Правила використання:

* Стартовий шаблон "Чертоги Фрілансера" 4 надається в знак вдячності за благодійну підтримку каналу "Фрілансер по життю" на патреоні
* Збірку, або її частини заборонено продавати у будь-якому вигляді
* Збірку заборонено виставляти у відкритий доступ, або передавати комусь
* Збірку можна використовувати для власних цілей в тому числі для розробки комерційних проєктів
* Збірку заборонено використовувати в росії

**Документація збірки (доповнюється)**: <https://www.patreon.com/collection/1439491?view=condensed>  
**Оновлення** **збірки:** <https://www.patreon.com/posts/126345239?collection=1439491>

При винекненні питань та пропозицій пишіть у чат патронів у дискорді!

Для додавання у **закритий чат патронів** виконайте інструкцію:

* Зареєструватись ([https://discord.com](https://discord.com/)) у дискорд (це  месенджер де знаходиться чат)
* На  патреон-сторінці  "Фрілансер по життю" натиснути на  вкладку "Моя  підписка" (<https://www.patreon.com/freelancerlifestyle/membership).>
* Перейшовши, пролиснути нижче та знайти розділ "Спільнота Discord", де є кнопка "Підключити Discord",
* Натиснувши яку, відбувається підключення вас до чату  "Патронес".
* Після цього чат "Патронес" з'явиться у вашому дискорді.

Загальний перелік функціоналу самої збірки без урахування готових JS/SCSS рішень:

▶️ Компонентна система з "гарячою" підключкою модулів  
▶️ Можливість розробляти під CMS WordPress (Локальний сервер Docker)  
▶️ Конвертація та підключення локальних шрифтів  
▶️ Конвертация та підключення WEBP/AVIF зображень з широким функціоналом  
▶️ Можливість розбиття JS / CSS файлів по сторінках та створення нестиснутих версій  
▶️ Можливість використання PHP рішень, таких як PHPMailer прямо на локальному сервері  
▶️ Створення критичних CSS стилів  
▶️ Навігаційна панель по сторінках проєкту  
▶️ Можливість використання REACT, VUE, Tailwindcss  
▶️ Автоматичне налаштування редактора VS Code - сніпети, аліаси  
▶️ Створення ZIP архіву, та відправка по FTP  
▶️ Та багато іншого

[02/06/2025] Оновлення

* Виправлення помилок панелі розробника, додавання версії файлів, підключення reset.css, плагіну header
* Допрацьований компонент обробки зображень

[29/05/2025] Оновлення

* Виправлення компоненту "Модальне вікно"

[27/05/2025] Оновлення

* Виправлення помилки в компоненті "До і після"
* Додана можливість працювати з власними окремими JS-файлами
* Додані сніпети для роботи з SVG-спрайтом

[22/05/2025] Оновлення

* Додана окрема сторінка для зручної розробки компонентів
* Виправлена помилка компонента "Статистика"
* Оновлена документація

[19/05/2025] Оновлення

* Виправлення роботи компоненту цифрового лічильника
* Додана можливість очищення статистики проєкту
* Додане підключення частих функцій в файл app.js
* Додана можливість підключення окремого файлу CSS-змінних в style.scss
* Змінена документація

**[17/05/2025] Оновлення**

* Виправлені компоненти "Селект" та "Календар"
* Допрацьований функціонал створення розмірів зображень

**[14/05/2025] Оновлення**

* Виправлена робота з WP та інших дрібних помилок

**[11/05/2025] Оновлення**

* Виправлення дрібних помилок

**[09/05/2025] Оновлення**

* Покращений вивід статистики
* Допрацьований компонент "Кількість"
* Виправлення дрібних помилок

**[05/05/2025] Оновлення**

* Покращена статистика - додана кількість редагованих файлів
* Вимкнена примусова конвертація SVG в data
* Виправлення дрібних помилок

**[03/05/2025] Оновлення**

* Додана можливість збору статистики вистраченого часу на проєкт
* Додана документація

**[01/05/2025] Оновлення**

* Додана можливість завантаження зібраного проєкту у GitHub репозиторій
* Допрацювання роботи компонентів у вкладених PHP-файлах теми WordPress

**[29/04/2025] Оновлення**

* Виправлення дрібних помилок
* Допрацьована взаємодія SCSS з модулями
* Допрацьований логінг в терміналі

**[25/04/2025] Оновлення**

* Виправлення дрібних помилок
* Архів ЧФ4 перенесений в [окрему статтю документації](https://www.patreon.com/posts/127347092?collection=1439491)

**[23/04/2025] Оновлення**

* Ім'я архіву ЧФ4 тепер має дату та час створення
* Допрацьована оптимізація SVG-іконок
* Доданий компонент "Кошик"
* Додана документація

**[21/04/2025] Оновлення**

* Виправлення помилки "Динамічного адаптиву" при збірці проєкту
* Дрібні покращення різних елементів та налаштувань
* Додана документація

**[20/04/2025] Оновлення**

* Дрібні покращення різних елементів та налаштувань
* Додана документація

**[15/04/2025] Оновлення**

* Виправлення помилки формування кінцевих файлів при збірці для CMS WodPress
* Додана можливість швидкого створення [власних компонентів](https://www.patreon.com/posts/126687222?collection=1439491) командою в терміналі
* Додана документація

**[14/04/2025] Оновлення**

* Виправлення помилки обробки вкладених зображень
* Виправлення помилки шляху завантажених відалених шрифтів
* Додана документація

**[13/04/2025] Оновлення**

* Покращена обробка імен шрифтів
* Додана оптимізація та чистка SVG-іконок для спрайту
* Додана документація

**[12/04/2025] Оновлення**

* Виправлення помилки при збірці сату на WordPress
* Додана документація

**[11/04/2025] Оновлення**

* Додана можливість створення копій зображеннь та інтеграція в HTML під різні DPI екранів
* Додана документація щодо роботи з зображеннями, оновленна стаття про налаштування ЧФ4

**[10/04/2025] Реліз**

Встановлення та запуск «Чертогів Фрілансера»

4. Підготовка до роботи

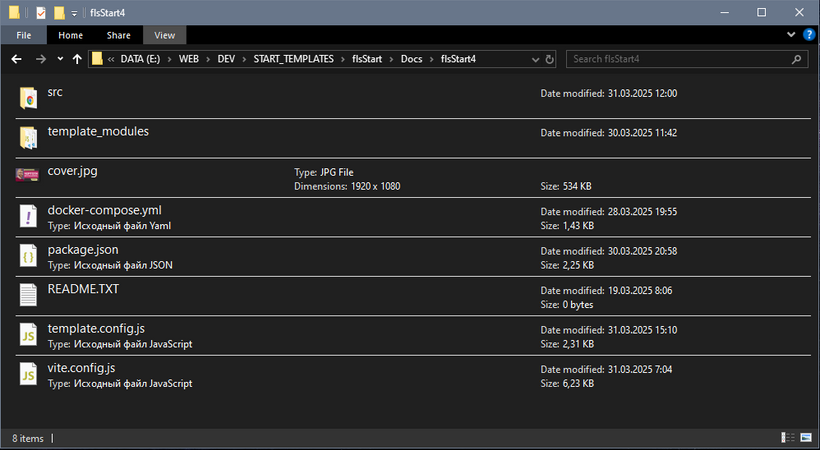
Нове

10 квітня

Отже, ви завантажили шаблон "Чертоги Фрілансера" 4 (далі ЧФ4), які далі кроки?

Встановлення шаблону

Насамперед слід розархівувати zip-архів у вашу папку проекту. Слідкуйте за тим, щоб папка та її батьківські папки не містили в назві кирилицю, пробіли, символи # та !

