Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
«ОДЕСЬКА ПОЛІТЕХНІКА»

КОНСПЕКТ ЛЕКЦІЙ

з дисципліни «Веб-технології та веб-дизайн»
для студентів 2 курсу
спеціальності 022 Дизайн
освітня програма − Архітектурний дизайн

Одеса
ОНПУ
2024

Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
«ОДЕСЬКА ПОЛІТЕХНІКА»

КОНСПЕКТ ЛЕКЦІЙ

з дисципліни «Веб-технології та веб-дизайн»
для студентів 2 курсу
спеціальності 022 Дизайн
освітня програма − Архітектурний дизайн

Затверджено
на засіданні кафедри ІТПД
Протокол №1 від 28.08.2024 р.

Одеса
ОНПУ
2024

Конспект лекцій з дисципліни «Веб-технології та веб-дизайн» для студентів 2 курсу першого (бакалаврського) рівня вищої освіти, що навчаються за спеціальністю 022 − Дизайн, освітньо-професійна програма – Архітектурний дизайн / Укл.: О.В. Рибак, Д.М. Солодкий. − Одеса: ОНПУ, 2024. – 71 с.

Укладач: Рибак О.В., к.т.н., доцент
Солодкий Д.М., асистент

Конспект лекцій містить систематизований виклад науково-методичних матеріалів з дисципліни «Веб-технології та веб-дизайн», передбачених навчальною програмою для вивчення протягом 4 семестру підготовки бакалаврів, які навчаються за спеціальністю 022 – Дизайн напрямку підготовки Архітектурний дизайн. Лекційні заняття покликані сформувати у здобувачів основи знань з веб-дизайну, інструментів веб-розробки та принципів роботи веб-сайтів. Конспект лекцій призначається для здобувачів денної та заочної форми навчання.

ЗМІСТ

стор

ВСТУП

Конспект лекцій з дисципліни «Веб-технології та веб-дизайн» містить науково-методичні матеріали, передбачені навчальною програмою до вивчення на 2 курсі протягом 4 семестру підготовки бакалаврів, що навчаються за спеціальністю 022 – Дизайн напрямку підготовки Архітектурний дизайн.

Метою вивчення дисципліни «Веб-технології та веб-дизайн» є засвоєння методологічних та концептуальних теоретичних відомостей про веб-технології, формування у здобувачів вмінь та навичок веб-дизайну та розробки веб-сторінок, а також підготовка фахівців, які вміють застосовувати сучасні методики роботи та технічної підтримки веб-сайтів для подальшої професійної діяльності.

В процесі вивчення дисципліни «Веб-технології та веб-дизайн» здобувачі опановують сучасні можливості інформаційних технологій, інструменти та шляхи розробки користувацьких веб-інтерфейсів та дизайну веб-сайтів. Вивчення теоретичного матеріалу та аналіз прикладів виконаних завдань, представлених у даному конспекті лекцій, забезпечить формування у здобувачів основних знань з веб-дизайну, інструментів веб-розробки та принципів роботи веб-сайтів.

Лекція 1. Принцип роботи пошукових систем.

План лекції:

1.1. Веб-сторінки, веб-сайти і пошукові системи.
1.2. Відмінності пошукової системи від браузера.
1.3. Схема обробки запиту в пошуковій системі.
1.4. Виведення результатів пошуку за запитом користувача.

1.1. Веб-сторінки, веб-сайти і пошукові системи.

У сучасному світі всесвітня мережа Інтернет є основним джерелом інформації.

Інформація у мережі Інтернет представлена на веб-сторінках. Але основною структурною одиницею Інтернету є не одна сторінка, а певна кількість сторінок, об’єднаних у сайт.

Сайт або веб-сайт (англ. website, від web (павутина) і site (місце)) — це сукупність логічно пов’язаних між собою веб-сторінок, доступних у мережі Інтернет, об’єднаних під єдиним доменним ім’ям.

Сервер – це комп’ютер, підключений до мережі Інтернет або локальної мережі, який працює цілодобово, надаючи доступ до даних, що на ньому містяться.

URL (скорочене від англ. Uniform Resource Locator, що в перекладі означає «уніфікований вказівник ресурсу»). URL – стандартизована адреса веб-ресурсу, що визначає його місцезнаходження у мережі Інтернет та спосіб доступу до нього.

Пошукова система – це програмний комплекс, що надає користувачам можливість доступу до необхідної інформації в інтернеті за допомогою пошуку в універсальному сховищі даних про веб-сайти, проіндексовані зазначеною пошуковою системою.

Рис. 1.1. Відомі пошукові системи різних часів

Рис. 1.2. Найпопулярніші пошукові системи світу

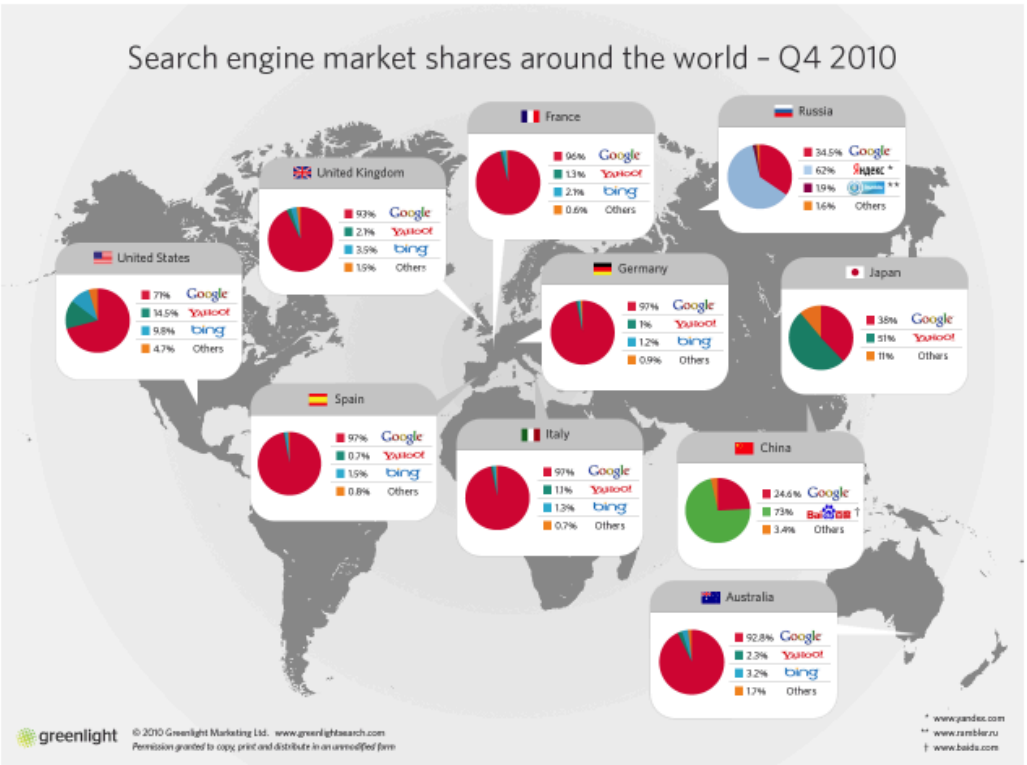

Рис. 1.3. Пошукові системи у різних країнах світу

1.2. Відмінності пошукової системи від браузера.

Браузер – це програма на комп'ютері, за допомогою якої можна переглядати власні веб-сторінки, результати видачі пошукових систем, сторінки різних сайтів в інтернеті, переходити між ними.

Прикладами браузерів є Google Chrome, Mozilla Firefox, Microsoft Edge, Safari, Opera, Internet Explorer та багато інших.

Ключові відмінності пошукової системи від браузера:

1. Неможливо отримати доступ до будь-якої інформації в інтернеті без веб-браузера. Знаючи адресу сайту, можна перейти на нього, не використовуючи пошукову систему.

2. Якщо користувач не має конкретної URL-адреси веб-сторінки, за допомогою пошукової системи можна знайти необхідну інформацію за фразою, словом або словосполученням, введеним у пошуковий рядок.

3. Браузер використовується для пошуку конкретного веб-сайту, а пошукова система – для сортування та фільтрації відповідної до запиту інформації.

4. Браузер працює як незалежна програма, натомість пошукова система не може працювати без браузера.

5. Для подальшої роботи, браузер необхідно встановити на комп'ютері або іншому пристрої. Користування пошуковою системою не потребує встановлення програмного забезпечення та передбачає доступ через браузер.

1.3. Схема обробки запиту в пошуковій системі.

Пошукові системи складаються з п'яти окремих програмних компонентів:

- Система видачі результатів (Search engine results engine) визначає результати пошуку з бази даних.
- Павук (Spider) – це програма, яка завантажує веб-сторінки. Вона працює так само, як браузер, коли користувач під'єднується до веб-сайту та завантажуєте сторінку.
- Краулер (Crawler) – програма, яка автоматично проходить по усім посиланням, знайденим на сторінці, та виділяє їх. Його задача – визначати, куди далі повинна рухатись програма-павук, ґрунтуючись на посиланнях або виходячи з заздалегідь заданого списку адрес. Краулер, слідуючи за знайденими посиланнями, здійснює пошук нових документів, ще не відомих пошуковій системі.
- Індексатор (Indexer) розділяє сторінку на частини і аналізує їх. Такі елементи, як заголовки сторінок, заголовки у тексті на сторінці, посилання, власне текст та його структурні елементи, виділяються й аналізуються окремо.
- База даних (Database) – сховище всіх даних, які пошукова система завантажує і аналізує. Зберігання даних часто потребує значних ресурсів.

1.4. Виведення результатів пошуку за запитом користувача.

Зробивши запит до пошукової системи, кожен користувач прагне якнайшвидше перейти на веб-сторінку з потрібною йому інформацією, тому пошукові сервіси є природніми розподілювачами потоків користувачів у мережі Інтернет. Таким чином, пошукові системи зосередили користувацький попит на інформацію і регулюють пропозицію інформації, перенаправляючи людей на ті або інші сайти.

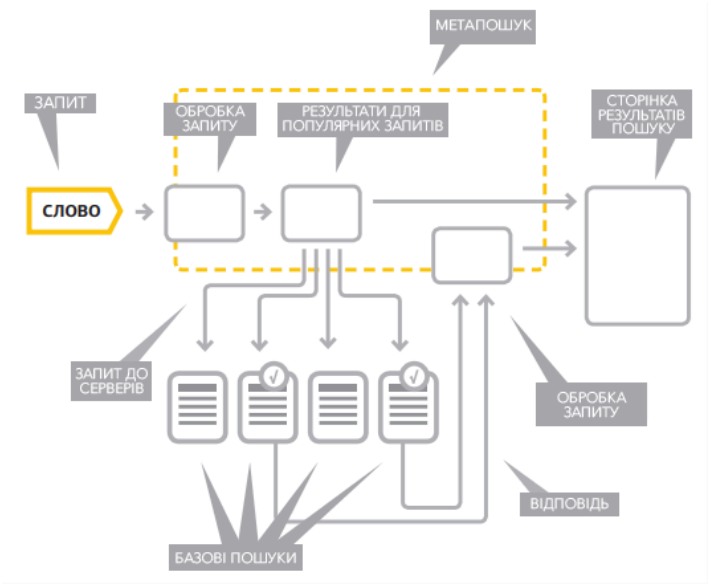

Рис. 1.4. Логічна схема обробки запиту в пошуковій системі

Контрольні питання

1. Дайте визначення поняттям «веб-сайт», «браузер», «пошукова система».
2. Подумайте, яка пошукова система є найпопулярнішою в усьому світі.
3. Поясніть різницю між цільовими типами пошукових запитів.
4. Опишіть принцип роботи пошукових систем.

Лекція 2. Індексація сайту в пошукових системах. Пошукова система Google.

План лекції:

2.1. Процес індексації сайту.
2.2. Релевантність і фактори, що на неї впливають.
2.3. Алгоритми ранжування сайтів.
2.4. Історія розвитку пошукових систем. Пошукова система Google.

2.1. Процес індексації сайту.

Індексація – це процес, під час якого пошукові роботи відвідують сайти, збираючи з їхніх сторінок різноманітну інформацію та заносячи її в спеціальні бази даних.

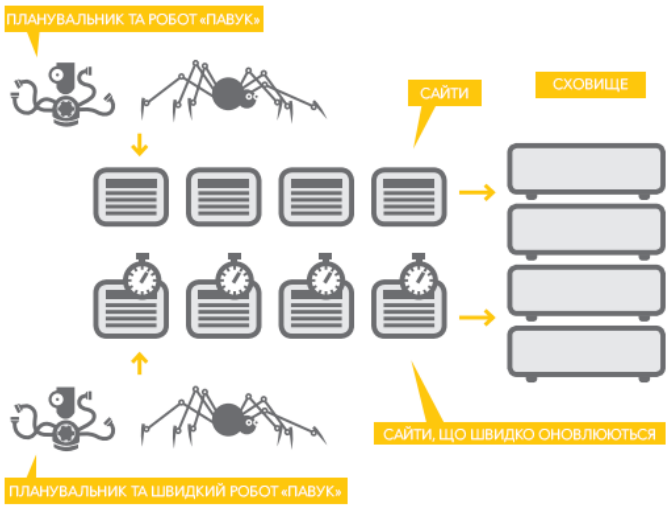

Рис. 2.1. Послідовність процесів індексації в пошуковій системі

2.2. Релевантність і фактори, що на неї впливають.

Релевантність означає ступінь відповідності знайденої веб-сторінки або веб-сайту запиту користувача.

Параметри, що впливають на порядок видачі сайту:

- наявність ключових слів в основному тексті сторінки і в мета-тегах;
- популярність сайту серед користувачів;
- концентрація ключових слів на сторінці;
- індекс цитованості;
- популярність та репутація сайтів, які посилаються на даний сайт;
- релевантність запиту сайтів, які посилаються на даний сайт.

2.3. Алгоритми ранжування сайтів.

Пошукові алгоритми засновані на математичних методах програмування, які сортують знайдені результати. Такі алгоритми ранжування є унікальними для кожної з пошукових систем. У цьому можна пересвідчитись, якщо ввести ключове слово або фразу у різних сервісах, наприклад, в Google та в Yahoo і, зберігши результати пошуку, співставити їх. У різних пошукових системах результати будуть відрізнятися.

Одним з найбільш складних алгоритмів ранжування сайтів у пошуковій видачі є Google PageRank. Google PageRank – система ранжування веб-сторінок, розроблена засновниками Google. Вона використовує структуру посилань для визначення релевантності. Сайти, які отримали великий PageRank, займають більш високі позиції у видачі Google. Хоча сам алгоритм є комерційною таємницею, вважається, що найважливішим чинником у визначенні PageRank є індекс цитованості, тобто кількість релевантних, тематично пов'язаних сайтів, які посилаються на даний сайт.

Окрім Google PageRank, для визначення релевантних до запиту сайтів з якісним контентом Google, як і інші пошукові системи, може застосовувати фільтри. Фільтри – це алгоритми, які використовуються пошуковою системою для визначення сайтів, що застосовують штучні методи підвищення позицій у видачі. У випадку виявлення порушень, пошукова система піддає такий сайт санкціям: песимізації (заниженню позиції

у пошуковій видачі) або видаленню зі списку результатів видачі. Одним з найважливіших фільтрів пошукової системи Google є Google Panda.

Google Panda – це алгоритм пошукової системи Google, який було запущено з метою знизити в результатах пошуку сайти з не унікальним, низькоякісним і дубльованим контентом. Загалом, є декілька типів фільтрів. Під дію одних сайти потрапляють через порушення, пов'язані з посиланнями, що ведуть на сайт, під дію інших – за дубльований контент, третіх – через порушення, пов'язані з поведінковими факторами, четверті песимізують сайт через технічні обмеження.

2.4. Історія розвитку пошукових систем. Пошукова система Google.

У 1990-ті роки, коли Інтернет лише починав розповсюджуватися у повсякденному житті, пошукових систем у сучасному розумінні цього слова не існувало. Пошук потрібного веб-сайту відбувався через каталоги сайтів, де наявні веб-ресурси впорядковувались за рубриками. Оскільки веб-сайтів в ті часи було небагато, такий спосіб структурування даних був досить зручним.

Першою комп’ютерною програмою для пошуку в інтернеті була програма Archie. Вона була створена у 1990 році студентами, які вивчали інформатику в університеті McGill University в Монреалі, Канада. Ця програма завантажувала списки усіх файлів з усіх доступних серверів та заповнювала базу даних, в якій можна було виконувати пошук за іменами файлів. Однак програма Archie не індексувала вміст цих файлів, оскільки об’єм даних був настільки незначним, що будь-яку інформацію можна було легко знайти вручну.

Наприкінці 1995 року було запущено найкращу пошукову систему свого часу – AltaVista. Тоді ця пошукова система вважалася надзвичайно швидкою, адже могла обробляти мільйони пошукових запитів на день. З цього часу і до 2000 року AltaVista стала найпоширенішою в усьому світі.

У 1998 році була створена компанія Google. Її розробники запропонували революційний алгоритм PageRank, який дозволяв враховувати вагу веб-сторінки в залежності від кількості посилань на неї. Такий підхід настільки покращив якість пошуку, що пошукова система Google швидкими темпами почала захоплювати аудиторію і через декілька років стала найпопулярнішою пошуковою системою у світі, якою залишається і дотепер.

Контрольні питання

1. Визначте, що собою являє процес індексації сайту.
2. Перерахуйте чинники, що впливають на релевантність веб-сторінки до запиту користувача.
3. Подумайте, чи враховує пошукова система Google знаки пунктуації, які не відносяться до операторів мови запитів.
4. Перелічіть відомі вам оператори інформаційно-пошукової мови.
5. Опишіть принцип роботи алгоритмів ранжування сайтів.
6. Розкажіть історію створення пошукової системи Google.

Лекція 3. Базові конструкції мови HTML.

План лекції:

3.1. Поняття тегу.
3.2. Структура HTML-документу.
3.3. Теги форматування тексту.
3.4. Теги заголовків.
3.5. Нумеровані й марковані списки.

3.1. Поняття тегу.

Мова HTML (HyperText Markup Language) – це мова розмітки документа, що описує форму відображення інформації на веб-сторінках у вікні веб-браузера.

Коди мови HTML, за допомогою яких проводять розмітку вихідного тексту, називають тегами. Під час відображення документа в браузері самих тегів не видно, але вони впливають на зовнішній вигляд документа. Виділяють парні – відкриваючий і закриваючий, та одиночні теги. Відкриваючі теги можуть містити атрибути, які впливають на ефект, що створюється тегом.

Атрибути – це додаткові ключові слова, які відділяються від основного ключового слова тегу і один від одного символами пробілу. Для більшості атрибутів слід задавати значення, яке відділяють від імені атрибута символом =. Значення атрибута зазвичай беруть у лапки. Закриваючі теги не містять атрибутів.

3.2. Структура HTML-документу.

Для створення веб-сторінки необхідно використовувати теги: <html>, <head>, <title>, <body>.

Код найпростішої веб-сторінки виглядає наступним чином:

```
<html>
  <head>
    <title>Моя перша сторінка </title>
  </head>
  <body>
    Привіт світе!!!
    <br>
    Записуємо довільний текст
  </body>
</html>
```

Усі документи повинні мати наступний шаблон коду:

<html> – початок документу
  <head> – початок службової інформації
  </head> – закриття службової інформації
  <body> – початок основного вмісту сторінки
  </body>– закриття основного вмісту сторінки
</html> – завершення документу

3.3. Теги форматування тексту.

Форматування тексту в HTML здійснюється за допомогою тегу <font>, а також його атрибутів size – для визначення розміру тексту та face – для задання типу шрифту.

Необхідно зазначити, що браузер використовує бібліотеку шрифтів, встановлену на комп'ютері користувача, і якщо раптом вказаного шрифту в бібліотеці не виявиться, то браузер замінить його на той шрифт, який є в наявності. Відтак, не завжди варто вказувати на веб-сторінках рідкісні нестандартні шрифти, оскільки такий шрифт на комп'ютері іншого користувача може перетворитися на типовий варіант з бібліотеки шрифтів.

Для виділення в тексті на веб-сторінці окремих параграфів необхідно скористатися тегом <p>. Вирівнювання параграфів, як і інших елементів веб-сторінки, по лівому краю, по правому краю, по центру або по ширині здійснюється шляхом додавання потрібного значення до атрибуту align.

Для задання накреслення курсивом використовується тег <i>, для напівжирного тексту – тег <b>, для підкреслення – тег <u>. Верхній індекс задається парними тегами , а нижній індекс – тегами .

Перед виведенням тексту на веб-сторінку, браузери обробляють його, видаляючи всі перенесення рядків і зайві пробіли. Проводиться така обробка для того, щоб на моніторах з різним розширенням екрану текст переносився на наступний рядок там, де це необхідно для даного пристрою, а не там, де раніше було вказано пробіли і перенесення рядків. Однак такий підхід не завжди є виправданим. Для того, щоб текст виводився браузерами на екран у тому вигляді, у якому він був набраний, тобто з усіма пробілами і перенесеннями рядків, можна скористатися спеціальним тегом <pre>. Текст, записаний всередині цього парного тегу, залишається попередньо відформатованим.

3.4. Теги заголовків.

Природні тексти зазвичай мають заголовки першого рівня. А якщо текст великий та логічно складний, то й підзаголовки для параграфів та розділів. Наявність заголовків дуже важлива для читання: при погляді на них користувач повинен зрозуміти, про що йдеться в даному тексті, і вирішити, чи варто читати його і з якого місця. Як і в книгах, у текстів на сайтах має бути своя логічна структура. Для її виділення часто використовуються заголовки, таким чином при роботі з текстом слід подбати про створення заголовків на веб-сторінці.

Виділення заголовків на веб-сторінці відбувається за допомогою парних тегів <h1>, <h2>, <h3>, <h4>, <h5> та <h6> для заголовків різного рівня. Заголовки на сторінці – це текстові елементи, виділені тегами <h1>, <h2>, …, <h6>. Дані теги створені для структурування тексту, тому використання їх у навігації ресурсу чи в службових елементах є прикладом погано розробленої сторінки.

Оскільки заголовок створюється автором тексту для привернення уваги читача, його смислова вага в середньому вища, ніж вага звичайного слова у тексті. Пошукові системи враховують цю обставину, тому при ранжуванні результатів пошуку слова у фразі, виділеній як заголовок, можуть мати більше значення, ніж простий текст. І якщо слова із запиту входять до якогось заголовку на сторінці, то вага такого входження буде оцінена пошуковою системою вище, ніж вага входження слів запиту в простий текст.

Теги <h1>, <h2>, <h3>,…<h6> можна використовувати лише в основному тексті веб-сторінки і лише у логічній послідовності. Заголовок рівня <h1> завжди вищий за інших, і повинен бути лише один на сторінці. Після нього в тексті можуть йти декілька підзаголовків рівня <h2>, потім – декілька підзаголовків рівня <h3> тощо. Така побудова коду забезпечує структуру тексту аналогічну до тексту із заголовками у книзі.

