Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ОДЕСЬКА ПОЛІТЕХНІКА»

МЕТОДИЧНІ ВКАЗІВКИ
до виконання лабораторних робіт з дисципліни «Веб–програмування»

Одеса
2024

Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ОДЕСЬКА ПОЛІТЕХНІКА»

МЕТОДИЧНІ ВКАЗІВКИ
до виконання лабораторних робіт з дисципліни «Веб–програмування» для студентів спеціальності 123 – «Комп'ютерна інженерія»

Затверджено на засіданні
кафедри комп'ютерних систем
Протокол № __ від _____

Одеса
2024

Методичні вказівки до виконання лабораторних робіт з дисципліни **«Веб–програмування»** для студентів спеціальності 123 усіх форм навчання – *«Комп'ютерна інженерія»* / Уклад.: *Дікусар К.В.* – Одеса: Національний університет «ОДЕСЬКА ПОЛІТЕХНІКА», 2024. – 71 с.

Укладачі: **К.В. Дікусар,** старший викладач.

Методичні вказівки містять десять лабораторних робіт згідно плану лекційного матеріалу.

Навчальний матеріал лабораторних робіт супроводжується основними теоретичними відомостями, графічними матеріалами та завданнями для обов'язкового виконання.

Поданий матеріал упорядковано за принципом від простого до складного, що сприяє кращому засвоюванню навчального матеріалу.

Зміст

ЛАБОРАТОРНА РОБОТА № 1

Тема роботи: Створення простого HTML-документу. Форматування шрифту і абзацу.

Мета роботи:

- Навчитися створювати простий гіпертекстовий документ засобами текстового редактора Блокнот.
- Навчитися використовувати теги форматування шрифту і абзацу.

Зміст протоколу до лабораторної роботи:

●Титульний аркуш (приклад у додатку 1);
●Мета роботи;
●Тексти HTML- документів;
●Копії екранів з відображенням HTML- документів у браузері;
●Висновки по роботі.

Основні довідкові матеріали (теги) для виконання лабораторної роботи представлені у таблицях 1.1 – 1.7.

Таблиця 1.1 – Основні теги HTML- документу

Призначення	Вид тегів	Примітка
Тип документу	<HTML></HTML>	Початок і кінець документу
Ім'я документу	<HEAD></HEAD>	Не відображається браузером
Заголовок	<TITLE></TITLE>	Вміст рядка заголовка вікна браузера
Тіло документу	<BODY></BODY>	Вміст WEB- сторінки

Таблиця 1.2 – Структура змісту документу

Призначення	Вид тегів	Примітка
Внутрішні заголовки різного рівня	<H№> текст </H№>	де № - номер рівня заголовка (від 1 до 6). Наприклад, <H1>.</H1> - заголовок 1-го рівня.
Заголовок з вирівнюванням	<H№ ALIGN=”LEFT| CENTER| RIGHT”> текст </H№>	LEFT – ліворуч, CENTER – по центру, RIGHT – праворуч.

Таблиця 1.3 – Теги форматування абзацу

Призначення	**Вид тегів**	**Примітка**
Створення абзацу (параграфа)	<P> текст </P>	Абзаци відділяються подвійним міжрядковим інтервалом
Перевод рядка усередині абзацу	 	Поодинокий тег
Вирівнювання абзацу	<P ALIGN=”LEFT”>текст </P> <P ALIGN= ”CENTER”>текст </P> <P ALIGN=”RIGHT”> текст</P> <P ALIGN=”JUSTIFY”> текст </P>	LEFT – ліворуч CENTER – по центру RIGHT – праворуч JUSTIFY – по ширині
Розділова горизонтальна лінія між абзацами	<HR SIZE=«?»>	Поодинокий тег. "?" – товщина лінії в пікселях. Товщину лінії можна не вказувати.

Таблиця 1.4 – Теги форматування шрифту

Призначення	**Вид тегів**	**Примітка**
Жирний	<B> текст </B>	<B> Жирний </B>
Курсив	<I> текст </I>	<I> *Курсив* </I>
Підкреслений	<U> текст </U>	<U> Підкреслений</U>
Перекреслений	<S> текст </S>	<S> ~~Перекреслений~~ </S>
Збільшений розмір	<BIG> текст </BIG >	
Зменшений розмір	<SMALL> текст </SMALL>	
Верхній індекс	^{текст}	^{Верхній індекс}
Нижній індекс	_{текст}	_{Нижній індекс}
Розмір шрифту	<FONT SIZE=«?»> текст </FONT>	? – значення від 1 до 7 або відносна зміна (наприклад, +2)
Базовий розмір шрифту	<BASEFONT SIZE=«?»>	Поодинокий тег ? – розмір від 1 до 7; за умовчанням рівний 3 і задається для усього документу в цілому
Гарнітура шрифту	<FONT FASE=«назва1, назва2, …»> текст </FONT>	Текст оформляється першим встановленим на комп'ютері шрифтом зі списку назв
Колір шрифту	<FONT COLOR=”$$$$$$”> текст </FONT>	Колір задається або ключовим словом, або шістнадцятирічним кодом з символом # RED – червоний, #FF0000 – шістнадцятирічний код червоного кольору

Таблиця 1.5 – Теги створення списків

Призначення	**Вид тегів**	**Примітка**
Нумерований	<OL>елементи списку </OL>	<OL> <LI>елемент списку 1 </LI> <LI> елемент списку2</LI> <LI> елемент списку3</LI> </OL>
Маркірований	<UL елементи списку </UL>	
Елемент списку	<LI елементи списку </LI>	

Таблиця 1.6 – Теги основних кольорів

Колір	**Назва**	**Шістнадцятирічний код кольору**		
		Red (R)	**Green (G)**	**Blue (B)**
Чорний	black	00	00	00
Темно-синій	navy	00	00	80
Блакитний	blue	00	00	FF
Зелений	green	00	80	00
Темно-зелений	teal	00	80	80
Салатовий	lime	00	FF	00
Блідо-голубий	aqua	00	FF	FF
Вишневий	maroon	80	00	00
Фіолетовий	purple	80	00	80
Олівковий	olive	80	80	00
Сірий	gray	80	80	80
Світло-сірий	silver	C0	C0	C0
Червоний	red	FF	00	00
Ліловий	fushsia	FF	00	FF
Жовтий	yellow	FF	FF	00
Білий	white	FF	FF	FF

Таблиця 1.7 – Основні параметри тега <BODY>

Параметр	**Значення**
text	Встановлює колір тексту документу, використовуючи значення кольору у вигляді RRGGBB. Наприклад: text=" 000000" – чорний колір.
link	Встановлює колір гіперпосилань, використовуючи значення кольору у вигляді RRGGBB. Наприклад: text="FF0000" – червоний колір.
vlink	Встановлює колір гіперпосилань, на яких користувач вже побував. Наприклад: text=" 00ff00" – зелений колір.
alink	Встановлює колір гіперпосилань при натисненні. Наприклад: text=" 0000ff" – синій колір.
bgcolor	Встановлює колір фону документу, використовуючи значення кольору у вигляді RRGGBB. Наприклад: text=" FFFFFF" – білий колір.
background	Встановлює зображення фону документу. Наприклад: Background="bg.gif"
Topmargin (marginheight*)	Встановлює величину верхнього поля документу. Наприклад: Topmargin="0"
Leftmargin (marginwidth*)	Встановлює величину лівого поля документу. Наприклад: Leftmargin="10"

*Параметри marginheight і marginwidth використовуються для установки величини верхнього і лівого поля у браузерах Netscape.

Escape-послідовності

У разі, якщо кодування або конфігурація обладнання і програмного забезпечення не дозволяє вводити певні символи, наприклад ©, ®, £ або §, можна використати посилання на символи, звані Escape- послідовностями. Escape- послідовності – це незалежний від кодування механізм введення будь-яких символів (див. таблицю 1.8).

Escape- послідовності в HTML можуть приймати дві форми:

- числові посилання на символи, наприклад, © (символ авторського права ©).
- буквені позначення символів, наприклад, © (символ авторського права ©).

Таблиця 1.8 – Список найчастіше використовуваних Escape- послідовностей

Символ	Назва символа	Escape-послідовності
<	менше	<
>	більше	>
&	амперсанд	&
"	лапки	"
	нерозривний пропуск	
©	символ авторського права	©
«	ліва лапка	«
»	права лапка	»

Завдання:

1. Створити текстовий документ, наприклад, з ім'ям index, з розширенням html. Відкрити його за допомогою програми Notepad++.
2. Ввести теги, що визначають структуру html- документу :

```
<HTML>
<HEAD> <TITLE> Прізвище</TITLE>
  </HEAD>
  <BODY>
Вітаю Вас на моїй першій web- сторінці!
  </BODY>
</HTML>
```

3. Зберегти документ і відкрити його у браузері.
4. Відредагувати документ – після тексту **"Вітаю Вас на моїй першій web-сторінці!"** текст підпису:

Студент групи NNN Прізвище Ім'я

5. Використовуючи одиночний тег
, відредагувати документ так, щоб підпис розпочинався з нового рядка, а **Прізвище Ім'я** – в наступному рядку. Проглянути у браузері новий варіант.

6. Оформити фрагменти тексту за допомогою стилів Заголовків – перший рядок документу оформити стилем **Заголовок 1-го** рівня за допомогою парного тега <H1> </H1>. Другий рядок оформити, як **Заголовок 6-го** рівня, а третій як **Заголовок 4-го рівня**.
7. Виконати форматування шрифту – після рядка **Прізвище Ім'я** додати ще один рядок тексту:

Нас ранок зустрічає прохолодою

8. Оформити фразу за приведеним нижче зразком:

~~Нас~~ *ранок* зустрічає прохолодою

У слові РАНОК усі букви повинні мати різні кольори. У слові ПРОХОЛОДОЮ оформити букви ПРО – червоним кольором, ОЮ – синім.

9. Оформити рядок з підписом **(Студент групи NNN Прізвище Ім'я)** курсивом.

10. Виконати оформлення списків – створити новий документ index2.html і ввести текст:

<HTML>
<HEAD> <TITLE> Прізвище </TITLE>
</HEAD>
<BODY>
Вітаю Вас на моїй другій web- сторінці!
</BODY>
</HTML>

11. Доповнити текст документу (між тегами <BODY>...</BODY>) наступним текстом:

Я знаю як оформляти:
Шрифти
Заголовки
Абзаци

12. Оформити три останні рядки як **список нумерований**. Для цього використати наступну конструкцію тегів:

<OL>
<LI> Шрифти </LI>
<LI> Заголовки </LI>
<LI> Абзаци </LI>
</OL>

13. Поміняти оформлення списку на **список маркірований**. Використати теги <UL>, </UL>.

14. Створити "змішаний" список:

Я знаю як оформляти:
1. **Шрифти**
- **Розмір;**
- **Колір;**
- **Гарнітуру;**
- **Індекси.**

2. **Заголовки**
- **Від 1-го до 6-го рівня.**

3. **Абзаци**
- **Вирівнювання;**
- **Розрив рядка всередині абзацу;**
- **З використанням переформатування**.

15. Оформити протокол до лабораторної роботи.

ЛАБОРАТОРНА РОБОТА № 2

Тема роботи: Створення і форматування таблиць. Створення закладок і гіперпосилань.
Мета роботи:
●Навчитися використати таблиці для оформлення веб-сторінок.
●Навчитися створювати закладки і гіперпосилання.

Зміст протоколу до лабораторної роботи:
●Титульний аркуш (приклад у додатку 1);
●Мета роботи;
●Тексти HTML- документів;
●Копії екранів з відображенням HTML- документів у браузері;
●Висновки по роботі.

Основні теоретичні відомості:

У таблицях 2.1 та 2.2 представлені основні теги для вставки та оформлення гіперпосилань.

Таблиця 2.1 – Вставка посилань

Призначення	Тег	Приклад
Посилання на іншу сторінку	<A HREF=” сторінка”> текст </A>	<A HREF=”str2.htm”> посилання1</A>
Посилання на закладку в іншому документі	<A HREF=” сторінка # ім'я закладки ”> текст</A>	<A HREF=” index.htm #met1”> На головну сторінку </A>”
Посилання на закладку в тому ж документі	<A HREF=”# ім'я закладки ”> текст </A>	<A HREF=” #metka1”> посилання2</A>
Визначити закладку	<A NAME=” ім'я закладки ”>текст</A>	<A NAME=”metka1”></A>

Таблиця 2.2 – Оформлення кольором фону і тексту посилань

Призначення	Теги	Приклад
Фонова картинка	<BODY BACKGROUND=” файл малюнка”>	<BODY BACKGROUND =”grafica.gif” TEXT=”black” (чорний) LINK=”#FF0000” (червоний) VLINK=”#FFFF00” (жовтий) ALINK=”#FFFFFF” (білий) </BODY>
Колір фону	<BODY BGCOLOR=”#$$$$$$”>	
Колір тексту	<BODY TEXT=”#$$$$$$”>	
Колір посилання	<BODY LINK=”#$$$$$$”>	
Колір пройденого посилання	<BODY VLINK=”#$$$$$$”>	
Колір активного посилання	<BODY ALINK=”#$$$$$$”>	

Створення таблиць

Таблиця складається з рядків та стовбців комірок, які можуть мати текст та рисунки. Зазвичай, таблиці використовуються для упорядкування і представлення даних, але можливості таблиць цим не обмежуються. За допомогою таблиць зручно верстати макети сторінок, розташував необхідним чином фрагменти тексту та зображень.

Для додавання таблиці на веб-сторінку використовується тег-контейнер TABLE. Таблиця повинна містити хоча б один рядок і стовпчик.

Для додавання рядків використовуються теги <tr> та </tr>. Щоб розділити рядки на колонки використовуються теги <td> та </td>.

У таблиці 2.3 вказані основні теги для створення та оформлення таблиць.

Таблиця 2.3 – Теги оформлення таблиці

<table>
<tr><td>Визначити таблицю</td><td><TABLE></TABLE></td><td rowspan="9">Приклад:
<TABLE border=”1” align
=«CENTER» width=«50%» >
<TR>
<TH >Товар</TH>
<TH>Ціна</TH>
</TR>
<TR>
<TD>Радиотелефон</TD>
<TD>2000 </TD>
</TR>
</TABLE>

<table><tr><th>Товар</th><th>Цена</th></tr><tr><td>Радиотелефон</td><td>2000</td></tr></table></td></tr>
<tr><td>Окантовка таблиці</td><td><TABLE BORDER =”?” </TABLE></td></tr>
<tr><td>Рядок таблиці</td><td><TR> </TR></td></tr>
<tr><td>Вирівнювання</td><td><TR ALIGN=left | right | center | middle | bottom ></td></tr>
<tr><td>Елемент таблиці</td><td><TD></TD></td></tr>
<tr><td>Вирівнювання по горизонталі</td><td><TD ALIGN=LEFT | RIGHT | CENTER></td></tr>
<tr><td>Вирівнювання по вертикалі</td><td><TD VALIGN = TOP|MIDDLE| BOTTOM></td></tr>
<tr><td>Установка ширини комірки (у пікселях або%)</td><td><TD WIDTH=«?»></td></tr>
<tr><td>Заливка кольором осередку</td><td><TD BGCOLOR = «# кольор»> </TD></td><td><TD BGCOLOR = «#FF0000»> </TD> червоний кольор</td></tr>
<tr><td>Заголовок стовпця або рядка</td><td><TH>текст </TH></td><td>Текст в комірці вирівнюється по центру, встановлюється жирний шрифт</td></tr>
</table>

Параметри таблиці

Для змінення виду і властивостей таблиці використовується безліч параметрів, які додаються у тег TABLE.

<table параметр1 =… параметр2 = …>

Опис параметрів таблиці та її властивості вказано в таблицях 2.4 – 2.6.

Таблиця 2.4 – Параметри таблиці та її властивості

Властивість	Значення	Опис	Приклад
align=	left, right, center	Вирівнювання таблиці	align=center
background=	URL	Фоновий рисунок	background=pic.gif
bgcolor=	#rrggbb	Колір фона таблиці	bgcolor=#FF9900
border=	n	Товщина рамки у пікселях	border=2
bordercolor=	#rrggbb	Колір рамки	bordercolor=#333333
bordercolordark=	#rrggbb	Тінь рамки	bordercolordark=#f0f0f0
cellpadding=	n	Відстань між коміркою та її змістом	cellpadding=7
cellspacing=	n	Дистанція між комірками	cellspacing=3
nowrap=		Забороняє переноси рядків у тексті	<table nowrap>
frame=	Void, below, above, rhs, lhs, hsides, vsides, box	Завдання типу рамки таблиці	frame=hsides
valign=	top, bottom	Вирівнювання по висоті	valign=top
width=	n, n%	Мінімальна ширина таблиці (в пікселях або %)	width=90%
height=	n, n%	Мінімальна висота таблиці (в пікселях або %)	height=18

Примітка:

1. Таблиця, якщо не вказано особо, завжди вирівнюється по лівому краю;
2. Параметр background, що відповідає за рисунок фону, своєрідно розуміється у різних браузерах. IE вставляє малюнок у всю таблицю, якщо таблиця по розміру більше фонового рисунку, він повторюється по горизонталі та вертикалі. Netscape додає фонове зображення у кожну комірку таблиці;
3. За замовчуванням, таблиця виводиться без рамки. Але Netscape добавляє тонку лінію між комірками. Щоб її не було, завжди вказуйте параметр border = 0;
4. Якщо ширина таблиці не вказана, вона підганяється під зміст комірок.