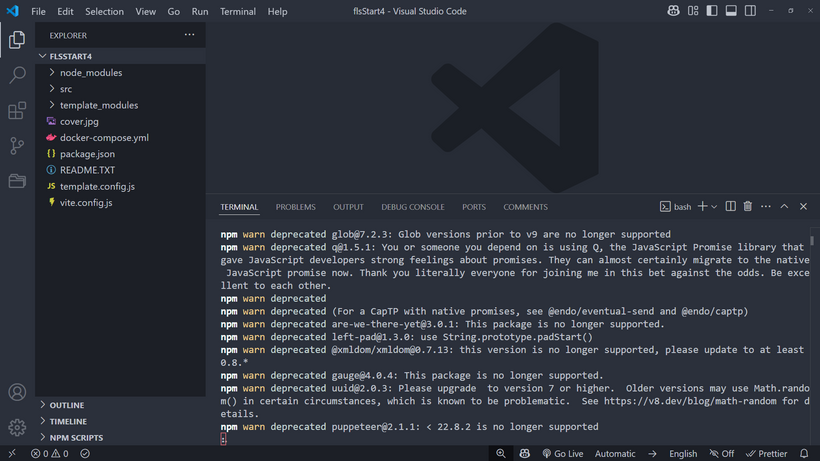
Вміст архіву в папці проекту

Якщо ви до цього моменту не користувалися NodeJS та пакетним менеджером NPM, слід [завантажити та встановити Node.js](https://nodejs.org/en/download). Завантажуйте версію рекомендовану для більшості.

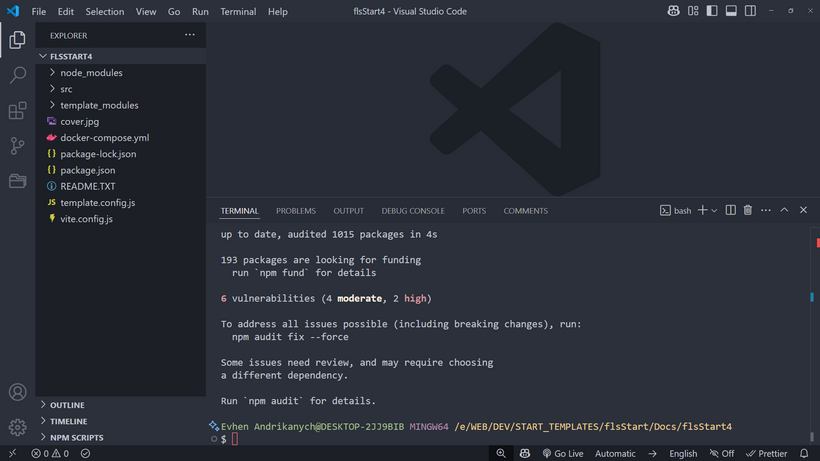
Далі відкриваємо термінал у цій папці проекту. Термінал може бути вбудований або відкритий окремо від редактора. Рекомендую використовувати термінал [GIT Bash](https://git-scm.com/downloads).

Після завершення встановлення необхідних інструментів введіть команду **npm i**

У процесі установки в терміналі можуть бути повідомлення з позначкою WARN жовтим кольором. Ці повідомлення можна ігнорувати. Але якщо ви отримуєте повідомлення ERR! червоним кольором — це критична помилка та її потрібно виправляти.

Попередження можна ігнорувати

Після успішного встановлення, у вас з'явиться папка **node\_modules** та файл **package-lock.json**

Встановлення завершено

Детальніше про [архітектуру папок та файлів](https://www.patreon.com/posts/126343576?collection=1439491) ЧФ ми поговоримо в наступному розділі, а поки що продовжуємо підготовку до роботи

Запуск шаблону та режими роботи

ЧФ4 може виконувати різні сценарії сценаріїв:

* Режим розробника. Команда запуску **npm run dev**
* Режим розробника з зовнішнім IP. Команда запуску **npm run online**
* Режим продакшену. Команда запуску **npm run build**
* Режим перегляду результату виконання продакшену. Команда запуску **npm run preview**
* Режим продакшена та надсилання результату на сервер по FTP. Команда запуску **npm run deploy**
* Режим продакшена та надсилання результату на GitHub. Команда запуску **npm run git**
* Режим продакшена та створення ZIP-архіву з результатом. Команда запуску **npm run zip**
* Режим розробника під CMS WordPress. Команда запуску **npm run wp**
* Режим продакшена під CMS WordPress. Команда запуску **npm run wpbuild**
* Зупинка Docker-серверу. Команда запуску **npm run wpstop**

Режим розробника

У режимі розробника виконуються лише необхідні для розробки завдання:

* Конвертація OTF/TTF шрифтів у WOFF2 та підключення до проєкту
* Створення іконкового шрифту та підключення до проєкту
* Опрацювання HTML-файлів, перейменування псевдонімів
* Опрацювання препроцесора PUG, перейменування псевдонімів
* Опрацювання JS-файлів
* Опрацювання SCSS/CSS-файлів
* Запускається локальний сервер, відкривається браузер з індексною сторінкою.
* Запускається спостерігач за зміною фалів. При кожній зміні файлів браузер оновлює сторінку
* Увага! В режимі розробника HTML CSS JS файли результату не записуються на диск (**папка dist не створюється**).

Режим продакшену

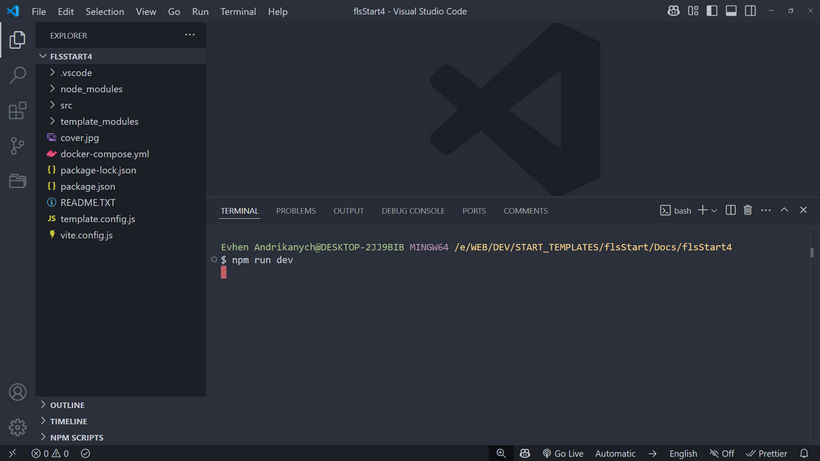
У режимі продакшену виконується фіналізація проекту, а саме:

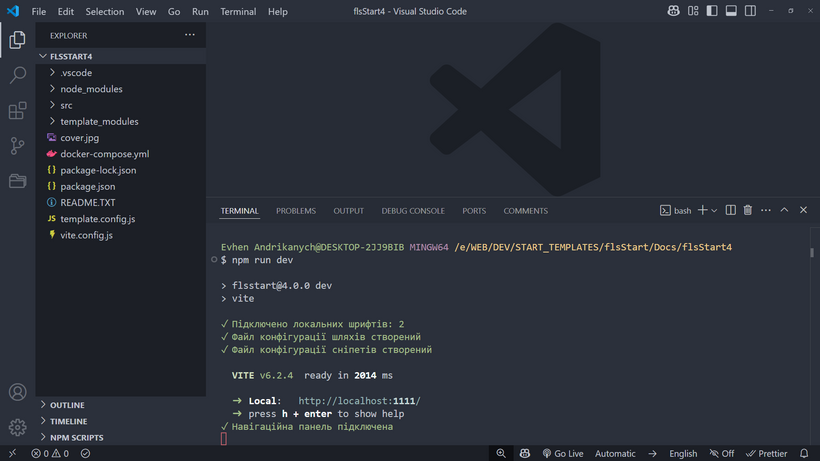
* Конвертація (за потреби) OTF/TTF шрифтів у WOFF2 та підключення до проєкту
* Конвертація зображень у WEBP/AVIF формат. Створення файлів різних розмірів. Стиснення та оптимізація JPG PNG SVG зображень
* Конвертація SCSS файлів у CSS файли, перейменування псевдонімів, групування медіа-запитів, додаються вендорні префікси для забезпечення кросбраузерності, обробляється підключення WEBP/AVIF зображень, проводиться стиснення. Можуть створюватись нестиснуті копії файлів. Можуть створюватись критичні стилі.
* Обробка та форматування HTML файлів, перейменування псевдонімів, обробляється підключення WEBP/AVIF зображень та різних розмірів файлів.
* При використанні PUG, файли перетворюються на HTML, перейменовуються псевдоніми, обробляється підключення WEBP/AVIF зображень та різних розмірів файлів.
* Збираються JS файли, проводиться стиснення та оптимізація кінцевого файла. У результат потрапляє лише використовуваний код. Можуть створюватись нестиснуті копії файлів.
* Копіюються файли із зазначеної папки
* Увага! Всі файли з результатом записуються на диск до папки **dist**, локальний сервер не запускається

Отже, давайте запустимо наш шаблон в режимі розробника, для цього в терміналі виконуємо команду **npm run dev**

Після запуску система виконає всі завдання режиму розробника, описані вище. Результатом роботи має стати відкрита у браузері сторінка змісту.

