3, \text{інакше} \end{cases}$
3	$F = \begin{cases} x^2 + y^2, (x < 10) \text{або} (y > 3) \\ x - y + x^2, x > 2 \\ x^3 + 3xy, \text{інакше} \end{cases}$	17	$F = \begin{cases} 2x^2 - 3y^2, (x < 10) \text{або} (y \neq 3) \\ 2x - 3y / x + x^2, x \neq 2 \\ x^3 + 3xy, \text{інакше} \end{cases}$
4	$F = \begin{cases} x / y + xy^2, (x > 0) \text{та} (y > 0) \\ y / x, (-x + y) / 2 > 0 \\ x^2, \text{інакше} \end{cases}$	18	$F = \begin{cases} 2x / y + xy^2, (x > 0) \text{та} (y \neq 0) \\ y / 2x, (2x + y) / 2 \neq 0 \\ x^2, \text{інакше} \end{cases}$
5	$F = \begin{cases} x - y^2, x - y > 2 \\ x / y, x > 0 \\ x^3 - y^2, \text{інакше} \end{cases}$	19	$F = \begin{cases} x - 2y^2, x - 3y \neq 2 \\ x - y / y, x > 0 \\ 2x^2 - 3y, \text{інакше} \end{cases}$
6	$F = \begin{cases} x / 2y, xy < -100 \\ x^2 - xy, xy > 20 \\ x^3 + 2, \text{інакше} \end{cases}$	20	$F = \begin{cases} 2x / y, xy \neq -100 \\ x^3 - 2xy, xy \neq 20 \\ x^2 + 4, \text{інакше} \end{cases}$
7	$F = \begin{cases} 2xy - 3, x / y > 0 \\ xy / 3, (x > 0) \text{та} (y < 3) \\ 2x - 3y, \text{інакше} \end{cases}$	21	$F = \begin{cases} xy - 1, x / y > 0 \\ 2xy / 2, (x \neq 0) \text{та} (y \neq 3) \\ x - y, \text{інакше} \end{cases}$

8	$F = \begin{cases} 2x + 3y, x < 3y \\ xy + x/3y, (y^2 > 2xy) \text{ або } (y > 0) \\ x^2 + y, \text{інакше} \end{cases}$	22	$F = \begin{cases} 10x + y, x \neq y \\ 2xy + 4x/y, (y^2 \neq 2xy) \text{ або } (y < 0) \\ x + y^2, \text{інакше} \end{cases}$
9	$F = \begin{cases} x^3 - \sqrt{x}, x^2 > 3 \\ xy, (x > 0) \text{ та } (y < 0) \\ x + y + xy - x/y, \text{інакше} \end{cases}$	23	$F = \begin{cases} x^3, x^3 \neq 3 \\ xy, (x \neq 0) \text{ та } (y < 0) \\ 2x + xy - x/2y, \text{інакше} \end{cases}$
10	$F = \begin{cases} 3xy - x, xy > 0 \\ 2x - 3y/x, (7x > 2y) \text{ або } (y > 2) \\ x - y, \text{інакше} \end{cases}$	24	$F = \begin{cases} xy - x, 2x/y \neq 0 \\ (x - 2y)/x, (14x \neq y) \text{ або } (2y < 0) \\ x - y, \text{інакше} \end{cases}$
11	$F = \begin{cases} 3xy, x > 0 \\ 2x/y, (x < 0) \text{ та } (y > 3/2) \\ x^2 + y^3, \text{інакше} \end{cases}$	25	$F = \begin{cases} x/y, x \neq 0 \\ 2x/y, (x \neq 0) \text{ або } (y > 3/2) \\ -x^3 + 2y^2, \text{інакше} \end{cases}$
12	$F = \begin{cases} x - y, xy > 0 \\ x^2 - y/x, (y < 0) \text{ та } (x > 3y) \\ 3xy - x^2y^2, \text{інакше} \end{cases}$	26	$F = \begin{cases} 2x - y, 2xy \neq 0 \\ x^2 + 2y/x, (y \neq 0) \text{ або } (x \neq 3y) \\ x/y - xy, \text{інакше} \end{cases}$
13	$F = \begin{cases} 2\sqrt{x} \cdot y, (y > 3) \text{ та } (x > 0) \\ x/y^2, (yx^2/2 > 3x) \text{ або } (x^2 > 2) \\ x^2/y, \text{інакше} \end{cases}$	27	$F = \begin{cases} (3/5)\sqrt{x} \cdot y, (y \neq 5) \text{ та } (x \neq 0) \\ x/y^2, (yx^2 \neq 3x) \text{ або } (x^3 \neq 5xy) \\ x/y^2, \text{інакше} \end{cases}$
14	$F = \begin{cases} -3x + y^3, (x > 0) \text{ та } (y < 0) \\ x + y - x^2y, xy - x < 3 \\ x^2 + y^2, \text{інакше} \end{cases}$	28	$F = \begin{cases} x + y^2, (x \neq 0) \text{ та } (y \neq 0) \\ x - y + x^3y^2, xy - x \geq 3 \\ x^2 - y^2, \text{інакше} \end{cases}$