Таблиця 2.5 – Основні атрибути (перше з перелічених значень – значення за замовчуванням)

Атрибут	Можливі значення	Дія атрибуту	В яких тегах використовується
COLOR=	1)Текстова назва кольору: GRAY (сірий), AQUA (аквамарин), BLACK (чорний), BLUE (синій), FUCHSIA (яркий пурпурно-червоний), GREEN (зелений), LIME (зеленуватий), MAROON (темно-бордовий), NAVY (темно-синій), OLIVE (оливковий), PURPLE (пурпурний), RED (червоний), SILVER (срібний), TEAL, YELLOW (жовтий), WHITE (білий). Можливі також ті ж слова з приставками LIGHT та DARK, наприклад LIGHTGREEN (світло-зелений), DARKBLUE (темно-голубий). 2) Шістнадцятирічний код у системі RGB	Задає колір лінії та шрифту у тексті або таблиці	<HR>, <FONT>
BGCOLOR=		Задає колір фону	<TABLE>, <TR>, <TH>, <BODY>
BORDERCOLOR=		Задає колір зовнішнього контуру таблиці	<TABLE>
TEXT=		Задає колір шрифту у документі в цілому	<BODY>
LINK= VLINK= ALINK=		Кольори відповідно невідвідуваних, відвідуваних та активних посилань	<BODY>
BACKGROUND=	«URL» файлу із зображенням для фону	Створює фон-рисунок	
BORDER=	Ціле число без розмірності	Задає товщину окантовки для зображення або таблиці	<IMG>, <TABLE>
ALIGN=	LEFT, CENTER, RIGHT	Горизонтальне вирівнювання текстового фрагменту або таблиці в цілому	<P>, <H1>…<H6>, <TABLE>, <HR>, <TH>, <TD>
ALIGN=	BOTTOM, TOP	Розташування заголовку над чи під таблицею	<CAPTION>
VALIGN=	MIDDLE, BOTTOM, TOP	Вертикальне вирівнювання фрагменту	<TABLE>, <TH>, <TD>

WIDTH=	Ціле число без розмірності або зі знаком %	Довжина/висота фрагменту у пік селях або % від ширини/висоти вікна. Для всіх комірок, що знаходяться у рядку/стовбці використовується максимальне значення через дані в її/його комірках	<TABLE>, <HR>, <TH>, <TD>, <IMG>
HEIGHT=			

Таблиця 2.6 – Основні атрибути (перше з перелічених значень – значення за замовчуванням)

Атрибут	Можливі значення	Дія атрибуту	В яких тегах використовується
SIZE=	Ціле число без розмірності (за замовчуванням 1)	Товщина лінії, розмір шрифту	<HR>, <FONT>
TYPE=	1, A, a, I, l	Тип нумерації елементів упорядкованого списку	<OL>
START	Номер першого елементу у вибраному типі нумерації		

Приклад № 1 – створення простої таблиці (рисунок 2.1)

```
<TABLE width=”100%” cellpadding=”0” cellspacing=”0” border=”1”>
    <CAPTION align=”top”>Статистика відвідувань сайтів онлайн-тестування
    </CAPTION>
    <TR>
        <TH>URL</TH>
        <TH>Хости</TH>
        <TH>Хіти</TH>
    </TR>
    <TR>
        <TD>www.brainbench.com</TD>
        <TD>12136</TD>
        <TD>22178</TD>
    </TR>
    <TR>
        <TD>www.transmarket.net/education</TD>
        <TD>12</TD>
        <TD>27</TD>
    </TR>
</TABLE>
```

Статистика відвідувань сайтів онлайн-тестування

URL	Хости	Хіти
www.brainbench.com	12136	22178
www.transmarket.net/education	12	27

Рисунок 2.1 Статистика у вигляді таблиці

Приклад № 2 – Реалізація двохколонної модульної сітки (таблиця 2.7)

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

Таблиця 2.7 – Двохколонна модульна сітка (приклад № 2)

Логотип	Основний матеріал
Меню	

Приклад № 3 – Реалізація трьохколонної модульної сітки (таблиця 2.8)

```
<TABLE width=”100%” cellpadding=”10” cellspacing=”10” border=”0”>
    <TR align=”center”>
        <TD colspan=”3”>Заголовок сайту</TD>
    </TR>
    <TR>
        <TD width=”200px”>Меню</TD>
        <TD width=”*”>Основний матеріал</TD>
        <TD width=”200px”>Додаткові функції</TD>
    </TR>
    <TR align=”center”>
        <TD colspan=”3”>Кінець сайту</TD>
    </TR>
</TABLE>
```

Таблиця 2.8 – Трьохколонна модульна сітка (приклад № 3)

Заголовок сайту		
Меню	Основний матеріал	Додаткові функції
Кінець сайту		

Завдання:

1. Створити таблицю за приведеним зразком (рисунок 2.2), зберегти документ під ім'ям **tabl_name.html.** Над таблицею помістити заголовок **Таблиця №1:**

1	2	3
4	5	6
7	8	9

Рисунок 2.2

При відображенні таблиці у браузері повинні задовольнятися наступні умови:

●таблиця має вирівнюватися по центру і бути правильної (симетричною) форми;

●в кожну ячейку помістити відповідний текст - цифру від 1 до 9.

2. Модифікувати таблицю (рисунок 2.3):

●додати два рядка знизу, перший з текстом - цифра 0, другий залишити порожнім;

●"розфарбувати" комірки в різні кольори.

1	2	3
4	5	6
7	8	9
0		

Рисунок 2.3

3. Створити новий HTML- документ - **rasp_name.html** з розкладом занять (рисунок 2.3).

●Документ повинен починатися заголовком

Розклад занять гр. АЕ-181 на осінній семестр 2018 г.

●Перший рядок таблиці має бути оформлений як заголовки полів (з використанням тегів**<TH>**).

●Таблиця по ширині повинна займати повний розмір вікна (рисунок 2.4). Ширину окремих стовпців задати у відносних одиницях (у %), з тим, щоб при зміні ширини вікна пропорції таблиці зберігалися. Зробити границі таблиці подвійними.

День тижня	Час	Предмет	Викладач	Аудиторія
Понеділок	8:00-9:35	Вища математика (лек)	доц. Іванов О.О.	320
	9:50-11:25	Вища математика (пр)	проф. Петрова І.А.	408
	11:40-13:15	Фізика (лаб)	доц. Сідоров О.И.	301
Вівторок	8:00-9:35	Веб-програмування (лек)	ст.в. Дікусар К.В.	206
	09:50--11:25	Історія (лек)	ст.в. Попов М.О.	602
	11:40-13:15	Фізика (лек)	доц. Сідоров О.І.	507
...	...	...	...	...

Рисунок 2.4

●Проглянути створений документ у браузері при різних розмірах вікна і різних налаштуваннях розміру шрифту.

4. Створити групу web- сторінок, об'єднаних меню:

●На робочому диску створити папку **My_raspisanie** для розміщення файлів розкладу.

●Помістити розклад на кожен день тижня і таблицю з меню в окремі файли.

Імена файлів: **menu.html** – для головної сторінки, назви днів тижня – для інших (**pn.html, vt.html, …**). Усі документи розмістити у папки **My_raspisanie**.

Таблиця меню для головної сторінки має вигляд (рисунок 2.5):

Розклад

Понеділок	Вівторок	Середа	Четверг	П’ятниця

Рисунок 2.5

●В таблиці меню створити гіперпосилання меню так, щоб виконувалися переходи на відповідний документ (відповідні дні тижня).

●У кінці кожного файлу з розкладом на день організувати гіперпосилання для повернення в головний документ з меню. Перевірити правильність виконання переходів по гіперпосиланням.

●Оформити фон кожного дня тижня власним кольором, співпадаючим з кольором елементу таблиці меню.

5. Оформити протокол до лабораторної роботи.

ЛАБОРАТОРНА РОБОТА № 3

Тема роботи: Створення фреймів. Вставка малюнків, аудіо і відео.
Мета роботи:

- Навчитися створювати фрейми і розділяти екран на області.
- Навчитися вставляти малюнки, аудіо- і видеофайли.

Зміст протоколу до лабораторної роботи:

- Титульний аркуш (приклад в додатку 1);
- Мета роботи;
- Тексти HTML-документів;
- Копії екранів з відображенням HTML- документів у браузері;
- Висновки по роботі.
- Протокол буде ухвалено тільки після демонстрації викладачу виконаного завдання.

Завдання:

1. Створити презентаційну сторінку групи або виконавця, розділивши вікно на фрейми (області) такого виду (таблиця 3.1):

Таблиця 3.1 – Орієнтований вигляд сторінки

Назва гурта (виконавця) Фото Біографія Фотогалерея Пісні Кліпи	Виведення інформації

2. У верхньому лівому фреймі вказати назву вибраної групи або виконавця. Назва групи або виконавця має бути великою і не чорного кольору.
3. У середній фрейм вставити зображення (фото гурту або виконавця) по центру.
4. У нижньому лівому фреймі розділи " Біографія", " Фотогалерея", " Пісні", " Кліпи" мають бути гіперпосиланнями з переходом на вікно "Виведення інформації".
5. За бажанням ви можете розмістити фрейми і розділи на власний розсуд.
6. У розділі " Фотогалерея" має бути не менше трьох фото.
7. У розділі " Пісні" повинно бути не менше 3 пісень.
8. У розділі " Кліпи" має бути хоча б одне відео.
9. Аудіо і відео повинне виводиться із смугою прокрутки.
10. При оформленні розділу " Біографія" використати навички оформлення тексту для кращої читаності.
11. Задній фон сторінки не має бути білим.
12. Завдання можна виконувати по 2 людини в групі, оформляє протокол один з вказівкою усіх прізвищ на титульному аркуші.
13. Оформити протокол до лабораторної роботи, показати викладачеві результат на ПК, відповісти на запитання.

Основні теоретичні відомості:

Створення фреймів

Фрейми ділять вікно браузеру на декілька областей, кожна з яких функціонує незалежно від інших і може відображати окремий HTML- документ. Для того, щоб сторінка могла містити фрейми, тег <BODY> замінюється на тег <FRAMESET>. Параметри тега <FRAMESET> приведені в таблиці 3.2.

Таблиця 3.2 – Параметри тега <FRAMESET>

Параметр	**Значення**
rows	Визначає розташування горизонтальних фреймів. Це розділений комами список пікселів, відсотків і відносних довжин. За замовчуванням використовується 100%, що означає один рядок. Наприклад, rows=”20%, 80%”
cols	Визначає розташування вертикальних фреймів. Це розділений комами список пікселів, відсотків і відносних довжин. За замовчуванням використовується 100%, що означає один рядок. Наприклад, cols=”*, *”

Допускається використання вкладених один в одного тегів <FRAMESET> для створення складної структури фреймів.

Після завдання структури фреймів, кожен фрейм описується за допомогою тега <FRAME>. Параметри тега <FRAME> приведенs в таблиці 3.3.

Таблиця 3.3 – Параметри тега <FRAME>

Параметр	**Значення**
name	Ім'я фрейма
marginwidth	Горизонтальний відступ праворуч і ліворуч (від 1 до 6) між фреймом і межею
marginheight	Вертикальний відступ згори і знизу (від 1 до 6) між фреймом і його межею
src	Шлях до гіпертекстового документа для фрейма. src= “filename"|_blank|_self|_parent|_top
scrolling	Прокрутка фрейма. scrolling=”YES|NO|AUTO”
noresize	Фіксований розмір фрейма
frameborder	Задає ширину межі фрейма
bordercolor	Задає колір межі фрейма

Приклад HTML-кода (рисунок 3.1), що реалізовує фрейми і браузеру, що розділяє вікно, на 3 області:

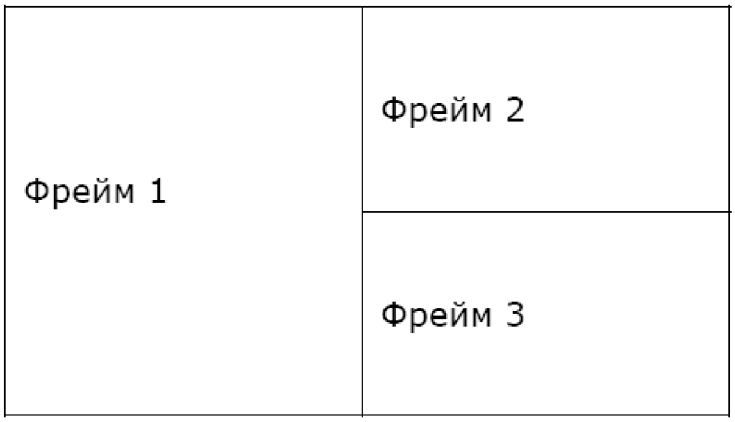


Рисунок 3.1

```
<html>
  <head><title>Сторінка з фреймами</title></head>
        <frameset cols="50%, 50%">
              <frame name="left" src="left.html">
        <frameset rows="100px, *">
              <frame name="top" src="top.html">
              <frame name="bottom" src="bottom.html">
        </frameset>
</frameset>
```

</html>

Приведемо ще один приклад, що складається з трьох html-документів (таблиця 3.4). Виведіть самостійно цей приклад у браузері, використовуючи дані з таблиці і переключаючись між коментаріями. Чим перший варіант відрізняється від другого?

Таблиця 3.4

frames.html	main.html	menu.html
<html> <!-- <frameset cols = "30%,70%" border = 10> --> <frameset rows = "30%,70%" border = 10> <frame src = "menu.html" name = "Menu"> <frame src = "main.html" name = "Main"> </frameset> </html>	html> <head> <title>Main page</title> </head> <body> Second frame </body> </html>	html> <head> <title>Menu</title> </head> <body> First frame </body> </html>

Використовуючи атрибут **target = "…"** можна зробити у фреймах гіперпосилання, по натисненню на які в ці ж або інші фрейми станеться завантаження інших сторінок.

Гіперпосилання

Основна потужність HTML походить з його можливості зв'язувати документи між собою за допомогою гіпертекстових посилань. Приклад гіпертекстового посилання:

<A HREF="news.html">Новини</A>

Цей вираз робить слово " Новини" гіпертекстовим посиланням на документ news.html, який знаходиться в тій же папці, що і основний документ. Щоб зробити посилання на документ, розташований в іншій папці, необхідно вказати *відносний шлях* від поточного документу до документу, на який робиться посилання. Наприклад, посилання на файл arhiv.html розташована в папці NewsArhiv виглядатиме так:

<A HREF="NewsArhiv/news.html"> Архів новин </A>

Для посилання на документ, розташований на іншому сайті необхідно вказувати адресу цього сайту і повний шлях до документу відносно папки сайту. Наприклад:

<A HREF="http://www.transmarket.net/education/lib/books.txt"> список книг з дизайну і програмування </A>

Посилання також можуть бути використані для переходу до певних частин документу. Припустимо, ви хочете зробити посилання в документі news.html на розділі «Анонс випуску», розташований нижче. Для цього необхідно:

1. Відмітити місце в документі, на яке вестиме посилання, таким чином:

**<A NAME = "point1"> Анонс випуску **

2. Створити посилання на помічену область:

<A HREF = "#point1"> Проглянути анонс </A>

Тут point1 – ім'я поміченої області, яке задається розробником.

Створення посилання всередину іншого документу виконується аналогічно, за винятком того, що в посиланні окрім імені поміченої області вказується ім'я іншого документу. Наприклад:

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

За умовчанням завантажуваний по посиланню документ відображається в тому ж вікні, що і попередній. Для того, щоб новий документ завантажувався в новому вікні, використовується атрибут target=”_blank”. Наприклад:

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

Для вставки графічного посилання замість текстової інформації в тег <A> поміщається графічний об'єкт.

Представлення графічної інформації

Для розміщення графічних елементів в HTML документах використовуються графічні файли 3 типів: *.JPG, *.GIF або *.PNG.

Розміщення графічного елементу в HTML документі здійснюється за допомогою тега <IMG>. Наприклад:

<IMG src=”file_name”>

де file_name це ім'я графічного файлу. У таблиці 3.5 перераховані додаткові параметри тега <IMG>.

Таблиця 3.5 – Основні параметри тега <IMG>

Параметр	**Значення**
align	Визначає положення об'єкту відносно навколишнього тексту. align=”top|middle|bottom|left|right”
alt	Задає альтернативний текст, що виводиться замість графічного об'єкту, якщо той не може бути відображений.
border	Задає ширину рамки навколо елементу в пікселях
height	Задає висоту елементу в пікселях
width	Задає ширину елементу в пікселях
usemap	Задає розташування і ім'я карти зображень

Карти зображень

Для розподілу посилань по картинці, наприклад, для створення графічного меню з однієї великої картинки, використовуються карти зображень. Для застосування карти зображень до графічного елементу необхідно:

• Вказати ім'я карти зображень для графічного елементу. Наприклад:

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

• Створити карту зображень з ім'ям map _ name і помістити її у файл з ім'ям file _ name. Якщо URL не вказаний, то пошук карти зображень map _ name ведеться в поточному документі. Код карти зображень записується в наступному виді:

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

Тут тег <AREA> визначає область на картинці. Додаткові параметри тега <AREA> приведені в таблиці 3.6.

Таблиця 3.6 – Основні параметри тега <AREA>

Параметр	**Значення**
shape	Визначає форму області. Можна задати одну з наступних областей: • default - стандартна форма • rect – прямокутник • circle –круг • poly - багатокутник довільної форми
coords	Задає координати області в пікселях. Круг має три координати, прямокутник – чотири, для багатокутника задаються координати кожного його кута.
href	Задає посилання для виділеної області
noref	Вказує, що в цій області картинки відсутнє посилання

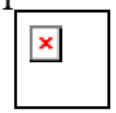
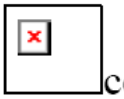
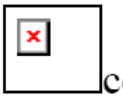
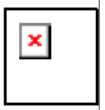
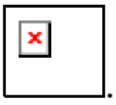
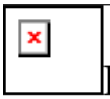
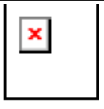
Наприклад, карта зображень, яка задає 2 активні області на картинці матиме наступний вигляд:

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

Вирівнювання зображень

Для зображень можна вказувати їх положення відносно тексту або інших зображень на веб-сторінці. Спосіб вирівнювання зображень задається параметром align тега IMG. У таблиці 3.7 розглянуті можливі значення цього параметру та результат його використання.

Таблиця 3.7 – Значення параметру align тега IMG

Код HTML	Опис	Приклад
<img src="hlpbell.gif" align=**texttop**>	Верхня межа зображення вирівнюється по самому високому текстовому елементу поточної строки	Lorem ipsum dolor sit amet,  consectetuer adipiscing elit...
<img src="hlpbell.gif" align=**top**>	Верхня межа зображення вирівнюється по самому високому елементу поточної строки	Lorem ipsum  dolor sit amet, consec-tetuer adipiscing elit...
<img src="hlpbell.gif" align=**middle**>	Вирівнювання середини зображення по базовій лінії поточної строки	Lorem ipsum dolor sit amet,  consec-
<img src="hlpbell.gif" align=**absmiddle**>	Вирівнювання середини зображення по середині поточної строки	Lorem ipsum dolor sit amet,  consec-tetuer adipiscing elit...
<img src="hlpbell.gif" align=**baseline**>	Вирівнювання зображення по базовій лінії поточної строки	Lorem ipsum dolor sit amet, consectetuer  adipiscing elit...
<img src="hlpbell.gif" align=**bottom**>	Вирівнювання нижньої межі зображення по навколишньому тексту	Lorem ipsum dolor sit amet, consectetuer adipiscing elit.  ..
<img src="hlpbell.gif" align=**left**>	Вирівнює зображення по лівому краю вікна	 Lorem ipsum do-lor sit amet, consec-tetuer adipiscing elit...
<img src="hlpbell.gif" align=**right**>	Вирівнює зображення по правому краю вікна	Lorem ipsum  dolor sit amet, consectetuer adipiscing elit...

Найбільш популярні параметри – left та right, що створюють обтікання навколо тексту. Щоб текст не прилягав щільно до рисунка, рекомендується у тегі IMG добавити параметр hspace та vspace, що задають відстань до тексту у пікселях.

ЛАБОРАТОРНА РОБОТА № 3А

Тема роботи: Каскадні таблиці стилів.

Мета роботи:
- Освоїти методи створення каскадних таблиць стилів для дизайну веб-сторінок;
- Розібратися в кожному наведеному прикладі;
- Виконати індивідуальне завдання за варіантом.