3.5. Нумеровані й марковані списки.

Існує 3 види списків, які можна розмістити на веб-сторінці: невпорядкований (маркований) список, впорядкований список, який ще називають нумерованим, а також список означень. Для створення невпорядкованого списку використовується тег <ul>, і кожен пункт списку виділяється тегом <li>. Впорядкований список, пункти якого виділяються арабськими чи латинськими цифрами або літерами, створюється за допомогою тегу <ol>. Як і для маркованого списку, кожен пункт нумерованого списку виділяється тегом <li>.

Списки означень складаються з двох частин: терміну та його визначення. Оголошується такий список тегом <dl>. Для вставки терміну застосовується тег <dt>, для вставки визначення −тег <dd>. Такий вид списків на веб-сторінках має вигляд словника і є найменш поширеним.

Приклади завдань:

Приклад 3.1.

```
<html>
  <head>
    <title>Мій перший сайт </title>
  </head>
  <body>
    <center>
    <h1>
      <font face="Times New Roman">Привіт світе!!!</font>
    </h1>
    </center>
      <p align="justify">
          <font size="+2"> Мене звати Карлсон. Це моя перша веб-сторінка. Тут я хочу знайти собі нових друзів. Я дуже сильно люблю <font size="+3" face="Arial"> варення! </font> З нетерпінням чекаю на запрошення на чай!
        </font>
    </p>
  </body>
</html>
```

Приклад 3.2.

```
<html>
  <head>
    <title>Стилі тексту</title>
  </head>
  <body>
    <big><b><u>Наукова стаття.</u></b></big>
    <br>
    <br>
     Повітря являє собою суміш газів азоту N<sub>2</sub> , частка якого складає 78 відсотків, кисню O<sub>2</sub>, який складає 21 відсоток, <strike> інших газів
```

</strike>, а також води H₂O при температурі 10^o за шкалою Цельсія.

*
*

Вперше дану пропорцію <s>предложил</s> запропонував у 1754 році <tt> Джозеф Блек.</tt>

*
*

*
*

*
*

<small>Концентрація водяної пари залежить від багатьох факторів.</small>

</body>

</html>

Приклад 3.3.

```
<html>
  <head>
    <title>Пробіли і перенесення</title>
  </head>
  <body>
    <pre>
     ПРИЧИННА

Реве та стогне Дніпр широкий,
Сердитий вітер завива,
Додолу верби гне високі,
Горами хвилю підійма.

       Т. Г. Шевченко.
    </pre>
  </body>
</html>
```

Приклад 3.4.

```
<html>
  <head>
    <title>Список означень</title>
  </head>
  <body>
    <dl>
      <dt> Бароко </dt>
          <dd> Архітектурний стиль, який вирізняється просторовим розмахом, плавністю й складним поєднанням криволінійних форм, злиттям об'ємів у динамічну масу, багату на скульптурний декор. </dd>
      <dt> Готика </dt>
          <dd> Тип сакральної і світської архітектури, розповсюдженої в період Середньовіччя, характерними тисами якої є витонченість, монументальність, спрямування вгору, багате декоративне оформлення </dd>
```

<dt> Ренесанс </dt>

<dd> Період розвитку архітектури в європейських країнах з початку XV до початку XVII століття, у якому особливе значення надавалося формам античної архітектури, тобто симетрії, пропорціонуванню, геометрії і порядку складових частин.

</dd>

</dl>

</body>

</html>

Контрольні питання

1. Наведіть загальний вигляд структури html-документа.
2. Розкажіть, яке призначення тега <font> та які параметри він може приймати.
3. Визначте, у яких тегах можна вставити заголовки на веб-сторінці.
4. Представте html-код, що задає виведення тексту у два рядки.
5. Опишіть загальний вигляд коду для порожньої веб-сторінки.
6. Перелічіть атрибути, що визначають тип вирівнювання.
7. Розкажіть, які види списків існують в html і якими тегами вони задаються.

Лекція 4. Робота з кольором та зображеннями на веб-сторінці.

План лекції:

4.1. Способи задання кольорів на веб-сторінці.
4.2. Заливка фону веб-сторінки кольором.
4.3. Формати зображень, які використовують в Інтернеті.
4.4. Вставка зображення на веб-сторінку.
4.5. Параметри та вирівнювання зображень.
4.6. Створення фонового зображення на веб-сторінці.

4.1. Способи задання кольорів на веб-сторінці.

Кольори на веб-сторінці можна задавати різними способами: використовуючи шістнадцяткове значення кольору у системі RGB або використовуючи константу кольору. Колір тексту на веб-сторінці можна задавати за допомогою відповідного значення атрибуту color, який вказується у відкриваючому тезі <font>. Інший спосіб встановлення кольору тексту для всієї веб-сторінки полягає у застосуванні атрибуту text всередині тегу <body>, у якому міститься основне наповнення веб-сторінки.

В мові HTML використовується своя палітра кольорів. Перерахуємо основні кольори:

Константа кольору	Код кольору	Колір
black	#000000	чорний
white	#FFFFFF	білий

yellow	#FFFF00	жовтий
silver	#C0C0C0	світло-сірий
red	#FF0000	червоний
green	#008000	зелений
gray	#808080	темно-сірий
blue	#0000FF	синій
navy	#000080	темно-синій
purple	#800080	фіолетовий
orange	#FFA500	помаранчевий
cyan	#00FFFF	бірюзовий

4.2. Заливка фону веб-сторінки кольором.

Колір фону веб-сторінки можна задавати за допомогою атрибуту bgcolor тегу <body>. Таким чином запис:

<body bgcolor="#40CAFF">

робить заливку фону всієї веб-сторінки блакитним.

У поєднанні з атрибутом text, який задає колір тексту на веб-сторінці, весь рядок коду виглядатиме наступним чином:

<body text=="#FFFF00" bgcolor="#40CAFF">

4.3. Формати зображень, які використовують в Інтернеті.

Веб-сторінка, на якій відсутні візуальні елементи, має небагато шансів зацікавити користувачів та заохотити їх провести на такому ресурсі якомога більше часу, що, своєю чергою, негативно впливатиме на розвиток сайту.

При підготовці зображень для розміщення на сайті слід враховувати декілька важливих чинників. Зображення повинно бути унікальним і не порушувати авторські права. Також якість зображень на веб-ресурсі має велике значення як для пошукових систем, так і для користувачів. Розмір графічних файлів, що розміщуються на веб-сторінках, повинен забезпечувати достатню якість зображень, але не бути завеликим. Використання зображень з великою вагою може значно збільшити час завантаження сторінки, на якій вони розміщені.

Важливо також, щоб формат графічного файлу дозволяв представляти його як зображення на веб-сторінці. В інтернеті можна розміщувати зображення у форматі JPG, GIF, PNG для растрової графіки та SVG для векторної.

4.4. Вставка зображення на веб-сторінку.

Зображення вставляються на веб-сторінку за допомогою одиночного тегу <img>. Всередині тегу <img> обов’язково повинен бути розміщений атрибут src із зазначенням місцезнаходження файлу із зображенням, що розміщується на веб-сторінці. Важливо, щоб шлях до графічного файлу із зображенням було прописано правильно із врахуванням усіх вкладених папок, всередині яких міститься файл. У випадку розміщення файлу із зовнішнього веб-ресурсу, необхідно зазначити його повну зовнішню адресу.

Наведемо низку прикладів, де вказано шлях до зображення:

<img src = "foto.jpg">

В даному прикладі foto.jpg − це назва графічного файлу. Шлях до цієї фотографії не вказується, оскільки файл з фото знаходиться в одній папці з html-документом.

<img src = "myfoto/foto.jpg">

Такий запис передбачає, що в директорії, де розміщено html-документ, є папка myfoto, у якій знаходиться файл foto.jpg.

<img src = "myfoto/graphics/foto.jpg">

Така адреса означає, що поруч з документом розміщена папка myfoto, у ній ще одна папка з назвою graphics, і вже в ній фотографія foto.jpg , яку потрібно викласти на сторінці.

<img src = "../foto.jpg">

Цей запис означає, що файл зображення розміщено на рівень вище від документа.

<img src = "../../foto.jpg">

Розміщення фотографії, відповідно, на два рівні вище. Отже, скільки поставите../ , на стільки рівнів і піднімитеся в ієрархічній структурі папок.

Вставка на своїй сторінці графічного файлу, який міститься на іншому інтернет-ресурсі, потребує запису повної URL-адреси з назвою протоколу включно.

<img src = "http://www.site.ua/foto/foto.jpg">

4.5. Параметри та вирівнювання зображень.

Ширина та висота зображення задається атрибутами width і height. Значення цих атрибутів вказується у пікселях. Якщо атрибути width та height не використовувати, тоді ширина і висота зображення за умовчанням буде рівною реальним його розмірам, без жодних спотворень.

Вирівнювання зображення на веб-сторінці здійснюється шляхом додавання відповідного значення до атрибуту align. Атрибути hspace та vspace дозволяють задати відстань від зображення до тексту по горизонталі та вертикалі відповідно. Рамка навколо зображення створюється за допомогою атрибуту border, у якому зазначається її товщина в пікселях. Колір рамки задає атрибут bordercolor.

Атрибути alt і title тегу <img> задають текст, який буде виводитися на місці, де має з'явитись зображення, до його завантаження або при відключеній графіці, а також якщо зображення завантажити не вдалося. Крім того, при наведенні курсору миші на зображення, текст, вказаний в параметрі title, можна побачити у якості тексту спливаючої підказки.

4.6. Створення фонового зображення на веб-сторінці.

Для того, щоб зображення зробити фоном веб-сторінки, використовується атрибут background відкриваючого тегу <body>:

<body background = "foto.jpg">

<body bgcolor = "#008000" background = "fon.jpg">

Поєднання атрибутів, що задають заливку фону кольором і встановлення фонового зображення, застосовують на випадок відключеного відображення графіки в браузері користувача або інших чинників, які можуть стати на заваді правильному завантаженню фонового зображення. За такої умови фон матиме вказане забарвлення, що попри все відповідатиме дизайнерському рішенню по оформленню веб-сторінки.

Приклади завдань:

Приклад 4.1.

<html>

<head>

```
    <title>Моя сторінка з фото</title>
  </head>
  <body text="#484800" background = "fon.jpg">
    <center>
      <h1> <font color="#008000"> Привіт світе!!!</font> </h1>
    </center>
    <p align="justify">
          <font size="+1" color="red"> Всім привіт! Це моя перша сторінка з фото</font>
        <img src="graphics/foto.jpg" align="top" width="180" height="240">
      <font size="-1" color="#800080"> Далі записуємо довільний текст…
      </font>
    </p>
  </body>
</html>
```

Контрольні питання

1. Визначте тег, яким задається колір тексту на веб-сторінці.
2. Поясніть, як вставити зображення на веб-сторінку.
3. Назвіть атрибути, що задають розміри зображення на веб-сторінці.
4. Згадайте, які графічні формати дозволяють розміщувати зображення на веб-сторінці.
5. Перелічіть можливі значення атрибуту align для вирівнювання зображення відповідно до тексту на веб-сторінці.

Лекція 5. Створення гіперпосилань на веб-сторінці.

План лекції:

5.1. Вставка текстових гіперпосилань.
5.2. Посилання через зображення.
5.3. Посилання на e-mail.
5.4. Закладки або якорі.
5.5. Динамічні ефекти.

5.1. Вставка текстових гіперпосилань.

Гіперпосилання – це вказівка для веб-браузера, до якого об'єкту в межах чи поза межами html-документа йому слід звернутися. За допомогою гіперпосилань користувачі можуть переходити з однієї веб-сторінки на іншу, завантажувати файли тощо. В якості гіперпосилання може бути оформлений фрагмент тексту або зображення. Коли веб-сторінка відображається у вікні веб-браузера, текстове посилання зазвичай виділене

синім кольором і підкреслюванням. Для переходу на сторінку, до якої веде посилання, користувачу достатньо натиснути кнопкою миші на текст посилання.

Для створення гіперпосилання необхідно використовувати теги <a> та в коді веб-сторінки, визначивши для відкриваючого тегу <a> атрибут href. Його значенням має бути вказаний шлях до html-файлу веб-сторінки, на яку веде посилання. Текст посилання розміщують між тегами <a> та .

Так само, як і у випадку зображень, важливо правильно прописувати місцезнаходження файлу html, на який відбувається перехід, із врахуванням усіх вкладених папок, всередині яких міститься файл. Якщо веб-сторінка, на яку вказує посилання, розміщена на іншому веб-сайті, то значенням атрибуту href повинна бути повна URL-адреса з назвою протоколу включно. Такі посилання називають зовнішніми. Якщо ж гіперпосилання вказує на іншу сторінку того ж сайту, тоді для пошуку документа достатньо задати лише відносний шлях. Таке посилання називають внутрішнім.

*Тут моя наступна сторінка *

Шлях посилання на документ, що відкривається, прописується таким самим чином, як і місцезнаходження файлу із зображенням, що розміщується на веб-сторінці.

5.2. Посилання через зображення.

У якості посилання може слугувати не лише текст, але й зображення. Для розміщення посилання через зображення всередині тегів <a>… слід записати тег <img>, в атрибуті src якого зазначити шлях до файлу із зображенням.

* <img src="kartynka.gif"> *

Принцип створення посилання через зображення такий самий, як і в текстовому гіперпосиланні, просто замість тексту всередині пари тегів <a>… записуємо тег для вставки зображення на веб-сторінку, яке потрібно зробити посиланням.

При переході за посиланням можна відкрити документ у новому вікні веб-браузера. Для цього застосовують атрибут target (ціль) і його значення _blank. Записується таке налаштування наступним чином:

* <img src="kartynka.gif"> *

5.3. Посилання на e-mail.

Якщо в атрибуті href задати адресу електронної пошти зі словом mailto перед нею, то після переходу за таким посиланням можна написати електронного листа, де в полі Кому буде записано зазначену адресу. Для того, щоб зробити текст або зображення посиланням на e-mail , його потрібно розмістити всередині парного тегу <a> з використанням вказівки mailto:

* Напишіть мені листа *

Окрім цього, у бланку листа можна заздалегідь прописати тему листа (?subject), текст повідомлення (&body), список електронних адрес, на які надсилаються копії листа (&cc) та список адрес прихованих копій листа. Адреси скриньок для надсилання копій та прихованих копій листа записуються через кому.

5.4. Закладки або якорі.

Гіперпосилання може вказувати на певне місце всередині сторінки, якщо туди заздалегідь вбудувати якір-мітку. Для визначення якоря також використовують теги <a> та , але замість атрибуту href задають атрибут name, значенням якого повинно бути ім'я

якоря. Воно може складатися з букв та цифр, але не повинно містити символів пробілу. Якщо на сторінці є декілька міток, то всі вони повинні мати різні назви.

Для створення посилання на встановлений якір у відкриваючому тезі <a> слід вказати ім'я якоря, відділивши його символом #. Символ # вказує на те, що після нього записано назву мітки, а не ім'я файлу. Посилання на мітку всередині поточного html-документа задають таким чином:

* Текст посилання*.

Щоб відвідувач сторінки, натиснувши на одне з посилань на розділи, перейшов одразу до потрібного місця в тексті, потрібно зробити дві речі. По-перше, слід присвоїти індивідуальне ім'я кожному розділу. Шукаємо в тексті потрібні розділи і робимо їх адресами посилань закладок, привласнюючи їм імена. До прикладу:

*Розділ1 *
*Розділ2 *
*Розділ3 *

Ім'я можна присвоїти будь-яке, не обов'язково glava1. По-друге, потрібно прописати посилання на якорі у змісті веб-сторінки. Зробимо це наступним чином:

* Розділ1*
* Розділ2*
* Розділ3*

Закладки використовують для навігації всередині сторінки, однак на них можна посилатися і з інших веб-сторінок або сайтів.

5.5. Динамічні ефекти.

Для створення найпростішого динамічного ефекту на веб-сторінці можна використовувати біжучий рядок. Розміщувати біжучий рядок можна в будь-якому місці веб-сторінки за допомогою парного тегу <marquee>, який змушує текст всередині рухатися в тому чи іншому напрямку. Біжучий рядок має низку налаштувань прокручування, які задаються атрибутами behavior, scrollamount, loop, direction, bgcolor, width і height.

Приклади завдань:

Приклад 5.1.

```
<html>
  <head>
    <title>Закладки</title>
  </head>
  <body>
    <h1>Т. Г. Шевченко</h1>
    <a href="#virsh1">Сон</a><br>
    <a href="#virsh2"> Гайдамаки </a><br>
    <a href="#virsh3"> Причинна</a><br>
    <h2><a name="virsh1">Сон</a></h2>
    <pre>
      У всякого своя доля
      І свій шлях широкий,
      ... ... ...
    </pre>
```

```
<h2><a name="virsh2"> Гайдамаки</a></h2>
<pre>
  Все йде, все минає – і краю немає.
  Куди ж воно ділось? відкіля взялось?
  ... ... ...
</pre>
<h2><a name="virsh3"> Причинна </a></h2>
<pre>
  Реве та стогне Дніпр широкий,
  Сердитий вітер завива,
  ... ... ...
</pre>
</body>
</html>
```

Приклад 5.2.

```
<html>
  <head>
    <title>Біжучий рядок</title>
  </head>
  <body>
    <p align="center"><h1> Біжучі рядки</h1></p>
    <marquee> Біжучий рядок за умовчанням </marquee>
    <marquee direction="right"> Біжучий рядок зліва направо </marquee>
      <marquee behavior="alternate"> Біжучий рядок рухається від краю до краю </marquee>
    <marquee scrollamount="10"> Біжучий рядок зі швидкістю 10 </marquee>
    <marquee scrollamount="1"> Біжучий рядок зі швидкістю 1 </marquee>
     <marquee direction="right" loop="2"> Цей рядок буде прокручуватися лише два рази </marquee>
    <marquee behavior="slide"> Біжучий рядок із зупинкою </marquee>
    <marquee bgcolor="#b40000"> Біжучий рядок з фоном </marquee>
      <marquee width=400> Біжучий рядок з обмеженням ширини прокручування </marquee>
    <marquee direction="up"> Біжучий рядок знизу вгору </marquee>
    <marquee hspace="300"> Біжучий рядок з відступами від границь </marquee>
  </body>
</html>
```

Контрольні питання

1. Опишіть, як вставити гіперпосилання на веб-сторінку.
2. Покажіть, яким чином графічне зображення зробити гіперпосиланням.
3. Вкажіть, за допомогою якого тегу задається мітка на веб-сторінці.
4. Перелічіть можливі види посилань у межах однієї веб-сторінки.
5. Дайте визначення, що таке якір і яким чином задати якір на веб-сторінці.

Лекція 6. Використання таблиць у HTML-документах.

План лекції:

6.1. Створення таблиць на веб-сторінках.
6.2. Встановлення ширини таблиць та комірок.
6.3. Форматування тексту та оформлення рамок таблиці.
6.4. Використання таблиць для розміщення об’єктів на веб-сторінці.

6.1. Створення таблиць на веб-сторінках.

Таблиця на веб-сторінці створюється за допомогою парного тегу <table>, що задає початок і кінець таблиці. Для виділення рядків та стовпців, з яких складається таблиця, існують теги <tr> і <td>. Кожен рядок таблиці оголошується тегом <tr>, який додатково може містити параметри align для горизонтального вирівнювання тексту в комірках таблиці та valign для вертикального вирівнювання. Стовпчики таблиці позначаються тегом <td> всередині кожного із рядків. Разом ці теги записуються наступним чином:

```
<table>
    <tr>
      <td> комірка</td>
    </tr>
</table>
```

Такий запис задає найменшу таблицю, в якій є всього один рядок, що містить один стовпчик, тобто, це одна комірка.

6.2. Встановлення ширини таблиць та комірок.

Ширину таблиці можна вказувати в параметрі width у пікселях або в процентах від розміру вікна веб-сторінки:

```
<table width="200">
<table width="100%">
```

Окрема комірка таблиці може мати свої власні значення параметрів вирівнювання align та valign, ширини width та висоти height, а також свій колір фону bgcolor. Крім того, всередині тегу <tr> можна вказувати атрибути colspan та rowspan. За допомогою цих атрибутів дві чи декілька комірок таблиці об’єднуються по горизонталі або вертикалі відповідно:

```
<td rowspan="3"> об’єднана комірка </td>
<td colspan="2"> об’єднана комірка </td>
```

При такому об’єднанні комірка розтягується на вказану кількість клітинок, але при цьому зсуває наступні комірки, які потрібно видалити.

6.3. Форматування тексту та оформлення рамок таблиці.

Тег <th> дозволяє вказати комірки із назвами стовпців таблиці. Вміст таких комірок виділяється напівжирним шрифтом і розміщується по центру. В усьому іншому тег <th> аналогічний тегові <td>.

Заголовок усієї таблиці створюється за допомогою тегу <caption>. Атрибут align цього тегу може приймати одне з двох значень: top, коли заголовок записується над таблицею, і bottom, коли заголовок розміщується під таблицею.

Рамка навколо таблиці створюється за допомогою атрибуту border тегу <table>. Цей атрибут також управляє відображенням ліній сітки таблиці. За умовчанням сітка всередині таблиці не відображається. Якщо записати тег <table> без зазначення атрибуту border або записати <table border="0">, сітка таблиці відображатися не буде. Якщо ж записати таким чином:

<table border = "5">

то цьому випадку сітка буде відображатися, а товщина рамки навколо таблиці дорівнюватиме 5 пікселям.

Товщину ліній сітки всередині таблиці можна регулювати, задаючи відповідну кількість пікселів як значення атрибуту cellspacing. По суті, цей атрибут вказує відстань між рамками сусідніх комірок. За умовчанням даний параметр має значення 2. Якщо атрибуту cellspacing присвоїти значення 0, то рамки суміжних комірок зливатимуться в одну лінію:

<table cellspacing = "0">

Атрибут cellpadding вказує розмір відступу між рамкою комірки і даними всередині неї. За умовчанням цей параметр має значення 1, але його можна змінити, збільшивши ширину полів:

<table cellpadding = "2">

6.4. Використання таблиць для розміщення об'єктів на веб-сторінці.

В html-документах таблиці використовуються або у явному вигляді як засіб представлення даних, або як елемент оформлення сторінки, за допомогою якого можна в потрібному місці розмістити на веб-сторінці текст і графіку. Таблична структура веб-сторінки передбачає вставку html-таблиці на всю ширину сторінки. В комірках цієї таблиці розміщується контент. Приклад табличного дизайну веб-сторінки:

Рис. 6.1. Частина веб-сторінки, структура якої створена на основі таблиці

Таким чином, візуально веб-сторінку можна розділити на необхідну кількість частин: шапка сайту, панель навігації, основний текст тощо. Табличний дизайн сайтів є дуже зручним і поширеним, особливо в середовищі розробників-початківців. Найбільшою перевагою такого підходу є його зрозумілість та можливість уникнути проблем сумісності.