Example:

```

<html>
<head>
<script>
function fun()
{
var a, b, c, d;
a = parseInt(main_form.t_a.value);
b = parseInt(main_form.t_b.value);
c = parseInt(main_form.t_c.value);

```

```

if(a*b*c>0) d=a*bc; else
if(a*b*c<0) d=ab*c; else
d=a*c-Math.sqrt(b);
main_form.t_d.value = "" + d;
}
</script>
</head>
<body>
<form id = "main_form">
a<input type="text" id = "t_a" value="10"><br>
b<input type="text" id = "t_b" value="-2"><br>
c<input type="text" id = "t_c" value="3"><br>
d<input type="text" id = "t_d" value=""><br>
<input type = "button" onClick = "fun();" value = "Calculate">
</form>
</body>
</html>

```

Performance results:

a	10
b	-2
c	3
d	16

Розрахувати

Laboratory work No. 10

Loop operators in Java Script

Theoretical information

The widespread use of loop operators when programming web pages is due not only to programming mathematical algorithms, but also to working with databases, creating interactive dynamic interfaces, etc. Loop operators in Java Script are of three types: For, while, do while. According to their functionality, they are called in the corresponding functions of the c/c++ language, the format of the for operator is as follows:

for(initialization; condition; increment)

in this case, "initialization" is a set of commands for initializing certain variables (if the variable occurs in the corresponding function, its type must be determined for the first time), separated by commas.

"condition" is a set of logical expressions separated from each other by commas

"increment" is a set of comma-separated commands executed after each cycle iteration. At the beginning of the execution of the loop operator, "initialization" operations take place, after which the "condition" is checked and if its value becomes true, all commands of the loop body are executed. After all operations are performed, the "increment" commands are executed, after which the "condition" is checked and the body of the loop is executed again. In most cases, the for loop statement is used

with a specified number of loop statements. The example below demonstrates the calculation of numbers from 1 to 1000 using the for operator:

```
var sum=0;
for(var i=1;i<=1000;i++)
sum+=i;
```

If the body of the loop must be executed while some condition is true, the while loop operator is used. The recording format of the while statement is as follows:

```
while(condition){...}
```

At the beginning of the execution of the loop operator, the "condition" is checked, if it is false, the loop will not be executed. If the "condition" takes the value true, the body of the loop is executed, after which the condition is checked again, and so on until the "condition" becomes false. One feature of the while function is that its body may not be executed. When the criterion for interrupting the execution of the loop occurs directly in its body, the do while loop operator can be used. Its construction is as follows:

```
do
{...}
while(condition)
```

At the beginning of the execution of the do while loop statement, the body of the loop is executed first, and only then the condition is checked. That is, the body of the cycle will occur at least once.