Рекомендоване обладнання і ПЗ:
- Комп'ютер з встановленим браузером Chrome або Opera будь-якої версії ;
- Текстовий редактор NotePad ++, або SublimeText, або будь-який інший, з можливістю підсвічування js і html синтаксису;
- Папка з файлами для лабораторної роботи.

Зміст протоколу до лабораторної роботи:
- Титульний аркуш;
- Мета роботи;
- Тексти CSS-, HTML-документів;
- Копії екранів з відображенням HTML-документів в браузері;
- Висновки по роботі.

Основні теоретичні відомості:

CSS (Cascading Style Sheets) – Каскадні таблиці стилів - це звід стильових описів, тих чи інших HTML тегів (далі елементів HTML), який може бути застосований як до окремого тегу – елементу, так і одночасно до всіх ідентичним елементів на всіх сторінках сайту. CSS по суті свого роду доповнення до HTML, яке значно розширює його можливості.

З огляду на те, що CSS дозволяє виносити повторювані стильові опису одних і тих же елементів в один файл відбувається значна "розвантаження" документів HTML, а це економія обсягу, трафіку, часу, грошей. HTML код стає легким, зручним для читання і редакції.

Впровадження CSS в HTML документ

Як впровадити CSS в документ HTML, тобто зв'язати стильовий опис елемента безпосередньо з самим елементом, яким-небудь HTML тегом?

Здійснити це завдання можна трьома способами:

1. Написати стильовий опис безпосередньо в самому елементі. Такий спосіб хорош лише в тому випадку, якщо такий елемент один єдиний в HTML документі, який потребує окремого стильового опису.

2. Написати стильовий опис для всіх ідентичних елементів HTML документа. Такий спосіб виправдовує себе, якщо стиль сторінки принципово відрізняється від загального дизайну сайту (групи взаємопов'язаних сторінок).

3. Винести стильовий опис елементів HTML в окремий файл CSS. Це дозволить управляти дизайном всього сайту цілком, кожною сторінкою сайту, в якій вказано звернення до CSS файлу. Цей спосіб є найбільш ефективним використанням таблиці каскадних стилів.

Давайте більш детально розглянемо кожен варіант, а заодно познайомимося з правилами синтаксису написання CSS.

Атрибут style

Практично кожен HTML тег має атрибут style, який говорить про те, що до цього тегу застосовується якись стильовий опис.

Пишеться так:

<P style = ""> це параграф з індивідуальним стилем </ p>

Все що буде написано між лапками атрибута style і буде стильовим описом для даного елемента, в даному випадку елемента <p>. Наприклад:

<P style = "color: # ff0000; font-size: 12px"> це параграф з індивідуальним стилем </ p>

В даному випадку ми вказали, що цей параграф повинен відображатися червоним кольором і мати розмір шрифту в 12 пікселів.

За таким же принципом можна вказати індивідуальний стиль практично для кожного HTML елемента.

Приклад (example1):

<Html>
<Head>
<Title> Атрибут style </ title>
</ Head>
<Body style = "background-color: # c5ffa0">
<H1 style = "color: # 0000ff; font-size: 18px"> Все про слонів </ h1>
<P style = "color: # ff0000; font-size: 14px"> На цьому сайті Ви знайдете будь-яку інформацію про слонів. </ P>
<H2 style = "color: # 0000ff; font-size: 16px"> Купити слона </ h2>
<P style = "color: # ff0000; font-size: 14px"> У нас Ви можете за вигідними цінами придбати кращих слонів !! </ p>
<H2 style = "color: # 0000ff; font-size: 16px"> Взяти слона на прокат </ h2>
<P style = "color: # ff0000; font-size: 14px"> Тільки у нас Ви можете взяти будь-яких слонів на прокат !! </ p>
</ Body>
</ Html>

Але ще раз повторюся – такий спосіб впровадження CSS хорош лише в тому випадку, якщо потрібно задати певний стиль малому числу HTML елементів.

Тег <style>

Для того, що б описати необхідні елементи одночасно на всій сторінці в заголовок HTML документа впроваджують тег <style> </ style> (не плутайте з однойменною атрибутом), в якому і відбувається опис потрібних нам елементів.

Погляньте на приклад, нижче до нього будуть коментарі (example2).

<Html>
<Head>
<Title> Тег style </ title>
<Style type = "text / css">
body {background-color: # c5ffa0}
h1 {color: # 0000ff; font-size: 18px}
h2 {color: # 0000ff; font-size: 16px}
p {color: # ff0000; font-size: 14px}
</ Style>
</ Head>
<Body>
<H1> Все про слонів </ h1>
<P> На цьому сайті Ви знайдете будь-яку інформацію про слонів. </ P>

<H2> Купити слона </ h2>
<P> У нас Ви можете за вигідними цінами придбати кращих слонів !! </ p>
<H2> Взяти слона на прокат </ h2>
<P> Тільки у нас Ви можете взяти будь-яких слонів на прокат !! </ p>
</ Body>
</ Html>

Як видно з прикладу, ми домоглися точно такого ж результату, що і в першому випадку, тільки тепер ми не прописуємо кожному елементу стиль індивідуально, а винесли його в "голову" документа тим самим вказавши, що всі заголовки <h1>, <h2> – будуть синіми, а параграфи <p> – червоними. Уявіть, як ми полегшили б собі роботу, будь на сторінці сотня таких параграфів і штук п'ятнадцять заголовків, так і сам документ став менше важити за рахунок "видалення" всіх повторюваних стильових описів для кожного окремо взятого елемента.

Тепер обіцяні коментарі:

Тег <style> прийнято впроваджувати в заголовок HTML документа між тегами <head> </ head>.

Атрибут тега <style> type - повідомляє браузеру, який синтаксис використовувати для правильної інтерпретації стилів. Для правильної інтерпретації браузерами CSS значення type (MIME тип даних) повинна дорівнювати text / css.

Усередині тега <style> </ style> йде безпосереднє оголошення стилів тих чи інших HTML елементів відповідно до наступного синтаксису:

Рисунок 3А.1

Якщо в блоці оголошення стилів вказується кілька властивостей елемента, то вони між собою розділяються крапкою з комою.

CSS в окремому зовнішньому файлі

Отже, підійшли ми до головного, гідності CSS, а саме можливості виносити всі відомості, що стосуються дизайну сайту, в окремий зовнішній файл. Відкриваємо блокнот (або інший редактор) і пишемо в ньому наступний текст (example3):

body {background-color: # c5ffa0}
a {color: # 000060; font-weight: bold;}
a: hover {color: # ff0000; font-weight: bold; text-decoration: none}
h1 {color: # 0000ff; font-size: 18px}
h2 {color: # ff00ff; font-size: 16px}
p {color: # 600000; font-size: 14px}

Далі зберігаємо цей невеликий файлик з розширенням * .css (назвемо його style3.css). Усе! файл зі стильовим описом створений! Тепер залишилося зовсім трохи, а саме змусити потрібні сторінки нашого сайту черпати інформацію з цього файлу. Робиться це за допомогою тега <link> (зв'язок). Тег <link> багатоцільовий і служить для "зв'язування" HTML документа з додатковими зовнішніми файлами, що забезпечують його належну роботу. Тег <link> є свого роду посиланням, тільки призначеної не для користувачів, а для програм оглядачів (браузерів).

Так як <link> несе в собі винятково службову інформацію, він розташовується в заголовку HTML документа між тегами <head> </ head> і не виводиться браузерами на екран.

Тег <link> має атрибути:

href - Шлях до файлу.

rel - Визначає відносини між поточним документом і файлом, на який робиться посилання.

shortcut icon - Визначає, що підключається файл є іконкою.

stylesheet - Визначає, що підключається файл містить таблицю стилів.

application / rss + xml - Файл в форматі XML для опису стрічки новин.

type - MIME тип даних підключається файлу.

Так як ми підключаємо в якості зовнішнього файлу каскадну таблицю стилів, то наша службова посилання набуває такого вигляду:

<Link rel = "stylesheet" href = "style3.css" type = "text / css">

Атрибуту rel присвоюємо значення stylesheet, так як підключаємо в якості зовнішнього файлу каскадну таблицю стилів, вказуємо шлях до файлу css (в цьому прикладі файл називається style3.css і лежить поруч з документом HTML, в якому прописується дане посилання). Так само вказуємо, що даний файл текстовий і містить в собі стильове опис type = "text / css". Тепер вставляємо цей рядок в заголовки сторінок нашого сайту і насолоджуємося результатом.

Приклад (example3):

файл style3.css

```
body {background-color: # c5ffa0}
a {color: # 000060; font-weight: bold;}
a: hover {color: # ff0000; font-weight: bold; text-decoration: none}
h1 {color: # 0000ff; font-size: 18px}
h2 {color: # ff00ff; font-size: 16px}
p {color: # 600000; font-size: 14px}
```

файл example3.html

```
<Html>
<Head>
<Title> каскадна таблиця стилів </ title>
<Link rel = "stylesheet" href = "style3.css" type = "text / css">
</ Head>
<Body>
<H2> Меню: </ h2>
<a href="example3.html"> Все про слонів. </a>
<a href="elephant3_1.html"> Купити слона. </a>
<a href="elephant3_2.html"> Взяти слона на прокат. </a>
<Hr>
<H1> Все про слонів </ h1>
<P> На цьому сайті Ви знайдете будь-яку інформацію про слонів. </ P>
</ Body>
</ Html>
```

файл elephant3_1.html

```
<Html>
<Head>
<Title> каскадна таблиця стилів </ title>
<Link rel = "stylesheet" href = "style3.css" type = "text / css">
</ Head>
<Body>
<H2> Меню: </ h2>
```

* Все про слонів. *
* Купити слона. *
* Взяти слона на прокат. *
<Hr>
<H1> Купити слона </ h1>
<P> У нас Ви можете за вигідними цінами придбати кращих слонів !! </ p>
</ Body>
</ Html>

файл elephant3_2.html
<Html>
<Head>
<Title> каскадна таблиця стилів </ title>
<Link rel = "stylesheet" href = "style3.css" type = "text / css">
</ Head>
<Body>
<H2> Меню: </ h2>
* Все про слонів. *
* Купити слона. *
* Взяти слона на прокат. *
<Hr>
<H1> Взяти слона на прокат </ h1>
<P> Тільки у нас Ви можете взяти будь-яких слонів на прокат !! </ p>
</ Body>
</ Html>

В наведеному вище прикладі, "сайт про слонів", на даний момент, є три сторінки, кожна з яких пов'язана з одним єдиним зовнішнім css файлом - style3.css. Таким чином, ми значно його "розвантажили" і зробили дизайн всього сайту "мобільним". Уявіть, скільки б кілобайт ми виграли, будь на цьому сайті сотня повноцінних сторінок !? А також, скільки б часу заощадили, якби нам знадобилося змінити що-небудь в його дизайні !?

Використовуйте атрибут style для будь-якого елемента, якщо цей елемент з відмінним від інших елементів стилем? один єдиний на всьому сайті.

Використовуйте тег <style> зі стильовим описом в тому випадку, якщо сторінка повинна мати індивідуальний дизайн в корені відмінний від інших сторінок сайту.

У більшості випадків розумно виносити каскадну таблицю стилів в окремий css файл.

Властивості тексту

Вирівнювання тексту

Якщо Ви пам'ятаєте, з курсу HTML, для того що б вирівняти текст, наприклад по центру екрана, ми застосовували до тегу містить в собі текст атрибут align (вирівнювання) і одне з його можливих значень center (по центру). Запис мала такий вигляд:

<P align = "center"> текст по центру </ p>

В CSS це завдання бере на себе властивість text-align, яке вирівнює текстовий зміст щодо елемента батька (наприклад, блоку div) або ж вікна браузера.

text-align (так само як і htmlловскій атрибут align) має такі значення:

left - Вирівняти текст по лівому краю елемента (за замовчуванням).

right - Вирівняти текст по правому краю.

center - Вирівняти текст по центру.

justify - Вирівняти текст по обох краях.

Тепер, для того, щоб вирівняти текст того ж параграфа по центру, слід писати так:
<P style = "text-align: center"> текст по центру </ p>

- це в цьому випадку, якщо ми, за допомогою атрибута style, впроваджуємо CSS безпосередньо в HTML тег. А ось в прикладі нижче використовується тег <style> в заголовку документа (example4):

```
<Html>
<Head>
<Title> Вирівнювання тексту </ title>
<Style type = "text / css">
h1 {text-align: center}
p {text-align: justify}
</ Style>
</ Head>
<Body>
<H1> Все про слонів </ h1>
<P> Слон - найбільша ссавець на нашій планеті! Найбільший слон з коли-небудь живуть на Землі був зареєстрований в Анголі в 1956 році. Цей самець важив близько 12 тон, а в висоту досягав 4,2 метра, що на метр вищий за середній Африканського слона. </ P>
<P> Слони є королівським символом Азіатської культури і відомі своєю відмінною пам'яттю і високим інтелектом. Аристотель одного разу сказав, що слон - "тварина, яке перевершує всіх інших в дотепності і інтелекті". </ P>
</ Body>
</ Html>
```

Оформлення тексту

Властивість text-decoration дозволяє декорувати текст, присвоївши йому одне або кілька значень з нижче представлених варіантів оформлення тексту.

Можливі значення:
blink - Текст буде блимати.
line-through - Робить текст перекресленим.
overline - Надчёрківаніе тексту.
underline - Підкреслення тексту.
none - Текст без оформлення.

Пишеться так:
* Посилання без підкреслення *

Приклад (example5):
файл style5.css

```
h1 {text-align: center; text-transform: capitalize}
h3 {text-align: left; text-decoration: underline; text-transform: uppercase}
a {text-decoration: underline}
a: hover {text-decoration: none}
p {text-align: justify; text-indent: 20px}
```

файл example5.html

```
<Html>
<Head>
<Title> Оформлення тексту </ title>
<Link rel = "stylesheet" href = "style5.css" type = "text / css">
</ Head>
```

<Body>
<H3> Меню: </ h3>
* Все про слонів.
 <! - фаил з попереднього прикладу ->*
* Купити слона. <! - фаил з попереднього прикладу ->*
<Hr>
<H1> Все про слонів </ h1>
<P> Слон - найбільша ... ... </ p>
<P> Слони є ... ... </ p>
</ Body>
</ Html>

Зверніть увагу на зовнішній файл CSS, в ньому ми "декорували" посилання елемент <a>, причому робили це двічі: перший раз а {text-decoration: underline} зробили її підкресленою, хоча можна було цього і не робити, так як тег <a> підкреслять за замовчуванням, а другий раз використовували так званий псевдоклас hover і заборонили підкреслення a: hover {text-decoration: none}

Даний псевдоклас вказує на те, що застосовувати до нього стильове опис варто лише в тому випадку, якщо користувач навів курсор на цей елемент. Так якщо в прикладі навести курсор на одну з посилань в меню? то підкреслення зникне, що створює певний динамічний ефект меню ставати "живим".

Відступ першого рядка

Властивість text-indent - задає відступ першого рядка в текстовому блоці з лівого боку, простіше кажучи робить "новий рядок".

Відстань від лівого краю вікна браузера або ж елемента батька (блоку, в який поміщений блок з текстом) може бути задано у відсотках від ширини вікна браузера або ж одиницях виміру, прийнятих в CSS.

У прикладі style5.css відстань відступу від лівого краю задається в пікселях (px).

Трансформація тексту

Властивість text-transform трансформує символи в зазначеному текстовому блоці, роблячи їх великими або прописними по одному з правил? в залежності від присудженого значення даній властивості. значення:

none - Текст відображається без будь-яких змін (за замовчуванням).
capitalize - Кожне слово в тексті відображається з заголовного символу.
lowercase - Все символи перетворюються в нижній регістр.
uppercase - Все символи перетворюються в великі букви.

Приклад (example6):
<Html>
<Head>
<Title> Трансформація тексту </ title>
</ Head>
<Body>
<P style = "text-transform: capitalize"> кафедра комп'ютерних систем </ p>
<P style = "text-transform: lowercase"> ІКС ОНПУ </ p>
<P style = "text-transform: uppercase"> АЕ / Ак-181 </ p>
</ Body>
</ Html>

Вертикальне вирівнювання

Вертикальне вирівнювання тексту в рядку встановлює властивість vertical-align. Можливі значення властивості vertical-align:

baseline - Вирівнює базову лінію елемента за базовою лінії батька.

bottom - вирівнює елемент по нижній частині рядка.

middle - вирівнює середину елемента за базовою лінії батька і додає половину висоти батьківського елементу.

sub - Нижній індекс (розмір шрифту не змінюється).

super - Верхній індекс (розмір шрифту не змінюється).

text-bottom - Нижня межа елемента вирівнюється по нижньому краю рядка.

text-top - Верхня межа елемента вирівнюється по верхньому краю рядка.

top - вирівнює елемент по верхній частині рядка.

Базова лінія - це лінія, на якій розташовуються "сидять" символи в текстовому рядку, Наприклад, буква "А" сидить прямо на цій лінії, а ось мала літера "у" сидить на ній же, але звісивши ноги.

Погляньте на малюнок з розміткою рядки:

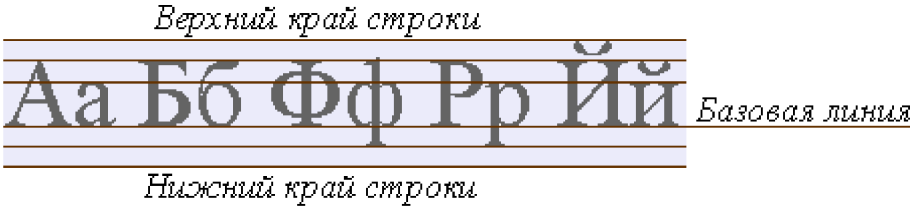


Рисунок ЗА.2

Так само вертикальне вирівнювання елемента щодо рядка може виражатися у відсотках, пікселях або будь-яких інших прийнятих в CSS одиницях виміру, причому ці одиниці можуть приймати як позитивні, так і негативні значення.

Приклад (example7):

```
<Html>
<Head>
<Title> Вертикальне вирівнювання тексту </ title>
</ Head>
<Body>
<Font size = "+ 3"> А і Б </ font>
<Span style = "vertical-align: + 5px"> сиділи на трубі </ span>
<Span style = "vertical-align: bottom"> А упало </ span>
<Span style = "vertical-align: top"> Б пропало .. </ span>
<Span style = "vertical-align: 50%"> що залишилося на трубі? </ Span>
<Hr>
формула води: Н <span style = "vertical-align: sub"> 2 </ span> О
<Hr>
<Span> н </ span>
<Span style = "vertical-align: -10px"> а </ span>
<Span style = "vertical-align: -20px"> і </ span>
<Span style = "vertical-align: -30px"> з </ span>
<Span style = "vertical-align: -40px"> до </ span>
<Span style = "vertical-align: -50px"> про </ span>
<Span style = "vertical-align: -60px"> з </ span>
<Span style = "vertical-align: -70px"> про </ span>
<Span style = "vertical-align: -80px"> до </ span>
```

</ Body>
</ Html>

Прогалини і перенесення рядка

Набраний текст, в будь-якому текстовому редакторі браузерами, за замовчуванням, виводиться на екран у вигляді суцільного тексту, де переноси рядків розставляються автоматично, а так само прибираються зайві (більше одного) прогалини між символами.