Приклади завдань:

Приклад 6.1.

<html>

```
  <head>
    <title>Таблиця</title>
  </head>
  <body>
    <table border="1">
      <tr>
        <td>рядок1 комірка1</td>
        <td> рядок1 комірка2</td>
        <td>рядок1 комірка3</td>
      </tr>
      <tr>
        <td> рядок2 комірка1</td>
        <td>рядок2 комірка2</td>
        <td>рядок2 комірка3</td>
      </tr>
      <tr>
        <td>рядок3 комірка1</td>
        <td> рядок3 комірка2</td>
        <td> рядок3 комірка3</td>
      </tr>
    </table>
  </body>
</html>
```

Приклад 6.2.

```
<html>
  <head>
    <title>Таблиця</title>
  </head>
  <body>
    <table border="1">
      <tr>
        <td rowspan = "3"> рядок1 комірка1</td>
        <td> рядок1 комірка2</td>
        <td> рядок1 комірка3</td>
      </tr>
      <tr>
        <td> рядок2 комірка2</td>
        <td> рядок2 комірка3</td>
      </tr>
      <tr>
        <td colspan = "2"> рядок3 комірка2</td>
      </tr>
    </table>
  </body>
</html>
```

Контрольні питання

1. Розкрийте призначення тегу <table> та які атрибути він може приймати.
2. Назвіть теги, які формують у таблиці рядки, клітинки-заголовки і звичайні клітинки.
3. Згадайте, за допомогою якого тегу створюється заголовок таблиці.
4. Вкажіть, за допомогою яких атрибутів створюється рамка навколо таблиці.
5. Опишіть структуру html-документу при табличному дизайні веб-сторінок.
6. Поясніть, яким чином об'єднати декілька комірок таблиці в одну.
7. Вкажіть атрибут, який використовують для вирівнювання елементів в комірках таблиці.
8. Подумайте, як можна регулювати товщину ліній сітки всередині таблиці.

Лекція 7. Фрейми, їхні теги й атрибути. Карти зображень.

План лекції:

7.1. Описання фреймової структури.
7.2. Описання окремих областей. Тег <frame>.
7.3. Тег <noframes>.
7.4. Карта зображень як панель навігації.
7.5. Структура карти зображень. Тег <map>.

7.1. Описання фреймової структури.

Зазвичай для всіх сторінок сайту деякі їхні частини, такі як назва сайту зверху і бічна панель навігації, містять одну й ту саму інформацію. При переході між сторінками змінюється лише основне наповнення кожної сторінки. Відтак, залишаючи в незмінному стані вміст однакових частин, при переходах між сторінками сайту можна завантажувати лише наповнення основної області. Інструментом, який дає змогу реалізувати таке рішення, є створення фреймової структури.

Фрейми дозволяють розбити вікно веб-браузера на декілька прямокутних областей, в кожну з яких можна завантажити окремий html-файл. Отже, фрейми створені для того, щоб визначити робочі області для кожного html-документа в єдиному вікні веб-браузера.

Для опису фреймової структури потрібно використовувати парний тег <frameset>. Всередині тегів <frameset> та </frameset> можуть міститись теги <frame>, <iframe> або інший набір фреймів, описаний тегами <frameset> та </frameset>.

Код html-документу, що задає фреймову структуру, має наступний вигляд:

```
<html>
  <head>
    <title>Фрейми </title>
  </head>
```

<frameset>

……..

</frameset>

</html>

Відсутність тегу <body> в даному випадку компенсує тег <frameset>, який дає вказівку встановити фрейм або набір фреймів.

Тег <frameset> може мати атрибути rows та cols, що вказують браузеру, яким чином слід розміщувати фрейми у його вікні. Атрибут rows описує розбиття на горизонтальні фреймові області, атрибут cols – розбиття на вертикальні області. В якості значень параметрів rows та cols вказуються розміри фреймів. Розміри можуть бути зазначені в абсолютних одиницях (у пікселях) або у відсотках.

7.2. Описання окремих областей. Тег <frame>.

Тег <frame> або <iframe> описує окремий фрейм і містить обов'язковий атрибут src, що визначає шлях до html-файлу, який повинен бути завантажений у фреймову область. Використання фреймової структури передбачає одночасне відображення вмісту різних веб-сторінок, адреси яких прописуються у параметрах фрейму.

<frame src="doc1.html">

<iframe src="logotype.html">

Крім атрибуту src, тег <frame> має атрибути name, який задає унікальне ім'я фрейму, та scrolling, який регулює відображення смуг прокручування у фреймі.

Поля фреймів або, простіше кажучи, відстань від межі фрейму до тексту чи зображення, задаються в пікселях за допомогою атрибутів marginwidth (по горизонталі) та marginheight (по вертикалі) тегу <frame> або <iframe>.

Атрибут frameborder вмикає або вимикає відображення межі між фреймами. Цей атрибут може приймати одне з двох значень: 1 , коли межа відображається (це значення встановлено за умовчанням), або значення 0, коли межа не відображається.

При наведенні курсору на межу фрейму він набуває іншого вигляду, що дозволяє захопити і перемістити її, і тоді цю межу можна пересунути в будь-який бік, утримуючи лівою кнопкою миші. Іноді така мобільність межі фреймів слугує на користь розробника, але частіше все ж заважає гарному дизайну веб-сторінок. Для того, щоб заборонити користувачу змінювати розміри вікон, для тегу <frame> застосовують атрибут noresize.

7.3. Тег <noframes>.

Деякі браузери не підтримують фреймову структуру документа або неправильно її інтерпретують, крім того, досить часто користувачі в налаштуваннях своїх браузерів навмисно відключають підтримку фреймової структури html-документу. Для того, щоб дати зрозуміти користувачу, що його браузер або налаштування браузера не підтримують фрейми, існує тег <noframes>.

Тег <noframes> виводить розміщений в ньому текст на екран у тому випадку, коли браузер користувача не підтримує фрейми або вони примусово вимкнені у його налаштуваннях. Якщо ж фрейми підтримуються браузером користувача, то даний тег просто ігнорується.

<noframes>

* Ваш веб-браузер не відображає фрейми *

</noframes>

Результат цього прикладу буде помітний, якщо браузер дійсно не підтримує фрейми або ж ви у якості експерименту вимкнете підтримку фреймів у своєму браузері. Тег <noframes> повинен бути розміщений всередині тегу <frameset>.

7.4. Карта зображень як панель навігації.

Карта зображень – це графічна панель на веб-сторінці з активними областями, через які можна здійснювати переходи на інші сторінки. Карти зображень створюються як навігаційні карти сайту. При цьому зображенню на веб-сторінці за допомогою атрибуту usemap тегу <img> присвоюється індивідуальне ім'я. Навігація через зображення визначається тегом <map>, у якого в атрибуті name слід вказати назву зображення. Активні області на карті зображень описуються тегом <area>, у параметрах якого задається форма кожної області, її координати, адреса файлу, на який здійснюється перехід через цю область. Карти зображень дозволяють реалізувати творчий дизайнерський підхід до створення навігаційної панелі сайту.

7.5. Структура карти зображень. Тег <map>.

Код для вставки карти зображень на веб-сторінку складається з двох частин. Перша частина за допомогою тегу <img> вставляє зображення на веб-сторінку. Атрибут usemap вказує, що зображення є картою. В якості значення параметру вказується URL-адреса опису конфігурації. Якщо опис карти розміщено в тому ж html-документі, тоді вказується назва розділу конфігурації карти, перед яким додається символ "#".

<img src="karta.jpg" usemap="#panel">

Друга частина, що являє собою опис конфігурації карти, розміщена між тегами <map> та </map>. Парний тег <map> слугує для опису конфігурації областей карти зображень. У тегу <map> є єдиний параметр – атрибут name. Значення параметру name повинно відповідати імені в параметрі usemap тегу <img>:

<map name = "panel">
…….
</map>

Активна область карти зображень описується за допомогою тегу <area>. Скільки на карті активних областей, стільки й тегів <area> повинно бути. Якщо одна активна область перекриває іншу, тоді буде реалізоване перше посилання зі списку областей.

Приклади завдань:

Приклад 7.1.

```
<html>
  <head>
    <title>Фрейми</title>
  </head>
  <frameset rows="20%,80%">
    <frame src="logotype.html">
    <frameset cols="30%,70%">
      <frame src="menu.html">
      <frame src="text.html">
    </frameset>
  </frameset>
</html>
```

Приклад 7.2.

```
<html>
  <head>
    <title>Фрейми</title>
  </head>
  <frameset cols="*, 800, *">
    <frame src="dekor.html" scrolling="no" noresize>
    <frameset rows="120, *">
        <frame src="logotype.html" scrolling="no" marginwidth="0" marginheight="0"
noresize>
      <frameset cols="200, 600">
        <frame src="menu.html" noresize>
        <frame src="text.html" marginwidth="10" marginheight="10" noresize>
      </frameset>
    </frameset>
    <frame src="dekor.html" scrolling="no" noresize>
  </frameset>
</html>
```

Приклад 7.3.

```
<html>
  <head>
    <title>Навігаційна карта</title>
  </head>
  <body>
    <center>
      <img src="karta.jpg" usemap="#panel">
    </center>
    <map name="panel">
      <area href="pryklad1.html" shape="rect" coords="15, 15, 82, 82">
      <area href="pryklad2.html" shape="circle" coords="60, 60, 30" alt="Підказка">
      <area href="pryklad3.html" shape="poly" coords="10, 100, 60, 10, 100, 100"
alt="Підказка 2">
    </map>
  </body>
</html>
```

Контрольні питання

1. Назвіть тег, який здійснює розподіл фреймової структури.
2. Вкажіть, який тег та які атрибути описують окремі фрейми.
3. Поясніть, як створюються складні конфігурації фреймів.
4. Розкрийте, як представити зображення у вигляді навігаційної карти.
5. Опишіть, яку будову має тег-контейнер <map>.

6. Згадайте, яке призначення має параметр name тегу <map>.
7. Розкрийте, яке призначення має тег-контейнер <noframes>…</noframes>.

Лекція 8. Форми в HTML.

План лекції:

8.1. Додавання форми в документ.
8.2. Збирання даних за допомогою форм.
8.3. Опис елементів управління.
8.4. Приклад заповнення форми.

8.1. Додавання форми в документ.

Форма на веб-сторінці представляє собою документ, створений за допомогою елементів HTML. Призначенням форми є збір інформації від користувачів. Для створення форми в html-документі застосовується парний тег <form>…</form>. Даний тег встановлює межі використання інших елементів, розміщених всередині форми.

Вміст форми на веб-сторінці визначається послідовністю елементів <input>, розміщених всередині пари <form> та </form>.

```
<form>
<p> Вкажіть E-mail</p><input type="email" name="email">
<p>Пароль</p><input type="password" name="password">
<p>Введіть пароль ще раз:</p><input type="password" name="password">
<input type="submit">
</form>
```

При цьому пару тегів <form> та </form> не слід записувати для кожного окремого елемента форми. Форма потрібна, щоб зберегти певний набір даних про користувача. Розбиваючи єдину форму на окремі частини, вона не виконуватиме свою основну функцію.

8.2. Збирання даних за допомогою форм.

Після того, як користувач заповнить форму і запустить процес її обробки, інформація з неї потрапляє в програму, що працює на сервері. Таким чином, користувач може інтерактивно взаємодіяти з веб-сервером через Інтернет. Елемент форми використовує як метод, так і дію для опису обробки формою даних, що вводяться користувачем. Метод GET або POST визначає, як повинні оброблятися вхідні дані з форми. Дія action вказує на URL програми, яка відповідає за обробку даних. Після обробки веб-сервером дані, передані через форму, записуються у базі даних сайту, де було розміщено форму.

8.3. Опис елементів управління.

Для визначення області всередині форми, де здійснюється збір даних, використовується елемент <input>. Цей елемент представляє собою поле для введення інформації користувачем. В атрибуті type тегу <input> слід вказати вид елементу форми, який потрібно вставити на веб-сторінку.

Значення text дозволяє створювати однорядкове текстове поле:

```
<input type ="text" name="fullName">
```

Значення password відповідає за поле для введення паролю:

<input type ="password" name="password" minlength ="5" maxlength ="5">

Для вставки елементів-прапорців у формах можна використовувати значення checkbox атрибуту type. Такий елемент передбачає можливість відмітити декілька пунктів в переліку значень. Об'єднання елементів checkbox у спільну групу здійснюється шляхом вказівки однакових значень в атрибуті name.

* Оберіть Вашу улюблену страву: *
<form>
*<input type = "checkbox" name = "food" value ="Вареники"> Вареники
*
*<input type ="checkbox" name ="food" value ="Голубці"> Голубці
*
*<input type ="checkbox" name ="food" value ="Котлети" checked> Котлети
*
</form>

Вибір лише одного варіанту з кількох можливих здійснюється через елементи-перемикачі, які задаються за допомогою компоненти radio.

* Вкажіть Ваш улюблений десерт: *
<form>
*<input type ="radio" name = "dessert" value="morozivo"> Морозиво
*
*<input type ="radio" name = "dessert" value="tort"> Торт
*
*<input type ="radio" name = "dessert" value="tistechka"> Тістечка
*
</form>

Створення кнопки, яка повертає початкові значення полів усіх елементів форми, відбувається шляхом додавання елементу reset.

<input type ="reset" value ="Відміна">

Кнопка submit використовується при завершенні введення користувачем інформації і відправці даних на сервер.

<input type ="submit" value ="Надіслати">

Багаторядкове текстове поле, яке передбачає введення великої кількості друкованої інформації у декілька рядків в межах форми, створюється за допомогою парного тегу <textarea>.

<textarea name = "comment" rows="7" cols="12" placeholder = "Поділіться своєю думкою"> </textarea>

Для організації списків з прокручуванням та випадаючих меню можна використовувати тег <select>. Визначення пунктів такого списку відбувається через теги <option> всередині конструкції <select>… </select>:

<form>
<select name ="Фрукти" size =3 >
<option value="Apricot"> Абрикоси
<option value="Pear"> Груші
<option value="Lemon_and_orange">Лимони й апельсини
<option value="Apple" selected> Яблука
</select>
</form>

Після того, як користувач заповнить форму і запускає процес її обробки, інформація з неї потрапляє в програму, що працює на сервері.

8.4. Приклад заповнення форми.

Наступний приклад містить більшість описаних вище елементів форми.

```
<html>
  <head>
    <title>Форма</title>
  </head>
  <body>
    <form>
    <h1> Анкета реєстрації</h1>
    <table cellpadding ="15px" align="center">
      <tr>
        <td> Введіть Ваш логін </td>
          <td> <input type = "text" id = "login" name = "login" placeholder = "Логін"
title = "Введіть Ваш логін"> </td>
      </tr>
      <tr>
        <td> Введіть Ваш пароль </td>
          <td> <input type = "password" name = "password" placeholder = "Пароль"
title = "Введіть Ваш пароль" maxlength = "5" value = "5"> </td>
      </tr>
      <tr>
        <td> Підписатись на поштову розсилку? </td>
        <td> <input type ="radio" name="choice" id="yes" checked>
             <label for="yes">Так</label>
             <input type="radio" name="choice" id="no">
             <label for="no">Ні</label>
        </td>
      </tr>
      <tr>
        <td> Які рецепти Вам рекомендувати?
             <p>(Цей вибір впливає на теги за якими Вам будуть пропонувати
страви*)
          </p>
        </td>
        <td align="left">
             <input type="checkbox" name="recipes"> Десерти <br>
             <input type="checkbox" name="recipes"> Гарніри/Основні страви<br>
             <input type="checkbox" name="recipes"> Вегетеріанське<br>
             <input type="checkbox" name="recipes"> Дієтичне
        </td>
      </tr>
      <tr>
        <td> Оберіть складність рецептів </td>
        <td> <select name ="recipes levels" size =1 >
             <option> Я новачок (можу зробити бутерброди)
             <option> Час від часу готую
             <option> Часто готую і маю базові знання
             <option selected > Я Майстер-Шеф
```

```
                    </select>
                </td>
            </tr>
            <tr>
                <td> Додайте відгук будь ласка як ви знайшли наш сайт </td>
                    <td> <textarea name ="commentary" cols="40" rows="5" placeholder="Ваш
коментар"> </textarea>
                </td>
            </tr>
            <tr>
                <td colspan="2"> <a href="file1.html">₊☆·₊˚ﻉ˚₊☆·₊</a> </td>
            </tr>
            <tr align="center">
                <td colspan="2" >
                    <input type="submit" value="Відправити">
                    <input type="reset" value="Скинути">
                </td>
            </tr>
        </table>
    </form>
</body>
</html>
```

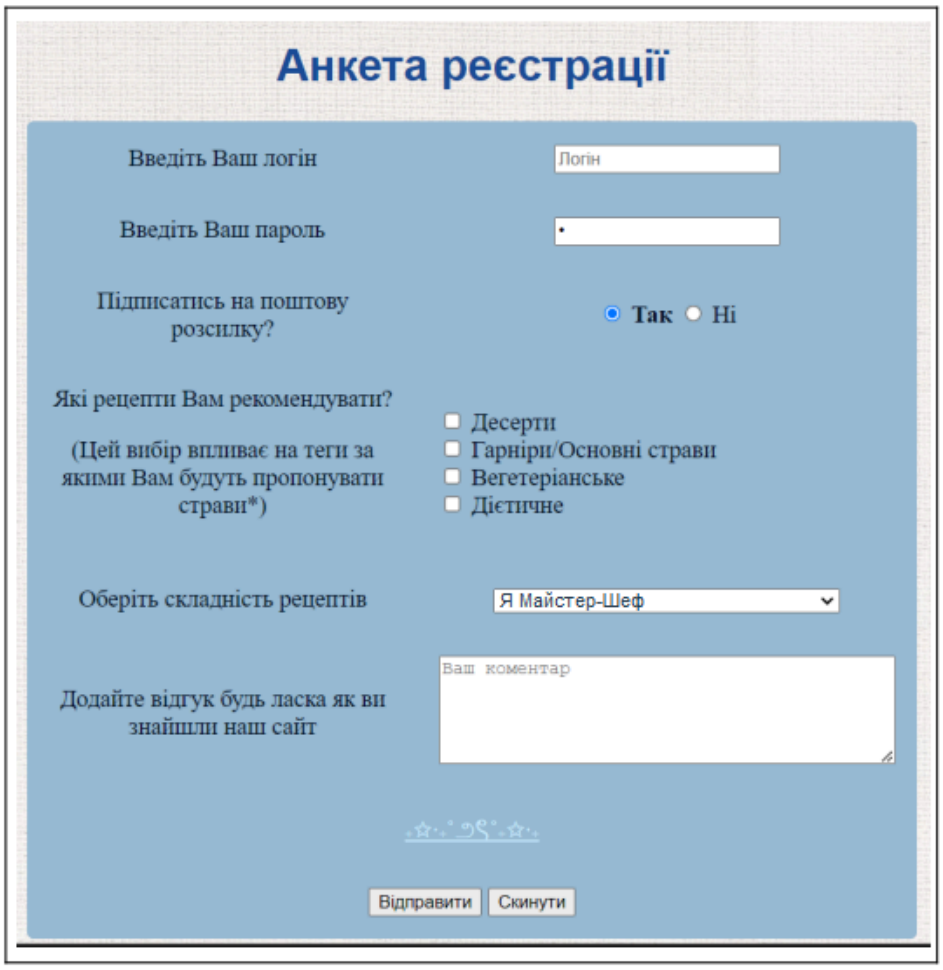

Рис. 8.1. Реалізація прикладу розміщення форми на веб-сторінці

Контрольні питання

1. Розкажіть, для чого використовуються форми.
2. Згадайте, за допомогою яких тегів створюються форми.
3. Назвіть спосіб застосування різних елементів форми.
4. Поясніть, як створити текстове поле для введення інформації.
5. Розкрийте, як у випадаючих списках встановити значення за умовчанням.
6. Опишіть спосіб зберігання даних, введених у форму.
7. Вкажіть, як розмістити прапорці та перемикачі у веб-формі.
8. Дайте визначення, що собою являють елементи checkbox та radio.
9. Поясніть, яким чином об’єднати декілька елементів прапорців або перемикачів в одну групу.
10. Розкрийте призначення кнопок reset і submit.
11. Перелічіть відомі вам атрибути тегу <textarea>.

Лекція 9. Використання таблиць каскадних стилів у HTML-документах.

План лекції:

9.1. Поняття про таблиці каскадних стилів.
9.2. Способи зв'язку каскадних стилів з HTML-документом.
  9.2.1. Вкладення визначення стилю в тег.
  9.2.2. Розташування опису стилів у заголовку HTML-документу.
  9.2.3. Підключення зовнішньої таблиці стилів.
9.3. Пріоритет застосування стилів.
9.4. Форматування шрифту.
9.5. Форматування тексту.

9.1. Поняття про таблиці каскадних стилів.

В стандартному HTML-документі для присвоєння якому-небудь елементу певних властивостей (колір, розмір, розміщення на сторінці тощо) доводиться щоразу описувати відповідні атрибути. Набагато зручніше було б довільно форматувати елементи веб-сторінки, один раз задавши стиль і потім лише використовуючи його. Досягнути подібного ефекту при створенні веб-сторінок дозволяють таблиці каскадних стилів, які додають до звичайних HTML-документів.

Каскадні таблиці стилів CSS (Cascading Style Sheets) – це набір правил оформлення та форматування, який може бути застосовано до різних елементів веб-сторінки. Використовуючи CSS, можна описати властивості елементів та визначити цей опис як стиль, і в подальшому вказувати, що елемент, який ви бажаєте оформити певним чином, повинен прийняти властивості вже описаного стилю.

Властивість стилю – це параметр стильового оформлення документу, який визначається специфікацією CSS. Елемент, якому належить визначений стиль, узагальнено називається селектором. Селектори поділяються на декілька типів: селектори тегів, ідентифікатори та класи. При заданні стилю після запису селектора йдуть фігурні дужки, в яких зазначається потрібна стильова властивість, а її значення вказується після двокрапки. У якості селектора може виступати будь-який тег HTML, для якого визначаються правила форматування, такі як колір, фон, розмір тощо.

Таким чином, в наборі правил CSS спочатку вказується ім'я тегу, оформлення якого буде перевизначено, без дужок, великими чи малими буквами. Потім всередині фігурних дужок записується властивість CSS, а після двокрапки – її значення.

```
Селектор
{
властивість1: значення1;
властивість2: значення2;
властивість3: значення3 значення4;
}
```

Набір параметрів розділяється між собою крапкою з комою і може розміщуватись як в один рядок, так і в декілька. Таблиці стилів – це, фактично, набори CSS-правил, які задають властивості форматування елементів веб-сторінки.