Увага! При конвертації шрифтів в терміналі буде виведено відповідне повідомлення, користуватись збіркою можна буде після завершення конвертації

Старт виконання команди

Команда успішно виконана

Можливі помилки та їх вирішення

Якщо браузер не запустився, а в терміналі видно помилки (ERR!), переконайтеся що:

* У вас встановлено Node.js та Python останньої версії
* Термінал відкритий з правами адміністратора
* У назвах папок на всьому шляху до проекту немає символу # або !
* Папки та файли мають бути названі латиницею без пробілів

Підготовка редактора до комфортної роботи із шаблоном

Для того щоб насолоджуватися можливостями шаблону ЧФ на повну, нам слід зробити деякі налаштування редактора. Як приклад представлений редактор VS Code

Налаштування псевдонімів

У статті [Архітектура шаблону. Файли та папки](https://www.patreon.com/posts/126343576?collection=1439491) ви дізнаєтеся що різні частини HTML, SCSS, JS файлів знаходяться на різних рівнях вкладеності, що створює певні незручності при підключенні, наприклад картинок, у процесі розробки.

Незручності ми не любимо тому налаштуємо так звані псевдоніми (аліаси) шляхів до папок. Для цього нам потрібно встановити плагін [Path Autocomplete](https://marketplace.visualstudio.com/items?itemName=ionutvmi.path-autocomplete).

За замовченням, ЧФ4 сам налаштує плагін відповидними псевдонімами вказаними у файлі налаштувань збірки. Там же можна вимкнути автоматичне налаштування.

Після цього ви можете сміливо використовувати псевдоніми при підключенні файлів, наприклад:

<img src="@img/cover.jpg" alt="Малюнок">

Редактор розпізнає псевдонім і виведе список файлів у зазначеній папці, а під час збирання система сама поміняє псевдонім на потрібний шлях!

Налаштування сніпетів

Сніпети — це короткі коди, які можуть викликати готові заготовки коду будь-якого обсягу. Це величезно підвищує швидкість розробки.

Наприклад сніпет **spollers + клавіша Tab** виведе код спойлерів:

<div data-fls-spollers class="spollers">  
<details class="spollers\_\_item">  
<summary class="spollers\_\_title">Заголовок спойлера</summary>  
<div class="spollers\_\_body">Контент спойлера</div>  
</details>  
</div>

Звичайно, я використав цю супер можливість у своїх чертогах. Тобто, побудова, наприклад, правильної HTML структури для того чи іншого JS модуля я додав у сніпет.

За замовченням, ЧФ4 автоматично додасть сніпети збірки у ваш проєкт і ви зможете ними користуватись під час розробки. Сніпети ЧФ4 будуть працювати паралельно з вашими сніпетами у редакторі. Автоматичне додавання сніпетів можна вимкнути у налаштуваннях збірки.

Чудово, сніпети ЧФ4 на борту! У цій документації, а також у коді шаблону ви часто будете зустрічати підказки із зазначенням сніпету.

У наступній статті ми познайомимося зі структурою файлів та папок шаблону «Чертоги Фрілансера» 4

Приємної роботи !

Архітектура «Чертогів Фрілансера» 4.

Файли та папки

Нове

10 квітня

У цьому розділі поговоримо про архітектуру, ієрархію папок та файлів, стартового шаблона «Чертоги Фрілансера» 4 (далі ЧФ4)

Розархівувавши збірку, в корені ви виявите такі файли та папки:

* Папка **src** - містить похідні файли та папки проекту. Саме тут працює розробник
* Файл **template.config.js**, містить користувацькі налаштування ЧФ4
* Файл **README.txt** - важлива коротка інформація щодо використання ЧФ4
* Файл **cover.jpg** - обкладинка ЧФ4
* Файл **package.json**, містить команди запуску збирачів, інформацію про встановлені плагіни та їх версії. А також іншу інформацію про автора та версію збірки. Не рекомендується редагувати без відповідних знаннь
* Файл **docker-compose.yml**, містись інструкції щодо встановлення та запуску серверу в режимі *fullstack* для роботи з [CMS WordPress](https://wordpress.com/). Не рекомендується редагувати без відповідних знаннь
* Папка **template\_modules** містить функціонал збірки. Не рекомендується редагувати
* Файл **vite.config.js**, містить інструкції для роботи системи VITE. Не рекомендується редагувати
* При першому запуску будь-якого режиму, з'явиться папка **.vscode**, яка буде містити налаштування редактора та сніпети (якщо це дозволено в налаштуваннях ЧФ4)
* При першому запуску команди npm run zip, з'явиться папка **zip**, яка буде містити zip-архів кінцевих файлів проєкту
* При першому запуску режима *fullstack* (npm run wp, npm run wpbuild), з'явиться папка **backend**, яка містить папки **wordpress** та **database**. Не рекомендується редагувати без відповідних знаннь
* Після інсталяції також автоматично з'являться папка **node\_modules** та файл **package-lock.json**. Не редагувати

Вміст папки src:

* Файл **index.html** - індексна (головна) сторінка проєкту що розробляється
* Папка **assets** - використовується для зберігання медіа-файлів проєкту - зображеннь, шрифтів, іконок тощо
* Папка **styles** - містить загальні файли стілів проєкту
* Папка **js** - містить загальні файли скриптів проєкту, а також, папки та файли бібліотек та фреймворків
* Папка **pug** - містить файли для роботи з препроцесором PUG
* Папка **php** - містить PHP-компоненти та є головною папкою PHP-серверу якщо таке налаштування увімкнене
* Папка **files** (порожня) -  всі файли цієї папки будуть скопійовані в папку з результатом (dist/files)
* Папка **components** - містить усі HTML/CSS/SCSS/JS [компоненти ЧФ4](https://www.patreon.com/posts/126687222?collection=1439491), а також тему для CMS WordPress, шаблони HTML-сторінок та їх головний контент

**Вміст папки src/components:**

Містить усі HTML/CSS/SCSS/JS компоненти ЧФ4, а також тему для CMS WordPress, шаблони HTML-сторінок та їх головний контент

* Папка **custom** (порожня) - тут можна створювати [власні компоненти](https://www.patreon.com/posts/126687222?collection=1439491)
* Папка **wordpress** - містить тему для CMS WordPress, плагін та допоміжні файли. Використовується для розробки інтерфейся під CMS WordPress в режимі *fullstack* (npm run wp, npm run wpbuild)
  + **fls-theme** - тема для CMS WordPress
  + **fls-plugin** - власний плагін для CMS WordPress
  + **JS-файли** - допоміжні файли, не редагувати, не видаляти
* Папка **templates** - містить шаблони HTML-сторінок, можна додавати свої
  + Папка **main** - містить HTML/SCSS/JS файли основного HTML-шаблону проєкту
  + Папка **projectpage** - містить HTML-шаблон сторінки проєкту
* Папка **pages** - містить HTML/SCSS/JS файли основного контенту сорінок проєкту
  + Папка **home** - містить HTML/SCSS/JS файли основного контенту інтексної (головної) сторінки проєкту
* Папка **effects** - містить папки та HTML/SCSS/JS файли компонентів які додають в проєкт різноманітні ефекти. Можна додавати свої
* Папка **forms** - містить папки та HTML/SCSS/JS файли компонентів які додають в проєкт різноманітний функціонал для роботи з формами та їх елементами. Можна додавати свої
* Папка **layout** - містить папки та HTML/SCSS/JS файли компонентів які додають в проєкт різноманітні частини інтерфейсу. Можна додавати свої