Властивість white-space імітує роботу тега <pre>, визначаючи показувати чи ні прогалини між символів, якщо таких більше ніж один, а так само дозволяє або забороняє перенесення рядка. Може мати наступні значення:

normal - текст виводиться як зазвичай (зайві прогалини прибираються), переноси рядків визначаються автоматично (за замовчуванням).

nowrap - забороняє автоматичне перенесення рядка.

pre - показує текст в тому вигляді в якому він був набраний. прогалини і переноси рядки не видаляються.

Приклад (example8):

<Html>
<Head>
<Title> Прогалини і перенесення рядка </ title>
</ Head>
<Body>
<P style = "white-space: pre">
Слон.
Далі туфельки слону.
Взяв він туфельку одну
І сказав: - Потрібні ширше,
І не дві, а всі чотири!
С. Я. Маршак.
</ P>
<Hr>
<P style = "white-space: nowrap">
Це довгий довжелезний текст, який навряд чи повністю поміститься в одному рядку, за замовчуванням в потрібному місці, браузер переніс би його на наступний рядок, проте ми примусово заборонили це робити, за допомогою значення nowrap властивості white-space. Так що тепер, по всій ймовірності, у вікні браузера з'явитися горизонтальна смуга прокрутки .. і навіщо, питається, ми це зробили?
</ P>
</ Body>
</ Html>

При використанні nowrap текст в потрібному місці можна переносити на наступний рядок використовуючи тег
.

Відстань між словами

Властивість word-spacing задає відстань між словами (групами Літери не розділеними пробілом) в рядку.

значення:

normal - Нормальне відстань (за замовчуванням).

px - Відстань задається в пікселях або будь-яких інших одиницях виміру, прийнятих в CSS.

Приклад (example9):

<Html>

<Head>
<Title> Відстань між словами </ title>
</ Head>
<Body>
<P align = "left" style = "word-spacing: 10px"> Відстань між словами дорівнює десяти пикселям </ p>
<P align = "left" style = "word-spacing: -10px"> Відстань між словами може мати від'ємне значення </ p>
</ Body>
</ Html>

Міжсимвольна відстань

А ось властивість letter-spacing визначає відстань між символами в тексті і так само як і може word-spacing бути задано такими значеннями:

normal - Нормальне відстань (за замовчуванням).

px - Відстань задається в пікселях або будь-яких інших одиницях виміру, прийнятих в CSS.

Приклад (example10):

<Html>
<Head>
<Title> Відстань між символами </ title>
</ Head>
<Body>
<P style = "letter-spacing: 5px"> Відстань між літерами дорівнює п'яти пікселям </ p>
<P style = "letter-spacing: -3px"> А тут літери, з за від'ємного значення, будуть насуватися один на одного </ p>
</ Body>
</ Html>

Інтерліньяж

Інтерліньяж - це відстань між рядками тексту.

Відстань між рядками тексту можна задати використовуючи властивість line-height, зробити це можна наступними способами:

normal - Норма (за замовчуванням).

% - Відсотки. за сто відсотків береться висота шрифту

0.5 - Множник. Може бути використано будь-яке число більше нуля. Так, наприклад множник 0.5 буде дорівнювати половинному міжрядкового відстані, а 2 - подвійному.

px - Пікселі та будь-які інші одиниці виміру, прийняті в CSS.

Приклад (example11):

<Html>
<Head>
<Title> Міжрядкові </ title>
</ Head>
<Body>
<Div style = "line-height: 150%">
*рядок перша
 рядок друга
 рядок третя
 рядок четверта
 рядок п'ята*
</ Div>
<Hr>
<Div style = "line-height: 0.5">
*рядок перша
 рядок друга
 рядок третя
 рядок четверта
 рядок п'ята*
</ Div>

<Hr>
<Div style = "line-height: 25px">
*рядок перша
 рядок друга
 рядок третя
 рядок четверта
 рядок п'ята*
</ Div>
</ Body>
</ Html>

Завдання на лабораторну роботу:

Зверстати веб-сайт, що складається з мінімум 5 сторінок, між якими можливий перехід по посиланнях.

Обов'язкове використання:

1. Стилю, вбудованого в сторінку 1;
2. Стилю, винесеного в окремий файл, для сторінки 2;
3. інтерліньяжу;
4. Міжсимвольного відстані;
5. Відстані між словами;
6. Прогалин і переносів рядка;
7. Вертикального вирівнювання;
8. Трансформації тексту;
9. Відступу першого рядка;
10. Оформлення тексту;
11. Стиль і розмір шрифту;
12. Колір фону і шрифту;
13. Вирівнювання тексту.

Завдання по варіантах на наступні тематики:

1. комп'ютери
2. навчання
3. одяг
4. туризм
5. погода
6. меблі
7. книги
8. музика
9. автомобілі
10. релігія
11. краса
12. історія
13. авіація / космонавтика
14. медицина
15. свята
16. тварини
17. казки
18. транспорт
19. фінанси
20. географія
21. їжа
22. архітектура

ЛАБОРАТОРНА РОБОТА № 4

Тема роботи: Методи підключення javascript до html-документу. Взаємодія з DOM.

Мета роботи:
●Освоїти методи підключення javascript-коду до html-документу;
●Ознайомитися з можливостями роботи з DOM за допомогою js;
●Ознайомитися з інструментами розробника, що поставляються разом з браузером.

Рекомендоване устаткування і ПЗ:
Комп'ютер зі встановленим браузером Chrome або Opera будь-якої версії;
Текстовий редактор NotePad++, або SublimeText, або будь-який інший, з можливістю підсвічування js і html синтаксису;
Архів з файлами лабораторної роботи.

Зміст протоколу до лабораторної роботи:
●Титульний аркуш (додаток 1);
●Мета роботи;
●Тексти JS-документів;
●Копії екранів з відображенням результату роботи у браузері;
●Висновки по роботі.

Основні теоретичні відомості:
В процесі "читання" браузером html-документу, при зустрічі тегу <script> – читання призупиняється, і в першу чергу виконується код, написаний усередині цього тегу.

Так, в наступному прикладі, буде показано початок сторінки, потім три рази виконається функція "alert", яка виводить віконце з інформацією, а тільки потім з'явиться інша частина сторінки.

```
<html>
<body>
   <h1> Рахуємо кроликів </h1>
   <script type="text/javascript">
      for(var i=1; i<=3; i++) {
         alert("З капелюха дістали "+i+" кролика!")
      }
   </script>
   <h1>... Порахували </h1>
</body>
</html>
```

Отже, перший спосіб виконати js-код – написати його усередині тега <script>.

Проте, записувати великі об'єми коду усередині html-документу, м'яко кажучи, незручно. Для вирішення подібної проблеми існує інший спосіб – винесення js-коду в окремий файл. Для цього, необхідно також використати тег <script>, присвоївши його атрибуту src шлях до виконуваного файлу. Наприклад:

```
<script src=”path/to/file.js”>
```

Шлях повинен вести до файлу що містить виконуваний код. Файл повинен мати розширення js. Код усередині файлу буде повністю виконаний у тому випадку, якщо він не

містить помилок. У разі, якщо в коді помилка є, він буде виконуватись до її появи – це наслідок інтерпретуємості мови.

Наприклад, є javascript-файл з наступним вмістом:

```
console.log('start execution')
console.log('I am counting')
for (var i = 0; i < someString.length; i ++){
console.log(i)
}
```

Якщо підключити його в html-документ, в консоль буде виведено два повідомлення: ‘start execution’, ‘I am counting’, а також повідомлення про помилку “myFile.js:3 Uncaught ReferenceError: someString is not defined”. Тіло циклу виконане не буде.

Розглянемо детальніше повідомлення про помилку. У загальному вигляді, ці повідомлення виглядають таким чином:

< ім'я файлу >:< номер рядка > < тип помилки > < текст помилки >,

де:

< ім'я файлу > – ім'я файлу, в якому сталася помилка. У нашому випадку – це myFile.js.

< номер рядка > – номер рядка, в якому сталася помилка. У нашому випадку – це третій рядок.

< тип помилки > – тип виниклої помилки. У нашому випадку, це Uncaught ReferenceError.

< текст помилки > – суть виниклої помилки. У наявному файлі немає нічого з ім'ям someString, що призводить до помилки “someString is not defined”.

Одне із завдань, яке вирішується за допомогою javascript є маніпулювання DOM (Document Object Model), іншими словами – зміни вже наявного html-документу – зміна атрибутів тегів, вставка нових тегів в документ, або ж їх видалення.

Для того, щоб виконати будь яку дію з DOM елементом (тобто html-тегом), необхідно якимось чином його “вибрати”.

Кожна веб-сторінка, яка завантажується у браузер, має свій власний об'єкт **document**. Для "вибору" тега можна скористатися одним з його методів, наприклад, такими як:

getElementById – повертає тег з певним значенням атрибуту “id”;
getElementsByClassName – повертає усі теги з певним значенням атрибуту “class”;
getElementsByTagName – повертає усі теги з певним ім'ям.

Наприклад, наступний код

```
var x = document.getElementsById("1");
```

збереже тег, у якого значення атрибуту id дорівнює одиниці в змінну x – як відомо в javascript в змінну можна скласти все що завгодно. В даному випадку після виконання коду типом змінної x буде “object”.

Повний список методів об'єкту **document** можна побачити на сайті:
https://developer.mozilla.org/ru/docs/Web/API/Document

Для того, щоб проставити DOM елементу значення будь якого атрибуту, можна скористатися методом самого елементу setAttribute.

Наприклад, наступний код

```
var x = document.getElementsById("1");
```

x.setAttribute(‘class’, ‘hidden’)

проставить вибраному елементу значення “hidden” атрибуту class.

Завдання на лабораторну роботу:

1. Розпакувати архів з файлами лабораторної роботи у будь-яку директорію. Ознайомитися з його вмістом. Відкрити у браузері файл lab.html.
2. У файлі counting.js за допомогою ключового слова var визначте змінну someString так, щоб тіло циклу виконалося 10 разів. За довідками звертайтеся на http://www.w3schools.com/jsref/jsref_obj_string.asp.
3. Відкрийте інструменти розробника вашого браузеру. Виберіть вкладку console. Переконайтеся, що зараз код працює правильно.
4. Відкрийте вкладку Elements інструментів розробника. За допомогою викладача, вивчіть вікно, що відкрилося.
5. Виберіть курсор. Клацніть курсором по тексту. Вивчіть властивості DOM елементу, що відкрилися.
6. У html-документі створіть тег <script>, усередині якого за допомогою методів document.getElementById, setAttribute проставте елементу, що містить текст клас “color-red”. Використайте дані в теоретичних відомостях як зразок.
7. Повторивши дії з пункту 5, подивіться, як змінилися властивості елементу.
8. Створіть файл в папці js, винесіть код з тегу в окремий файл, і підключіть його.
9. Відкрийте вкладку source. У лівому вікні, відкрийте вміст папки js, в списку, що з'явився, відкрийте файл rotate.js. Функція, яку ви бачите, містить декілька логічних помилок, яка заважає єхидному колобку зробити повне обертання при натисненні кнопки rotate.
10. Встановіть брейкпоинт на рядках 16 і 20, і дайте відповідь на наступні питання:

- Чи спрацьовує функція rotate при натисненні на кнопку (чи заходимо ми в неї?)
- Чи виконує тіло циклу потрібну кількість разів?
- Чи вибираємо ми на кожній ітерації циклу правильний елемент з масиву назв класів?

11. Відредагуйте файл так, щоб відповідь на кожне питання була ствердною.
12. Оформіть протокол про виконану роботу.

ЛАБОРАТОРНА РОБОТА №5

Тема роботи: Функція і обробка події.
Мета роботи: навчитися використовувати функції та події.

Зміст протоколу до лабораторної роботи:
- Титульний аркуш (додаток 1);
- Мета роботи;
- Тексти JS-документів;
- Копії екранів з відображенням результату роботи у браузері;
- Висновки по роботі.

Основні теоретичні відомості:

Основним елементом мови JavaScript є функція. Опис функції має вигляд

function F (V) {S},

де F – ідентифікатор функції, який задає ім'я, по якому можна звертатися до функції;
V – список параметрів функції, що розділяються комами;
S – тіло функції, в ньому задаються дії, які треба виконати, щоб отримати результат. Необов'язковий оператор return визначає повернене функцією значення. Зазвичай усі визначення і функції задаються в розділі <head> документа. Це забезпечує інтерпретацію і збереження в пам'яті усіх функцій при завантаженні документу у браузер.

Приклад 1. Знаходження площі трикутника.

У попередніх прикладах користувачеві не надавалася можливість вводити значення, і залежно від них отримувати результат. Інтерактивні документи можна створювати, використовуючи форми. Припустимо, що ми хочемо створити форму, в якій поля «Основа» і «Висота» служать для введення відповідних значень. Крім того, у формі створимо кнопку «Вичислити». При клацанні мишею по цій кнопці ми хочемо отримати значення площі трикутника. Дія користувача (наприклад, клацання кнопкою миші) викликає подію. Події в основному пов'язані з діями, вироблюваними користувачем з елементами форм HTML. Зазвичай перехоплення і обробка події задається в параметрах елементів форм. Ім'я параметра обробки події розпочинається з приставки on, за якою йде ім'я самої події. Наприклад, параметр обробки події click буде мати вигляд onclick.

Лістинг 1. Реакція на подію Click.

```
<HTML>
<HEAD>
<title> Обробка значень з форми </title>
<script language="JavaScript">
<!--//
function care (a, h)
{
var s=(a*h)/2;
document.write ("Площа прямокутного трикутника дорівнює ",s);
return s
}
//-->
</script>
```

```
</HEAD>
<BODY>
<P> Приклад сценарію зі значеннями з форми </P>
<FORM name="form1">
Основа: <input type="text" size=5 name="st1"><hr>
Висота: <input type="text" size=5 name="st2"><hr>
<input type="button" value= Вичислити
onClick="care(document.form1.st1.value, document.form1.st2.value)"> /* По кліку миші на кнопці у функцію care передаються два параметри – вміст полів введення */
</FORM>
</BODY>
</HTML>
```

При інтерпретації HTML-сторінки браузером створюються об'єкти JavaScript. Взаємозв'язок об'єктів між собою представляє ієрархічну структуру. На самому верхньому рівні ієрархії знаходиться об'єкт windows, що представляє вікно браузеру. Об'єкт windows є предком або "батьком" усіх інших об'єктів. Кожна сторінка окрім об'єкту windows має об'єкт document. Властивості об'єкту document визначаються вмістом самого документу: колір фону, колір шрифту і т. д. Для набуття значення основи трикутника, введеного в першому полі форми, має бути виконана конструкція

```
document.forml.st1.value
```

тобто, кажучи українською мовою (при цьому читаємо з кінця), використовуємо дані value з поля введення з ім'ям st1, що знаходиться на формі form1 об'єкту document.

Приклад 2. Обчислення площі квадрата.

Напишемо сценарій, що визначає площу квадрата по заданій стороні. Площа повинна обчислюватися у той момент, коли змінилося значення його сторони. Нехай форма містить два текстових поля: одне – для довжини сторони квадрата, інше – для вичисленої площі. Кнопка «Оновити» очищає поля форми. Площа квадрата обчислюється при виникненні події change, яка відбувається у той момент, коли значення елементу форми з ім'ям num1 змінилося, і елемент втратив фокус. HTML-код представлений в лістингу 2.

Лістинг 2. Реакція на подію Change

```
<HTML>
<HEAD>
<title> Обробка події Change – зміна значення елементу </title>
<script>
function srec(obj)
{obj.res.value=obj.num1.value* obj.num1.value}
</script>
</HEAD>
<BODY>
<P> Обчислення площі квадрата </P>
<FORM name="form1">
Сторона: <input type="text" size=7 name="num1"
onChange="srec(form1)">
<hr>
Площа: <input type="text" size=7 name="res"><hr>
```

```
<input type="reset" value= Оновити >
</FORM>
</BODY>
</HTML>
```

Подія Focus виникає в мить, коли користувач переходить до елементу форми за допомогою клавіши <Tab> чи клацання миші.

Подія "втрата фокусу" (Blur) відбувається у той момент, коли елемент форми втрачає фокус.

Подія select викликається вибором частини або усього тексту в текстовому полі. Наприклад, клацнувши двічі мишкою по полю, ми виділимо поле, настане подія select, обробка якого приведе до обчислення необхідного значення.

У мові JavaScript визначені деякі стандартні об'єкти і функції, користуватися якими можна без попереднього опису. Одним із стандартних об'єктів є об'єкт Math. У властивостях згаданого об'єкту зберігаються основні математичні константи, а його методи можна використати для виклику основних математичних функцій.

Вираз y=log x запишеться y=Math.log(x).

Завдання:

1. Перевірити приклади з лабораторної роботи.
2. На площині задані координати трьох точок. Створіть окремий файл logic.js. Підключіть його до документу. Помістіть в нього функцію, яка обчислює площу трикутника (використати подію Focus).
3. Напишіть функцію, яка для точки, заданої координатами на площині, визначає відстань до початку координат (використати подію Select).
4. Напишіть функцію, яка міняє місцями значення двох введених змінних (використати подію Blur).

ЛАБОРАТОРНА РОБОТА № 6

Тема роботи: Дата і час в JavaScript.

Мета роботи: навчитися працювати з об'єктом Date.

Зміст протоколу до лабораторної роботи:
- Титульний аркуш (приклад в лабораторній роботі № 1);
- Мета роботи;
- Тексти JS-документів;
- Копії екранів з відображенням результату роботи у браузері;
- Висновки по роботі.

Основні теоретичні відомості

При роботі з JavaScript часто виникає потреба вивести на екран поточну дату і час. В основному, це використовують для довідки для клієнта або ж для запису в яку-небудь змінну. Для виведення повної інформації про дату і час використовується функція (об'єкт) **date()**.

Об'єкт Date призначений для маніпуляцій з датами і часом. Його примітивним значенням є число, рівне кількості мілісекунд відносно базового часу, рівного півночі 1 січня 1970 р. по Гринвіцькому меридіану (UTC, Universal Coordinated Time). Якщо це значення рівне NaN, то воно вважається невизначеним.

Створення

Цей об'єкт створюється таким чином:

```
var date = new Date();
```

Тепер змінна date – це об'єкт з датою, який зберігає в собі теперішній момент часу (секунду, хвилину, годину і так далі). За допомогою спеціальних функцій можливо отримати потрібні характеристики часу, наприклад, поточна година, поточний день або поточний місяць.