Таблиці стилів можуть містити текстові коментарі подібно до тих, які розміщуються в html-документах (<!-- коментар html -->). Проте в CSS коментарі мають інший синтаксис: початок коментаря позначається як слеш із зірочкою праворуч (/*), а завершення коментаря – зірочка зі слешем зліва (*/). Між ними розміщується довільне текстове повідомлення (наприклад: /* Це коментар в CSS*/).

9.2. Способи зв'язку каскадних стилів з HTML-документом.

Стилі CSS можуть включатися в html-документ 3 різними способами. Бувають внутрішні стилі, стилі рівня документу і зовнішні стилі.

9.2.1. Вкладення визначення стилю в тег.

Внутрішній стиль записується в атрибуті style та застосовується лише до вмісту вказаного тегу. Значення атрибуту style обов'язково роміщують у лапках, до того ж властивості відділяються одна від іншої крапкою з комою. Загальний вигляд синтаксису тегу з атрибутом style має наступний вигляд:

<тег style=" найменування властивості: значення">

Внутрішній стиль має більш високий пріоритет, аніж зовнішні стилі та стилі рівня документу, проте такий спосіб задання стилю не відповідає принципу розділення змісту веб-сторінки та її оформлення.

9.2.2. Розташування опису стилів у заголовку HTML-документу.

Стилі рівня документу застосовуються до всіх однакових елементів на веб-сторінці. Такі таблиці стилів записуються всередині тегу <style>…</style>, який вкладається в тег <head>…</head> у структурі коду HTML. Синтаксис тегів <style>…</style> при розміщенні інформації про стиль має загальний вигляд:

```
<style type ="text/css">
  Опис таблиці стилів
</style>
```

Такий спосіб задання стилів використовується, коли потрібно застосувати однакові стилі одразу до багатьох тегів в одному документі.

9.2.3. Підключення зовнішньої таблиці стилів.

У зовнішніх файлах потрібно зберігати стилі, спільні для всього сайту. Вони впливають одразу на велику кількість тегів у багатьох документах. Це виявляється дуже зручним, якщо сайт містить багато сторінок. При застосуванні зовнішньої таблиці стилів опис селекторів та їхніх властивостей розміщується в окремому файлі з розширенням .css, а для того, щоб зв'язати html-документ з цим файлом, використовується тег <link>. Даний тег поміщається в контейнер <head>…</head>.

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

Значення атрибутів rel та type тегу <link> залишаються незмінними, атрибут href задає шлях до CSS-файлу зі стилями. Використання зовнішнього файлу зі стилями дозволяє спростити задачу редагування стилів для великих сайтів. Якщо усі сторінки підключають один і той самий зовнішній стиль CSS, достатньо в ньому задати нове значення властивості для певного виду елементів на всіх сторінках. Інакше довелося б редагувати кожну сторінку окремо.

9.3. Пріоритет застосування стилів.

Припустимо, що для тексту визначено властивість color в атрибуті style одного кольору, в тегах <style> – іншого кольору, а в окремому файлі – третього кольору. Крім того, у параметрі color тегу <font> встановлено четвертий колір. Розглянемо приклад:

```
<html>
  <head>
    <title>Пріоритет застосування стилів</title>
    <link rel="stylesheet" type="text/css" href="style.css">
    <style type="text/css">
      p { color: blue }
    </style>
  </head>
  <body>
    <p style="color: green"><font color="yellow">Текст1</font></p>
    <br>
    <p style="color: green">Текст2</p>
    <p> Текст 3</p>
  </body>
</html>
```

Вміст зовнішнього файлу зі стилями style.css матиме лише один запис:

```
p { color: red }
```

Виникає питання: якого кольору буде абзац із Текстом 1? І якого кольору буде абзац із Текстом 2? Для відповіді на ці питання існує правило каскадування, що задає пріоритет застосування стилів:

- Стиль, заданий таблицею стилів, буде скасовано, якщо в html-коді явно описано атрибути форматування засобами html.
- Стиль, заданий у тегах <style>, буде скасовано, якщо в атрибуті style тегу html вказано інший стиль.
- Стиль, заданий в окремому файлі, буде скасовано, якщо в тезі <style> вказано інше визначення стилю.

Саме через таку структуру пріоритетів таблиці стилів називають каскадними.

Іншими словами, найменший пріоритет має стиль, описаний в окремому файлі, а найвищий пріоритет – стиль, вказаний атрибутом. У представленому прикладі, до параграфу зі словом "Текст1" буде застосовано форматування, вказане властивістю color тегу <font>, тобто абзац буде жовтого кольору. А параграф зі словом "Текст2" матиме колір, вказаний в атрибуті style, тобто зелений. В параграфі із вмістом "Текст3" колір буде синім, як це вказано в стилі рівня документу.

За допомогою властивості !important можна змінити пріоритет стилю. Наприклад, змінимо вміст файлу style.css таким чином:

```
p { color: red !important }
```

В результаті було перевизначено всі стилі, і параграф зі словом "Текст2" матиме червоний колір, а не зелений. Однак абзац зі словом "Текст1" так і залишиться жовтого кольору, оскільки колір було вказано у властивості color тегу <font>, а не задано через стиль.

9.4. Форматування шрифту.

Каскадні таблиці стилів CSS дозволяють керувати властивостями шрифту і тексту. Властивість font-family задає тип шрифту. Можна вказати декілька значень назви шрифтів через кому. На комп'ютері користувача браузер перевірить перший шрифт зі списку: якщо шрифт встановлено в бібліотеці шрифтів, тоді його буде застосовано, якщо ні – браузер перейде до наступного вказаного шрифту. Останнім у списку шрифтів зазвичай вказується загальний тип родини шрифтів: serif (шрифти із зарубками), sans-serif (рублені шрифти), cursive (курсивні шрифти), fantasy (декоративні шрифти) або monospace (моноширинні шрифти). Приклад:

font-family: Georgia, 'Times New Roman', serif

Якщо на комп'ютері користувача встановлено шрифт Georgia, тоді буде використовуватись він, якщо немає – тоді Times New Roman. Якщо ж і Times New Roman відсутній, тоді браузер буде використовувати шрифт із зарубками, який встановлено на комп'ютері.

Властивість font-size вказує розмір шрифту. Може задаватися у вигляді абсолютного значення в пунктах (pt) або пікселях (px), а також відносною величиною – у відсотках (%) або в em. Приклад:

font-size: 12pt

font-size: 150%

Властивість font-style задає накреслення тексту: normal (звичайне), italic (курсивне) або oblique (нахилене). Рівень жирності тексту регулюється властивістю font-weight, яка дозволяє приймати значення normal (звичайний) та bold (напівжирний текст). Останнє значення діє аналогічно тегові <b>.

9.5. Форматування тексту.

Властивість color задає колір тексту. Наприклад, задати червоний колір для усіх видів заголовків можна таким чином:

h1, h2, h3, h4, h5, h6 {color: #ff0000}

h1, h2, h3, h4, h5, h6 {color: red}

Міжрядковий інтервал, який вказує відстань між рядками тексту, задає властивість line-height. Значення може вказуватися числом як множник від поточного розміру шрифту, у відсотках, а також у пунктах (pt), пікселях (px) та інших одиницях вимірювання CSS:

line-height: 1.5;

Властивість text-decoration задає спосіб оформлення тексту. Варіанти значень: line-through (перекреслений), overline (лінія над текстом), underline (підкреслювання), none (відключення ефектів). Відключити підкреслювання у посилань можна, вказавши CSS-правило:

a {text-decoration: none}

За вирівнювання тексту на веб-сторінці відповідає властивість text-align, а властивість text-indent задає відступ першого рядка («червоний рядок»). Властивості font-style, font-variant, font-weight, font-size, font-family та lineheight можна задати в одному правилі. При форматуванні будь-якого тексту засобами CSS значення font-size та font-family є обов'язковими, інші властивості можна не вказувати.

Контрольні питання

1. Опишіть особливості застосування таблиць каскадних стилів.

2. Розкрийте сутність основної концепції CSS.
3. Вкажіть способи зв'язку каскадних стилів з html-документом.
4. Перелічіть одиниці вимірювання у CSS.
5. Вкажіть псевдо-клас, що дозволяє оформити посилання, на які наведений курсор миші.

Лекція 10. CSS-властивості: рамки, списки, фон, позиціонування.

План лекції:

10.1. Стиль, товщина та колір лінії рамки.
10.2. CSS-властивості списків.
10.3. Фон елементу. Колір фону. Фонове зображення.
10.4. Внутрішні та зовнішні відступи.
10.5. Форматування блоків. Позиціонування.

10.1. Стиль, товщина та колір лінії рамки.

В CSS існує універсальна властивість для задання всіх індивідуальних особливостей оформлення межі елементів – це властивість border. Для опису всіх властивостей рамки в одному правилі можна використовувати визначення border-width, що задає ширину рамки, border-style, яка задає стиль рамки та border-color, яка встановлює її колір. Загальна конструкція виглядатиме наступним чином:

border: border-width border-style border-color

Наприклад:

```
td { border-left-style: none;
     border-right-style: none;
     border-top-style: none;
     border-bottom-style: none;
     text-align: center }
caption { border-top-style: solid }
```

Можливими значеннями властивості border-width є: thin (тонка лінія), medium (середня – за замовчуванням), thick (товста), а також стандартні значення ширини. Можна також встановлювати значення ширини рамки для певної сторони. Для цього використовуються властивості border-top-width, border-right-width, border-bottom-width, border-left-width з аналогічними значеннями.

Властивість border-style задає стиль рамки. Її варіантами значень є: none (відсутність стилю, що є значенням за замовчуванням), hidden (прихована рамка), dotted (пунктир), solid (суцільна рамка), double (подвійна рамка), dashed (штрих-пунктир), groove (подвійна борозна), ridge (гребінь), inset (врізання), outset (орнамент). Можна також

встановити значення стилю рамки для певної сторони. Для цього використовуються властивості border-top-style, border-right-style, border-bottom-style, border-left-style.

Властивість border-color визначає колір рамки елемента веб-сторінки. Значення кольору встановлюється за правилами, що відповідають стандарту: шістнадцятковим значенням кольору у системі RGB або з використанням константи кольору. Можна також вказати значення кольору рамки для певної сторони окремо. Для цього використовуються властивості border-top-color, border-right-color, border-bottom-color, border-left-color.

10.2. CSS-властивості списків.

В HTML є три основних типи списків: невпорядковані списки <ul>, коли елементи списку помічені маркерами; впорядковані списки <ol>, у яких елементи списку помічені цифрами або буквами, та списки означень <dl>, що складаються з терміну та його визначення.

Каскадні таблиці стилів CSS дозволяють встановити різні маркери елементів списку. Тип маркера елемента списку визначає властивість list-style-type. В залежності від значень, ця властивість може задати, наприклад, квадратні або круглі маркери для невпорядкованого списку, або числа, букви чи римські цифри для впорядкованого списку.

Властивість list-style-position встановлює, чи буде з'являтися маркер всередині пунктів списку або ззовні перед початком кожного пункту. Значення цієї властивості за умовчанням – outside, що зумовлює маркери знаходитися попереду пунктів списку. При встановленні значення цієї властивості на inside, маркери будуть знаходитися всередині рядків.

Властивість list-style-image дозволяє використовувати власне зображення у якості маркера списку. Однак ця властивість дещо обмежена з точки зору управління положенням, розмірами та іншими характеристиками маркерів.

Зазначені властивості списків можуть бути вказані одразу, використовуючи лише один короткий запис властивості list-style. Розглянемо приклад задання стилю списку з перерахуванням усіх трьох властивостей CSS:

```
ul {
  list-style-type: square;
  list-style-image: url(punkt.png);
  list-style-position: inside;
}
```

Даний запис може бути замінено наступним:

```
ul {
  list-style: square url(punkt.png) inside;
}
```

Значення при цьому можуть бути перераховані у довільному порядку. Дозволено використовувати одне, два або усі три значення. За умовчанням значення властивостей list-style-type, list-style-image та list-style-position встановлені disc, none та outside. Якщо буде вказано одразу і тип маркера, і зображення у якості маркера, тип використовуватиметься у якості запасного варіанту, якщо зображення з якоїсь причини не може бути завантажено.

10.3. Фон елементу. Колір фону. Фонове зображення.

Як і в HTML, у CSS фоном веб-сторінки або інших її елементів слугує заливка кольором або зображення. Фонове зображення може повторюватись.

Властивість background-color встановлює колір фону веб-сторінки або якогось конкретного елементу. Розглянемо приклад:

td.head {background-color: #ffff00}

Фонове зображення на веб-сторінці встановлюється за допомогою властивості background-image. Важливо правильно прописувати адресу графічного файлу із зображенням та використовувати зображення у форматі JPG, GIF, PNG для растрової графіки або SVG для векторної. Розглянемо приклад:

body {background-image: url(images/pryklad.jpg)}

Окрім зазначених, в CSS існують додаткові властивості для встановлення характеристик фонового зображення. Властивість background-attachment задає поведінку фонового зображення при прокрутці. За умовчанням його значенням є scroll, яке означає, що фон прокручується разом зі змістом веб-сторінки. Значення fixed властивості background-attachment робить фон веб-сторінки непорушним.

Властивість background-position задає початкове положення фонового зображення по горизонталі (left, center, right) і вертикалі (top, center, bottom). Замість ключових слів можна вказувати відстань в пікселях або відсотках.

Для фонового зображення невеликого розміру, яке займає лише частину веб-сторінки, є можливість задати його повторення. Властивість background-repeat вказує, в якому напрямку повинно повторюватись фонове зображення. Можливі значення цієї властивості: repeat – повторення по горизонталі і вертикалі (це значення за умовчанням), repeat-x – повторення лише по горизонталі; repeat-y – повторення лише по вертикалі; значення no-repeat дає вказівку відключити повторення.

Приклад CSS коду, де використовуються різні властивості фонового зображення:

```
body {
background-image: url('pryklad.png');
background-repeat: repeat-x;
background-position: 120px 180px;
}
```

10.4. Внутрішні та зовнішні відступи.

Будь-який елемент веб-сторінки займає у вікні веб-браузера деяку прямокутну область. Причому ця область має як внутрішні, так і зовнішні відступи. Внутрішній відступ – це відстань між контентом всередині та реальною або уявною межею області. Зовнішній відступ – це відстань між реальною або уявною межею та іншим елементом веб-сторінки, точніше сказати, між межею і крайньою точкою зовнішнього відступу іншого елемента веб-сторінки.

За допомогою атрибутів margin-left, margin-right, margin-top і margin-bottom можна встановити відступи одного елемента веб-сторінки від іншого. При цьому дозволено задавати абсолютне чи відносне значення. Крім того, атрибути відступів можуть мати

від'ємні значення. Атрибути margin-top, margin-right, margin-bottom, margin-left позначають ширину верхнього, правого, нижнього та лівого зовнішнього поля. Значення за умовчанням у них 0. В якості значень вказують відстані у стандартних одиницях вимірювання в CSS або відсотковий показник, який позначає відношення ширини поля до ширини елемента.

body { margin-left: 0; margin-right: 0; margin-top: 0; margin-bottom: 0}

За допомогою атрибутів padding-left, padding-right, padding-top і padding-bottom можна встановити ширину відстані між вмістом елемента веб-сторінки та певною областю його межі або його рамки, якщо вона є. Наприклад, ці атрибути задають відстань між текстом та рамкою комірки таблиці.

td { padding-bottom: 20px }

td { padding-right: 50px }

Ширину зовнішніх полів для всіх сторін елемента веб-сторінки можна вказати у загальній властивості margin. Ця властивість може мати від 1 до 4 значень. У випадку запису одного значення, воно присвоюється всім полям. При вказівці двох значень перше присвоюється верхньому та нижньому полю, а друге – лівому та правому. При трьох значеннях перше відповідатиме верхньому полю, друге – лівому та правому, а третє – нижньому. У прикладі показано задання властивості margin, коли для кожної сторони вказані різні значення у різних одиницях вимірювання:

body { margin: 15mm 5% 20px 0 }

Ширина внутрішнього поля для всіх сторін елемента веб-сторінки може бути вказана у загальній властивості padding. У цієї властивості також може бути від 1 до 4 значень. Одне значення привласнюється внутрішнім полям з усіх сторін. Для двох значень перше привласнюється верхньому й нижньому внутрішньому відступу, а друге – лівому й правому. При трьох значеннях перше позначає верхнє поле, друге відповідає лівому й правому, а третє – нижньому. Розглянемо приклад:

td { padding: 15mm 50px 20px 0 }

До появи останніх стандартів CSS для обтікання текстом зображень користувалися атрибутом align елемента img зі значеннями left і right. Зараз стандартизована властивість float зі значеннями left і right може бути застосована не тільки до зображень, але й до інших елементів HTML. У комбінації з властивістю float часто застосовують властивість clear зі значеннями left, right або both для того, щоб звільнити місце ліворуч або праворуч від інших елементів веб-сторінки. Властивість float також можна використовувати при необхідності виведення тексту у декілька стовпців.

10.5. Форматування блоків. Позиціонування.

Встановлення властивостей елементів, розміщених на веб-сторінці, засобами CSS передбачає застосування стилів до обмежених областей, які називають блоками-контейнерами, та їхнє позиціонування.

Блок-контейнер – це невидима прямокутна область, що визначена браузером. Таблиці стилів дозволяють управляти цією областю, встановлюючи її положення на сторінці з використанням абсолютних або відносних значень позиціонування.

Позиціонування блоку за допомогою властивості position за замовчуванням має значення static, іншими значеннями є absolute (абсолютне), relative (відносне) і fixed (фіксоване) позиціонування.

Абсолютне позиціонування дозволяє задавати координати внутрішнього блоку щодо зовнішнього контейнера, яким може бути батьківський елемент документа або інший блок відповідно до вікна перегляду браузера. При цьому блок елемента повністю видаляється з потоку документа й позиціонується щодо його блока-контейнера.

Відносне позиціонування дозволяє задавати координати блоку щодо його незміщеного положення в потоці документа. При відносному позиціонуванні елемент зміщується зі свого звичайного положення, але простір, який він повинен був займати, не зникає.

Фіксоване позиціонування дозволяє задавати координати блоку щодо початкового вікна перегляду браузера. Фіксований елемент прив’язується до певної точки у вікні браузера й не переміщується, навіть при прокручуванні веб-сторінки. Такі елементи часто застосовують для створення навігаційних панелей та рекламних блоків.

Порядок розташування блоків на веб-сторінці визначає z-index. Значеннями z-індексу є цілі числа (позитивні й негативні), причому блоки з більшими значеннями z-індексу будуть з’являтися над блоками з меншими значеннями.

Елемент використовується для визначення властивостей прошарку. Він є контейнерним елементом і дозволяє застосувати позиціонування й z-index для одного або кількох елементів, розміщених у ньому.

Контрольні питання

1. Охарактеризуйте CSS-властивості, за допомогою яких можна оформити границі елемента.
2. Розкрийте призначення тегів та .
3. Опишіть, які види позиціонування елементів існують в CSS.
4. Назвіть CSS-властивість, за допомогою якої можна змінювати прозорість елементів.
5. Згадайте властивості, що задають внутрішні та зовнішні відступи елемента веб-сторінки.

Лекція 11. Блоковий дизайн веб-сторінок.

План лекції:

11.1. Теги для позначення блокових елементів веб-сторінки.
11.2. Управління відображенням елементу.

11.3. Блокова верстка сайтів.
11.4. Переповнення контейнерів.

11.1. Теги для позначення блокових елементів веб-сторінки.

Блоковий підхід при верстці та дизайні сайтів полягає у використанні блокових елементів для розміщення контенту. Блоки-контейнери можуть відображати певний контент, і до кожного з них може бути застосований індивідуальний стиль. Основним елементом, що використовується в блоковій верстці, є тег . Частина коду, відокремлена цим тегом, називається шаром. Тег − це так званий контейнер (блок), який може містити форматований текст, зображення та ін. Верстка на основі активно використовує CSS для гнучкого й зручного форматування елементів веб-сторінок. За допомогою CSS можна з точністю до пікселя задати розташування блоків на сторінці, необхідні відступи, колір, розмір шрифтів і багато іншого.

У новій версії мови гіпертекстової розмітки, HTML5, з'явилися нові теги для позначення блокових елементів верстки сайту: <header>, <footer>, <nav>, <aside>, <section>, <article>, <main> тощо. Таким чином, різні частини веб-сайту отримали свої уніфіковані імена, що дозволяє уникнути плутанини між різними блоками.

11.2. Управління відображенням елементу.

Для того, щоб блокові елементи розташувалися разом по горизонталі, необхідно задати однакове значення властивості float для слідуючих один за одним елементів. З метою розміщення блоків нижче вирівняних по горизонталі, необхідно використовувати атрибут стилю clear, який вказує, що даний блок повинен розташовуватися нижче контейнерів, які передують йому.

#top { height: 20px; width:412px}
#left { height: 200px; width:50px; float: left}
#content { height: 200px; width:300px; float:left}
#right { height: 200px; width:50px; float:left}
#bottom { height: 20px; width:412px; clear: left}

11.3. Блокова верстка сайтів.