Додатково, папки компонентів можуть містити папки та файли плагінів конкретного компоненту, наприклад src/components/layout/header/plugins/scroll

Вміст папки src/assets:

* Папка **fonts** (порожня) - використовується для конвертації та додавання локальних шрифтів в проєкт.
* Папка **img** - використовується для зберігання та використання зображень. Містить файл favicon.ico який підключений до HTML-сторінок (замінюємо на свій)
* Папка **svgicons** (порожня) - використовується для конвертації та додавання в проєкт іконкового шрифту та/або спрайту

Вміст папки src/styles:

* Файл **style.scss** - головний файл стилів, який підключається до усіх HTML-сторінок.
* Файл **settings.scss** - файл налаштуваннь загальних значень стилів проєкту
* Папка **fonts** - містить *автоматично заповнювані*, при запуску будь-якого режиму, CSS-файли стилів підключення локальних та іконкового шрифтів
* Папка **includes** - містить загальні CSS/SCSS-файли стілів
  + Файл **index.scss** - а*втоматично підключається* до кожного SCSS-файлу проєкту та містить підключення налаштувань, SCSS міксінів та шаблонів. Можна підключати свої
  + Файл **extends.scss** - містить набір SCSS-шаблонів. Можна додавати свої
  + Файл **mixins.scss** - містить набір SCSS-міксінів. Можна додавати свої
  + Файл **spritemap.scss** - містить SCSS-міксін для роботи з SVG-спрайтом. *Заповлюється автоматично*
* Папка **libs** - містить а*втоматично підключаємі* до HTML-сторінок CSS-файли
  + Файл **reset.css** - а*втоматично підключається* до кожної HTML-сторінки, якщо не використовується інша CSS-бібліотека/фреймворк
  + Файл **tailwind.css** - а*втоматично підключається* до кожної HTML-сторінки, якщо використання вказано в налаштуваннях ЧФ4

Вміст папки src/js:

* Файл **app.js** (порожній) - основний файл, загальних для усього проєкту, скриптів. Підключається до кожної HTML-сторінки
* Папка **common** - містить файли загальних скриптів
  + Файл **functions.js** - містить корисні функції, які використовуються в компонентах ЧФ4
* Папка **react** - містить стартові файли та папки для роботи з React
* Папка **vue** - містить стартові файли та папки для роботи з Vue

Налаштування збірки «Чертоги Фрілансера» 4

Нове

10 квітня

На цій сторінці описані налаштування збірки «Чертоги Фрілансера» 4 (Далі ЧФ4).

ЧФ4 побудована так, щоб якомога більше можливостей можна було налаштувати під потреби користувача.

Для того щоб налаштувати збірку відкриваємо файл **template.config.js** в папці проєкту.

Перелік налаштувань:

* **lang** - Налаштування мови повідомлень ЧФ4 в терміналі та консолі: ua/en
* **vscode** - Налаштування редактора VS Code
  + **settings** - Автоматичне налаштування псевдонімів для проєкту (плагін [Path Autocomplete](https://marketplace.visualstudio.com/items?itemName=ionutvmi.path-autocomplete))
  + **snippets** - Автоматичне додавання сніпетів ЧФ4 в проєкт
* **devcomponents** - Налаштування сторінки розробки компонентів
  + **enable** - Вмикач: true/false
  + **filename** - Ім'я файлу сторінки
* **git** - Налаштування завантаження на GitHub
  + **repo** - Шлях репозиторію
  + **branch** - Бранч
* **navpanel** - Налаштування навігаційної панелі
  + **dev** - У режимі розробника. Вмикач: true/false
  + **build** - У режимі продакшена. Вмикач: true/false
  + **position** - Позиція виводу панелі: left/right
  + **color** - Колір тексту та посилань
  + **background** - Колір фону
  + **transition** - Швидкість анімації відкриття у мілісекундах
* **statistics** - Керування збором та виводом статистики проєкту
  + **enable** - Дозволити збір статистики. Вмикач: true/false
  + **showonbuild** - Автоматичний вивід статистики в режимі продакшн. Вмикач: true/false
* **server** - Налаштування локального серверу
  + **path** - Налаштування шляхів до файлів: ./ - відносний, / - абсолютний
  + **copyfiles** - Копіювання файлів з папки *src/files* у *dist/files* в режимі продакшн. Вмикач: true/false
  + **version** - Додавання унікальної версії до кінцевих CSS/JS файлів. Вмикач: true/false
  + **hostname** - Назва серверу
  + **port** - Порт серверу
* **html** - Налаштування обробки HTML-файлів
  + **beautify** - Форматування в режимі продакшн
    - **enable** - Вмикач: true/false
    - **indent** - Налаштування типу відступів
* **styles** - Налаштування роботи зі стилями та обробки SCSS/CSS-файлів
  + **tailwindcss** - Підключити Tailwindcss. Вмикач: true/false
  + **pxtorem** - Переводити PX в REM в режимі продакшн. Вмикач: true/false
  + **critical** - Створити та додати в HTML-файли критичні CSS-стилі. Вмикач: true/false
  + **codesplit** - Розділяти CSS-фали на окремі та підключати відповідним HTML-сторінкам в режимі продакшн. Вмикач: true/false
  + **devfiles** - Створення нестиснутих копій файлів в режимі продакшн. Вмикач: true/false
* **fonts** - Налаштування обробки шрифтів
  + **iconsfont** - Створення шрифту з SVG іконок (Іконкового шрифту) та підключення його в проєкт. Вмикач: true/false
  + **download** - Завантажувати та робити локальними шрифти які підключені віддалено (наприклад з Google Fonts). Вмикач: true/false
* **images** - Налаштування обробки зображень
  + **svgsprite** - Створення спрайту з SVG-іконок та підключення його в проєкт. Вмикач: true/false
  + **optimize** - Налаштування оптимізації зображень в режимі продакшн
    - **enable** - Вмикач: true/false. Якщо оптимізаця вимкнена зображення просто копіюються без конвертації та оптимізації
    - **edithtml** - Вносити зміни в HTML-файли. Якщо увімкнено, то HTML-файли будуть змінені згідно з налаштуваннями конвертації та створення додаткових розмірів зображень
    - **sizes** - Розміри файлів, які будуть додатково створені
    - **dpi** - Множники для створення додаткових файлів
    - **attrignore** - Атрибут тега IMG при наявності якого зображення не буде оброблено
    - **modernformat** - Налаштування конвертації в сучасні формати
      * **enable** - Вмикач: true/false
      * **type** - Формат зображення в який потрібно конвертувати: webp/avif
      * **only** - Залишати лише файли в сучасному форматі після конвертації. Вмикач: true/false
      * **quality** - Якість зображення
    - **jpeg** - Налаштування оптимізації зображень
      * **quality** - Якість зображення,
    - **png** - Налаштування оптимізації зображень
      * **quality** - Якість зображення
* **js** - Налаштування обробки JavaScript-файлів
  + **hotmodules** - "Гаряче" підключення компонентів. Вмикач: true/false
  + **devfiles** - Створення нестиснутих копій файлів в режимі продакшн. Вмикач: true/false
  + **bundle** - Налаштування формування кінцевого файлу в режимі продакшн
    - **enable** - Система збирає весь код в один JS та один CSS файли незалежно від налаштування styles -> codesplit. Вмикач: true/false
* **react** - Вмикає обробку React. Вмикач: true/false
* **vue** - Вмикає обробку Vue. Вмикач: true/false
* **php** - Налаштування PHP-серверу
  + **enable** - Вмикач: true/false
  + **base** - Шлях до папки де знаходяться файли PHP
  + **hostname** - Ім'я хосту
  + **port** - Порт
  + **binary** - Шлях до папки де встановлений інтерпритатор PHP
  + **ini** - Шлях до файлу налаштувань php.ini
* **pug** - Налаштування препроцесору PUG
  + **enable** - Вмикач: true/false
* **ftp** - Налаштування відправлення кінцевих файлів (з папки dist) на сервер (хостинг) по FTP в режимі продакшена
  + **host** - Ім'я FTP хосту
  + **port** - FTP Порт
  + **remoteDir** - Віддалена папка на сервері
  + **user** - Ім'я користувача FTP
  + **password** - Пароль користувача FTP
* **logger** - Логи роботи модулів та компонентів збірки
  + **terminal** - Логи роботи модулів в терміналі. Вмикач: true/false
  + **console** - Логи роботи компонентів в консолі
    - **enable** - Вмикач: true/false
    - **removeonbuild** - Прибирає логи в режимі продакшн. Вмикач: true/false
* **projectpage** - Cтворення сторінки проєкту в режимі продакшн. Сторінка буде містити посилання на усі HTML-файли проєкту для зручної демонстрації замовнику
  + **enable** - Вмикач: true/false
  + **projectname** - Назва проєкту, яка виводиться на сторінці замість %projectname%. Якщо порожньо, замінеться назвою папки проєкту
  + **template** - Шлях до шаблону сторінки проєкту
  + **outfilename** - Ім'я файлу. Якщо порожньо то створиться HTML-файл з назвою папки проєкту
* **aliases** - Налаштування псевдонімів
* **coffee** - Налаштування повідомлення про перерву
  + **enable** - Вмикач: true/false
  + **text** - Текст повідомлення
  + **interval** - Інтервал виводу повідомлення в хвилинах
* **novaposhta** - Налаштування компонента Нова Пошта
  + **enable** - Вмикач: true/false
  + **key** - Особистий ключ