Для того, щоб зручніше було виконувати лабораторну роботу, рекомендується використати базову операцію **alert** у форматі «alert(повідомлення)». Вікно повідомлення, яке виводиться, є модальним вікном. Слово "модальне" означає, що відвідувач не може взаємодіяти із сторінкою, натискати інші кнопки і тому подібне, поки не розбереться з вікном. В даному випадку поки не натисне на "OK". Наприклад:

```
var date = new Date();
alert( date );
```

new Date (year, month, date, hours, minutes, seconds, ms)

Дату можна створити, використовуючи компоненти в місцевій часовій зоні. Для цього формату обов'язкові тільки перші два аргументи. Відсутні параметри, починаючи з hours вважаються рівними нулю, а date – одиниці.

Варто відмітити, що рік year має бути з 4 цифр, відлік місяців month розпочинається з нуля. Наприклад:

```
new Date(2011, 0, 1, 0, 0, 0, 0); // // 1 січня 2011, 00:00:00
new Date(2011, 0, 1); // те ж саме, годинник/секунди за умовчанням рівний 0
```

Дата задана з точністю до мілісекунд:

```
var date = new Date(2011, 0, 1, 2, 3, 4, 567);
alert( date ); // 1.01.2011, 02:03:04.567
```

Отримання компонентів дати

Для доступу до компонентів дати-часу об'єкту Date використовуються наступні методи:

- getFullYear() – отримати рік (из 4 цифр);
- getMonth() – отримати місяць, от 0 до 11;
- getDate() – отримати число місяця, от 1 до 31;
- getHours(), getMinutes(), getSeconds(), getMilliseconds() – отримати відповідні компоненти.

Додатково можна отримати день тижня:

- getDay() - Отримати номер дня в тижні. Тиждень в JavaScript розпочинається з неділі, так що результат буде числом від 0 (неділя) до 6 (субота).

Усі методи, вказані вище, повертають результат для місцевої часової зони.

Існують також UTC-варіанти цих методів, що повертають день, місяць, рік і т.п. для зони GMT+0 (UTC): getUTCFullYear(), getUTCMonth(), getUTCDay().

Тобто, відразу після "get" вставляється "UTC".

Якщо ваш локальний час зрушений відносно UTC, то наступний код покаже різні години:

```
//поточна дата
var date = new Date();
//година в поточній часовій зоні
alert( date.getHours() );
//скільки зараз часу в Лондоні?
//час у зоні GMT+0
alert( date.getUTCHours() );
```

Окрім описаних вище існують два спеціальні методи без UTC- варіанту:

- getTime() – повертає число мілісекунд, що пройшли з 1 січня 1970 року GMT+0, тобто того ж виду, який використовується в конструкторі new Date(milliseconds).
- getTimezoneOffset() – повертає різницю між місцевим і UTC-часом в хвилинах.

```
alert(new Date().getTimezoneOffset()); // Для GMT-1 виведе 60
```

Установка компонентів дати

Наступні методи дозволяють встановлювати компоненти дати і часу:

- setFullYear(year [, month, date])
- setMonth(month [, date])
- setDate(date)
- setHours(hour [, min, sec, ms])
- setMinutes(min [, sec, ms])
- setSeconds(sec [, ms])
- setMilliseconds(ms)
- setTime(milliseconds)(встановлює усю дату по мілісекундам з 01.01.1970 UTC)

Усі вони, окрім setTime(), володіють також UTC-варіантом, наприклад: setUTCHours().

Як видно, деякі методи можуть встановлювати декілька компонентів дати одночасно, зокрема, setHours. При цьому, якщо якась компонента не вказана, вона не змінюється. Наприклад:

```
var today = new Date;
today.setHours(0);
alert( today ); // сьогодні, але година змінена на 0
```

```
today.setHours(0, 0, 0, 0);
alert( today ); // сьогодні, рівно 00:00:00.
```

Автовиправлення дати

Автовиправлення – дуже зручна властивість об'єктів Date. Вона полягає в тому, що можна встановлювати свідомо некоректні компоненти (наприклад 32 січня), а об'єкт сам себе виправить.

```
var d = new Date(2013, 0, 32); // 32 січня 2013 ?!?
alert(d); // ... это 1 лютого 2013!
```

Неправильні компоненти дати автоматично розподіляються на інших.

Наприклад, треба збільшити на 2 дні дату «28 лютого 2011». Може бути так, що це буде 2 березня, а може бути і 1 березня, якщо рік високосний. Але нам про усе це думати не треба. Просто додаємо два дні. Решту зробить Date:

```
var d = new Date(2011, 1, 28);
d.setDate(d.getDate() + 2);
alert( d ); // 2 березня, 2011
```

Також це використовують для отримання дати, віддаленої від наявної на потрібний проміжок часу. Наприклад, отримаємо дату на 70 секунд більшу поточної:

```
var d = new Date();
d.setSeconds(d.getSeconds() + 70);
alert( d ); // виведе коректну дату
```

Можна встановити і нульові, і навіть від'ємні компоненти. Наприклад:

```
var d = new Date;
d.setDate(1); // поставити перше число місяця
alert( d );
d.setDate(0); // нульового числа немає, буде останнє число попереднього місяця
alert( d );
```

Перетворення до числа, різниця дат

Коли об'єкт Date використовується в числовому контексті, він перетвориться в кількість мілісекунд:

```
alert(+new Date) // +date те ж саме, що: +date.valueOf()
```

Важливий побічний ефект: дати можна віднімати, результат віднімання об'єктів Date – їх тимчасова різниця, в мілісекундах. Це використовують для виміру часу:

```
var start = new Date; // засікли час
// щось зробити
for (var i = 0; i < 100000; i++) {
  var doSomething = i * i * i;
}
var end = new Date; // кінець виміру
alert( "Цикл зайняв" + (end - start) + " ms" );
```

Метод Date.now()

Метод Date.now () повертає дату відразу у вигляді мілісекунд. Технічно, він аналогічний виклику +new Date (), але на відміну від нього не створює проміжний об'єкт дати, а тому у багато разів швидше. Його використання особливо рекомендується там, де продуктивність при роботі з датами критична. Звичайно це не на веб-сторінках, а, приміром, в розробці ігор на JavaScript.

Рекомендується запам'ятати наступне:

●Дата і час представлені в JavaScript одним об'єктом: Date. Створити "тільки час" при цьому не можна, воно має бути з датою. Список методів Date ви можете знайти в довіднику Date або вище.

●Відлік місяців розпочинається з нуля.

●Відлік днів тижня (для getDay()) теж розпочинається з нуля (і це неділя).

●Об'єкт Date зручний тим, що сам коректується. Завдяки цьому легко зрушувати дати.

●При перетворенні до числа об'єкт Date дає кількість мілісекунд, що пройшли з 1 січня 1970 UTC. Побічне слідство – дати можна віднімати, результатом буде різниця в мілісекундах.

●Для отримання поточної дати в мілісекундах краще використати Date.now (), щоб не створювати зайвий об'єкт Date.

Завдання:

Завдання 1. Створіть функцію getWeekDay (date), яка виводить поточний день тижня в короткому форматі „пн“, „вт“, … „нд“.

Наприклад:

```
var date = new Date(2012,0,3);  // 3 січня 2012
alert( getWeekDay(date) );      // Повинно вивести 'вт'
```

Завдання 2. Напишіть функцію getSecondsToday (), яка повертає скільки секунд пройшло з початку сьогоднішнього дня.

Наприклад, якщо зараз 10:00 і не було переходу на зимовий/літній час, то:

```
getSecondsToday() == 36000 // (3600 * 10)
```

Функція повинна працювати у будь-який день, тобто в ній не повинно бути конкретного значення сьогоднішньої дати.

Підказка. Для виведення досить згенерувати об'єкт Date, що відповідає початку дня, тобто "сьогодні" 00 годин 00 хвилин 00 секунд, і відняти його з поточної дати.

Додатково. Що треба додати в коді, щоб функція повертала скільки секунд залишилося до завтра?

Завдання 3. Напишіть функцію formatDate (date), яка виводить дату date у форматі дд.мм.рр:

Наприклад:

```
var d = new Date(2011, 0, 20); // 20 січня 2011
alert( formatDate(d) ); // '20.01.11'
```

Зверніть увагу, провідні нулі мають бути присутніми, тобто 1 січня 2001 повинно бути 01.01.01, а не 1.1.1.

Підказка. Компоненти треба вивести один за іншим. Використайте date.getDate(), date.getMonth(), date.getFullYear().

ЛАБОРАТОРНА РОБОТА № 7

Тема роботи: Звернення до елементів форми – прапорці, радіо кнопки, списки.
Мета роботи:
●Навчитися створювати звернення до елементів форми.
●Навчитися перевіряти значення властивостей об'єктів форми: прапорців, радіо кнопок, випадних списків.
Зміст протоколу до лабораторної роботи:
●Титульний аркуш (додаток 1);
●Мета роботи;
●JavaScrip-коди до кожного завдання;
●Копії екранів з відображенням результатів роботи у браузері для кожного завдання;
●Висновки по роботі.

Властивість checked об'єкту input

Прапорці і радіо кнопки визначаються в документі тегом <input> з атрибутами:

●type="checkbox" – прапорець;
●type="radio" – радіо кнопка.

Прапорці не залежать один від одного. Їх можна встановлювати і скидати у будь-якій комбінації. Радіо кнопки призначені для вибору одного варіанту з декількох альтернативних.

Прапорці і радіо кнопки - різновиди об'єкту input. У об'єкті input є властивість checked. Ця властивість має значення true, коли прапорець встановлений (радіо кнопка вибрана), і false –в іншому випадку. Ми повинні навчитися перевіряти значення цієї властивості.

Зверніть увагу, що властивість checked об'єкту input має зовсім інший сенс ніж однойменний атрибут тегу input. Атрибут вказує, що прапорець встановлений за умовчанням, а властивість зберігає поточне положення прапорця, яке може змінюватися користувачем або програмою.

Приклад 1: Напишіть додаток для перевірки таблиці множення. Правильні приклади необхідно відмітити прапорцями. Якщо помилок немає, в полі форми вивести слово "вірно", інакше "невірно" (рисунок 7.1)

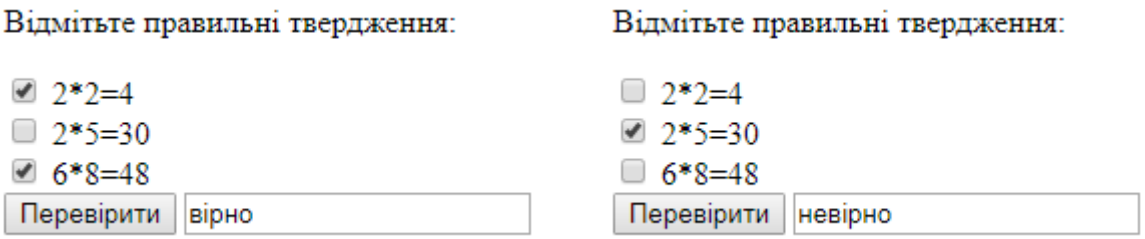


Рисунок 7.1

```
<head>
<script>

function checkAll()
{
var ans='невірно';
// змінна p1 прапорець з ім'ям c1
var p1=document.getElementById('c1');
```

```
// змінна p2 прапорець з ім'ям c2
var p2=document.getElementById('c2');
// змінна p3 прапорець з ім'ям c3
var p3=document.getElementById('c3');
// змінна p4 текстове поле з ім'ям result
var p4=document.getElementById('result');
// якщо прапорець c1 піднятий, прапорець c2 не піднятий і прапорець c3 піднятий
if (p1.checked && !p2.checked && p3.checked) ans='вірно';
// виведення результату в текстове поле
p4.value=ans;
}
</script>
</head>

<body>
<p> Відмітьте правильні твердження:</p>
<form>
<input type="checkbox" id="c1" name="c1"> 2*2=4 <br/>
<input type="checkbox" id="c2" name="c2"> 2*5=30<br/>
<input type="checkbox" id="c3" name="c3"> 6*8=48<br/>
<input type="button" value="Перевірити" onClick="checkAll();">
<input type="text"id="result" name="result" value="">
</form>
</body>
```

Приклад 2: Той самий додаток переробити так, щоб результат виводився в нове вікно Рисунок 7.2.

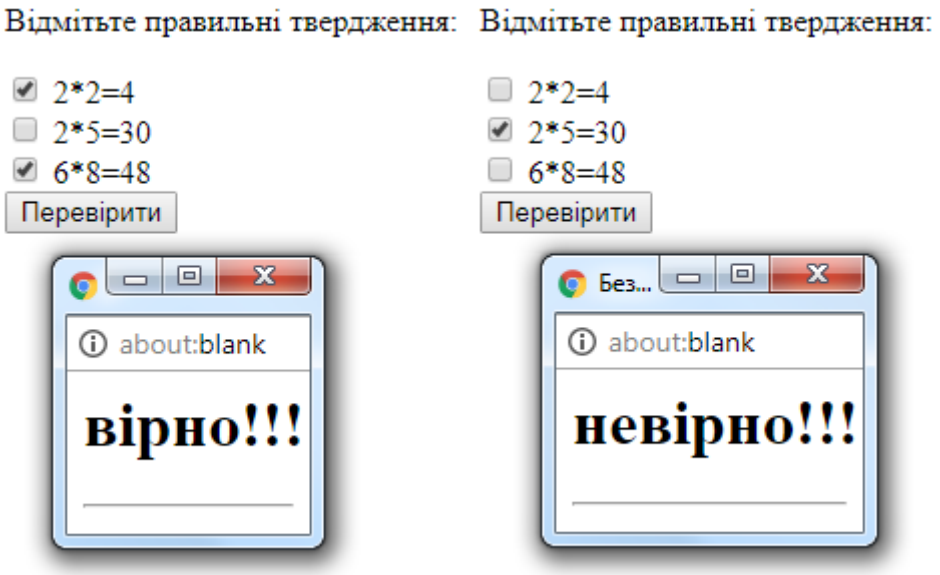


Рисунок 7.2

```
<head>
<script>

function checkAll()
{
var ans='невірно';
```

```
// змінна p1 прапорець з ім'ям c1
var p1=document.getElementById('c1');
// змінна p2 прапорець з ім'ям c2
var p2=document.getElementById('c2');
// змінна p3 прапорець з ім'ям c3
var p3=document.getElementById('c3');
// якщо прапорець c1 піднятий, прапорець c2 не піднятий і прапорець c3 піднятий
if (p1.checked && !p2.checked && p3.checked) ans='вірно';
// виведення результату в нове вікно
// змінна win екземпляр об'єкту window
// створюється порожнє вікно розмірами 100 на 100,
var win=window.open("","","width=100,height=100");
// відкриваємо запис в документ вікна win
win.document.open();
// формування рядка str для виведення
// вміст в заголовок передається зі змінної ans
var str = "<h1>"+ans+"!!!</h1><hr>";
// виведення рядка в документ вікна win
win.document.write(str);
// закриваємо виведення в документ вікна win
win.document.close();
}
</script>
</head>

<body>
<p> Відмітьте правильні твердження:</p>
<form>
<input type="checkbox" id="c1" name="c1"> 2*2=4 <br/>
<input type="checkbox" id="c2" name="c2"> 2*5=30<br/>
<input type="checkbox" id="c3" name="c3"> 6*8=48<br/>
<input type="button" value="Перевірити" onClick="checkAll();">
</form>
</body>
```

Приклад 3: Напишіть додаток, який перевіряє, чи правильно вибрана радіо кнопка. Вивести результат методом alert (рисунок 7.3).

Рисунок 7.3

```
<html>
<head>
<script type="text/JavaScript">
```

```
// отримати значення з радіокнопок
function validate()
{
// змінна p1 радіокнопка з ім'ям k2
var p1=document.getElementById('k2');
// якщо вибрана друга радіокнопка
if (p1.checked) alert("вірно");
else alert("невірно");
}
</script>
</head>

<body>
<h2>
що повертає метод prompt об'єкту window?
</h2>
<form>
<h3>
<input type='radio' id='k1' name='v' checked />   ціле число <br />
<input type='radio' id='k2' name='v' />   строку <br />
<input type='radio' id='k3' name='v' />   дійсне число <br />
</h3>
<input type="button" value="Перевірити" onClick="validate()" />
</form>
</body>
</html>
```

Приклад 4: Напишіть додаток, який створює нове вікно і виводить в нього введений в текстове поле форми текст у вигляді заголовка. Вид вирівнювання для заголовка вибирається за допомогою радіо кнопки (рисунок 7.4).