Task : Develop an html page that would organize the calculation of the sum of the series given by the formula according to the option:

No	Task option	No	Task option
1	$\sum_{i=1}^5 (-1)^i \frac{x+y-i!}{(i-1)!}$	15	$\sum_{i=2}^6 (-1)^{i+1} \frac{2xy-x^2}{(i+1)!}$
2	$\sum_{i=3}^5 (-1)^{i+1} \frac{x^2-y^3+3}{i!-5x}$	16	$\sum_{i=2}^7 (-1)^i \frac{i!-xy}{(i+1)!}$
3	$\sum_{i=7}^{11} (-1)^i \frac{x+y-i!}{(i-1)!}$	17	$\sum_{i=3}^5 (-1)^{i+1} \frac{2xy-y^2+x}{3xi!}$
4	$\sum_{i=5}^8 (-1)^{i+1} \frac{x/y+x^2/3}{x^2y^2(i+1)!}$	18	$\sum_{i=6}^{12} (-1)^i \frac{-2x+y^2}{i!}$
5	$\sum_{i=1}^5 (-1)^i \frac{2xy-i!}{1-x}$	19	$\sum_{i=3}^7 (-1)^{i+1} \left(\frac{7i-xy}{i+12} i^2 - (i+2)! \right)$
6	$\sum_{i=3}^6 (-1)^{i+1} \frac{x-y}{i!}$	20	$\sum_{i=2}^6 (-1)^i \frac{3x-y}{i!}$
7	$\sum_{i=2}^6 (-1)^i \frac{i+1}{i-2} i!$	21	$\sum_{i=2}^7 (-1)^i \frac{i!-2xy}{(i+2)!}$
8	$\sum_{i=7}^9 (-1)^{i+1} \frac{i!/x+y}{(i+1)!}$	22	$\sum_{i=1}^5 (-1)^i \frac{3y/x+y^2}{(i+1)!}$

9	$\sum_{i=4}^6 (-1)^i \frac{2x-y}{i!}$	23	$\sum_{i=2}^7 (-1)^{i+1} \frac{0.85i + xy/(1+x^2)}{i!}$
10	$\sum_{i=1}^{12} (-1)^{i+1} \frac{i+3}{(x+y)!}$	24	$\sum_{i=3}^5 (-1)^i \frac{3xy-x}{2y(i+x)!}$
11	$\sum_{i=3}^5 (-1)^i \frac{2xy-x^2}{i+2} i!$	25	$\sum_{i=7}^{11} (-1)^{i+1} \frac{+y}{2+x} (i+3)!$
12	$\sum_{i=2}^4 (-1)^{i+1} \frac{-x+xy^2}{2x} i!$	26	$\sum_{i=5}^8 (-1)^i \frac{12y/2x-x^2y^2(i+1)!}{i!}$
13	$\sum_{i=6}^{12} (-1)^i \frac{x^y - \sqrt{x}}{i!}$	27	$\sum_{i=1}^5 (-1)^{i+1} \frac{3xy-5}{i!}$
14	$\sum_{i=2}^5 (-1)^{i+1} \frac{x+2xy-y^2}{(i+1)!}$	28	$\sum_{i=7}^{10} (-1)^i \frac{y-x+xy+\sqrt{x}}{(i+y)!}$

Example:

```

<html>
<head>
<script>
function fun()
{
var a, b, sum=0.0, i=0.0, cur=0.0;
a = parseInt(main_form.t_a.value);
b = parseInt(main_form.t_b.value);
var to4nost = 0.001;
do
{
i++;
var factorial = 1;
for(var j=1;j<=i;j++)
factorial*=j;

cur = (2*a*i*i+i/b)/(factorial*factorial);
sum+=cur;
}while(cur>to4nost)
main_form.t_c.value = "" + sum;
}
</script>
</head>
<body>
<form id = "main_form">
a<input type="text" id = "t_a" value="10"><br>
b<input type="text" id = "t_b" value="-2"><br>
Result<input type="text" id = "t_c" value=""><br>

```



```

<input type = "button" onClick = "fun();" value = "Calculate">
</form>
</body>
</html>

```

Performance results:

```

a 10
b -2
Результат 44.7963868
Розрахувати

```

Laboratory work No. 11

Creation of dynamic interfaces using Java Script

Theoretical information : in most cases, informational html sites work online, continuously displaying information coming from certain databases, archives and other sources. From the very beginning, the developer does not know how the information will be selected, how many information feeds will be displayed on the main page, etc. That is why the use of static objects is inconvenient when creating interactive sites. Dynamic interfaces are one of the recognized tools for developing a flexible platform adapted to a specific activity. The essence of the method of dynamic interfaces is to place special <div> elements on the page. In the process of work, when the amount and essence of information that should be displayed on the page at a certain time is determined, the html code is automatically generated by the program itself. After that it is encapsulated in a <div>. Encapsulation of html code in a div occurs using the innerHTML function. In this way, it is possible to automatically create questionnaire forms by reading information from the database, while its editing becomes surprisingly simple and error-proof.