Робота зі шрифтами. Локальні, іконкові шрифти та підключення з Google Fonts

Нове

10 квітня

Підключення шрифтів з Google Fonts:

Найпростіший спосіб - це скористатися плагіном для VS Code [Google Fonts](https://marketplace.visualstudio.com/items?itemName=Pavlo-K.better-google-fonts)

* Натискаємо **F1** і шукаємо плагін **Google Fonts**
* Якщо хочемо підключити шрифт окремим тегом link у HTML файл, вибираємо **Google Fonts: insert <link>**
* Якщо хочемо підключити шрифт до файлу стилів (зазвичай це **src/styles/style.scss**), вибираємо **Google Fonts: insert CSS @import**
* Зі списку обираємо потрібний шрифт, можна скористатися пошуком
* Зі списку обираємо потрібні варіації шрифту
* Тиснемо "Ok"

*Приклад підключення шрифту Montserrat + варіації:*

@import url("<https://fonts.googleapis.com/css?family=Montserrat:regular,600,700&display=swap>");

Шрифти, підключені з Google Fonts, не повинні потрапляти в папку з результатом (dist), вони підвантажуються з сервера Google. Але, якщо ви хочете щоб підключені віддалені шрифти підключились локально, скористайтесь налаштуванням ЧФ4: **fonts -> download: true**

Також варто зазначити, що отримати рядок для підключення шрифту ви можете і без плагіна на сайті [Google Fonts](https://fonts.google.com/).

Підключення локальних файлів шрифтів

У стартовому шаблоні "Чертоги Фрілансера" 4 (далі - ЧФ4) весь процес підключення локальних та іконочних шрифтів максимально автоматизований. Вихідні файли шрифтів конвертуються з сучасного [веб-формату .WOFF2](https://www.w3.org/TR/WOFF2/), а також виконується запис підключення шрифтів у файл стилів, включаючи значення font-weight та font-style на основі імені файлу.

Правила назв файлів шрифтів

Оскільки ЧФ4 автоматизує процес підключення родин шрифтів а також їх варіацій, важливо правильно називати файли за схемою:

назва родини-варіація

Наприклад, якщо файл з назвою KoPubBatang-Bold.ttf, то система підключе родину (font-family) KoPubBatang та варіацію (font-weight) 700.

Покрокова інструкція підключення локальних шрифтів:

* Завантажити або отримати від дизайнера/замовника файли шрифтів у форматі .TTF, .OTF або .WOFF2 і покласти їх у папку **src/assets/fonts**
* Перевірити назви файлів враховуючи інформацію вище
* Запустити режим розробки
* Вказати сімейство шрифту за замовчуванням у змінній **$fontFamily** у файлі **src/style/settings.scss**, та, за наявності інших родин та/або варіацій шрифту, іншим селекторам у стилях проєкту

Як ЧФ4 обробляє шрифти?

При запуску збірки в будь-якому режимі, ЧФ4 перевірить, чи є файли шрифтів у форматах .TTF, .OTF або .WOFF2 у папці **src/assets/fonts**

Далі відбуваються наступні етапи:

* Файли .OTF (якщо вони є) конвертуються у .TTF і зберігаються у папці src/assets/fonts
* Файли .TTF (якщо вони є) конвертуються у .WOFF2 і записуються у папку src/assets/fonts
* Файли .WOFF2 (якщо вони є) просто скануються для наступного етапу підключення
* Файли-похідники .OTF та .TTF (якщо вони є) видаляються
* Після конвертації ЧФ4 запише в файл **src/styles/fonts/fonts.css** конструкції @font-face для всіх файлів, включаючи значення font-weight та font-style, засновані на імені файлу шрифту. Заповнений файл стилів автоматично підключиться до усіх HTML-сторінок проєкту

Створення іконкового шрифту та підключення його в проєкт:

* Помістити іконки в форматі SVG в папку **src/assets/svgicons**
* Увімкнути в налаштуваннях ЧФ4 створення іконкового шрифту (fonts->iconsfont: true)
* Запустити режим розробки
* Використовувати спеціальні класи в HTML розмітці для виводу іконки з набору. Подивитись увесь набір іконок в шрифті та їх класи ви можете за посиланням в панелі розробника (якщо вона увімкнена в налаштуваннях ЧФ4).

Класи іконкового шрифту можна використовувати як окремо:

<button class="button --icon-some">Some text</button>

Так і як модифікатор:

<button class="button button--icon-some">Some text</button>

Після додавання класу, в об'єкті створиться псевдоелемент ::before який і буде виводити іконку

Налаштування ЧФ4 для роботи зі шрифтами

В файлі налаштувань ЧФ4 в розділі fonts ви знайдете:

* iconsfont - вмикає створення та підключення іконкового шрифту
* download - вмикає завантаження та підключення локально віддалених шрифтів

Робота зі стилями. Файли Settings.scss та Style.scss

Нове

10 квітня

В "Чертогах Фрілансера" 4 (далі ЧФ4) використовується [препроцесор SASS в синтаксісі SCSS](https://sass-lang.com/), що дає змогу використовувати широкий та потужний функціонал для створення стилів проєкту та їх керування.

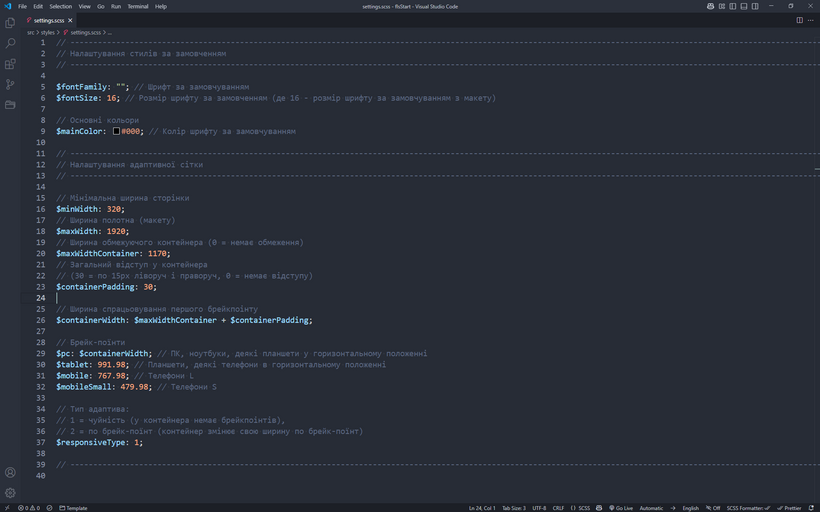
Завдяки компонентному підходу ЧФ4, є можливість описувати стилі для кожної сторінки та кожного блоку окремо, у файлах відповідного компоненту. Це надасть змогу системі розділяти стилі по сторінках, що позитивно впливає на оптимізацію проєкту. Окремі файли стилів компонентів ви знайдете у папці **src/components**