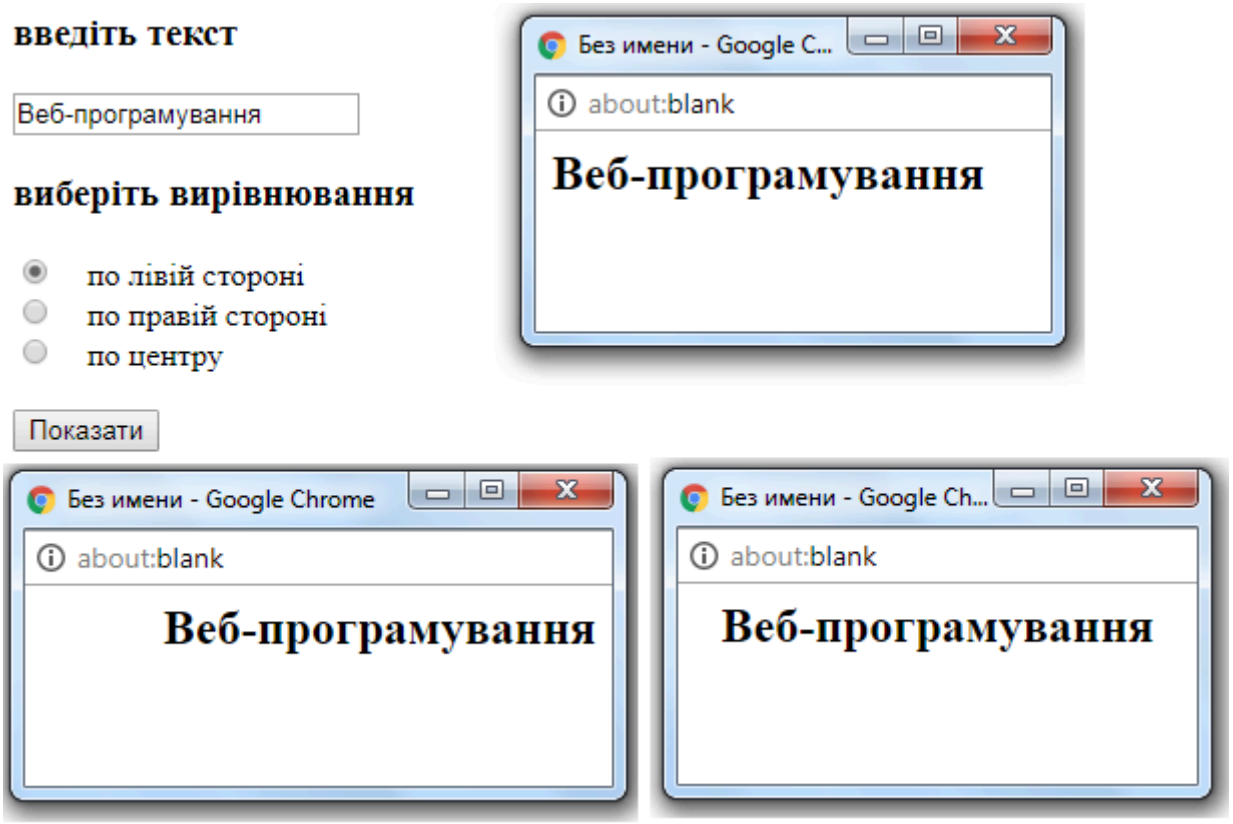


Рисунок 7.4

```
<html>
<head>
<script type="text/JavaScript">
// отримати значення з радіо кнопок
function validate()
{
// змінна p текстового поля з ім'ям tx
var p=document.getElementById('tx');
// змінна textt містить текст з текстового поля
var textt=p.value
// змінна p1 радіо кнопка з ім'ям k1
var p1=document.getElementById('k1');
// змінна p2 радіо кнопка з ім'ям k2
var p2=document.getElementById('k2');
// змінна p3 радіокнопка з ім'ям k3
var p3=document.getElementById('k3');
// значення змінної vt - вид вирівнювання
// якщо вибрана 1-а радіо кнопка
if (p1.checked) vt='left'
// якщо вибрана 2-а радіо кнопка
if (p2.checked) vt='right'
// якщо вибрана 3-я радіо кнопка
if (p3.checked) vt='center'
var win=window.open("","","width=100,height=50");
win.document.open();
// змінна st містить
// сформовану таблицю стилів нового документу
var st="<head><style> h2 {text-align:"+vt+";}</style></head>"
```

```
win.document.write(st);
// змінна st містить
// сформовану тіло нового документу
var st="<body><h2>"+textt+"</h2></body>"
win.document.write(st);
win.document.close();
}
</script>
</head>
<body>
<form>
<h3>введіть текст</h3>
<input type="text"id="tx" name="tx" value="">
<h3> виберіть вирівнювання </h3>
<input type='radio' id='k1' name='v' checked />    по лівій стороні <br />
<input type='radio' id='k2' name='v' />    по правій стороні <br />
<input type='radio' id='k3' name='v' />    по центру <br />
<br />
<input type="button" value="Показати" onClick="validate()" />
</form>
</body>
</html>
```

Властивості об'єкту option

Для створення списку, що розкривається, використовується елемент <select>, який включає декілька елементів <option>, по одному елементу на кожен рядок списку.

Кожен з цих об'єктів option має наступні властивості:

- text – містить текст відповідного рядка списку;
- selected – рівне true, якщо рядок вибраний, і false – в іншому випадку.

Приклад 5: Напишіть додаток, який вибирає картинку зі списку і при натисненні на кнопку виводить в нове вікно вибрану назву у вигляді заголовка і саму картинку (рисунок 7.5).

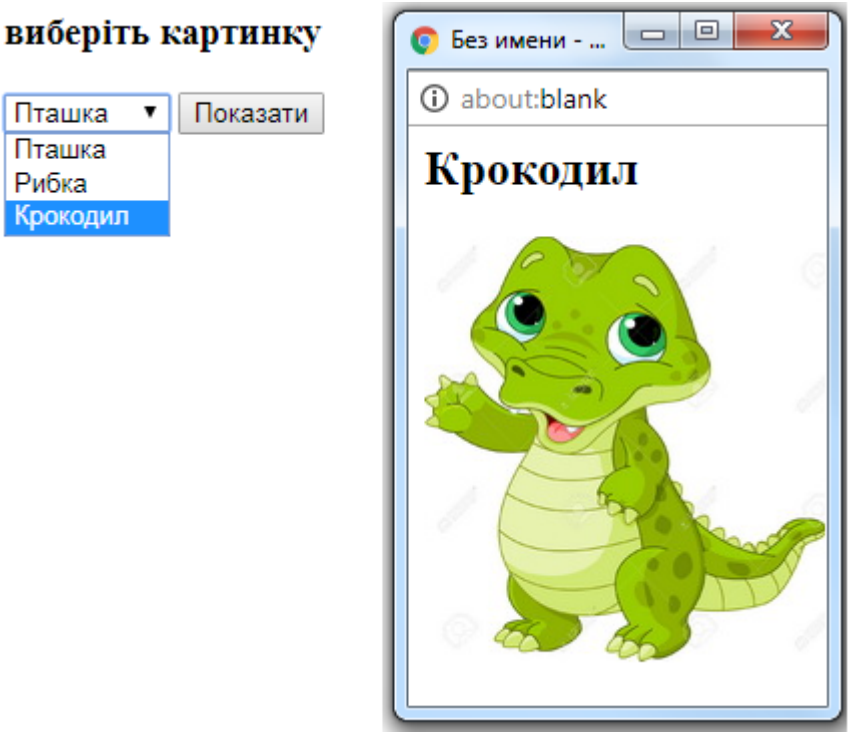


Рисунок 7.5

```
<html>
<head>
<title> Програмування списків </title>
<script type="text/JavaScript">
function GO()
{
// змінна p1 елемент списку з ім'ям k1
var p1=document.getElementById('k1');
// змінна p2 елемент списку з ім'ям k2
var p2=document.getElementById('k2');
// змінна p3 елемент списку з ім'ям k3
var p3=document.getElementById('k3');
// формування постійної частини нового документу
st1="<body><h2>"
st3="</h2><img src="
st5=" /></body>"
// формування змінної частини нового документу
if (p1.selected)
{ st2=p1.text ; st4="'Bird.gif'" }
if (p2.selected)
{ st2=p2.text; st4="'Fish.png'"}
if (p3.selected)
{ st2=p3.text; st4="'Crocodile.jpg'"}
// відкриття нового вікна
var win=window.open("","","width=200,height=300");
win.document.open();
// сформований документ
str=st1+st2+st3+st4+st5
// висновок у вікно
win.document.write(str);
win.document.close();
}
</script>
</head>

<body>
<form>
<h3> виберіть картинку </h3>
<select>
<option id='k1'> Пташка </option>
<option id='k2'>Рибка</option>
<option id='k3'>Крокодил</option>
</select>
<input type="button" value="Показати" onclick="GO()">
</form>
</body>
</html>
```

Завдання

Варіант 1 (непарні номери за списком в журналі)

Створіть HTML- додаток з такою формою як показано на рисунку 7.6:

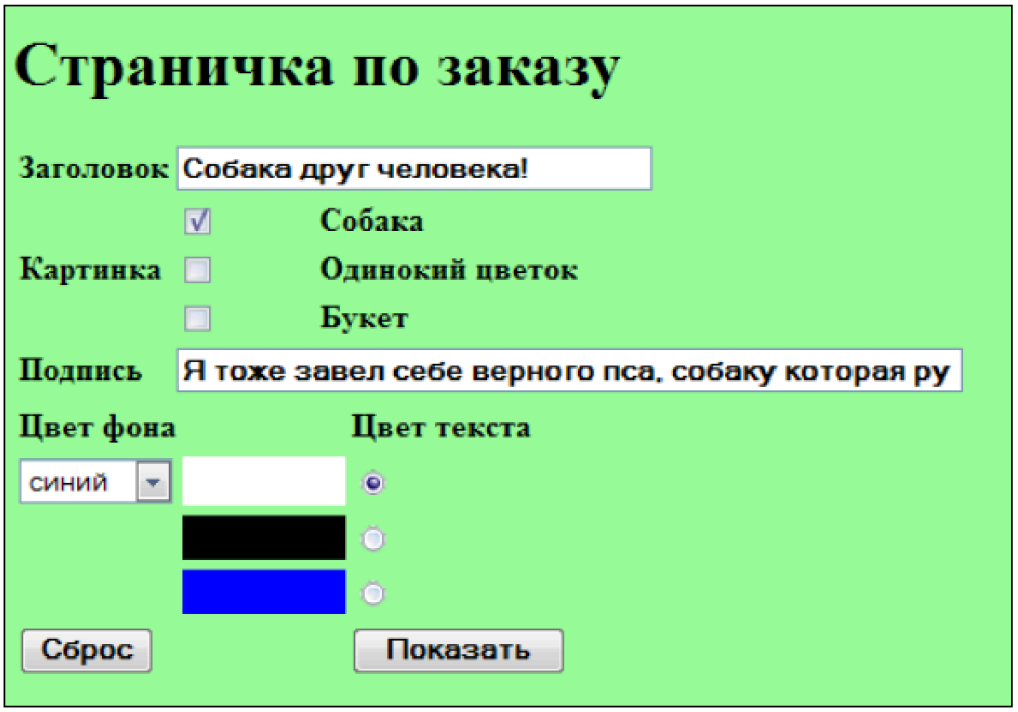


Рисунок 7.6

По натисненню на кнопку "Показати", додаток повинен відкрити нове вікно і показати там замовлені картинки з короткими підписами. Нове вікно повинне створюватися "на льоту" з використанням інформації, яку ввів у форму користувач.

Приклад можливої сторінки показано на рисунку 7.7:

Рисунок 7.7

Варіант 2 (парні номери за списком в журналі)

Створіть HTML-додаток з такою формою ск показано на рисунку 7.8:

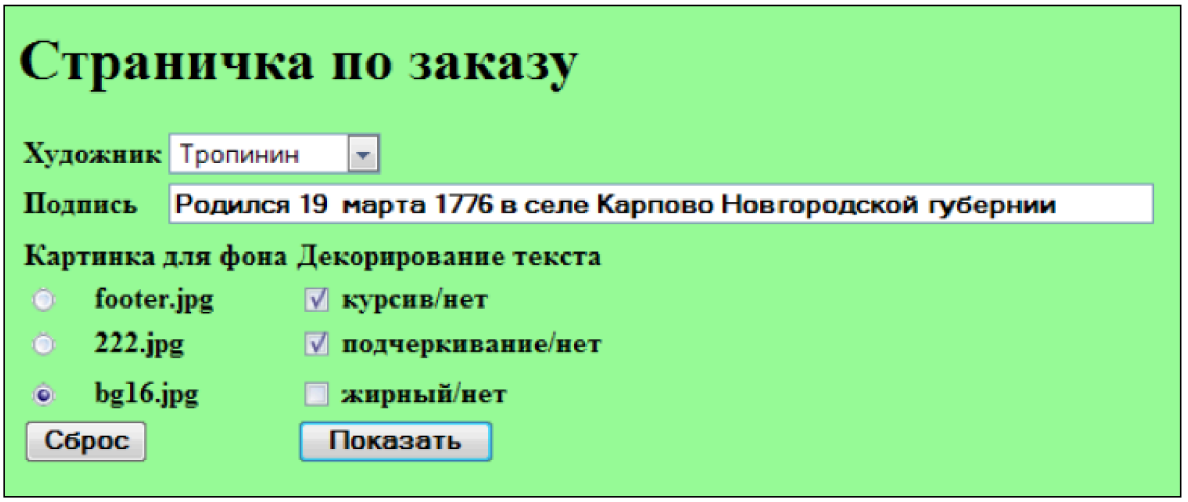


Рисунок 7.8

По натисненню на кнопку "Показати", додаток повинен відкрити нове вікно і показати там портрети вибраного художника з короткими підписами. Нове вікно повинне створюватися "на льоту" з використанням інформації, яку ввів у форму користувач.

Приклад можливої сторінки вказано на рисунку 7.9:

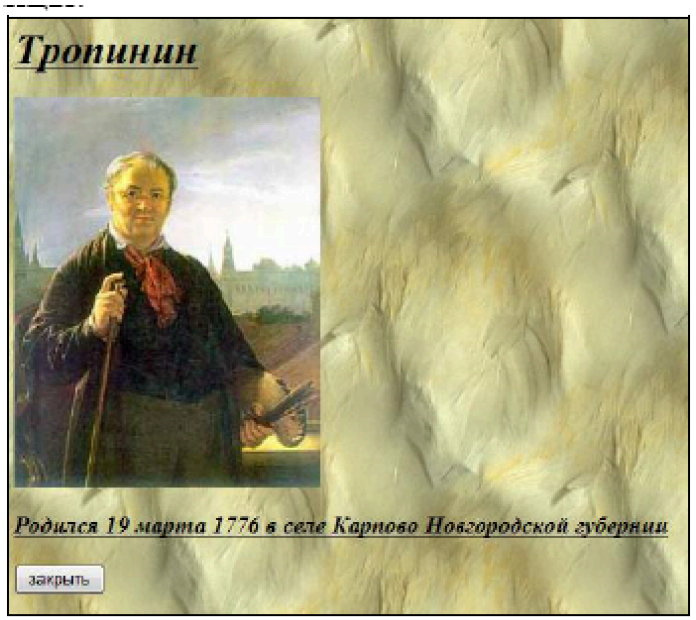


Рисунок 7.9

ЛАБОРАТОРНА РОБОТА № 8

Тема роботи: Об'єкт Image.

Мета роботи:

- Навчитися використовувати об'єкт Image;
- Вивчити події MouseOver, MouseOut.

Зміст протоколу до лабораторної роботи:

- Титульний аркуш (додаток 1);
- Мета роботи;
- JavaScrip-коди до кожного завдання;
- Копії екранів з відображенням результатів роботи у браузері для кожного завдання;
- Висновки по роботі.

Усі картинки (елементи img) в документі є екземплярами об'єкту Image. У кожного об'єкту Image існує властивість src, яку можна міняти. Використовуючи це, можна вносити зміни в графічні образи, присутні на web-сторінці.

Події миші MouseOver і MouseOut

Події mouseover і mouseout відбуваються, коли курсор миші переміщається на елемент або відповідно йде за його межі. Обробники подій мають імена onmouseover і onmouseout відповідно. У прикладі 1 будемо змінювати цю властивість залежно від того, яка настає подія mouseover або mouseout.

Приклад 1: У цьому прикладі при наведенні курсора миші на малюнок, на його місці з'являється інший малюнок (рисунок 8.1 – 8.2).

Для обробки подій створена функція doEvent з логічним параметром. При наведенні курсора на картинку (виникає подія MouseOver) викликається функція doEvent з параметром true. В цьому випадку у картинки міняється значення властивості src на flower.jpg. При відведенні курсора за картинку (виникає подія onMouseOut) викликається функція doEvent з параметром false і значення цієї властивості відновлюється на bigdaisy.jpg.

Рисунок 8.1

Рисунок 8.2

```
<head>
 <style>
  body {background-color:#dfd8c5;}
  img {width:200; height:300}
 </style>
```

```
<script>
  function doEvent(type) {
   // змінна cv об'єкт img з ім'ям pic
   var cv=document.getElementById('pic');
   // якщо параметр функції рівний true, то
   // міняємо властивість src об'єкту cv на flower.jpg
   if(type) cv.src="flower.jpg";
   // в іншому випадку показуємо
   // картинку з ім'ям bigdaisy.jpg
   else cv.src="bigdaisy.jpg";
  }
</script>
</head>
 <body>
  <H1> Події MouseOver і MouseOut</H1>
  <p> Перенеси курсор миші на квітку!</p>
  <img name ="pic" id='pic' src="bigdaisy.jpg"
     onMouseOver="doEvent(true);"
     onMouseOut="doEvent(false);">
 </body>
```

Приклад 2: У цьому прикладі при наведенні курсору миші на кнопку, в текстовому полі форми з'являється підказка про призначення цієї кнопки. При відведенні курсору у бік підказка зникає (рисунок 8.3).

У скрипті створений масив mess, що містить усі варіанти підказок.

Для обробки подій створена функція doEvent з параметром, який співпадає з одним з індексів масиву. Функція виводить значення елементу масиву з цим індексом в полі info форми.

При наведенні курсору на будь-яку з кнопок (виникає подія onMouseOver) параметру привласнюється значення 1, 2 або 3. По цій події викликається функція, якій передаються ці параметри і виводиться в поле відповідний елемент масиву mess. При відведенні курсору з кнопки (виникає подія onMouseOut) функції передається значення 0 і в поле виводиться елемент масиву з індексом 0, а саме порожній рядок.

Рисунок 8.3

```
<head>
  <style>body{background-color:#dfd8c5;}
  </style>
<script>
  var mess = new Array("","Виключення комп'ютера ",
  " Видалення файлів з системного диска ",
  " Форматування вінчестера ");
  function doEvent(num)
  {
  // змінна cv об'єкт з ім'ям info
  var cv=document.getElementById('info');
  // міняємо властивість value об'єкту cv на відповідний елемент масиву
  cv.value= mess[num];
  }
</script>
</head>
  <body>
        <h1>onMouseOver та onMouseOut</h1>
        <P> Перед тим, як натиснути кнопку, добре подумай!
        <form>
              <input type="button" value=" 1 "
              onMouseOver="doEvent(1);" onMouseOut="doEvent(0);">
              <input type="button" value=" 2 "
              onMouseOver="doEvent(2);" onMouseOut="doEvent(0);">
              <input type="button" value=" 3 "
              onMouseOver="doEvent(3);" onMouseOut="doEvent(0);">
              <br/><br/>
              <input name="info" id='info' type="text" value="" size="50">
        </form>
  </body>
```

Приклад 3: У цьому прикладі після натиснення на кнопку, картинка у вікні міняється автоматично кожні декілька секунд (рисунок 8.4).