При розробці сайту з блоковою структурою передусім слід зрозуміти, який шаблон веб-сторінки нам потрібен, тобто намалювати її будову. В прикладі будемо використовувати просту структуру, яка зустрічається у більшості сайтів. Припустимо, що ми хочемо отримати веб-сторінку такого вигляду:

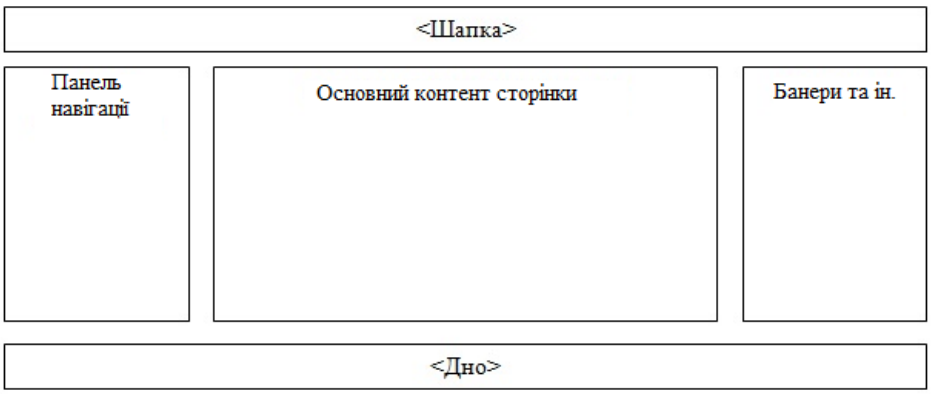

Рис.11.1. Макет веб-сторінки

Додаємо наступний html-код для розміщення п'яти контейнерів:

```
<div id="top">top</div>
<div id="left">left</div>
<div id="content">content</div>
<div id="right">right</div>
<div id="bottom">bottom</div>
```

До таблиці стилів додаємо відповідні стилі для кожного контейнера:

```
#top { height: 20px; width:412px; border: 3px double black; }
#left { height: 200px; width:50px; border: 3px double black; }
#content { height: 200px; width:300px; border: 3px double black; }
#bottom { height: 20px; width:412px; border: 3px double black; }
```

Для наочності, для кожного контейнера вказані параметри його меж. В результаті отримаємо наступне:

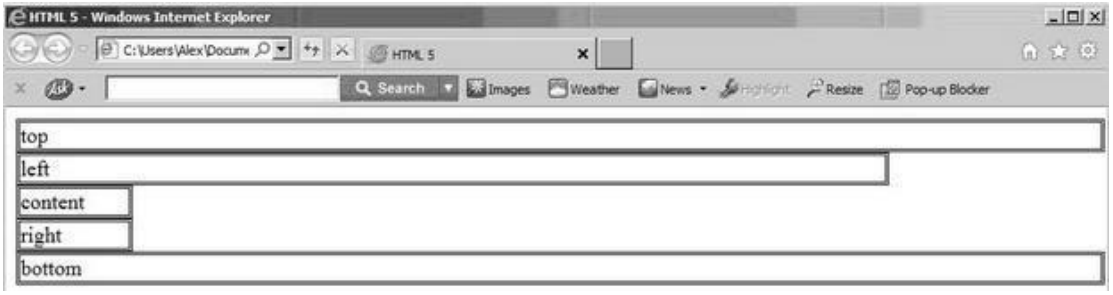

Рис.11.2. Результат розміщення контейнерів веб-сторінки за умовчанням

Очевидно, що розміщення контейнерів, яке виводиться за умовчанням, не підходить для створення дизайну веб-сторінки. З метою керування розміщенням елементів використовуються атрибути стилю float та clear. Змінимо стилі контейнерів з попереднього прикладу таким чином:

```
#top { height: 20px; width:412px; border: 3px double black}
#left { height: 200px; width:50px; border: 3px double black; float: left}
#content { height: 200px; width:300px; border: 3px double black; float:left}
```

#right { height: 200px; width:50px; border: 3px double black; float:left}
#bottom { height: 20px; width:412px; border: 3px double black; clear: left}

Внесених змін достатньо для того, щоб отримати бажаний результат:

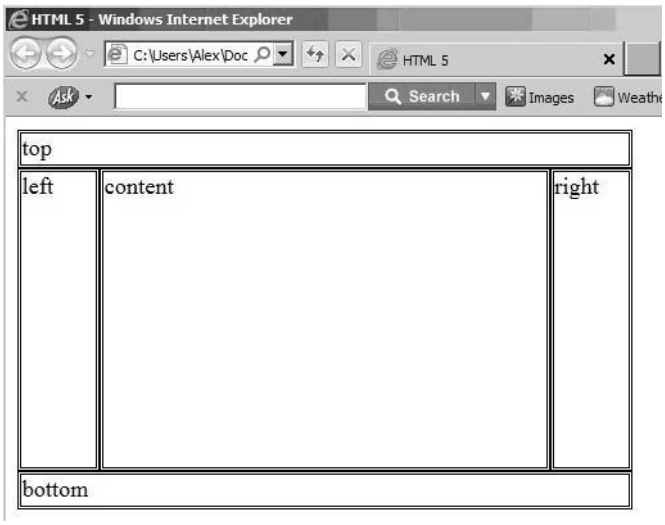

Рис. 11.3. Результат розміщення контейнерів за допомогою атрибутів стилів float та clear

Розглянемо інший приклад блокового дизайну на основі нових тегів для позначення блокових елементів верстки сайту <header>, <nav>, <aside>, <footer>. Структура веб-сторінки складатиметься з шапки сайту, горизонтального меню, лівого сайдбару, основної частини та футеру:

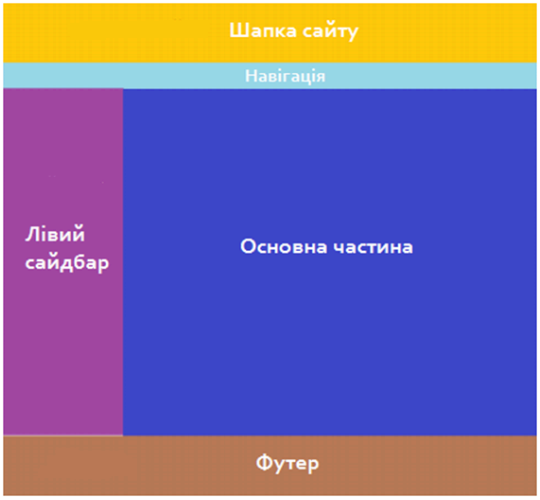

Рис. 11.4. Макет веб-сторінки з використанням блоків <header>, <nav>, <aside>, <footer>

Відповідно до того, як було передбачено структуру у графічному вигляді, можна прописати html-код для створення веб-сторінки.

<html>
<head>
<title>Блокова структура </title>

</head>
<body>
<header>
Шапка сайту
</header>
<nav>
Горизонтальне меню навігації
</nav>
<aside>
Лівий сайдбар
</aside>
* *
Основна частина
* *
* *
<footer>
Футер
</footer>
</body>
</html>

Для того, щоб веб-сторінка набула потрібного вигляду, додаємо стилі CSS у розділ <head> розробленого html-документу.

<style>
body {
background: #f5f5f5;
color: #000000;
font-family: Arial, Times New Roman;
font-size: 16px;
}
header {
background: yellow;
height: 100px;
width: 100%;
}
nav {
background: #330044;
color: white;
width: 100%;
height: 50px;
}
aside {
background: #1baf5d;

```
float: left;
width: 10%;
height: 400px;
}
#content {
background: gray;
float: left;
width: 90%;
}
#clear {
clear:both;
}
footer {
background: #ff0404;
color: white;
height: 80px;
width: 100%;
}
</style>
```

11.4. Переповнення контейнерів.

Для керування відображенням вмісту контейнера у випадках, коли розмір контенту перевищує межі розмірів контейнера, використовується атрибут стилю overflow. Існує можливість управління відображенням вмісту контейнера окремо по горизонталі та вертикалі відповідно за допомогою атрибутів overflow-x і overflow-y, значення яких аналогічні значенням overflow.

При встановленні розмірів блоків слід правильно розраховувати їхню ширину з огляду на інші блоки, розташовані поруч. Загальна ширина екрану складає 100%, тобто щоб розмістити два блоки поруч, необхідно задати їхню ширину таким чином, щоб сума значень ширини двох блоків завжди дорівнювала 100%. Інакше може з'явитися пустий простір, або блоки не будуть вміщуватись на веб-сторінці, що порушить їхню структуру. Тоді один з них може переміститися нижче або вище за інший, або зміститися поверх нього.

Контрольні питання

1. Розкажіть, що таке успадковування властивостей у CSS.
2. Перерахуйте призначення та застосування властивості прокручувань.
3. Опишіть структуру документу при блоковому дизайні веб-сторінок.
4. Вкажіть, за допомогою яких тегів реалізується блокова верстка сайтів.
5. Поясніть, чому необхідно використовувати CSS при розробці веб-сторінок на основі блокової структури.

6. Назвіть атрибут, який слід використовувати, коли розмір контенту в блоці перевищує межі розмірів контейнера.

Лекція 12. Розробка інтерфейсу сайтів в програмному середовищі Figma.

План лекції:

12.1. Знайомство з графічним редактором Figma.
12.2. Основні можливості програми.
12.3. Початок роботи в сервісі Figma.
12.4. Особливості редагування файлів.
12.5. Спільна робота над проєктами.

12.1. Знайомство з графічним редактором Figma.

Figma – це сучасний хмарний інструмент для проєктування інтерфейсів, що використовується у веб-дизайні, UI/UX-дизайні та створенні прототипів. Програма працює за принципом візуального редактора, що дозволяє створювати макети веб-сайтів, мобільних додатків та інших цифрових продуктів у реальному часі.

Figma є графічним редактором, що працює в хмарному середовищі і являє собою універсальний інструмент для вирішення широкого спектра завдань у веб-дизайні та розробці інтерфейсів. Її функціонал дозволяє проєктувати інтерфейси різної складності: від веб-сайтів до мобільних додатків і десктопних програм. Цей інструмент здобув велику популярність серед дизайнерів завдяки своїй універсальності, легкості у використанні та можливостям для спільної роботи. На відміну від традиційних десктопних програм, таких як Adobe Photoshop або Sketch, Figma може працювати безпосередньо в браузері, що усуває необхідність встановлення додаткового програмного забезпечення та надає доступ до проєктів з будь-якого пристрою.

Однією з основних переваг Figma є її інтеграція з хмарними технологіями, що дозволяє користувачам отримувати миттєвий доступ до своїх проєктів і синхронізувати зміни в реальному часі. Це особливо важливо у сучасному дизайні, де швидкість і адаптивність стають вирішальними факторами. Крім того, Figma підтримує кросплатформеність, що означає, що вона працює на різних операційних системах, таких як Windows, macOS та Linux, а також має мобільний додаток для перегляду проєктів.

Функціонал Figma включає багато зручних та гнучких інструментів для проєктування інтерфейсів. Вони дозволяють швидко створювати фрейми, фігури, текстові блоки та інші елементи, необхідні для дизайну. Завдяки можливості створення інтерактивних прототипів, користувачі можуть тестувати роботу інтерфейсу та оцінювати його зручність на етапі проєктування. Це також дозволяє вчасно виявити недоліки і внести необхідні корективи ще до початку розробки.

Крім того, Figma підтримує створення дизайн-систем, що включають уніфіковані стилі, компоненти та шаблони. Такий підхід надзвичайно зручний для великих проєктів, де необхідно підтримувати візуальну та функціональну цілісність продукту. Вбудовані

можливості експорту дозволяють легко передавати графічні ресурси у різних форматах, що спрощує співпрацю між дизайнерами та розробниками на фінальних етапах роботи.

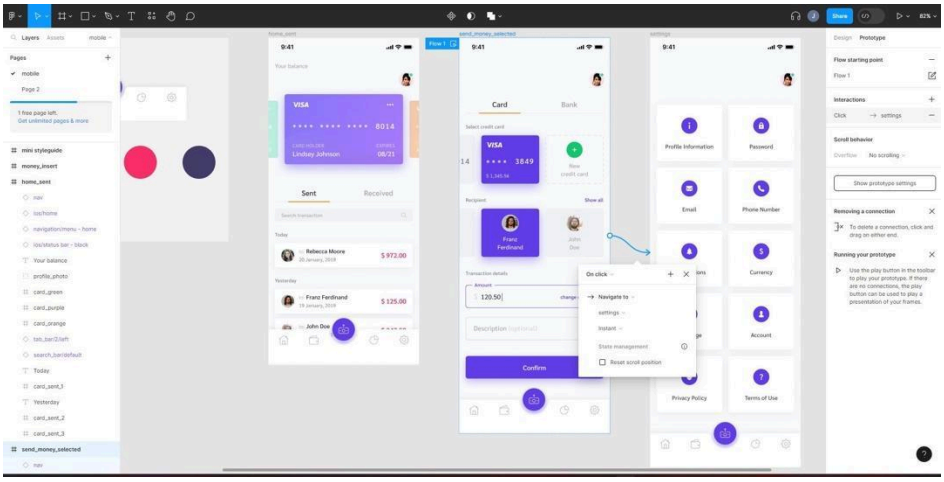

Рис. 12.1. Інтерфейс середовища розробки Figma

При запуску програми Figma користувача зустрічає стартовий екран, який має чистий і структурований інтерфейс, орієнтований на швидкий доступ до основних функцій. У верхній частині екрана розташована панель навігації, де можна знайти поле для пошуку проєктів, а також доступ до профілю користувача та опцій створення нових файлів. У правому верхньому куті розміщена кнопка "New", яка відкриває меню для створення нового Design file, FigJam file, імпорту файлів або перегляду готових шаблонів.

Інтерфейс Figma інтуїтивно зрозумілий і доступний навіть для новачків у сфері дизайну. Панель інструментів програми включає в себе всі основні функції, необхідні для створення графічних елементів, що дозволяє користувачам швидко освоїти роботу з програмою. Завдяки цим характеристикам, Figma може бути ідеальним вибором для дизайнерів, які прагнуть до ефективної та зручної роботи над своїми проєктами. Таким чином, Figma стала стандартом індустрії веб-дизайну.

12.2. Основні можливості програми.

Figma пропонує широкий набір функцій, які дозволяють дизайнерам ефективно реалізовувати дизайн-проєкти та оптимізувати робочий процес. Інструменти Figma охоплюють усі етапи створення інтерфейсів, починаючи від розробки базових концепцій до створення інтерактивних прототипів і підготовки проєктів для передачі розробникам.

Однією з найсильніших сторін Figma є підтримка адаптивного дизайну. Інструменти програми дозволяють створювати макети, які автоматично підлаштовуються під різні розміри екранів. Це реалізується через використання фреймів із гнучкими параметрами та функцій Auto Layout. Зазначені функції дозволяють задавати правила розташування елементів, що значно спрощує роботу над адаптивними версіями дизайну для різних пристроїв.

При створенні нового проєкту відкривається екран програми з робочим інтерфейсом. У верхній частині екрана розташована головна панель інструментів, де зібрані базові функції для роботи. Тут можна знайти інструменти для створення фреймів, фігур, текстових блоків, а також опції для редагування та налаштування об’єктів. Крім

цього, на панелі є кнопки для перемикання режимів перегляду та функція "Share" для спільного доступу до файлу.

Інструменти для роботи з текстом у Figma є одним із ключових елементів її функціоналу. Програма підтримує використання змінних шрифтів, які дозволяють гнучко налаштовувати товщину, ширину чи інші параметри тексту. Крім того, функція стилів тексту забезпечує швидке застосування єдиних параметрів форматування в усьому проєкті, а це, в свою чергу, сприяє узгодженості дизайну. Ще однією важливою функцією є можливість створення інтерактивних компонентів. Це включає не лише статичні макети, але й складні елементи, які реагують на дії користувача, наприклад, випадаючі меню чи кнопки зі станами наведення. Завдяки цьому можна демонструвати реалістичну поведінку інтерфейсу без необхідності залучення програмістів.

Figma також має вбудовані інструменти для створення графіки та іконок. Векторний редактор дозволяє малювати довільні форми, налаштовувати криві та працювати з шарами. Завдяки підтримці експорту у векторні формати, готові іконки можуть бути легко інтегровані в інші проєкти чи передані розробникам.

Інтеграція плагінів – ще одна важлива перевага Figma. Платформа підтримує велику кількість додаткових інструментів, які можуть автоматизувати рутинні завдання, наприклад, генерацію фіктивного тексту, створення сіток чи оптимізацію зображень. Плагіни значно розширюють функціонал програми та дозволяють адаптувати її під конкретні потреби користувача.

Для команд, які працюють над великими проєктами, Figma пропонує можливості організації файлів і ресурсів. Вбудовані бібліотеки дозволяють централізовано зберігати компоненти, стилі та інші елементи дизайну, забезпечуючи їх доступність для всіх учасників команди. Це сприяє стандартизації та зменшує кількість помилок, пов’язаних із використанням застарілих або несумісних елементів. Завдяки вбудованим функціям прототипування, адаптивного дизайну, інтерактивних компонентів і підтримки плагінів, Figma є універсальним інструментом для вирішення найрізноманітніших дизайнерських завдань. Багатий функціонал Figma дозволяє створювати високоякісні інтерфейси, які відповідають сучасним стандартам і вимогам до дизайну сайтів.

12.3. Початок роботи в сервісі Figma.

Figma є хмарним сервісом, тому для початку роботи достатньо зареєструватися на офіційному сайті (figma.com) за допомогою електронної пошти або облікового запису Google. Після входу користувач потрапляє на стартову сторінку (або відкриває десктоп версію програми), де представлено список доступних проєктів, панель навігації та кнопки для розробки нових файлів. Для створення нового проєкту потрібно натиснути кнопку "New File", після чого відкривається робоча область, що складається з полотна (Canvas), панелі інструментів, панелі властивостей і списку шарів. Полотно є основним простором для роботи, де дизайнер створює та редагує макети, причому розмір полотна необмежений, що дозволяє працювати над кількома екранами чи компонентами в одному файлі.

Ліворуч розташована панель шарів і сторінок. У верхній частині цієї панелі відображається перелік сторінок проєкту, що дозволяє користувачеві керувати кількома робочими областями в межах одного файлу. Це дозволяє зручно організовувати проєкт і розділяти його на логічні частини. Під кожною сторінкою згодом відображатимуться шари (Layers), що містять усі елементи дизайну на обраній сторінці.

З правого боку екрану розташована панель властивостей. Дана панель змінюється залежно від обраного інструмента чи елемента. У верхній частині цієї панелі можна побачити вкладки "Design", "Prototype" і "Inspect", які відповідають за налаштування дизайну, створення прототипів та перевірку властивостей для розробників. У нижній частині панелі можна налаштувати фон робочого полотна та знайти опції для експорту готових елементів.

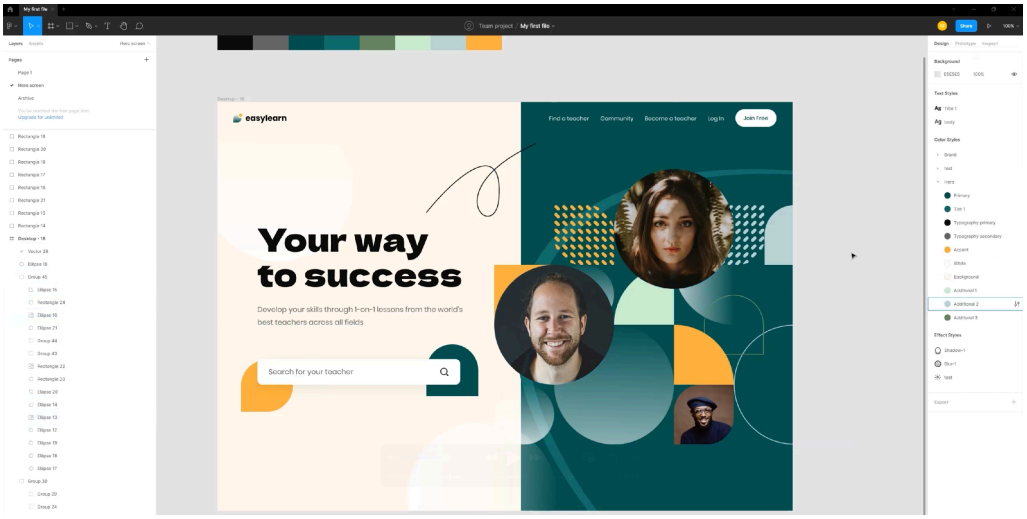

Рис. 12.2. Вигляд робочого простору при розробці проєкту у Figma

На початковому етапі важливо налаштувати параметри проєкту: вибрати розміри фреймів (наприклад, для мобільних пристроїв чи десктопів), встановити кольорову палітру та визначити стилі тексту. Figma пропонує шаблони для популярних розмірів екранів, що спрощує цей процес. Панель інструментів містить базові інструменти для створення геометричних фігур, роботи з текстом, використання ліній та перо (Pen Tool) для малювання довільних форм. Усі інструменти підтримують гарячі клавіші, що значно пришвидшує роботу. Програма також дозволяє імпортувати ресурси, такі як зображення чи іконки, із зовнішніх джерел, а також інтегрувати елементи з інших графічних редакторів, наприклад, із сервісу Sketch.

Особливістю Figma є автоматичне збереження проєктів у хмарі, що дозволяє уникнути втрати даних. Усі зміни зберігаються в реальному часі, а функція історії дозволяє повернутися до попередніх версій файлу. Важливим аспектом початкової роботи є налаштування доступу: у Figma можна запрошувати інших користувачів до розробки проєкту через електронну пошту або посилання, встановлюючи ролі для кожного учасника (перегляд, коментування чи редагування). Це забезпечує ефективну командну роботу з чітким розподілом обов’язків. Для новачків сервіс пропонує інтерактивні гіди та навчальні матеріали, які також допомагають швидко освоїти основи роботи та ознайомитися з ключовими функціями.

12.4. Особливості редагування файлів.

Редагування файлів у Figma відзначається гнучкістю, зручністю та широким спектром інструментів, які дозволяють реалізовувати найскладніші дизайнерські завдання.

Завдяки хмарній архітектурі сервісу, всі зміни вносяться в реальному часі, незалежно від того, чи працює над файлом одна людина, чи команда. Це дозволяє уникнути дублювання файлів, проблем із синхронізацією версій і втрати даних.

Однією з ключових особливостей Figma є можливість роботи з шарами. Усі елементи дизайну організовані в ієрархічну структуру, яка відображається у вигляді списку. Користувач може переміщувати шари, групувати їх, перейменовувати та приховувати, що значно спрощує навігацію та редагування складних проєктів. Панель властивостей, розташована праворуч, дозволяє швидко змінювати параметри вибраного елемента, такі як розмір, колір, прозорість чи ефекти.

Редагування текстових елементів у Figma є інтуїтивно зрозумілим. Програма підтримує змінні шрифти, стилі тексту та можливість налаштування міжлітерного і міжрядкового інтервалу. Завдяки цьому можна створювати текстові блоки, які відповідають сучасним вимогам до типографіки. Крім того, текстові стилі можна зберігати та застосовувати до інших елементів у проєкті, що забезпечує узгодженість оформлення. Ще однією важливою функцією є можливість редагування векторних елементів. Вбудований інструмент Pen Tool дозволяє створювати та змінювати довільні форми, а також налаштовувати їхні криві, кути та вузли. Це відкриває широкі можливості для створення графічних елементів, таких як іконки чи ілюстрації, без необхідності використання додаткових програм.

Також Figma підтримує налаштування стилів і ефектів для будь-яких елементів. Користувачі можуть застосовувати тіні, градієнти, розмиття та інші візуальні ефекти, які легко налаштовуються через панель властивостей. Крім того, ці стилі можна зберігати в бібліотеках, що дозволяє повторно використовувати їх у різних файлах чи проєктах.

Ще однією унікальною особливістю є функція Smart Selection, яка автоматично вирівнює та розподіляє вибрані елементи. Це значно спрощує процес редагування макетів, особливо якщо потрібно швидко внести зміни в розташування об’єктів. Додатково функція Auto Layout дозволяє налаштовувати динамічне розташування елементів, що автоматично адаптуються до змін у розмірах або вмісті. Редагування файлів у Figma також передбачає можливість роботи з компонентами. Компоненти є повторюваними елементами, які можна редагувати централізовано. Наприклад, зміна одного компонента автоматично оновить всі його копії в проєкті. Це особливо корисно для великих проєктів, де потрібно підтримувати узгодженість дизайну.