Task : to develop an html document in which there will be two text elements "number of rows" and "number of columns". According to their meaning, create a table of text elements (see example), each element of which should be filled in according to an individual option. For variants whose serial number is a multiple. For options 1, 6, 11, 16, 21, 26, calculate the sum of the elements of the obtained two-dimensional array. For options 2, 7, 12, 17, 22, 27, calculate the sum of the elements of the second row. For options 3, 8, 13, 18, 23, 28, calculate the sum of the elements of the third column. For options 4, 9, 14, 19, 24, calculate the number of elements whose row and column serial numbers match. For options 5, 10, 15, 20, 25, calculate the product of the elements of the last column.

No	Task option	No	Task option
1	$A_{ij} = 2i + j$	15	$A_{ij} = 7j + 2i$
2	$A_{ij} = 2i / j$	16	$A_{ij} = 2j - i$
3	$A_{ij} = 11i + j$	17	$A_{ij} = -5j + 3i$
4	$A_{ij} = 2i + 13j$	18	$A_{ij} = 3i - 2j$

5	$A_{ij} = 3i + 2j$	19	$A_{ij} = 11j - i$
6	$A_{ij} = 2i / 5j$	20	$A_{ij} = 2j / 5i$
7	$A_{ij} = 1/(2 + j)$	21	$A_{ij} = 2j + i$
8	$A_{ij} = 11j + i$	22	$A_{ij} = -7j / 3i$
9	$A_{ij} = 5i + 3j$	23	$A_{ij} = 11i - j$
10	$A_{ij} = 2i - j$	24	$A_{ij} = 2j / i$
11	$A_{ij} = -2i / 7j$	25	$A_{ij} = 3j + 2i$
12	$A_{ij} = 3i + j$	26	$A_{ij} = i - j$
13	$A_{ij} = 3j - 2i$	27	$A_{ij} = 5j + 3i$
14	$A_{ij} = -5i + 3j$	28	$A_{ij} = (3 - i)/(j + 5)$

Example:

```

<html>
<head>
<script>
var str, stb;
function fun()
{
str = parseInt(main_form.t_str.value);
stb = parseInt(main_form.t_stb.value);
var res_str = "<table>\n";
for(var i=1;i<=str;i++)
{
res_str+="

```

```

{
var sum = 0;
for(var j=1;j<=stb;j++)
{
res_str = "sum += parseInt(main_form._" + i + "_" + j + ".value);";
eval(res_str);
}
str_report += "Sum of " + i + " terms = " + sum + ";\n"
}
alert(str_report);
}
</script>
</head>
<body>
<form id = "main_form">
<table>
<tr>
<td>Number of terms</td>
<td> <input type="text" id = "t_str" value="5"></td>
</tr>
<tr>
<td> Number of columns</td>
<td><input type="text" id = "t_stb" value="5"></td>
</tr>
<tr>
<td> <input type = "button" onClick = "fun();" value = "Build matrix"><br>
<td>
</tr>
<tr>
<td> <input type = "button" onClick = "fun_build();" value = "Calculate"> <td>
</tr>
</table>
<div id = "main_div"></div>
</form>
</body>
</html>

```

Performance results:

Кількість строк

Кількість стовбців

Побудувати матрицю

Розрахувати

11	12	13	14	15
21	22	23	24	25
31	32	33	34	35
41	42	43	44	45
51	52	53	54	55

JavaScript

Сумма 1 строки = 65;
Сумма 2 строки = 115;
Сумма 3 строки = 165;
Сумма 4 строки = 215;
Сумма 5 строки = 265;

OK

Laboratory work No. 12

Working with databases using Java Script

Example : to develop an Internet page, when loaded, the content of one of the tables is unloaded from the database.