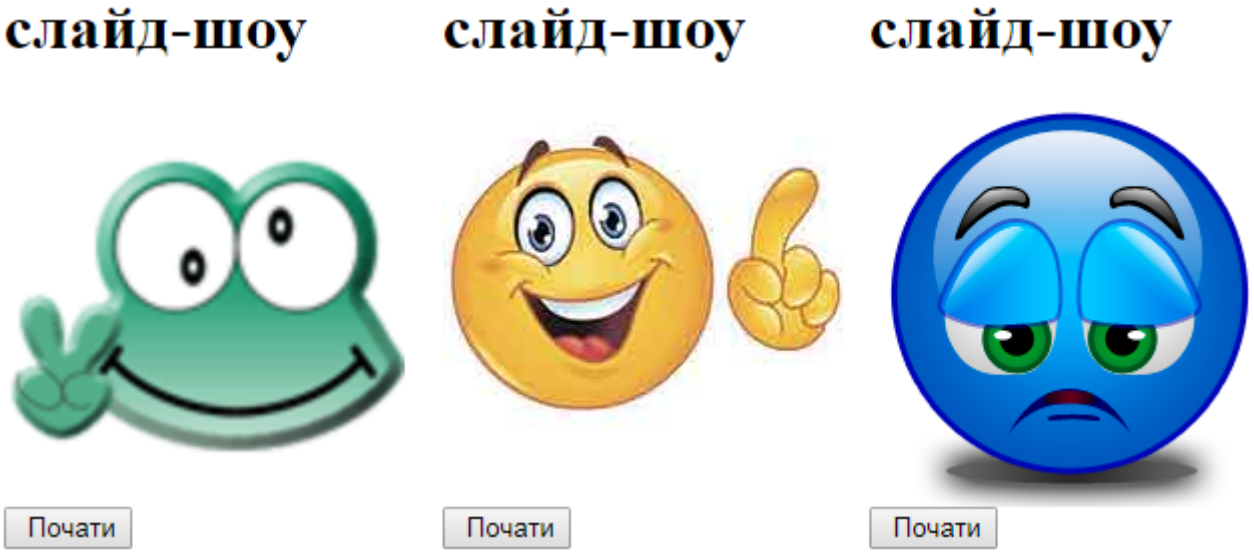


Рисунок 8.4

```
<head>
  <style>
  img{width:200;height:200}
  </style>
<script>
  i=0;
  function slideShow()
  {
  // масив картинок
  ris = new Array('7.jpg',   '8.jpg',   '9.png', '6.jpg');
  // Перевіряємо, чи не вийшов лічильник за межі масиву
  if  (i >= 4)      i = 0;  // якщо вийшов, обнуляємо лічильник
  // змінна r об'єкт з ім'ям slide
  r=document.getElementById('slide')
  // Міняємо властивість src цього об'єкту на елемент з масиву
  r.src=ris[i];
  // Збільшуємо лічильник
  i=i+1;
  // викликаємо цю ж функцію через кожні 1500 мс
  setTimeout("slideShow()",1500);
  }
</script>
  </head>
  <body>
          <h1> слайд-шоу </h1>
          <img id='slide'    src="3.gif"/>
  <form>
          <input type="button" value=" Почати" onClick="slideShow();">
  </form>
  </body>
```

Завдання

Варіант 1 (непарні номери за списком в журналі)

1. У завданні при наведенні курсору миші на фото автомобіля (рисунок 8.5), в текстовому полі форми з'являється повідомлення про його марку і вартість. При відведенні курсору у бік повідомлення зникає. Тематика малюнків і інформація, що виводиться, в полі форми може бути будь-якою. Фон сторінки – будь-який світлий. Колір тексту темний, але не чорний.

Рисунок 8.5

2. Організувати показ слайд-шоу з цих картинок, щоб кожна наступна картинка показувалася не автоматично, а при натисненні кнопки (рисунок 8.6).

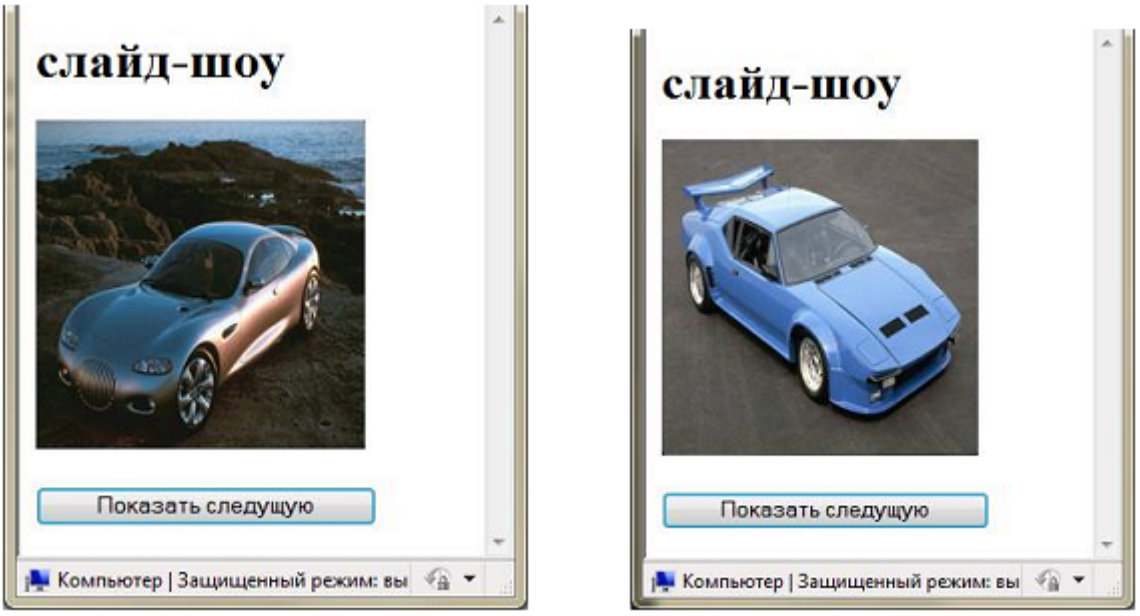


Рисунок 8.6

Варіант 2 (парні номери за списком в журналі)

1. У завданні при наведенні курсору миші на назву тварини (рисунок 8.7), в елементі таблиці з'являється малюнок цієї тварини. При відведенні курсору у бік малюнок зникає (тобто підставляється порожній малюнок, наприклад empty-1.gif). Тематика малюнків може бути будь-якою. Фон сторінки – будь-який світлий. Колір тексту – темний, але не чорний.

Рисунок 8.7

2. Організувати показ слайд-шоу з цих же картинок, щоб кожна наступна картинка показувалася не автоматично, а при натисненні кнопки (рисунок 8.8).

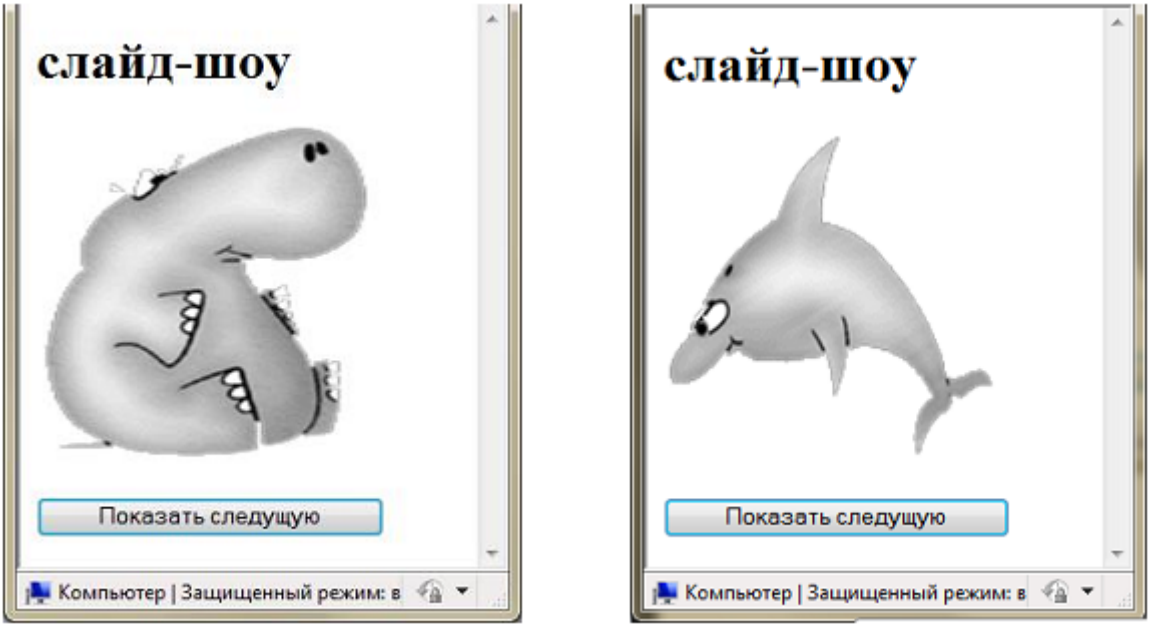


Рисунок 8.8

ЛАБОРАТОРНА РОБОТА № 9

Тема роботи: Властивість style. Об'єкт style і його властивості.

Мета роботи:

- Навчитися використовувати об'єкт style;
- Вивчити властивість display;
- Вивчити основи переміщення елементів.

Зміст протоколу до лабораторної роботи:

●Титульний аркуш (додаток 1);
●Мета роботи;
●JavaScrip-коди до кожного завдання;
●Копії екранів з відображенням результатів роботи у браузері для кожного завдання;
●Висновки по роботі.

В усіх HTML-елементів є властивість style. Значенням цієї властивості є об'єкт style, чиї власні властивості дозволяють змінювати установки стилів.

Властивості об'єкту style

Властивості об'єкту style дозволяють змінити стиль будь-якого елементу Web-сторінки, просто присвоївши потрібній властивості необхідне значення. Існує чітка відповідність між властивостями стилю CSS і властивостями об'єкту style в JavaScript.

Властивості об'єкту style в JavaScript мають майже такі ж імена що і в CSS за тим виключенням, що символи "-" забираються, а перші букви усіх слів, що утворюють ім'я атрибуту, окрім першого, робляться прописними. У наступній таблиці 9.1 показані приклади перетворення імен атрибутів стилю в імена властивостей об'єкту style, що встановлюють стиль елементу.

Таблиця 9.1

CSS	JavaScript
background-attachment	backgroundAttachment
font-family	fontFamily
z-index	zIndex

Приклад 1: По натисненню кнопки змінюватимемо колір букви і напис на кнопці (рисунок 9.1).

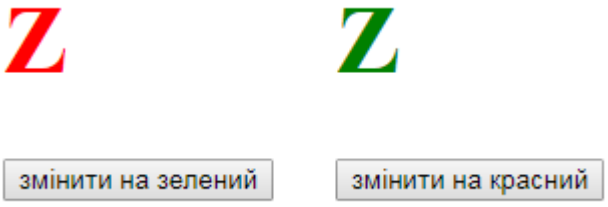


Рисунок 9.1

```
<head>
 <style>
  h1
   {color:red; font-size:36pt;}
 </style>
<script>
 i=0;
 function f()
 {
```

```
 // об'єкт буква Z
 r1=document.getElementById('e1')
 // об'єкт кнопка
 r2=document.getElementById('e2')
 if (i==0)
  {
  у об'єкту r1 є властивість style. Це теж об'єкт
  // міняємо у об'єкту style властивість color
  r1.style.color='green';
  // міняємо у об'єкту r2 властивість value
  r2.value='змінити на червоний';
  i=1;}
 else
  {
  r1.style.color='red'; r2.value='змінити на зелений';
  i=0
  }
 }
</script>
</head>
```

Властивість display

Властивість display задає спосіб відображення елементу на сторінці. Деякі значення:

- inline — елемент поводиться як лінійний;
- block — елемент поводиться як блоковий;
- none — "видаляє" елемент із сторінки разом з вмістом. Елемент не видимий, на сторінці блоки розташовуються так, немов елементу немає.

Приклад 2: По натисненню кнопки картинка з'являтиметься і зникатиме (рисунок 9.2). Для цього будемо користуватися властивістю display об'єкту style.

Натисни мене

Рисунок 9.2

```
<head>
 <script>
  function displ()
  {
   p=document.getElementById('st')
   // у об'єкту p є властивість style. Це теж об'єкт
   // міняємо у об'єкту style властивість display
   if (p.style.display == 'none')
    {p.style.display = 'block'}
   else {p.style.display = 'none'}
  }
 </script>
</head>

<body>
 <img src="FlyingBird.gif" id='st' />
 <br />
 <input type="button" value="Натисни мене" onclick="displ();" >
</body>
```

Основи переміщення елементів

Рух об'єкту на Web- сторінці здійснюється шляхом зміни властивостей, які задають його координати. У CSS координати об'єкту задаються за допомогою властивостей left і top.

Властивість left задає горизонтальну координату об'єкту, яка у разі позиціонування в абсолютних координатах (position: absolute) задає відстань в пікселях від лівої межі екрану. Властивість top задає вертикальну координату об'єкту, яка у разі позиціонування в абсолютних координатах задає відстань в пікселях від верхньої межі екрану. Якщо задати елементу абсолютне позиціонування і змінювати його координати top і left, то елемент рухатиметься.

Метод setTimeout (expression, msec)

Метод setTimeout виконує вираження або функцію після закінчення встановленої кількості мілісекунд.

Expression – строкове вираження, що містить ім'я функції, що викликається, msec – числове значення в мілісекундах. Вираження виконується одноразово і для його повторного виконання потрібно черговий виклик методу.

Приклад 3: Створимо документ, в якому малюнок Nature.jpg займає 75% вікна і є нижнім шаром. На верхньому шарі другий малюнок FlyingBird.gif, який рухається в право. Коли другий малюнок досягає краю першого малюнка, він зупиняється (рисунок 9.3).

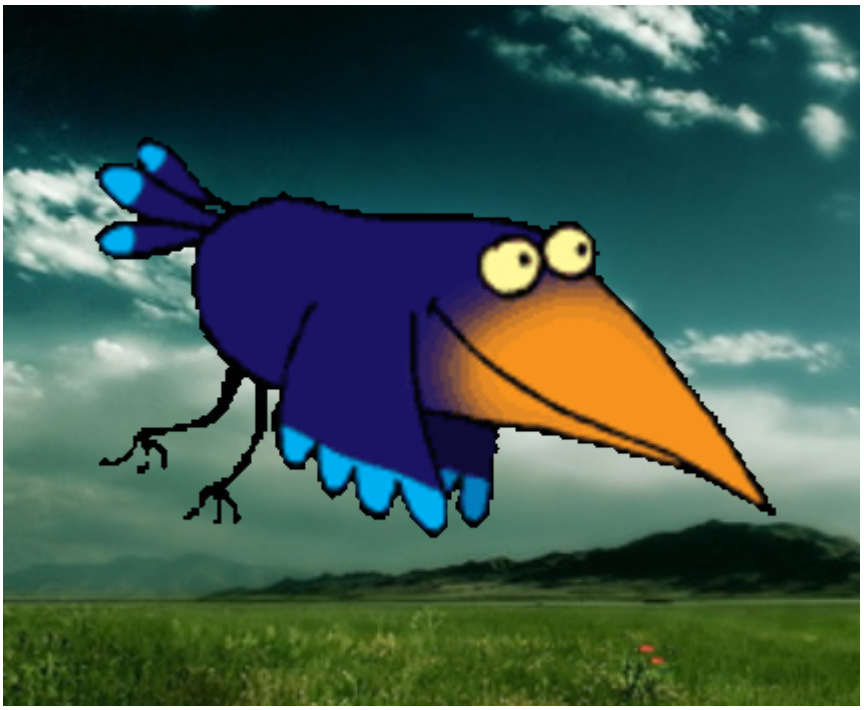


Рисунок 9.3

```
<head>
 <style>
  #im2 {position:absolute;left:0;top:100}
 </style>
</head>

<body>
 <img src="Nature.jpg" id="im1" width="75%" />
 <img src="FlyingBird.gif" id="im2" />
 <script type="text/JavaScript">
  function f()
   {
   // зрушення малюнка на 10
   x=x+10;
   // змінюємо властивість left 2-го малюнка
   r2.style.left=x
   // якщо він не дійшов до краю 1-го малюнка
   // то викликаємо кожної 1000 мс цю ж функцію
   if (x<x1-x2) setTimeout("f()",700);
   }
  // початок скрипта
  x=0
  r1=document.getElementById('im1')
  r2=document.getElementById('im2')
  x1=r1.width // ширина фону
  x2=r2.width //ширина пташки
  f();
 </script>
</body>
```

Приклад 4: Змінимо попередній документ так, щоб малюнок FunnySun.gif рухався по головній діагоналі малюнка Nature2.jpg вниз. Досягнувши правого нижнього кута, малюнок рухається вгору по цій ж діагоналі. Досягнувши лівого верхнього кута, цей малюнок знову рухається вниз. І так далі (рисунок 9.4).

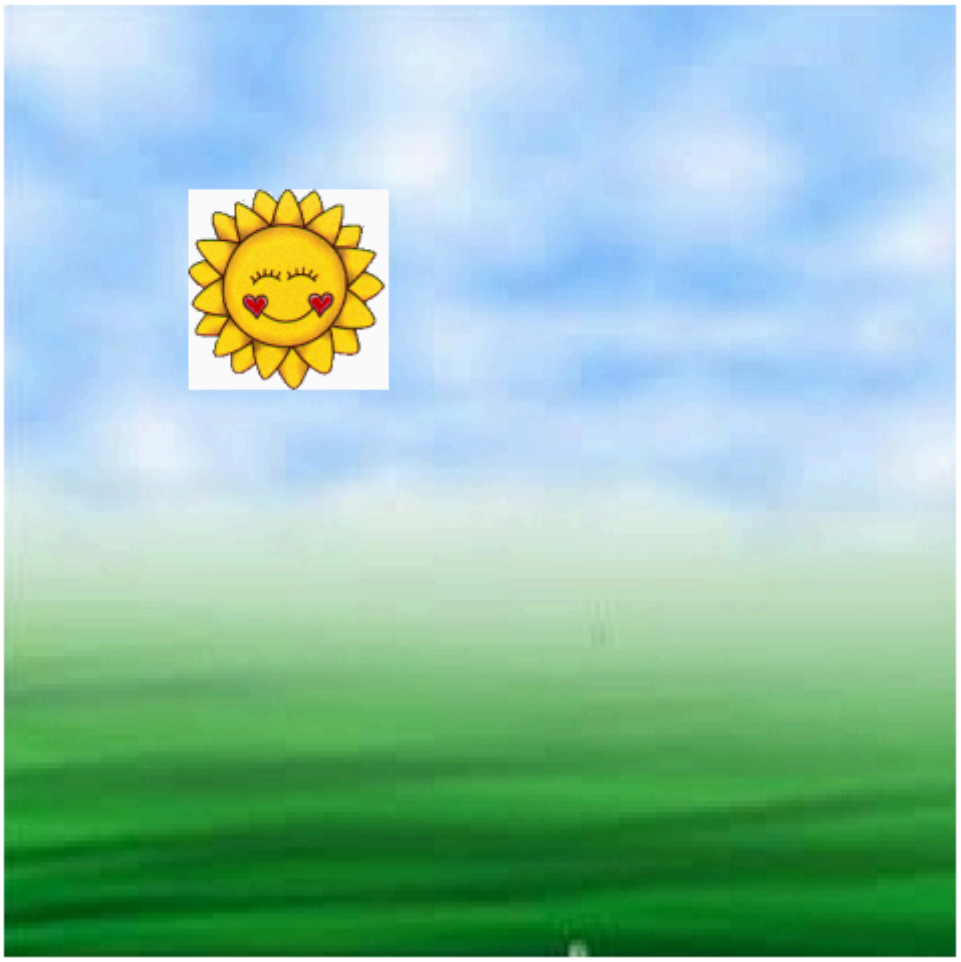


Рисунок 9.4

Крок – відстань ліворуч і згори від лівого верхнього кута нерухомого малюнка до малюнка, що рухається. Якщо малюнок рухається управо і вниз, то крок позитивний. Якщо малюнок рухається вліво і вгору, то крок негативний.

```
<html>
 <head>
  <style>
   #im2 {position:absolute;left:0;top:0}
  </style>
 </head>

 <body>
  <img src="Nature2.jpg" id="im1" width="30%" />
  <img src="FunnySun.gif" id="im2" />
  <script type="text/JavaScript">
   function f()
   {
   // зрушення малюнка на крок s
   x=x+s; y=y+s;
```

```
  r2.style.left=x
  r2.style.top=y
  // якщо долетів до правого краю, міняємо крок на негативний
  if (x>=x1) s=-10
  // якщо долетів до лівого краю, міняємо крок на позитивний
  if (x<=0) s=10
  setTimeout("f()",50);
  }
  x=0 ; y=0; s=10;
  r1=document.getElementById('im1')
  r2=document.getElementById('im2')
  x1=r1.width //ширина фону
  f();
 </script>
</body>
</html>
```

Завдання
Варіант 1 (непарні номери за списком в журналі)

1. Створіть документ, в якому малюнок, на ваш вибір, поміщений в рамку сірого кольору типу inset і завтовшки 50. По натисненню кнопки змінюватиме колір рамки на синій і напис на кнопці. При наступному натисненні змінюватиме колір рамки на сірий і напис на кнопці (рисунок 9.5).