12.5. Спільна робота над проєктами.

Figma є одним із найпотужніших інструментів для організації спільної роботи над проєктами завдяки своїй хмарній архітектурі. Усі учасники команди можуть працювати над одним файлом одночасно, а зміни, внесені кожним користувачем, відображаються в реальному часі. Це усуває потребу в ручному злитті файлів або синхронізації версій, що є типовою проблемою для багатьох традиційних програм.

Для організації командної роботи у Figma передбачено декілька рівнів доступу: редагування, коментування та перегляд. Власник файлу може налаштувати права доступу для кожного учасника, запрошуючи їх через електронну пошту або за допомогою посилання. Такий підхід дозволяє чітко розподілити обов’язки між членами команди та забезпечити безпеку даних. Однією з ключових функцій для командної роботи є система коментування. Користувачі можуть залишати коментарі безпосередньо на полотні, прикріплюючи їх до конкретних елементів дизайну. Це значно спрощує процес

обговорення, дозволяє швидко вирішувати питання та уникати непорозумінь. Повідомлення про нові коментарі надходять учасникам проєкту у реальному часі.

Figma також пропонує можливість створення команд і проєктів, що організовують роботу з файлами. Усі файли, пов'язані з певним проєктом, зберігаються в одній папці, доступній усім учасникам команди. Це дозволяє зберігати структуру та уникати хаосу в управлінні файлами. Крім того, у платних версіях сервісу доступна функція Design Systems, яка дозволяє створювати та використовувати спільні бібліотеки компонентів, стилів і кольорів. Це забезпечує узгодженість дизайну у великих проєктах і значно скорочує час на виконання повторюваних завдань.

Ще однією перевагою Figma є інтеграція з іншими сервісами, такими як Slack або Jira. Це дозволяє автоматизувати процеси комунікації та трекінгу завдань, зберігаючи єдину екосистему для роботи над проєктом. Завдяки функціям командної роботи, Figma є незамінним інструментом для сучасних спеціалістів, які прагнуть працювати ефективно, злагоджено та без затримок.

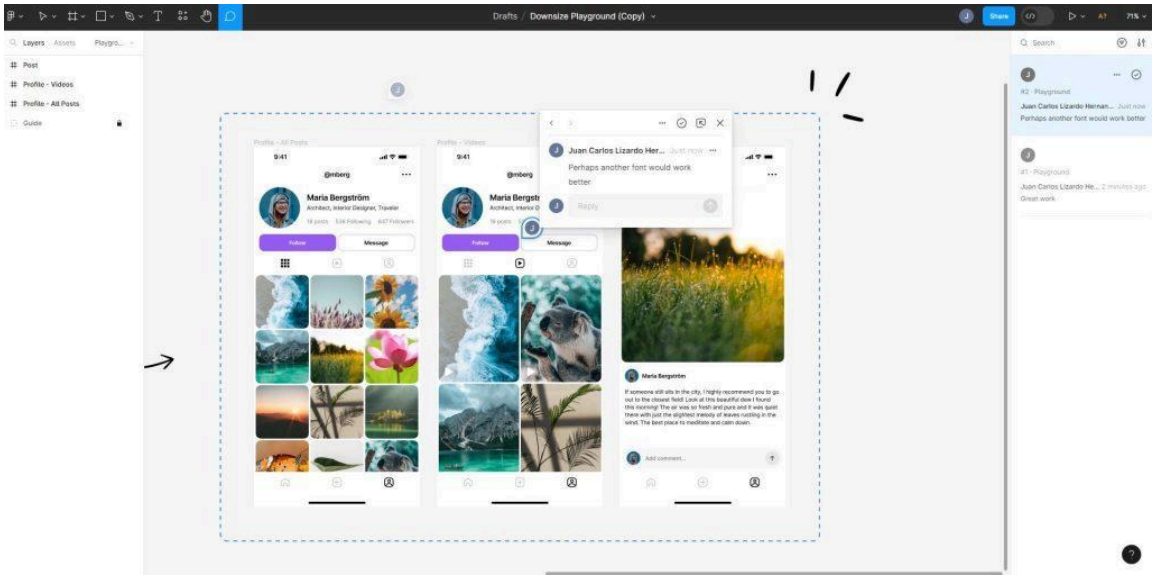

Рис. 12.3. Приклад командної роботи над проєктом у Figma

Контрольні питання

1. Розкрийте, для чого використовується програма Figma і які її основні переваги.
2. Охарактеризуйте базові інструменти дизайну в програмному середовищі Figma.
3. Опишіть основні інструменти Figma для створення графічних об'єктів.
4. Поясніть, як здійснюється групування об'єктів у Figma.
5. Розкажіть, яким чином забезпечується спільна робота над проєктами у Figma.

Лекція 13. Базові інструменти редактора Figma.

План лекції:

13.1. Робота зі стилями у Figma.
13.2. Робота з кольором і налаштування відтінків.
13.3. Прозорість елементів.
13.4. Градієнтні переходи кольорів.

13.1. Робота зі стилями у Figma.

Стилі у Figma є важливим інструментом для забезпечення узгодженості та ефективності в дизайні. Вони дозволяють зберігати і застосовувати однакові параметри до різних елементів, що значно спрощує роботу над великими проєктами та покращує організацію дизайну. Ключовою перевагою стилів є можливість централізовано змінювати параметри із автоматичним оновленням всіх елементів, до яких цей стиль застосовано. Така можливість особливо корисна в умовах командної роботи, коли багато дизайнерів працюють над одним проєктом, і важливо зберігати єдність у вигляді тексту, кольорів чи ефектів на всіх етапах розробки.

Основними типами стилів у Figma є текстові стилі, стилі кольорів, стилі ефектів та ліній. Текстові стилі дозволяють встановлювати шрифт, розмір, інтерліньяж, міжлітерний інтервал та інші параметри типографіки, що забезпечує консистентність тексту на всіх екранах і компонентах. Стилі кольорів дозволяють визначати палітри, які використовуються у проєкті, що спрощує вибір кольорів і підтримку брендових стандартів. Стилі ефектів (наприклад, тіні чи розмиття) дозволяють додавати візуальні ефекти до елементів, що підвищує естетичну цілісність дизайну. Стилі ліній дають можливість встановлювати товщину, тип і колір ліній, що часто використовується для створення рамок чи роздільників.

Використання стилів у Figma значно спрощує редагування та внесення змін. Якщо потрібно змінити шрифт чи колір у всьому проєкті, достатньо оновити стиль, і всі елементи, до яких він застосований, автоматично оновляться. Це економить час, знижує ймовірність помилок і дозволяє швидко адаптувати дизайн до нових вимог чи змін у брендових стандартах. Крім того, стилі можна зберігати в бібліотеках для подальшого використання, завдяки чому вдається організувати доступ до них для всієї команди та забезпечити узгодженість у багатьох проєктах одночасно.

У Figma існує кілька основних типів стилів, кожен з яких виконує свою специфічну функцію в процесі створення дизайну. Це текстові стилі, стилі кольорів, стилі ефектів та стилі ліній. Кожен тип стилю має свою роль у забезпеченні узгодженості та зручності роботи з проєктами, що є особливо важливим для великих команд або складних дизайнів.

Текстові стилі в Figma дозволяють стандартизувати всі параметри типографіки, що використовуються в проєкті. Це включає вибір шрифтів, їхнього розміру, міжлітерного інтервалу, міжрядкового інтервалу, а також кольору тексту. Текстові стилі є надзвичайно корисними, коли необхідно підтримувати однаковий вигляд заголовків, підзаголовків, абзаців чи кнопок на всіх екранах проєкту. Використання таких стилів дозволяє легко змінювати вигляд тексту в усьому проєкті, змінюючи лише один параметр стилю, що значно заощаджує час та зменшує ймовірність помилок.

Стилі кольорів дають можливість централізовано управляти палітрою кольорів, що використовується в проєкті. Це особливо важливо для забезпечення узгодженості в дизайні, оскільки кольори можуть використовуватися в різних елементах – від фону до кнопок чи іконок. Користувач може створювати стилі для основних кольорів, акцентів, фонових кольорів тощо, і це дозволяє не лише зберігати єдність кольорової палітри, але й швидко змінювати кольори в усьому проєкті при необхідності, наприклад, при зміні брендингу.

Стилі ефектів охоплюють різноманітні візуальні ефекти, які можна застосовувати до елементів дизайну, такі як тіні, розмиття, градієнти чи прозорість. Ці стилі дозволяють створювати глибину, контраст та візуальний інтерес у проєкті. Наприклад, тіні можуть використовуватися для надання елементам об’єму, а градієнти – для створення плавних переходів кольорів. Стилі ефектів також дають можливість централізовано редагувати ефекти, що застосовуються до елементів, і змінювати їх усього в декілька кліків.

Стилі ліній визначають параметри ліній, які використовуються для рамок, роздільників чи контурів. Вони включають товщину, колір, тип лінії (суцільна, пунктирна тощо). Використання стилів ліній дозволяє стандартизувати вигляд усіх ліній у проєкті та забезпечити їх узгодженість.

Процес створення стилів досить простий, але вимагає чіткого розуміння того, як і коли їх застосовувати. Стиль можна створити для тексту, кольору, ефектів або ліній, вибравши відповідний елемент і налаштувавши його параметри, після чого цей стиль можна зберегти для подальшого використання. Це дозволяє створювати стандартизовані елементи дизайну, які можна швидко застосовувати до інших частин проєкту, забезпечуючи тим самим узгодженість у всіх його аспектах.

Створення текстового стилю, наприклад, починається з вибору елемента тексту і налаштування всіх необхідних параметрів: шрифт, розмір, інтерліньяж, колір та інші характеристики. Після цього ці налаштування можна зберегти як стиль, який буде доступний для подальшого використання. Це дає змогу зберігати єдність типографіки на всіх екранах і компонентах проєкту. Точно так само можна створювати стилі кольорів, ефектів та ліній, налаштовуючи відповідні параметри і зберігаючи їх для швидкого застосування до інших елементів. Це особливо корисно при роботі з великими проєктами, де важливо підтримувати єдину кольорову палітру або однакові візуальні ефекти на всіх екранах.

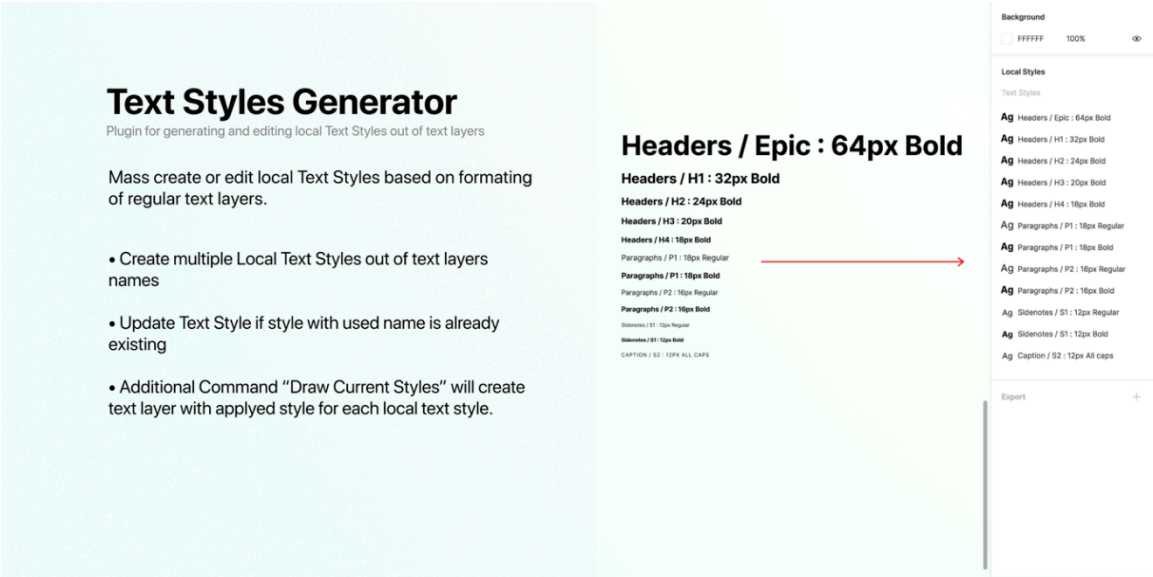

Рис. 13.1. Робота із текстовими стилями у Figma

Після створення стилю його можна застосувати до будь-якого елемента, вибравши відповідний стиль із панелі стилів і натиснувши на нього. Це дозволяє швидко і ефективно оновлювати елементи дизайну, адже всі зміни, внесені до стилю, автоматично відображаються на всіх елементах, до яких цей стиль був застосований. Наприклад, якщо потрібно змінити шрифт або колір у всьому проєкті, достатньо оновити стиль, і всі елементи, до яких цей стиль застосовано, будуть змінені одночасно. Це значно економить час, знижує ймовірність помилок і дозволяє швидко адаптувати дизайн до нових вимог. Крім того, стилі можна зберігати в бібліотеках для подальшого використання в інших проєктах, що дає змогу ефективно використовувати однакові елементи в різних проєктах і забезпечує узгодженість у всіх роботах, що виконуються.

Бібліотеки стилів у Figma є потужним інструментом для організації та обміну стилями між різними проєктами чи командами. Використання бібліотек стилів дозволяє створювати централізовані сховища для всіх стилів, що використовуються в проєкті, і забезпечує зручний доступ до них для всіх учасників проєкту. Бібліотеки дозволяють підтримувати єдність дизайну на всіх етапах розробки, адже всі елементи будуть використовувати однакові стилі, навіть якщо проєкт містить велику кількість різних екранів або компонентів. Крім того, бібліотеки стилів дозволяють зберігати стандартизовані елементи для подальшого використання в інших проєктах.

Для ефективної роботи з бібліотеками стилів важливо дотримуватися кількох правил. По-перше, варто чітко організувати бібліотеки, створюючи окремі категорії для різних типів стилів, наприклад, для текстових стилів, кольорів, ефектів. Це дозволить швидко знаходити необхідні стилі та зменшить ймовірність плутанини. По-друге, важливо регулярно перевіряти та оновлювати бібліотеки, щоб переконатися, що всі стилі відповідають вимогам проєкту і не містять застарілих чи непотрібних елементів. По-третє, для великих проєктів можна створювати окремі бібліотеки для різних частин проєкту або для різних етапів розробки, що дозволить більш ефективно організувати роботу над проєктом.

Використання бібліотек стилів також дозволяє працювати з глобальними стилями, які можуть бути застосовані до всіх елементів проєкту. Це особливо важливо для великих

проєктів, де необхідно підтримувати єдину палітру кольорів, типографіку та ефекти на всіх екранах.

13.2. Робота з кольором і налаштування відтінків.

Кольори є одним із найважливіших елементів дизайну, оскільки вони впливають на сприйняття інтерфейсу та взаємодію користувача з продуктом. У Figma робота з кольором забезпечує широкий спектр можливостей для створення візуально привабливих та функціональних дизайнів. Кольори можуть використовуватися не лише для естетичного оформлення, але й для створення ієрархії, акцентів та забезпечення зручності користування інтерфейсом. Налаштування відтінків дозволяє точно контролювати кольорові параметри, що є важливим для досягнення бажаного ефекту та відповідності брендовим вимогам.

Основним інструментом для роботи з кольором у Figma є палітра кольорів, яка дозволяє обирати кольори для елементів дизайну, таких як фон, текст, кнопки та інші компоненти. Figma підтримує кілька форм введення кольорів: через шістнадцятковий код, RGB або HSL, що дає змогу точно налаштувати відтінки і досягати потрібного результату. Крім того, Figma дозволяє працювати з градієнтами, що дає можливість створювати плавні переходи між кольорами, додаючи глибину та динамічність елементам дизайну. Використання градієнтів допомагає зробити інтерфейс більш живим та привабливим, а також може бути використано для підкреслення важливих елементів, таких як кнопки або банери.

Налаштування відтінків у Figma є важливим етапом роботи з кольором, оскільки дозволяє створювати різні варіації одного кольору, що підходять для різних елементів інтерфейсу. Відтінки можуть бути налаштовані шляхом зміни яскравості, насиченості або відтінку основного кольору. Це дозволяє зберігати єдність кольорової палітри, при цьому створюючи достатньо контрасту для різних елементів. Наприклад, один і той самий колір може бути використаний для фону, тексту або кнопок, але за допомогою налаштування відтінків можна створити варіації, які будуть добре сприйматися в контексті конкретного елемента.

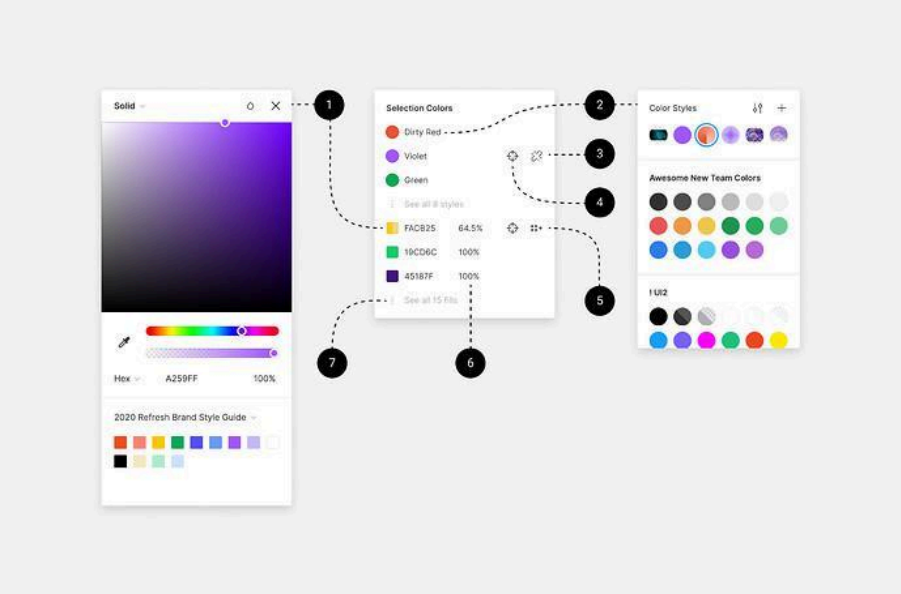

Рис. 13.2. Налаштування відтінків у Figma

Також важливою складовою роботи з кольором є використання кольорових стилів у Figma. Це дозволяє створювати стандартизовані палітри, які можна використовувати в усьому проєкті. Кольорові стилі можуть включати основні кольори бренду, акцентні кольори, фонові кольори та кольори для різних елементів інтерфейсу. Важливо, щоб кольори були взаємопов'язані та відповідали одному стилю, що забезпечить узгодженість дизайну. Для цього можна створювати окремі палітри для різних типів елементів, наприклад, для кнопок, тексту, фону або іконок, що дозволяє швидко застосовувати кольори та зберігати єдність дизайну.

13.3. Прозорість елементів.

Прозорість дозволяє створювати візуальні ефекти, які додають глибину, контекст і динамічність. У Figma цей параметр використовується для налаштування рівня прозорості окремих елементів, що дозволяє досягти ефектів накладання, створення тіней або забезпечення більш м'яких переходів між елементами. Використання прозорості дозволяє дизайнерам маніпулювати видимістю елементів, контролюючи їх взаємодію з іншими об'єктами на екрані та створюючи більш естетично привабливі композиції.

Прозорість у Figma може бути налаштована через параметр "Opacity", що дозволяє змінювати рівень видимості елемента від 0% (повністю прозорий) до 100% (повністю непрозорий). Це дає змогу точно контролювати, наскільки видимим буде елемент у контексті інших об'єктів на екрані. Прозорість може бути застосована до різних елементів дизайну, таких як фони, текстури, іконки, зображення та інші компоненти. Наприклад, при роботі з фонами можна налаштувати прозорість так, щоб елементи, які знаходяться за фоном, були злегка видні, що створює ефект накладання і дозволяє зберігати контекст.

Одним із основних застосувань прозорості є створення ефектів накладання, де один елемент частково перекриває інший, дозволяючи частково бачити зображення або фон, що знаходяться під ним. Цей ефект особливо корисний для створення складних візуальних композицій, де важливо зберегти видимість кількох елементів одночасно. Прозорість також активно використовується для створення ефектів тіней або затемнення, коли необхідно додати елементам глибину або підкреслити важливі ділянки інтерфейсу.

Важливою особливістю роботи з прозорістю є її вплив на взаємодію елементів з користувачем. Прозорі елементи можуть бути використані для м'якого підсвічування важливих кнопок або акцентних елементів, не відволікаючи увагу від основного контенту. Наприклад, кнопки з низьким рівнем прозорості можуть здаватися більш м'якими і менш нав'язливими, що підвищує зручність використання інтерфейсу.

Прозорість також може бути використана для створення плавних переходів між елементами, особливо при використанні анімацій або ефектів при наведенні курсору. Це дозволяє зробити інтерфейс більш динамічним і привабливим для користувачів, надаючи їм відчуття взаємодії з елементами, що змінюються в реальному часі. Використання прозорості в таких контекстах допомагає уникнути різких переходів і створює більш органічні та естетичні ефекти.

13.4. Градієнтні переходи кольорів.

Градієнтні переходи дозволяють створювати плавні переходи між різними кольорами, додаючи глибину, об'єм і динамічність елементам інтерфейсу. У Figma

градієнти використовуються для створення ефектів, які підвищують естетичну привабливість дизайну, а також можуть виконувати функціональні завдання, такі як виділення важливих елементів або створення контрасту. Градієнти дозволяють дизайнеру працювати з кольоровими переходами, що забезпечує більш плавні та органічні зміни кольору, ніж просте використання одного кольору.

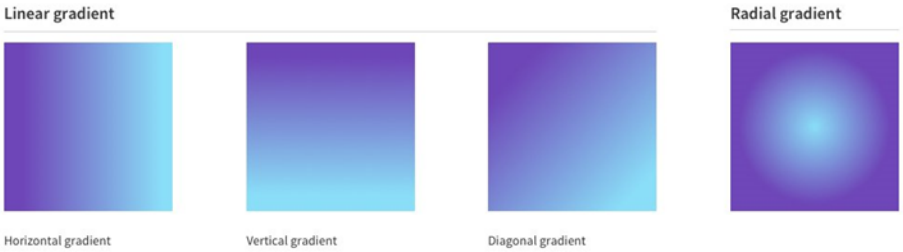

Рис. 13.3. Види градієнтних переходів у Figma

Основною характеристикою градієнтних переходів є їхня здатність поєднувати два або більше кольорів в одну плавну лінію, що створює ефект змішування. Figma підтримує різні типи градієнтів, серед яких найбільш поширеними є лінійні та радіальні. Лінійний градієнт змінює кольори по прямій лінії, що дає змогу створювати горизонтальні, вертикальні або діагональні переходи. Радіальний градієнт, в свою чергу, створює плавний перехід кольорів від центральної точки до країв елемента, що дозволяє створювати ефекти освітлення або підсвічування. Кожен тип градієнта має свої переваги в залежності від контексту, і їхнє правильне використання дозволяє створити ефект, який відповідає вимогам дизайну.