Для зручності працювати зі стилями варто використовуючи значення у PX (пікселях), та, де це потрібно, у % (відсотках) та EM. Система автоматично переведе усі значення з PX у відносні одиниці REM в режимі продакшн (якщо це увімкнено в налаштуваннях ЧФ4)

Файл settings.scss

Файл **src/styles/settings.scss** є головним файлом налаштувань значень стилів за замовченням, роботи адаптивної сітки та міксінів, зокрема таких як "Чуйна властивість"

Файл автоматично підключається до кожного SCSS-проєкту, та саме з цього файлу варто починати роботу над інтерфейсом

Файл **settings.scss**

Налаштування сімейства, розміру та кольору шрифту за замовчуванням

Для того, щоб працювати зі шрифтами, спочатку потрібно їх підключити. Всі дії докладно описані у [цій статті.](https://www.patreon.com/posts/robota-zi-ta-z-126343813)

Після виконання дій щодо підключення шрифтів необхідно вказати значення для наступних змінних

**$fontFamily** - ім'я сімейства шрифт за замовчуванням. Вказуємо ім'я основного шрифту у проекті та його тип, наприклад:

$fontFamily: "Montserrat, sans-serif";

**$fontSize** - розмір шрифту за замовчуванням. Слід вказати значення розміру шрифту за замовчуванням у пікселях (тільки число без px):

$fontSize: 16;

**$mainColor** - колір шрифту за замовчуванням. Вказуємо код кольору:

$mainColor: #000;

Також, тут можна додавати й інші змінні, у тому числі з кодами кольорів створюючи кольорову палітру проєкту

Налаштування адаптивної сітки

У ЧФ4 є можливість налаштувати обмежуючий контейнер на роботу як з адаптивною(чуйною) версткою, так і з версткою по брейкпоїнтах.

Перед початком робіт налаштовуємо наступні змінні:

**$minWidth** - мінімальна ширина вьюпорта (екрана), що підтримується проектом. Зазвичай це 320px, але з відмиранням старих пристроїв, це значення можна змінювати на будь-яке потрібне, вказуємо тільки число без px:

$minWidth: 320;

**$maxWidth** - ширина всього макета (полотна), не плутати з шириною контейнера, що обмежує. Як правило, дизайнери надають макети шириною 1920 чи 1440 пікселів, але це значення може бути будь-яким. Вимірюємо макет і вказуємо лише число без px:

$maxWidth: 1920;

**$maxWidthContainer** - ширина обмежуючого контейнера. Власне, це ширина контенту в макеті дизайну. Вимірюємо макет і вказуємо лише число без px:

$maxWidthContainer: 1170;

Якщо в макеті немає обмеження контенту, тобто контент розташований на всю ширину полотна (з відступами), то слід вказати значення **0** (нуль):

$maxWidthContainer: 0;

**$containerPadding** - загальний відступ (сума відступів ліворуч і праворуч) у обмежуючого контейнера. Вказуємо лише число без px:

$containerPadding: 30;

Якщо відступів немає, або ви хочете використовувати адаптивну властивість, слід вказати **0** (нуль):

$containerPadding: 0;

**$containerWidth** - ширина спрацьовування першого брейкпоїнту. Власне, це сума ширин обмежуючого контейнера та його відступів. Як правило, міняти тут нічого не потрібно.

$containerWidth: $maxWidthContainer + $containerPadding;

Вищезгадані змінні впливають і на функціонал чуйної властивості, який описаний в окремій статті

Налаштування брейкпоінтів

Налаштовуємо "стандартні" граничні ширини певних груп пристроїв:

**$pc** - ПК, ноутбуки, деякі планшети в горизонтальному положенні. Як правило, тут вказується змінна **$containerWidth**:

$pc: $containerWidth;

**$tablet** - планшети, деякі телефони у горизонтальному положенні. Зазвичай значення дорівнює 991.98px:

$tablet: 991.98;

**$mobile** - великі телефони. Зазвичай значення дорівнює 767.98px:

$mobile: 767.98;

**$mobileSmall** - маленькі телефони. Зазвичай значення дорівнює 479.98px:

$mobileSmall: 479.98;

Для швидкого виклику медіа-запиту з потрібним брейкпоінтом можна використовувати [сніпети](https://template.fls.guru/template-docs/install.html#snippets) md1, md2, md3, md4. Или, для Mobile First, mmd1, mmd2, mmd3, mmd4

Налаштування типу адаптива (поведінка обмежуючого контейнера)

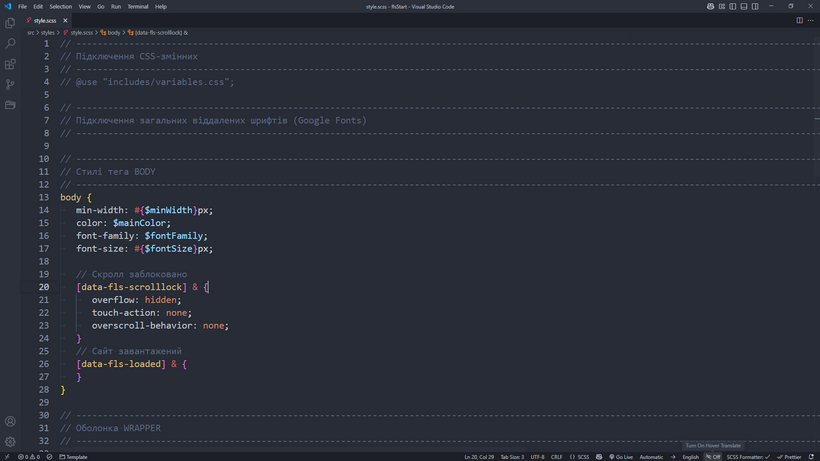
**$responsiveType** - налаштування типу адаптива (поведінка обмежуючого контейнера):

* **1** - чуйна. У контейнера немає брейкпоінтів, він звужується разом із браузером
* **2** - по брейкпоїтам. Контейнер змінює свою ширину за налаштованими брейкпоїнтами.

$responsiveType: 1;

Файл style.scss

Файл **src/styles/style.scss** є основним та загальним файлом стилів проєкту у стартовому шаблоні "Чертоги Фрілансера" 4. Тут знаходяться основні стилі тегу <body>, елементу з класом .wrapper, та обмежуючого контейнеру (клас \*\_\_container) на основі налаштувань в файлі settings.scss

Файл style.scss

**Підключення окремого файлу CSS-змінних**

На початку файлу є секція де за потреби можна підключити окремий файл CSS-змінних - src/styles/includes/variables.css

**Підключення віддалених шрифтів**

Далі є секція де, за потреби, слід підключати віддалені шрифти, наприклад з [Google Fonts](https://fonts.google.com/). Для цього зручно використовувати плагін для VS Code - [Better Google Fonts for VS Code](https://marketplace.visualstudio.com/items?itemName=Pavlo-K.better-google-fonts)

**Селектори та стилі**

**body {}** - стилі основного тега <body>.

Також додано підготовку для появи у тега <html> двох класів:

* **[data-fls-scrolllock]** - блокування скролу. Для цього вже написано відповідні стилі
* **[data-fls-loaded]** - сайт завантажений. За цим класом ми можемо впливати на <body> для відображення контенту після повного завантаження даних

.**wrapper {}** - обгортка всього контенту на сторінці. Для неї написані стилі притискання підвалу до низу сторінки, важливий параметр overflow: clip; який не дасть з'явитися горизонтальному скролу сторінки, а також вирішення інших проблем

Нижчевказано селектор обмежуючого контейнера та його стилі, значення яких багато в чому складаються з налаштувань в файлі settings.scss

Стилі обмежувального контейнера будуть застосовуватися до будь-якого елемента, у класі якого є рядок **"\_\_container"**. Для зручності можна використовувати сніпет **cnt**

<div class="block\_\_container"> ... </div>

Загальні стилі проєкту

Нижче усіх предвстановлених стилів ви можете додавати власні загальні стилі які застосуються до *усіх сторінок проєкту*.