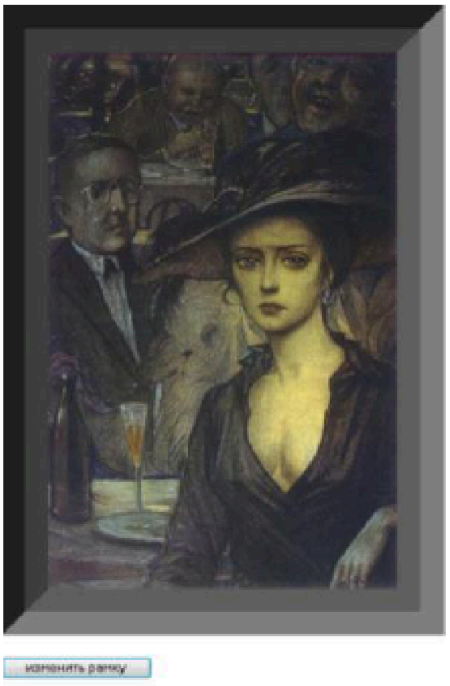


Рисунок 9.5

2. Спочатку на сторінці тільки текст вірша і кнопка з написом "показати свічку". При натисненні на кнопку текст вірша ховається, на його місці з'являється малюнок, і напис на кнопці міняється на "показати текст" (рисунок 9.6). Якщо ще раз натиснути на кнопку, малюнок зникає і з'являється текст. Текст вірша і малюнок будь-які. Параметри форматування сторінки:

- Колір фону темно-синій, розмір шрифту 22, курсив, колір шрифту білий.
- Рамка навколо тексту: колір #ffA089, тип outset, товщина 5.

- Картинка розміром 200 на 200.
- Рамка навколо картинки: колір #ffA089,тип outset, товщина 50, відступи по 5.

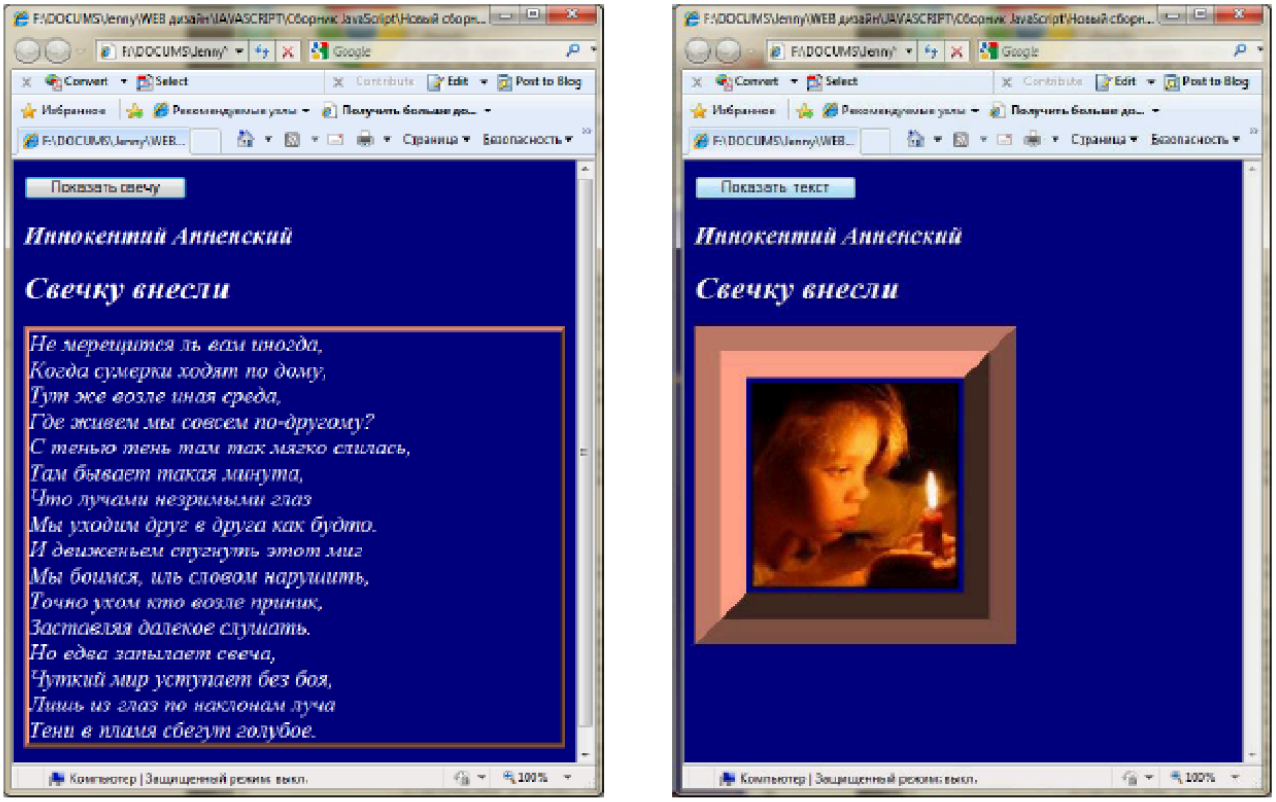


Рисунок 9.6

3. Створіть документ, в якому малюнок, на ваш вибір, займає 50% вікна і є нижнім шаром. На верхньому шарі другий малюнок, також на ваш вибір (*.gif), який рухається по горизонталі до правої межі першого малюнка, потім на рядок (висоту малюнка) вниз і назад, до лівої межі, на рядок вниз і до правої межі, і так далі до нижньої межі першого малюнка (рисунок 9.7).

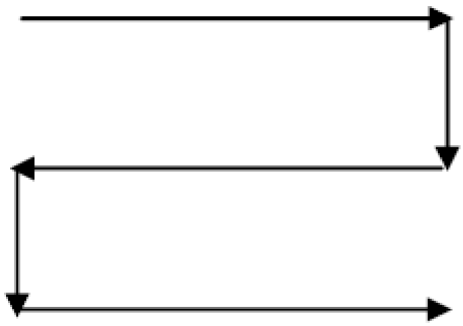


Рисунок 9.7

Варіант 2 (парні номери за списком в журнал)

1. Створіть документ, в якому блоковий елемент розмірами 250 на 250 і заливкою червоного кольору поміщений в рамку темно-синього кольору типу inset і завтовшки 10. По натисненню кнопки змінюватиме колір заливки на блакитній і напис на кнопці. При наступному натисненні змінювати колір заливки на червоний і напис на кнопці (рисунок 9.8).

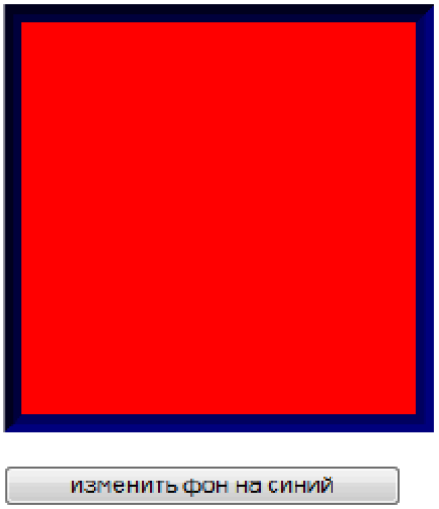


Рисунок 9.8

2. Спочатку на сторінці дві картинки у рамках і дві кнопки з написом "сховай мене". При натисненні на кнопку картинка ховається, і напис на кнопці міняється на "покажи мене" (рисунок 9.9). Якщо ще раз натиснути на кнопку, картинка з'являється і міняється напис на кнопці. Картинки вибираємо будь-які. Параметри форматування сторінки:

- Колір фону темно-синій.
- Картинки розміром 200 на 200.
- Рамка навколо першої картинки: колір #ffA089, тип outset, товщина 50, відступи по 5.
- Рамка навколо першої картинки : колір #ffA089, тип outset, товщина 50, відступи по 5.
- Рекомендується: картинки і кнопки помістити в таблицю, функцію зробити з параметром. Параметр - номер картинки.

Рисунок 9.9

3. Створіть документ, в якому малюнок, на ваш вибір, займає 50% вікна і є нижнім шаром. На верхньому шарі другий малюнок, також на ваш вибір (*.gif), який рухається по вертикалі до нижньої межі першого малюнка, потім на стовпець (ширину малюнка) управо і вгору, до верхньої межі, на стовпець управо і до нижньої межі, і так далі до правої межі нижнього малюнка (рисунок 9.10).

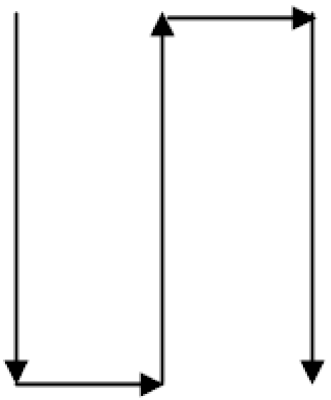


Рисунок 9.10

ЛАБОРАТОРНА РОБОТА № 10

Тема роботи: Створення калькулятора
Мета роботи:

- Вивчити метод eval;
- Розглянути роботу пропонованого в якості прикладу калькулятора;
- Виконати завдання.

Зміст протоколу до лабораторної роботи:

- Титульний аркуш;
- Мета роботи:
- JavaScrip код до завдання;
- Копія екрану з відображенням результатів роботи в браузері виконаного завдання;
- Висновки по роботі.

Об'єкт Math

Об'єкт Mathє вбудованим об'єктом, що зберігає в своїх властивостях і методах різні математичні константи і функції. об'єктMath не є функціональним об'єктом.

На відміну від інших глобальних об'єктів, об'єкт Math не є конструктором. Всі властивості і методи об'єктаMathє статичними. Ви посилаєтеся на константу π черезMath.PI і викликаєте функцію синуса через Math.sin (x), де x є аргументом методу. Константи в JavaScript визначено з повною точністю дійсних чисел.

Властивості:

Math.E - число Ейлера або Непера, основа натуральних логарифмів, приблизно рівне 2,718.
Math.LN2 - натуральний логарифм з 2, приблизно дорівнює 0,693.
Math.LN10 - натуральний логарифм з 10, приблизно дорівнює 2,303.
Math.LOG2E - двоічний логарифм з E, приблизно дорівнює 1,443.
Math.LOG10E - десятічний логарифм з E, приблизно дорівнює 0,434.
Math.PI - протношеніе довжини окружності кола до його діаметру, приблизно дорівнює 3,14159.
Math.SQRT1_2 - довадратний корінь з 1/2; або, що те ж саме, 1, поділена на квадратний корінь з 2, приблизно дорівнює 0,707.
Math.SQRT2 - довадратний корінь з 2, приблизно дорівнює 1,414.

Методи:

- Зверніть увагу, що тригонометричні функції (sin (), cos (), tan (), asin (), acos (), atan () і atan2 ()) приймають в параметрах або повертають кути в радіанах. Для перетворення радіанів в градуси, поділіть їх на величину (Math.PI / 180); для перетворення в зворотному напрямку, помножте градуси на цю ж величину.
- Зверніть увагу, що точність більшості математичних функцій залежить від реалізації. Це означає, що різні браузери можуть дати різні результати, більш того, навіть один і той же движок JavaScript на різних операційних системах або архітектурах може видати різні результати.

Math.abs (x) - повертає абсолютне значення числа.
Math.acos (x) - повертає арккосинус числа.

Math.acosh (x) - повертає гіперболічний арккосинус числа.
Math.asin (x) - повертає арксинус числа.
Math.asinh (x) - повертає гіперболічний арксинус числа.
Math.atan (x) - повертає арктангенс числа.
Math.atanh (x) - повертає гіперболічний арктангенс числа.
Math.atan2 (y, x) - повертає арктангенс від приватного своїх аргументів.
Math.cbrt (x) - повертає кубічний корінь числа.
Math.ceil (x) - повертає найменше ціле число, більше, або дорівнює зазначеному числу.
Math.clz32 (x) - повертає кількість провідних нулів 32-бітного цілого числа.
Math.cos (x) - повертає косинус числа.
Math.cosh (x) - повертає гіперболічний косинус числа.
Math.exp (x) - повертає Ex, де x - аргумент, а E - число Ейлера (2,718 ...), підстава натурального логарифма.
Math.expm1 (x) - возвращает exp (x), з якого відняли одиницю.
Math.floor (x) - повертає найбільше ціле число, менше, або дорівнює зазначеному числу.
Math.fround (x) - повертає найближчим число з плаваючою комою одинарної точності, Представляюще це число.
Math.hypot ([x [, y [...]]]) - повертає квадратний корінь з суми квадратів своїх аргументів.
Math.imul (x) - повертає результат множення 32-бітних цілих чисел.
Math.log (x) - повертає натуральний логарифм числа (loge, також відомий як ln).
Math.log1p (x) - повертає натуральний логарифм числа 1 + x (loge, також відомий як ln).
Math.log10 (x) - повертає десятковий логарифм числа.
Math.log2 (x) - повертає двійковий логарифм числа.
Math.max ([x [, y [...]]]) - повертає найбільше число зі своїх аргументів.
Math.min ([x [, y [...]]]) - повертає найменше число зі своїх аргументів.
Math.pow (x, y) - повертає підставу в ступеня експоненти, тобто, значення виразу основаніеекспонента.
Math.random () - повертає псевдовипадкове число в діапазоні від 0 до 1.
Math.round (x) - повертає значення числа, округлене до найближчого цілого.
Math.sign (x) - повертає знак числа, що вказує, чи є число позитивним, негативним або нулем.
Math.sin (x) - повертає синус числа.
Math.sinh (x) - повертає гіперболічний синус числа.
Math.sqrt (x) - повертає позитивний квадратний корінь числа.
Math.tan (x) - возвращает тангенс числа.
Math.tanh (x) - возвращает гіперболічний тангенс числа.
Math.toSource () - возвращает рядок "Math".
Math.trunc (x) - возвращает цілу частину числа, прибираючи дробові цифри.

Приклад: Лістинг самого найпростішого калькулятора.

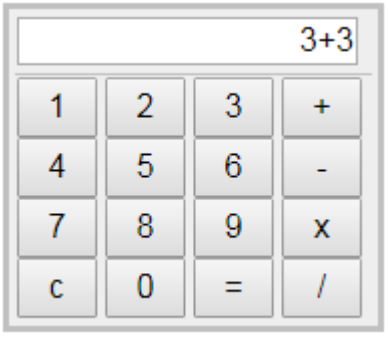


Рисунок 10.1

```
<! DOCTYPE html>
<Html>
<Head>
 <Meta charset = "utf-8" />
 <Title> Калькулятор </ title>
 <Style type = "text / css">
 #calculator * {font-size: 16px;}
 #calculator table {border: solid 3px silver; border-spacing: 3px; background-color: #EEE; }
 #calculator table td {border-spacing: 3px;}
 input.display {width: 166px; text-align: right;}
 td.buttons {border-top: solid 1px silver;}
 input [type = button] {width: 40px; height: 30px;}
 </ Style>
</ Head>

<Body>
<Form name = "calc" id = "calculator">
 <Table>
 <Tr>
 <Td>
 <Input type = "text" name = "input" size = "16" class = "display">
 </ Td>
 </ Tr>
 <Tr>
 <Td class = "buttons">
 <Input type = "button" name = "one" value = "1" OnClick = "calc.input.value + = '1'">
 <Input type = "button" name = "two" value = "2" OnClick = "calc.input.value + = '2'">
 <Input type = "button" name = "three" value = "3" OnClick = "calc.input.value + = '3'">
 <Input type = "button" name = "add" value = "+" OnClick = "calc.input.value + = '+'">
 <br>
 <Input type = "button" name = "four" value = "4" OnClick = "calc.input.value + = '4'">
 <Input type = "button" name = "five" value = "5" OnClick = "calc.input.value + = '5'">
 <Input type = "button" name = "six" value = "6" OnClick = "calc.input.value + = '6'">
 <Input type = "button" name = "sub" value = "-" OnClick = "calc.input.value + = '-'">
 <br>
 <Input type = "button" name = "seven" value = "7" OnClick = "calc.input.value + = '7'">
 <Input type = "button" name = "eight" value = "8" OnClick = "calc.input.value + = '8'">
 <Input type = "button" name = "nine" value = "9" OnClick = "calc.input.value + = '9'">
 <Input type = "button" name = "mul" value = "x" OnClick = "calc.input.value + = '*'">
 <br>
 <Input type = "button" name = "clear" value = "c" OnClick = "calc.input.value = "">
 <Input type = "button" name = "zero" value = "0" OnClick = "calc.input.value + = '0'">
 <Input type = "button" name = "doit" value = "=" OnClick = "calc.input.value = eval
(calc.input.value)">
 <Input type = "button" name = "div" value = "/" OnClick = "calc.input.value + = '/'">
 </ Td>
 </ Tr>
 </ Table>
</ Form>
</ Body>
</ Html>
```

Завдання:

- Поміняйте дизайн калькулятора – колір, розмір та розташування кнопок, шрифт.
- Зробіть так, щоб при діленні на 0 результат був не Infinity, а повідомлення про помилку, наприклад, «поділ на нуль неможливо». Також врахуйте й інші можливі помилки.
- Додайте за своїм варіантом арифметичні дії в калькулятор і математичні функції (використовуйте об'єкт Math):

Таблиця 10.1 – Варіанти завдань для індивідуального виконання

Варіант	Math	
1	sin	cbrt
2	cos	exp
3	tan	log1p
4	asin	log10
5	acos	log2
6	atan	sqrt
7	atan2	abs
8	asinh	ln10
9	acosh	expm1
10	atanh	sign

Додаток 1

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ОДЕСЬКА ПОЛІТЕХНІКА»

Інститут штучного інтелекту і робототехніки

Кафедра Комп'ютерних Систем

Лабораторна робота № 1

З дисципліни: Веб-програмування

Тема: Створення простого HTML- документу. Форматування шрифту і абзацу.

Зробив: студент групи УК-_____

Прізвище Ім'я По батькові

Оцінка:

Перевірив: ст.вик. Дікусар Е.В.

Одеса

2024