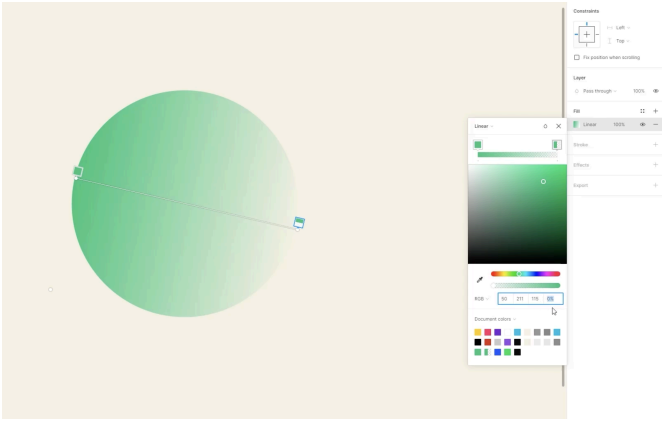

Рис. 13.4. Створення ефекту заливки «Градієнт»

Для налаштування градієнтних переходів у Figma можна обирати декілька кольорів для створення багатокольорових градієнтів, а також визначати точку початку і кінця переходу. Це дає можливість точно контролювати, як кольори будуть змінюватися в межах заливки градієнта. Крім того, Figma дозволяє налаштовувати напрямок переходу, його інтенсивність та розташування кольорових точок, що дає ще більше можливостей для створення унікальних ефектів. Для створення складних градієнтів можна комбінувати кілька кольорів або використовувати різні типи градієнтів.

Градієнтні переходи використовуються для створення різноманітних візуальних ефектів, таких як тіні, світлові ефекти, об’єми або навіть фони. Вони допомагають зробити інтерфейс більш живим і привабливим, а також можуть підкреслити важливі елементи, такі як кнопки, банери або логотипи. Наприклад, кнопки з градієнтним фоном виглядають більш привабливо і привертають увагу користувача, що покращує взаємодію з інтерфейсом. Градієнти також можуть бути використані для створення текстур або фонів, що додає складності та глибини елементам інтерфейсу.

Використання градієнтів вимагає уважного підходу, оскільки неправильне їхнє застосування також може призвести до перевантаження інтерфейсу або створення ефекту хаосу. Тому важливо дотримуватися принципів гармонії кольорів та контрасту, щоб градієнти виглядали естетично і не відволікали увагу від основного контенту. Градієнти повинні бути використані так, щоб вони підкреслювали важливі елементи, не порушуючи загальну композицію дизайну.

Контрольні питання

1. Визначте, як встановлюються налаштування стилю у Figma.
2. Поясніть, яким чином налаштувати базові ефекти шарів.
3. Назвіть, які параметри можна зберегти як стилі у Figma.
4. Представте способи встановлення кольорів і налаштування відтінків у Figma.
5. Опишіть, що можна налаштувати на панелі Effects у середовищі Figma.

Лекція 14. Типографіка в програмному середовищі Figma.

План лекції:

14.1. Правила типографіки.
14.2. Створення текстових об’єктів.
14.3. Налаштування стилю тексту.
14.4. Бібліотека шрифтів.

14.1. Правила типографіки.

Шрифт є не лише інструментом для передачі тексту, але й важливим елементом візуального оформлення, що допомагає створити атмосферу, підкреслити стиль та забезпечити зручність читання. Вибір шрифтів залежить від багатьох факторів, таких як контекст використання, тип проєкту, цільова аудиторія та вимоги до читабельності. Існує кілька основних типів шрифтів, серед яких найбільш поширеними є шрифти із зарубками (serif), рублені шрифти (sans-serif), курсивні шрифти (cursive), декоративні шрифти (fantasy), моноширинні шрифти (monospace), рукописні шрифти (handwriting).

Шрифти із зарубками мають невеликі штрихи або «завитки» на кінцях літер, що робить їх ідеальними для друкованих матеріалів, де важливо забезпечити легкість читання на великих обсягах тексту. Вони створюють відчуття класичності та стабільності, їх часто

використовують для газет, книг і офіційних документів. Рублені шрифти, у свою чергу, мають чисті, прості лінії без додаткових елементів, що робить їх більш сучасними та мінімалістичними. Вони широко використовуються у веб-дизайні, оскільки на екранах вони забезпечують високу читабельність, особливо на малих розмірах шрифтів. Моноширинні шрифти, де кожна буква займає однакову кількість простору, зазвичай використовуються для відображення коду або технічних текстів, оскільки вони допомагають забезпечити чіткість і структуру.

При виборі шрифтів важливо враховувати такі параметри, як вага шрифту, висота літер, інтерліньяж та кернінг. Вага шрифту визначає його товщину, що впливає на видимість і акцентування елементів тексту. Легші шрифти часто використовуються для основного тексту, тоді як важчі шрифти можуть застосовуватися для заголовків або акцентів. Висота літер (або висота x) визначає відстань між верхньою і нижньою частинами літер, що важливо для забезпечення зручності читання. Інтерліньяж – це відстань між рядками тексту, яка повинна бути достатньою для того, щоб текст не виглядав стиснутим, але й не був занадто розсіяним. Кернінг визначає відстань між окремими літерами, що дозволяє створити більш гармонійний вигляд тексту.

Вибір шрифтів має бути обґрунтованим і відповідати вимогам проєкту. Для досягнення гармонії в дизайні важливо використовувати не більше двох або трьох різних шрифтів, щоб уникнути перевантаження і забезпечити чіткість та єдність елементів веб-сторінки. Крім того, важливо пам’ятати про контекст і цільову аудиторію: для молодіжних проєктів можуть підійти сучасні рублені шрифти, тоді як для більш офіційних або традиційних проєктів варто обирати шрифти із зарубками, що додають солідності.

Правильне використання типографічної ієрархії дозволяє чітко розрізняти важливі елементи, такі як заголовки, підзаголовки, основний текст та допоміжні елементи, що забезпечує легкість навігації та зручність сприйняття. Ієрархія створюється за допомогою різних шрифтів, розмірів, ваги, кольору та стилю тексту, що дозволяє акцентувати увагу на важливих частинах контенту і направляти погляд користувача у правильному порядку.

Основним інструментом для створення типографічної ієрархії є різниця в розмірах шрифтів. Заголовки, як правило, мають більший розмір, ніж основний текст, що дозволяє їм привертати увагу і допомагає користувачам швидко орієнтуватися в структурі контенту. Підзаголовки, в свою чергу, повинні бути дещо меншими за заголовки, але все ще достатньо великими, щоб виділятися на фоні основного тексту. Важливо, щоб різниця між заголовками та підзаголовками була чіткою, але не надмірною, щоб не порушувати єдність дизайну. Крім того, зміна ваги шрифтів також сприяє створенню ієрархії: жирний шрифт може використовуватися для заголовків або акцентів, тоді як для основного тексту зазвичай вибираються більш легкі варіанти.

Типографічна ієрархія також включає використання кольору для підкреслення важливих елементів. Наприклад, заголовки можуть бути визначені за допомогою більш насиченого кольору, що дозволяє їм виділятися на фоні інших елементів. Водночас основний текст зазвичай має нейтральний колір, що сприяє комфортному читанню на великих обсягах контенту. Використання різних стилів, таких як курсив або підкреслення, може додатково акцентувати увагу на окремих частинах тексту, але їх варто використовувати обережно, щоб не порушити загальний баланс і не перевантажити інтерфейс.

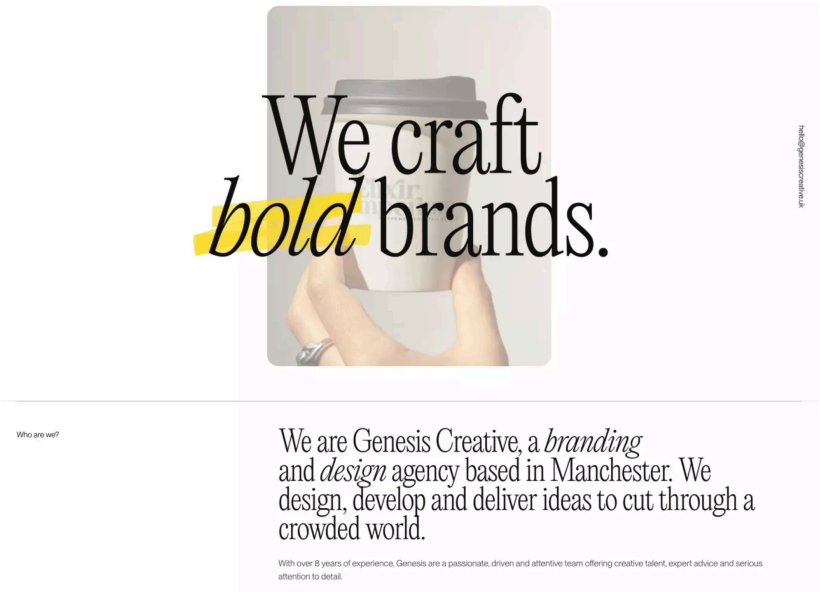

Рис.14.1. Сторінка із різними типами шрифтів, розроблена у Figma

Важливою складовою типографічної ієрархії є також відстань між елементами, зокрема між заголовками, підзаголовками та основним текстом. Інтерліньяж і відступи між абзацами дозволяють створювати чітку межу між різними блоками інформації, що полегшує сприйняття тексту. Правильне використання відступів допомагає організувати контент і зробити його більш зручним для читання. Крім того, важливо враховувати контекст, у якому буде використовуватися текст, і цільову аудиторію, оскільки типографічна ієрархія може варіюватися залежно від специфіки проєкту: для наукових статей або технічних інтерфейсів ієрархія буде суворішою, а для творчих або маркетингових проєктів – більш гнучкою та експериментальною.

Адаптивність шрифтів означає їхню здатність коректно відображатися на різних пристроях, незалежно від розміру екрана чи роздільної здатності. Це особливо важливо в умовах сучасного дизайну, коли користувачі взаємодіють із веб-сайтами через смартфони, планшети, ноутбуки та великі монітори. Читабельність, у свою чергу, визначає легкість сприйняття тексту і залежить від таких факторів, як вибір шрифтів, розмір тексту, інтерліньяж, контрастність та відстані між елементами.

Одним із ключових аспектів адаптивності є використання відносних одиниць вимірювання, таких як em або rem, для визначення розмірів шрифтів. Це дозволяє тексту автоматично масштабуватися залежно від параметрів пристрою, на якому його переглядають. Наприклад, базовий розмір шрифту може бути встановлений на 16px, а всі інші розміри – пропорційно до цього значення. Такий підхід забезпечує гармонійне масштабування тексту та його зручність для читання на будь-якому екрані. Крім того, важливо враховувати адаптивність міжрядкових інтервалів (інтерліньяжу), щоб текст залишався легким для сприйняття навіть на невеликих екранах.

Контрастність тексту також є критичним фактором для забезпечення читабельності. Текст повинен мати достатній контраст із фоном, щоб бути добре видимим за різних умов освітлення. Наприклад, світлий текст на темному фоні може бути ефективним для нічного режиму, тоді як темний текст на світлому фоні зазвичай краще сприймається при денному освітленні. Для забезпечення відповідності стандартам доступності (WCAG)

рекомендується використовувати коефіцієнт контрастності не менше 4.5:1 для основного тексту та 3:1 для великих заголовків.

Ще одним важливим правилом є використання лаконічних шрифтів, які забезпечують легкість сприйняття інформації. Надмірно декоративні або складні шрифти можуть створювати труднощі для користувачів, особливо на невеликих екранах або при великому обсязі тексту. Для основного тексту зазвичай обирають рублені шрифти, такі як Arial, Roboto або Open Sans, які добре читаються на екранах. Заголовки можуть використовувати більш виразні шрифти, але їх також слід підбирати з урахуванням загальної стилістики сайту.

Особливу увагу слід приділяти організації тексту на сторінці. Відступи між абзацами, списками та іншими елементами тексту повинні бути достатніми, щоб забезпечити чітку візуальну структуру. Надто щільний текст виглядає перевантаженим і складним для читання, тоді як надмірно великі відступи можуть порушити цілісність дизайну. Крім того, для великих блоків тексту доцільно використовувати вирівнювання за лівим краєм, оскільки воно забезпечує природний потік читання.

Підсумовуючи вищесказане, можна сформулювати перелік основних правил типографіки у веб-дизайні:

– Читабельність та зручність сприйняття тексту.

Читабельність є однією з основних вимог до тексту в веб-дизайні. Текст має бути легким для сприйняття, що досягається шляхом правильного вибору шрифтів, розмірів, міжрядкових інтервалів та іншого налаштування типографії. Вибір шрифту повинен забезпечувати його чіткість і комфортне читання, особливо на екранах різного розміру.

– Ієрархія тексту та його структурованість.

Ієрархія тексту є важливою складовою організації інформації на веб-сторінці. Вона дозволяє користувачам швидко орієнтуватися і знаходити необхідну інформацію, а також зрозуміти її важливість. Чітке виділення заголовків, підзаголовків і основного тексту за допомогою різних шрифтів, стилів, розмірів та ваги шрифтів допомагає створити візуальну структуру, яка полегшує сприйняття контенту. Заголовки та підзаголовки повинні бути помітними та контрастними, в той час як основний текст має залишатися комфортним для читання, не відволікаючи увагу від основних ідей.

– Консистентність та використання стилів.

Консистентність у використанні типографічних елементів на всіх сторінках веб-сайту є важливим аспектом у створенні зручного інтерфейсу. Використання однакових шрифтів, стилів та налаштувань типографії на різних сторінках забезпечує узгодженість дизайну, що допомагає користувачам легше орієнтуватися в контенті. У веб-дизайні доцільно використовувати визначені стилі для заголовків, підзаголовків і основного тексту, а також для інших елементів, таких як списки або цитати. Це не тільки зберігає єдність оформлення, але й полегшує редагування тексту, дозволяючи змінювати стиль тексту в одному місці, що автоматично оновлюється на всіх сторінках.

– Контрастність та колірна палітра.

Контраст між текстом і фоном є ключовим фактором у забезпеченні зручності читання. Недостатній контраст може призвести до труднощів у сприйнятті тексту, що негативно позначається на користувацькому досвіді. Веб-дизайнери повинні обирати кольори тексту та фону так, щоб вони були достатньо контрастними для легкого сприйняття навіть при слабкому освітленні або на екранах з низькою роздільною

здатністю. Оскільки кольори можуть впливати на емоційне сприйняття інформації, важливо використовувати кольорову палітру, яка відповідає бренду та дизайну сайту.

- Адаптивність типографіки.

Адаптивність тексту є важливою складовою у сучасному веб-дизайні, оскільки користувачі звертаються до веб-ресурсів з різних пристроїв (настільних комп'ютерів, планшетів, мобільних телефонів). Текст на веб-сторінках повинен адаптуватися до різних розмірів екрану та забезпечувати зручність читання на будь-якому пристрої. Окрім того, адаптивність типографії включає в себе оптимізацію інтервалів між рядками, міжлітерного інтервалу та розмірів шрифтів, що змінюються в залежності від розміру екрану.

- Мінімалізм та чіткість в оформленні тексту.

Мінімалізм є важливою складовою сучасного веб-дизайну. Занадто багато тексту чи надмірні декоративні шрифти можуть перевантажити сторінку та знизити зручність навігації. Дизайнери повинні забезпечити чіткість та лаконічність текстових елементів, щоб користувачі могли швидко знайти необхідну інформацію без зайвих зусиль. Слід уникати використання великої кількості різноманітних шрифтів або занадто складних шрифтів, оскільки це може зробити дизайн перевантаженим і менш естетичним. Простота у виборі шрифтів і стилів дозволяє тексту бути більш зрозумілим для користувача.

14.2. Створення текстових об'єктів.

Figma надає широкий спектр інструментів для роботи з текстом, які дозволяють створювати, редагувати й налаштовувати текстові елементи відповідно до вимог проєкту. Завдяки інтуїтивному інтерфейсу та можливостям хмарного середовища, цей процес є зручним і ефективним.

Для створення текстового об'єкта у Figma необхідно скористатися інструментом Text, який можна активувати за допомогою панелі інструментів або клавіші T. Клікнувши на полотні, дизайнер створює текстовий об'єкт із фіксованою шириною, а введення тексту автоматично адаптує його висоту. Якщо необхідно створити текстовий блок із заданими межами, слід натиснути й перетягнути мишу, щоб визначити область, у межах якої текст буде автоматично переноситися.

Figma дозволяє гнучко налаштовувати параметри тексту. У панелі властивостей доступні опції вибору шрифту, його розміру, стилю (жирний, курсив, підкреслення), а також вирівнювання тексту та міжрядкових інтервалів. Однією з переваг Figma є можливість застосування відносних одиниць, таких як Auto для міжрядкових інтервалів, що забезпечує динамічне налаштування тексту залежно від його розміру. Крім того, у Figma доступна інтеграція з бібліотеками шрифтів, такими як Google Fonts, що значно розширює можливості вибору типографіки.

Особливу увагу слід приділити налаштуванню текстових стилів. Figma дозволяє створювати й зберігати стилі тексту, які можуть бути використані повторно в межах одного проєкту або навіть у кількох проєктах. Це особливо корисно для підтримання єдиної стилістики, наприклад, у дизайні кнопок, заголовків і підзаголовків. Зміни в текстових стилях автоматично застосовуються до всіх елементів, які їх використовують, що значно спрощує процес редагування.

Інструменти вирівнювання та організації текстових об'єктів також є важливою частиною роботи у Figma. Можна легко розташовувати текстові елементи відносно інших об'єктів на полотні, використовуючи сітки, направляючі або функцію автоматичного

вирівнювання. Наприклад, текст може бути прив'язаний до певної сітки, що забезпечує точність і акуратність у макеті.

Додатково Figma підтримує багатомовний текст і дозволяє працювати з різними системами письма. Це зручно для міжнародних проєктів, де важливо перевіряти, як текст відображається різними мовами. Для цього можна використовувати функцію вставки тексту в реальному часі або додавати текстові зразки для перевірки адаптивності.

14.3. Налаштування стилю тексту.

Основою налаштування стилю тексту є панель властивостей, яка надає доступ до таких параметрів, як шрифт, розмір, жирність, курсив, міжрядкові інтервали, вирівнювання та відступи. Figma підтримує широкий вибір шрифтів, у тому числі локальні шрифти, встановлені на комп'ютері, та шрифти з бібліотек, таких як Google Fonts. Це дозволяє адаптувати стиль тексту до потреб конкретного проєкту, враховуючи його аудиторію, мету та контекст.

Однією з найпотужніших функцій Figma є створення текстових стилів, які можна зберігати та повторно використовувати. Щоб створити стиль, дизайнер налаштовує параметри тексту для конкретного об'єкта, а потім зберігає їх як стиль, доступний у бібліотеці проєкту. Наприклад, можна створити стилі для заголовків, підзаголовків і основного тексту, кожен із яких буде мати унікальні налаштування розміру, шрифту та міжрядкових інтервалів. Це забезпечує узгодженість типографіки в усьому дизайні, а також дозволяє швидко оновлювати стилі, якщо виникає потреба у зміні.

Налаштування міжрядкових інтервалів і відступів є важливим аспектом створення текстових стилів. Figma дозволяє використовувати відносні одиниці, такі як Auto, для автоматичного підлаштування міжрядкових інтервалів залежно від розміру тексту. Це забезпечує динамічність і адаптивність тексту в різних умовах відображення. Відступи між абзацами та вирівнювання тексту також налаштовуються індивідуально, в залежності від композиції дизайну.

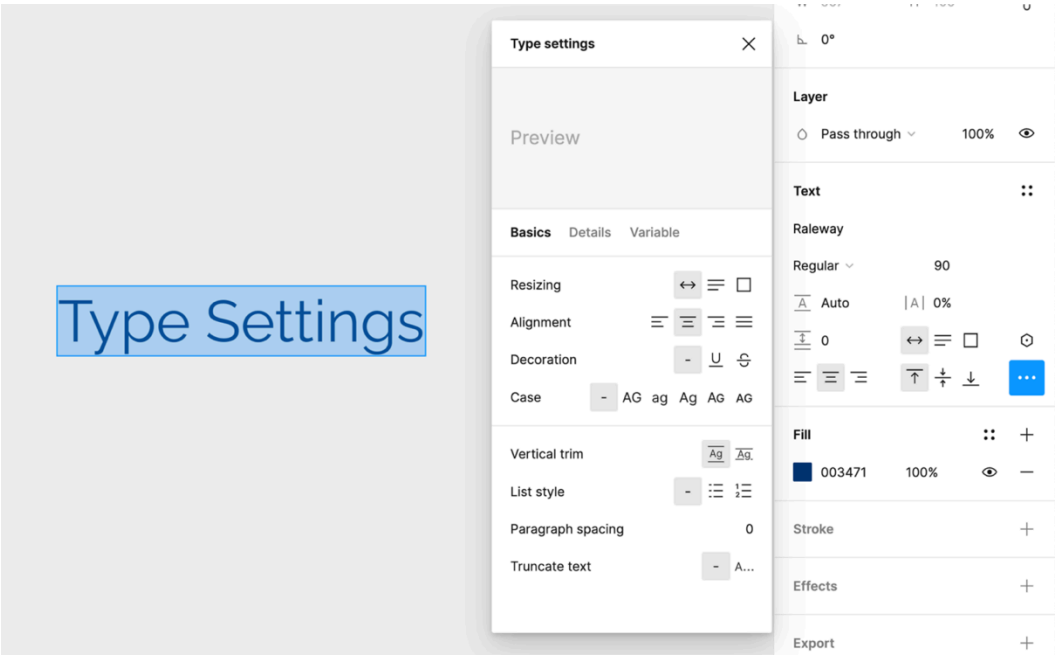

Рис. 14.2. Панель для налаштування шрифтів у Figma

Іншим важливим аспектом є інтеграція текстових стилів із компонентами. У Figma текстові стилі можуть бути прив'язані до компонентів, таких як кнопки, картки або заголовки. Це означає, що зміна стилю автоматично оновлює всі компоненти, які його використовують, що значно зменшує час на ручне редагування. Наприклад, якщо потрібно змінити шрифт у всіх заголовках, достатньо оновити відповідний стиль, і зміни будуть застосовані автоматично.

Також можна працювати з різними рівнями прозорості, кольорами тексту та ефектами, такими як тіні або обведення. Усі ці параметри можна зберігати в текстових стилях, що забезпечує їх повторне використання та узгодженість у дизайні.