Theoretical information : when developing real sites, online stores, catalogs, or just an information page of a private entrepreneur, information about goods? services, company structure, information should be stored separately, and access to it should be controlled and limited in some way. The structure of requests should be simple and clear. For the most part, the information base of the site is located in the Access database (the type of database may be different, the differences in the software code will consist only in the connection of the appropriate driver). When working with a database, the main query language is the SQL language. The connection to the database is as follows:

```
ADO.Open("Provider=Microsoft.Jet.OLEDB.4.0;Data Source=db.mdb", "", "", 0);
```

where ADO is an ActiveXObject of type ADODB.Connection.

The request to the database is performed using the Execute method (SQL request). The result of the query will be a table, the columns of which correspond to those specified in the query, and the terms correspond to the corresponding tuples of the database. The MoveNext() method of the received table increments the sequence number of the current tuple of the received table. The EOF property of the resulting table indicates the end of the number of tuples in the queue. There is no possibility of reversing the tuples of the resulting table in JavaScript.

Task : create an “HTML Application” (*.hta) document that creates a connection to an existing database, executes a request to the tables and displays the table fields on the screen according to the individual option:

No	Task option <table name>.<field name>	No	Task option <table name>.<field name>
1	Books.Name, Books.Pages	15	Shop.name, Shop.Boss_name
2	Authors.Name, Authors.Date	16	Student.Name, Student.Group
3	Shop.name, Shop.Address	17	Country.Name, Country.Continent
4	Student.Name, Student.b_date	18	Computer.Proprietor, Computer.HDD
5	Country.Name, Country.Amount_People	19	Books.Nname, Books.Publisher

6	Computer.Proprietor, Computer.IP	20	Authors.Name, Authors.Address
7	Books.Name, Books.Authors	21	Shop.name, Shop.Code
8	Authors.Name, Authors.Surname	22	Student.Name, Student.Course
9	Shop.name, Shop.Tel_num	23	Country.Name, Country.Language
10	Student.Name, Student.Faculty	24	Computer.Proprietor, Computer.OS
11	Country.Name, Country.Capital	25	Books.Name, Books.Country
12	Computer.Proprietor, Computer.RAM	26	Authors.Name, Authors.Tel_num
13	Books.Name, Books.Year	27	Shop.name, Shop.Thematic
14	Authors.Name, Authors.m_town	28	Student.Name, Student.Hobby

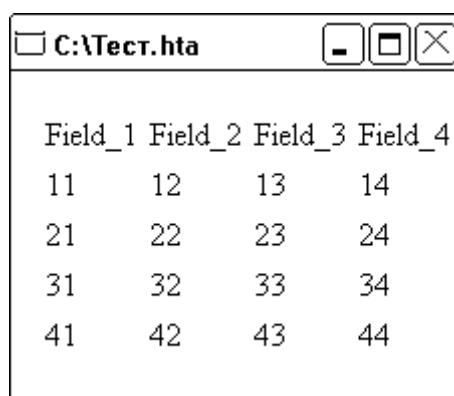
Example:

```

<html>
<head>
<script>
function pr_load()
{
var ADO = new ActiveXObject("ADODB.Connection");
var res_str = "";
ADO.Open("Provider=Microsoft.Jet.OLEDB.4.0;Data Source=db.mdb", "", "", 0);
GR = ADO.Execute("SELECT * FROM table_1");
if(GR.BOF && GR.EOF)
{
WS.Popup("There are no records in the database.", 60, "Error", 0+48);
window.close();
}
res_str += "<table border =
\"1\"><tr><td>Field_1<\td><td>Field_2<\td><td>Field_3<\td><td>Field_4<\
/td><\tr>";
do
{
res_str += "<tr><td>" + GR.Fields(0).Value + "<\td><td>" +
GR.Fields(1).Value + "<\td><td>" + GR.Fields(2).Value + "<\td><td>" +
GR.Fields(3).Value + "<\td><\tr>"
}while(GR.MoveNext(),!GR.EOF);
main_div.innerHTML = res_str;
ADO.Close();
}
</script>
</head>
<body onLoad = "pr_load();">
<form id = "main_form">
<div id = "main_div"></div>
</form>
</body>
</html>

```

Performance results:



Field_1	Field_2	Field_3	Field_4
11	12	13	14
21	22	23	24
31	32	33	34
41	42	43	44