Можливі налаштування ЧФ4 для роботи зі стилями

У файлі налаштувань ЧФ4 в розділі styles ви знайдете:

* tailwindcss - вмикає використання tailwindcss, при цьому файл src/styles/libs/reset.css буде вимкнений
* pxtorem - вмикає автоматичну конвертацію PX в REM
* critical - вмикає створення та інтеграцію в HTML критичних стилів
* codesplit - розділяє стилі по сторінках проєкту
* devfiles - створює не стиснуті файл стилів

Робота з HTML. Огляд функціоналу.

Нове

10 квітня

В збірці "Чертоги Фрілансера" 4 інтегровані широкі можливості роботи з HTML-файлами які, за бажанням, можна використовувати під час розробки того чи іншого проєкту.

Підключення окремих HTML-файлів

В процесі розробки HTML-конструкцій часто виникають ситуації коли є потреба перевикористати той чи інший відрізок коду в різних місцях, особливо враховуючи компонентний підхід ЧФ4.

Для вирішення цих задач інтегрований функціонал "Вставка" (include).

<include src="шлях до HTML-файлу"></include>

Давайте розглянемо цей функціонал на простому прикладі блоків <head>, <header> та <footer> проєкту, які мають бути підключені на усіх HTML-сторінках:

<!doctype html>  
<html lang="en">  
<!-- підключення head -->  
<include src="@components/layout/head/head.html"></include>  
<body>  
<div class="wrapper">  
<!-- підключення header -->  
<include src="@components/layout/header/header.html"></include>  
<main class="page"> <!-- контент сторінки --> </main>  
<!-- підключення footer -->  
<include src="@components/layout/footer/footer.html"></include>  
</div>  
<script type="module" src="@js/app.js"></script>  
</body>  
</html>

Таким чином, редагуючи файл компонента, наприклад header.html, зміни відбудуться по всьому сайту де цей компонент підключений.

Використання HTML-шаблонів

Для більш складних та великих проєктів, буде зручно використовувати HTML-шаблони (templates). Зберігаються шаблони в папці src/components/templates. Звісно можна додавати свої.

Давайте розглянемо приклад коли на сайті є різні структури - на головній сторінці і ще декількох немає бокової панелі, а ще на декількох є. Відповідно ми можемо підготувати і потім використовувати різні структурні шаблони.

Ось приклад шаблону main.html:

<!doctype html>  
<html lang="en">  
<include src="@components/layout/head/head.html"></include>  
<link rel="stylesheet" href="@components/templates/main/main.scss">  
<body>  
<div class="wrapper">  
<block name="header"></block>  
<block name="main"></block>  
<block name="footer"></block> </div> <block name="popup"></block>  
<script type="module" src="@js/app.js"></script>  
</body>  
</html>

Ось приклад шаблону inner.html:

<!doctype html>  
<html lang="en">  
<include src="@components/layout/head/head.html"></include>  
<link rel="stylesheet" href="@components/templates/inner/inner.scss">  
<body>  
<div class="wrapper">  
<block name="header"></block>  
<div class="inner">  
<block name="main"></block>  
<block name="aside"></block>  
</div>  
<block name="footer"></block>  
</div>  
<block name="popup"></block>  
<script type="module" src="@js/app.js"></script>  
</body>  
</html>

Ми бачемо, що структура цих шаблонів відрізняється, та до кожного підключений свій унікальний файл стилів.

**<block name="ім'я блоку">** - це комірка в яку ми можемо "вкладати" потрібний контент.

Ось приклад головной сторінки (index.html) яка використовує шаблон "main.html":

<template src="@components/templates/main/main.html">  
<block name="header">  
<include src="@components/layout/header/header.html"></include>  
</block>  
<block name="main">  
<include src="@components/pages/home/home.html"></include>  
</block>  
<block name="footer">  
<include src="@components/layout/footer/footer.html"></include>  
</block>  
<block name="popup"></block>  
</template>

А ось приклад внутрішньої сторінки (contacts.html) яка будується по шаблону "inner.html":

<template src="@components/templates/inner/inner.html">  
<block name="header">  
<include src="@components/layout/header/header.html"></include>  
</block>  
<div class="inner">  
<block name="main">  
<include src="@components/pages/contacts/contacts.html"></include>  
</block>  
<block name="aside">  
<include src="@components/layout/aside/aside.html"></include>  
</block>  
</div>  
<block name="footer">  
<include src="@components/layout/footer/footer.html"></include>  
</block>  
<block name="popup"></block>  
</template>

Як ми бачимо, в "комірки" шаблону вставлені різні компоненти, в тому числі, головний контент конкретної сторінки - @components/pages/home/home.html та @components/pages/contacts/contacts.html

Використання елементів програмування в HTML-файлах

Використання *include* та *template* дуже допомагають в роботі з HTML-файлами, а щоб можливості були ще ширші я додав елементи програмування

Використання змінних

В ЧФ4 є можливість передавати дані в окремі HTML-файли, підключені через *include* та *template*, за допомогою **locals** (змінних)

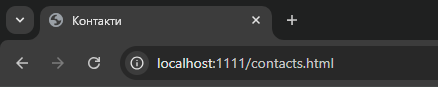
Наприклад ми підключаємо head сайту за допомогою *include* і на тій чи іншій сторінці (наприклад "Контакти"), хочемо вивести відповідний текст у тег <title>. Для цього передаємо дані у [форматі JSON](https://developer.mozilla.org/en-US/docs/Learn_web_development/Core/Scripting/JSON):

<!DOCTYPE html>  
<html lang="uk">  
<include src="@components/layout/head/head.html" locals='{"title":"Контакти"}'></include>  
<body>  
<div class="wrapper">  
<include src="@components/layout/header/header.html"></include>  
<include src="@components/pages/home/home.html"></include>  
<include src="@components/layout/footer/footer.html"></include>  
</div>  
</body>  
</html>

А у файлі head.html у тегу <title> виводимо змінну у подвійних квадратних дужках [[...]]:

<head>  
<meta charset="UTF-8" />  
<meta name="viewport" content="width=device-width, initial-scale=1.0">  
<link rel="stylesheet" href="@styles/style.scss">  
<title>[[title]]</title>  
</head>

Результат:



Таким чином можна передавати і виводити безліч інформації:

<!DOCTYPE html>  
<html lang="uk">  
<include src="@components/layout/head/head.html" locals='{  
"title":"Контакти",  
"keywords":"ключове слово, ключове слово, ключове слово...",  
"description":"опис сторінки..."  
}'></include>  
<body>  
<div class="wrapper">  
<include src="@components/layout/header/header.html"></include>  
<include src="@components/pages/home/home.html"></include>  
<include src="@components/layout/footer/footer.html"></include>  
</div>  
</body>  
</html>

Використання умов

Передача інформації за допомогою змінних (locals) відкриває ще ширші можливості разом з умовним вітвленням <if> <elseif> <else>

Наприклад, нам необхідно додати клас пункту меню який відповідає сторінці сайту. Для цього передаємо інформацію у файл header.html за допомогою locals:

<!DOCTYPE html>  
<html lang="en">  
<include src="@components/layout/head/head.html" locals='{"title":"Home Page"}'></include>  
<body>  
<div class="wrapper">  
<include src="@components/layout/header/header.html" locals='{"active":"Home"}'></include>  
<include src="@components/pages/home/home.html"></include>  
<include src="@components/layout/footer/footer.html"></include>  
</div>  
</body>  
</html>

У файлі header.html, при побудові меню, використовуємо умову:

<div class="header\_\_menu menu">  
<button type="button" data-fls-menu class="menu\_\_icon icon-menu"><span></span></button>  
<nav class="menu\_\_body">  
<ul class="menu\_\_list">  
<if condition="'[[active]]' === 'Home'">  
<li class="menu\_\_item"><a href="#" class="menu\_\_link menu\_\_link--active">Home</a></li>  
</if>  
<else>  
<li class="menu\_\_item"><a href="[[item.href]]" class="menu\_\_link">Home</a></li>  
</else>  
...  
</ul>  
</nav>  
</div>

Використання циклів

Для додаткової оптимізації побудов конструкцій додана можливість використання циклів <each>.