14.4. Бібліотека шрифтів.

Завдяки інтеграції локальних і хмарних шрифтів, Figma надає гнучкість у виборі та налаштуванні типографіки відповідно до потреб конкретного проєкту.

Figma підтримує використання локальних шрифтів, які встановлені на комп'ютері користувача. Для цього необхідно завантажити та встановити спеціальний додаток Figma Font Helper, який забезпечує доступ до локальних шрифтів у браузерній версії Figma. У настільній версії програми цей функціонал доступний автоматично. Це дозволяє використовувати корпоративні або спеціально створені шрифти, які не входять до стандартних бібліотек.

Ще однією важливою особливістю є інтеграція з Google Fonts, яка забезпечує доступ до великої кількості хмарних шрифтів. Це особливо зручно для командної роботи, оскільки всі учасники проєкту можуть працювати з однаковими шрифтами без необхідності їх завантаження на локальні пристрої. Google Fonts пропонує безкоштовні шрифти з різними стилями, що дозволяє легко знайти рішення для будь-якого проєкту, від мінімалістичних інтерфейсів до складних дизайнерських концепцій.

У Figma також доступні інструменти для управління шрифтами в межах проєкту. Наприклад, дизайнери можуть створювати текстові стилі, які прив'язуються до певних шрифтів, їх розмірів і властивостей. Ці стилі зберігаються у бібліотеках і можуть бути використані в різних файлах проєкту. Завдяки цьому забезпечується узгодженість типографіки та спрощується процес внесення змін. Якщо потрібно змінити шрифт у всьому проєкті, достатньо оновити відповідний стиль, і зміни автоматично застосуються до всіх елементів, які його використовують.

Крім того, Figma підтримує багатомовні шрифти, що дозволяє працювати з текстом на різних мовах, включаючи мови з нелатинськими системами письма, такими як арабська, китайська чи кирилиця. Ще одним корисним аспектом є можливість попереднього перегляду шрифтів. У Figma можна швидко переглянути, як виглядає текст із різними шрифтами, що значно спрощує процес вибору. Також доступні опції пошуку за назвою шрифту та фільтрації за стилями, що допомагає швидко знайти потрібний варіант.

Контрольні питання

1. Дайте визначення поняттю типографіки у веб-дизайні.
2. Згадайте правила типографіки у веб-дизайні.
3. Поясніть, як створюються текстові об'єкти у Figma.

4. Опишіть, які основні інструменти роботи із текстом існують в програмному середовищі Figma.

Лекція 15. Робота з системою шарів у Figma.

План лекції:

15.1. Створення і налаштування фреймів в середовищі Figma.
15.2. Поняття масок об’єктів.
15.3. Групування об’єктів.
15.4. Базові ефекти шарів та їх налаштування.

15.1. Створення і налаштування фреймів в середовищі Figma.

Інструмент Frame дозволяє створювати області, що визначають межі та структуру інтерфейсу, забезпечуючи логічне розташування елементів і зручність їх подальшого редагування.

Фрейм – це базовий контейнер у Figma, який використовується для організації елементів дизайну на робочому полотні. Фрейми є невід’ємною частиною структури проєктів, оскільки вони дозволяють створювати окремі робочі області для компонентів, сторінок, секцій або повноцінних екранів інтерфейсу.

Фрейми виконують функцію контейнерів, які можуть містити інші об’єкти, такі як текст, фігури, зображення, групи та вкладені фрейми. Вони є аналогом “артбордів” у інших графічних редакторах, проте мають розширений функціонал, зокрема підтримку адаптивного дизайну та обмежень. У межах фрейму також можна задавати обмеження (Constraints), які регулюють розташування та поведінку об’єктів при зміні розмірів фрейму. Це особливо корисно для створення інтерфейсів, що адаптуються до різних розмірів екранів (мобільних пристроїв, планшетів, десктопів тощо).

Фрейми можуть містити інші фрейми або компоненти, що дозволяє створювати складні, багаторівневі структури. Фрейми також підтримують функцію експорту як окремих елементів дизайну у форматах PNG, SVG, JPG або PDF. Крім того, фрейми є основою для побудови інтерактивних прототипів, де кожен фрейм може виступати окремим екраном користувацького інтерфейсу.

Для створення фреймів у Figma використовується спеціальний інструмент, який доступний у верхньому меню або через гарячу клавішу (за замовчуванням клавіша F). Після активації інструменту користувач може вибрати готові шаблони фреймів із панелі властивостей, наприклад, стандартні розміри для мобільних пристроїв, планшетів, настільних комп’ютерів чи соціальних мереж. Це значно спрощує початкову роботу, дозволяючи швидко адаптувати дизайн під конкретний формат. Крім того, є можливість створювати фрейми з довільними розмірами.

Фрейми виконують кілька важливих функцій у дизайні. По-перше, вони забезпечують структурну організацію, дозволяючи групувати елементи, такі як текст, зображення, іконки та кнопки, у межах одного контейнера. Це сприяє кращій візуальній

ієрархії та зручності управління елементами. По-друге, фрейми виступають основою для створення адаптивних дизайнів, оскільки їх можна налаштовувати так, щоб вони змінювали свої розміри залежно від контексту або пристрою, на якому відображатиметься інтерфейс.

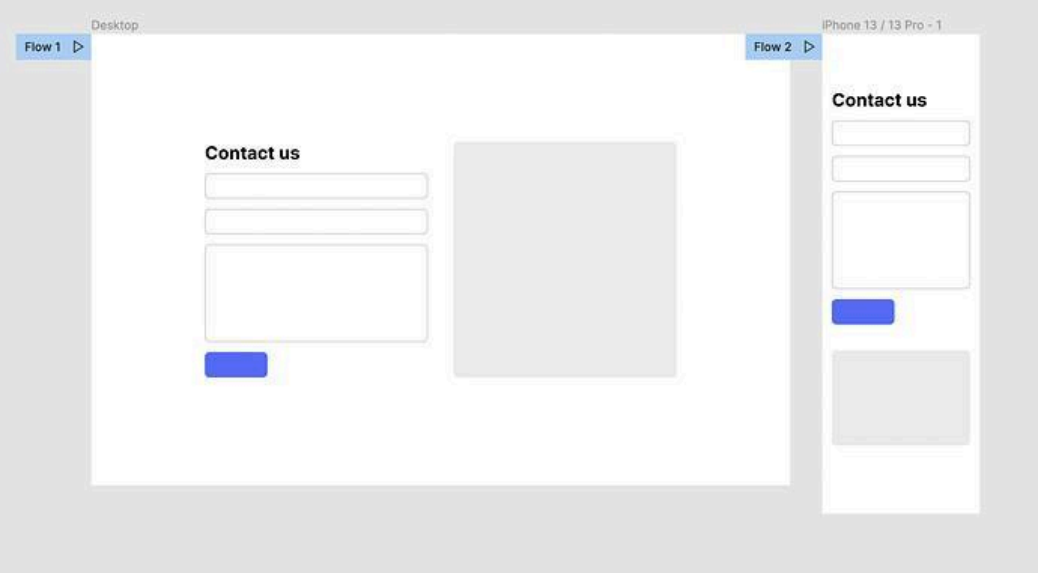

Рис. 15.1. Створення фрейму у середовищі Figma

Особливістю фреймів у Figma є їх багатофункціональність. Вони можуть виконувати роль звичайних контейнерів для вмісту або слугувати окремими сторінками, секціями чи компонентами. Наприклад, один фрейм може представляти екран мобільного додатка, а інший – спливаюче вікно або меню. Завдяки цьому, дизайнери можуть створювати багаторівневі структури, які легко масштабуються та адаптуються до змін у проєкті. Також фрейми легко інтегруються з іншими елементами дизайну. Наприклад, їх можна вкладати один в одного, створюючи складні композиції. Крім того, фрейми підтримують використання напрямних і сіток, що допомагає забезпечити точне вирівнювання та симетрію елементів.

Налаштування фреймів у Figma є ключовим етапом роботи над проєктом. Усі властивості фреймів доступні через панель інструментів праворуч, де користувач може змінювати їхні розміри, позицію, зовнішній вигляд і взаємодію з іншими елементами.

Однією з основних можливостей є зміна розмірів фреймів. Figma дозволяє вручну задавати ширину та висоту або використовувати функцію автоматичного масштабування, яка підлаштовує розміри фрейма під його вміст. Крім того, інструменти вирівнювання допомагають точно розташовувати фрейми на полотні. Це особливо важливо для створення складних макетів, де точність розташування елементів має вирішальне значення.

Також фрейми підтримують налаштування візуальних властивостей, таких як колір заповнення, обведення та тіні. Наприклад, користувач може задати однотонний фон, градієнт або навіть використати зображення як заповнення. Опції обведення дозволяють створювати рамки різної товщини та стилю, а функція додавання тіней додає глибини та об’єму. Усі ці властивості легко налаштовуються через інтуїтивно зрозумілий інтерфейс.

Для більш складних проєктів важливим аспектом є робота з адаптивністю. Фрейми можуть бути налаштовані так, щоб змінювати свої розміри залежно від параметрів батьківського контейнера або екрану. Це забезпечується через функції Auto Layout, які дозволяють задавати правила поведінки елементів усередині фрейма: вирівнювання, відступи, розтягнення або фіксація розмірів. За допомогою цих інструментів дизайнери можуть створювати макети, які автоматично адаптуються до різних розмірів екранів і пристроїв.

Ще однією важливою функцією є інтерактивність фреймів. Figma дозволяє створювати інтерактивні прототипи, додаючи до фреймів переходи, анімації та тригери. Наприклад, користувач може встановити налаштування таким чином, щоб при натисканні на кнопку відкривався інший фрейм або виконувався певний анімаційний ефект. Ці можливості роблять фрейми не лише статичними контейнерами, а й інтерактивними елементами, які підвищують якість презентації та тестування дизайну веб-сторінки.

15.2. Поняття масок об’єктів.

Маски – це інструмент, що дозволяє приховувати або виділяти певні частини вмісту, забезпечуючи точний контроль над видимістю елементів у межах веб-дизайну. Маски використовуються для створення складних композицій, акцентування уваги на окремих деталях або обмеження видимості зображень, текстів чи інших графічних об’єктів у межах заданої форми.

У Figma маска працює за принципом визначення області видимості: елементи, що знаходяться в межах форми маски, залишаються видимими, тоді як усе, що виходить за її межі, стає прихованим. Будь-який об’єкт, наприклад, прямокутник, коло або навіть складна векторна фігура, може бути використаний як маска. Для застосування маски достатньо вибрати бажаний об’єкт і активувати функцію Mask через контекстне меню або гарячу клавішу Ctrl + Alt + M (на Windows) чи Cmd + Option + M (на Mac).

Маски у Figma інтегруються в робочий процес завдяки своїй гнучкості. Вони можуть застосовуватися до одного або кількох шарів одночасно, що дозволяє створювати багатошарові композиції. Наприклад, можна накласти зображення на текст, використовуючи текстовий об’єкт як маску, що створює ефект заповнення тексту зображенням. Аналогічно, маски часто застосовуються для обрізання фотографій у межах заданої форми, наприклад, для створення аватарів або іконок.

Ще однією перевагою масок у Figma є їхня динамічність. Маски не є остаточними і можуть редагуватися в будь-який момент. Користувач може змінювати форму маски, її розташування або розміри, а також додавати нові об’єкти до групи, на яку поширюється дія маски. Це забезпечує високу гнучкість і зручність у роботі, особливо на етапах прототипування та внесення змін до дизайну. Figma також підтримує комбінування масок із іншими інструментами, такими як градієнти, ефекти тіней або прозорості, що відкриває додаткові можливості для створення унікальних візуальних ефектів. Наприклад, можна використовувати маску для часткового затемнення зображення, створення плавного переходу або виділення певної області без зміни самого зображення.

15.3. Групування об’єктів.

Групування об’єктів дозволяє об’єднувати кілька об’єктів у одну групу, що спрощує їхнє переміщення, масштабування, вирівнювання та інші маніпуляції. Групи є особливо

корисними при роботі з великими проєктами, де потрібно підтримувати порядок і зручність редагування.

Створення групи у Figma здійснюється простим вибором кількох елементів і активацією функції Group через контекстне меню або гарячу клавішу Ctrl + G (на Windows) чи Cmd + G (на Mac). Усі об'єкти всередині групи залишаються незалежними, але об'єднуються в один шар, що відображається в панелі шарів. Це дає змогу виконувати операції з усією групою, не впливаючи на її внутрішню структуру.

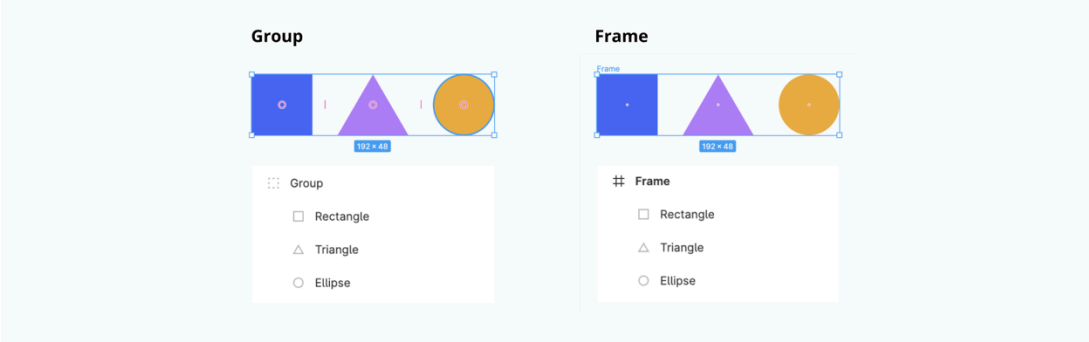

Рис. 15.2. Створення групи у Figma

Групи забезпечують гнучкість у налаштуванні дизайну. Наприклад, можна змінювати положення або розміри групи, зберігаючи відносне розташування елементів усередині неї. Це особливо зручно при роботі з комплексними інтерфейсами, де об'єкти часто формують логічні блоки, такі як кнопки, панелі навігації чи картки контенту.

Figma також підтримує вкладене групування, коли одна група може містити інші групи. Це дозволяє створювати багаторівневу структуру, яка полегшує роботу з великими макетами. Наприклад, вкладені групи можуть використовуватися для організації компонентів, що повторюються, таких як списки або таблиці.

Крім того, групування полегшує взаємодію з іншими функціями Figma. Зокрема, Auto Layout може бути застосований до груп, забезпечуючи автоматичне вирівнювання та адаптацію елементів у залежності від їхнього вмісту. Групи також можуть бути стилізовані: до них можна застосовувати кольори, градієнти, ефекти тіней і навіть маски, що додає гнучкості у створенні дизайну.

Окремо варто зазначити можливість розгрупування об'єктів. Якщо група більше не потрібна, її можна легко розпустити, зберігаючи всі елементи. Це дає змогу вносити зміни до структури макета без втрати даних або взаємозв'язків між об'єктами.

15.4. Базові ефекти шарів та їх налаштування.

Тінь дозволяє симулювати взаємодію елементів із джерелами світла, надаючи їм реалістичності та відокремлюючи від фону або інших об'єктів. У контексті інтерфейсного дизайну, тіні широко використовуються для виділення активних елементів, таких як кнопки, картки або модальні вікна, а також для створення візуальної ієрархії.

У Figma тінь налаштовується через панель Effects, де користувач може обрати параметри, що впливають на її вигляд. Серед основних налаштувань тіні – Offset X і Offset Y, які визначають горизонтальне та вертикальне зміщення тіні відносно об'єкта. Це дозволяє створювати ефект напрямку світла, наприклад, імітуючи тінь, що падає зліва або справа.

Параметр Blur регулює ступінь розмиття тіні. Чим вищий показник, тим м’якшою і менш різкою виглядає тінь, що ідеально підходить для створення природного вигляду. Навпаки, низький рівень розмиття робить тінь чіткішою, що може використовуватися для створення технічного або графічного ефекту.

Opacity або прозорість дозволяє контролювати інтенсивність тіні. Знижена прозорість робить тінь менш помітною, що підходить для ненав’язливих елементів, тоді як висока прозорість акцентує увагу на об’єкті. Крім того, Figma дозволяє змінювати колір тіні, що дає змогу адаптувати її до загальної кольорової гами проєкту. Наприклад, кольорові тіні можуть створювати сучасний і динамічний вигляд, особливо в креативних дизайнах.

Figma також підтримує можливість накладення кількох тіней на один об’єкт. Це відкриває додаткові можливості для експериментів із глибокими шарами або складними світловими ефектами. Наприклад, можна створити ефект подвійного освітлення або м’якого відбиття світла.

Ефект розмиття – ще один популярний ефект шарів у Figma, що є потужним інструментом для створення візуальної глибини, фокусування уваги та надання дизайну естетичної привабливості. Цей ефект використовується для виділення ключових елементів інтерфейсу, створення фону, що не відволікає, або імітації фізичних властивостей, таких як матове скло. У Figma доступно два основних типи розмиття: Layer Blur та Background Blur, кожен із яких має свої унікальні властивості та застосування.

Layer Blur впливає безпосередньо на шар або об’єкт, до якого застосовується. Цей тип розмиття рівномірно розсіює контури об’єкта, роблячи його менш чітким. Layer Blur ідеально підходить для створення ефекту фокусування, коли увага користувача спрямовується на більш чіткі елементи дизайну, залишаючи розмиті деталі у фоновому режимі. Інтенсивність розмиття регулюється через параметр Blur Radius, який задає ступінь розсіювання пікселів.

Background Blur, на відміну від Layer Blur, працює з об’єктами, які знаходяться за межами вибраного шару. Цей ефект дозволяє розмивати фон, зберігаючи чіткість переднього плану. Background Blur широко використовується для створення сучасних інтерфейсів, наприклад, у модальних вікнах або спливаючих панелях, де важливо зберегти видимість фону, але зменшити його вплив на сприйняття основного контенту. Налаштування Background Blur включає регулювання інтенсивності розмиття та можливість додавання прозорості для створення більш реалістичного вигляду.

Використання ефекту розмиття вимагає врахування кількох дизайнерських аспектів. По-перше, важливо дотримуватися балансу між розмитими та чіткими елементами, щоб не перевантажити інтерфейс і забезпечити комфортне сприйняття. По-друге, варто враховувати контекст використання: для функціональних елементів, таких як кнопки чи текст, надмірне розмиття може ускладнити взаємодію. По-третє, розмиття слід використовувати з урахуванням загальної стилістики проєкту, щоб воно гармонійно поєднувалося з іншими елементами дизайну.

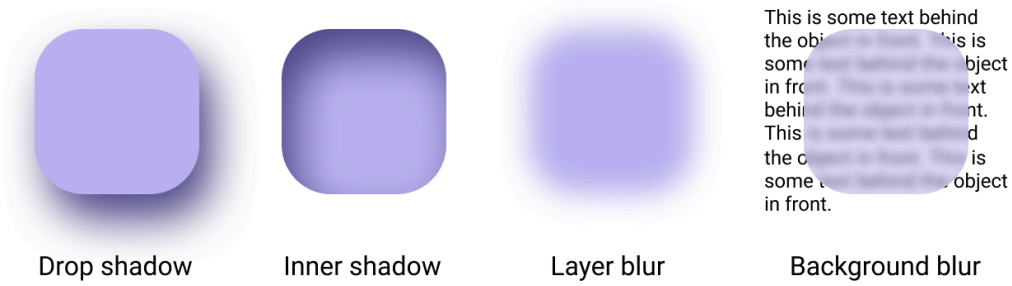

Рис. 15.3. Тіні та ефект розмиття у середовищі Figma

Прозорість визначає ступінь видимості шару або ефекту, що впливає на сприйняття композиції, а налаштування кольору дозволяє адаптувати елементи до загальної стилістики дизайну.

Прозорість в Figma регулюється через параметр Opacity, який задається у відсотках. Значення 100% відповідає повній непрозорості об’єкта, тоді як 0% робить його повністю прозорим. Ця функція є корисною для створення багатошарових композицій, де об’єкти накладаються один на одного, формуючи складні візуальні ефекти. Наприклад, зниження прозорості може використовуватися для створення підлеглих елементів, які не відволікають увагу від основного контенту.

Налаштування кольору ефектів, таких як тіні або розмиття, дозволяє дизайнерам досягати гармонії між елементами інтерфейсу. У Figma це реалізується через палітру кольорів, яка включає як стандартні кольори, так і можливість створення власних відтінків. Використання кольорових ефектів може додати дизайну унікальності та сучасності. Наприклад, кольорові тіні або світіння використовуються в креативних проєктах для створення футуристичного або грайливого вигляду.

Особливу увагу варто приділити комбінуванню прозорості та кольору. Наприклад, прозорі кольорові тіні можуть створювати м’які переходи між елементами, забезпечуючи при цьому глибину та об’єм. Для таких ефектів важливо враховувати контрастність між фоном і об’єктом, щоб зберегти читабельність і зручність використання інтерфейсу.

Контрольні питання

1. Опишіть, що таке фрейм і як його створити у середовищі Figma.
2. Розкрийте, які основні відмінності груп об’єктів та фреймів.
3. Поясніть, який принцип ієрархії шарів використовується у Figma.
4. Згадайте, які основні параметри обмежень (Constraints) передбачені у Figma.
5. Вкажіть, яким чином здійснюється групування об’єктів у Figma.

СПИСОК ЛІТЕРАТУРИ

Основна література:

1. Бородкіна І.Л., Бородкін Г.О. Web-технології та Web-дизайн: застосування мови HTML для створення електронних ресурсів. − Київ: Ліра-К, 2020. − 212 с. (28 примірників)
2. Пасічник В.В., Пасічник О.В., Угрин Д.І. Веб-технології. − Львів: Видавництво "Магнолія", 2024. − 336 с. (11 примірників)
3. Мельник Р.А. Програмування веб-застосувань (фронт-енд та бек-енд). – Львів: Видавництво Львівської політехніки, 2018. – 247 с. (19 примірників)

4. Стаяно Ф. FIGMA. Проектування і прототипування інтерфейсів відповідно до принципів UX/UI. – Харків: Фоліо, 2021. – 370 с. (7 примірників)
5. Пасічник О.Г., Пасічник О.В., Стеценко І.В. Основи веб-дизайну: [Навч. посіб.]. – К.: Вид. група BHV,2009. – 336 с. (16 примірників)

Додаткова література:

6. Емброуз Г., Леонард Н. Основи графічного дизайну 02. Дизайнерське дослідження. Пошук успішних креативних рішень. − Пер. з англ. – К.: ArtHuss, 2019. – 192 с.
7. Moore S., Wilkins A. Strategic Web Design & e-Commerce. – Mujo Learning Systems, 2021. – 385 p.
8. McGrath M. HTML, CSS & JavaScript in easy steps. – UK: In Easy Steps Limited, 2020. – 480 p.