Ост приклад побудови меню за допомогою циклу на основі певного масиву даних:

<div class="header\_\_menu menu">  
<button type="button" data-fls-menu class="menu\_\_icon icon-menu"><span></span></button>  
<nav class="menu\_\_body">  
<ul class="menu\_\_list">  
<each loop='item in [  
{"href": "#","ancor": "Home"},  
{"href": "#","ancor": "About"},  
{"href": "#","ancor": "Blog"},  
{"href": "#","ancor": "Contacts"}  
]'>  
<if condition="'[[active]]' === '[[item.ancor]]'">  
<li class="menu\_\_item">  
<a href="[[item.href]]" class="menu\_\_link menu\_\_link--active">[[item.ancor]]</a>  
</li>  
</if>  
<else>  
<li class="menu\_\_item">  
<a href="[[item.href]]" class="menu\_\_link">[[item.ancor]]</a>  
</li>  
</else>  
</each>  
</ul>  
</nav>  
</div>

Налаштування ЧФ4 для роботи з HTML

У файлі налаштувань ЧФ4 в розділі html ви знайдете:

* beautify - налаштування форматування в режимі продакшн
  + enable - вмикач
  + indent - тип відступів

Робота з зображеннями. Огляд функціоналу

Нове

11 квітня

[Завантажити "Чертоги Фрілансера" 4](https://www.patreon.com/freelancerlifestyle/shop/startovii-shablon-chertogi-frilansera-4-1444862)

В збірці "Чертоги Фрілансера" 4 (ЧФ4) реалізована автоматизація широких можливостей оптимізації зображень:

* Оптимізація (зменьшення ваги) JPEG/JPG, PNG, GIF зображень
* Конвертація в сучасні формати WEBP/AVIF, що дуже суттєво зменьшує вагу зображень
* Створення меньших за розміром копій під різні ширини екранів
* Створення більших за розміром копій для виводу на екранах з високою щильністю пікселів на дюйм (DPI)

Усіма опціями можна керувати в [налаштуваннях ЧФ4](https://www.patreon.com/posts/126343727?collection=1439491), в розділі **images -> optimize**:

Увага! ЧФ4 не застосовує оптимізацію зображень при розробці під CMS WordPress

**enable** - дозволяє вимкнути будь-яку оптимізацію, це може бути потрібно у випадку подільшої інтеграції на бекенд, де існують власні методи оптимізації.

**edithtml** - за потреби, дозволяє вимикати автоматичне редагування HTML-коду (вставка копій зображень під різні екрани, заміна назви файлу на сучасний формат при конвертації тощо)

**sizes** - дозволяє через кому вказати ширини зображень які треба створити для виводу на відповідних екранах, наприклад якщо вказано [600, 1200] то:

В режимі розробки:

<img src="@img/cover.jpg" alt="Image">

Після збірки проєкту:

<picture>  
<source media="(max-width: 600px)" srcset="./assets/img/cover-600.webp" type="image/webp">  
<source media="(max-width: 1200px)" srcset="./assets/img/cover-1200.webp" type="image/webp">  
<img alt="Image" src="./assets/img/cover.webp">  
</picture>

Якщо ми не хочемо щоб ЧФ4 проводив таку оптимізацію, лишаємо квадратні дужки порожніми: []

Також є можливість задавати свої розміри конкретному зображенню, для цього додайте атрибут data-fls-image-sizes з вказаними розмірами через кому до тегу <img>:

<img data-fls-image-sizes="1000, 600" src="..." alt="...">

**dpi** - дозволяє через кому вказати типи щилькості пікселів екранів (2x, 3x тощо) для яких треба створити зображення, наприклад при [2,3] попередній приклад буде виглядати вже ось так:

<picture>  
<source media="(max-width: 600px)" srcset="./assets/img/cover-600.webp 1x,./assets/img/cover-600-2x.webp 2x,./assets/img/cover-600-3x.webp 3x" type="image/webp">  
<source media="(max-width: 1200px)" srcset="./assets/img/cover-1200.webp 1x,./assets/img/cover-1200-2x.webp 2x,./assets/img/cover-1200-3x.webp 3x" type="image/web  
<img alt="Image" src="./assets/img/cover.webp" srcset="./assets/img/cover.webp 1x,./assets/img/cover-2x.webp 2x,./assets/img/cover-3x.webp 3x">  
</picture>

Якщо ми не хочемо щоб ЧФ4 проводив таку оптимізацію, лишаємо квадратні дужки порожніми: []

**attrignore -** назва атрибути, при додаванні якого до тегу IMG оптимізація зображення, вказаного в цьому тезі буде ігноруватись.

**modernformat -** розділ конвертації в сучасні формати WEBP/AVIF

**modernformat -> enable -** керує потребою конвертувати зображення в сучасний формат

**modernformat -> type -** вказуємо в який формат конвертувати WEBP чи AVIF

**modernformat -> only -** якщо true то оригінальне зображення буде видалено, якщо false то буде інтегроване в конструкцію <picture>

**modernformat -> quality -** налаштування якості зображення від 0 до 100, чим вище число тим краща якість, але більша вага

**jpeg** - керування налаштуваннями формату JPEG

**jpeg -> quality -** налаштування якості зображення від 0 до 100, чим вище число тим краща якість, але більша вага

**png**- керування налаштуваннями формату PNG

**png -> quality -** налаштування якості зображення від 0 до 100, чим вище число тим краща якість, але більша вага

Навігаційна панель розробника

Нове

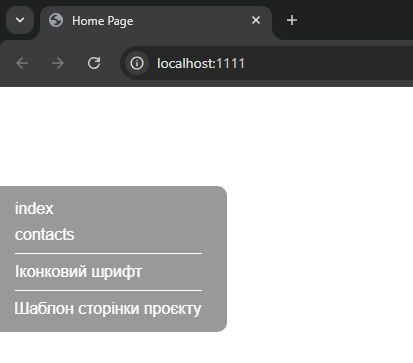
11 квітня

В збірці "Чертоги Фрілансера" 4 (далі ЧФ4) реалізована можливість використання **навігаційної панелі розробника**, скорочено **НПР**.

Що таке НПР?

Навігаційна панель розробника (НПР) - це "плаваюче" меню яке в автоматичному режимі виводить наступну інформацію:

* Посилання на усі сторінки проєкту. А саме усі HTML-файли які розміщені в корні папки *src*, для зручності навігації по проєкту.
* При використання [іконкового шрифту](https://www.patreon.com/posts/robota-zi-ta-z-126343813), посилання на сторінку з виводом усіх SVG-іконок та їх класів, для зручності їх копіювання. Посилання відкриється в новій вкладці
* При використанні сторінки проєкту, посилання на шаблон сторінки проєкту для зручності її розробки. Посилання відкриється в новій вкладці



Налаштування НПР

Увімкнути, або вимкнути НПР, в тому числі в режимі продакшн, можна в [налаштуваннях ЧФ4](https://www.patreon.com/posts/126343727). Крім того, є можливість налаштування положення та зовнішнього вигляду панелі:

* **navpanel** - Налаштування навігаційної панелі
  + **dev** - У режимі розробника. Вмикач: true/false
  + **build** - У режимі продакшена. Вмикач: true/false
  + **position** - Позиція виводу панелі: left/right
  + **color** - Колір тексту та посилань
  + **background** - Колір фону
  + **transition** - Швидкість анімації відкриття у мілісекундах

Статистика проєкту

Нове

11 квітня

В стартовому шаблоні "Чертоги Фрілансера" 4 (далі ЧФ4) реалізована можливість збирати та переглядати статистику вистраченого часу на проєкт та інші дані

Статистика збирається під час роботи режиму розробника

Для того щоб почати збирати статистику необхідно увікнути цей функціонал в налаштуваннях ЧФ4:

statistics: {  
enable: true,  
...  
}

Для перегляду статистики ви можете використовувати команду терміналу **npm run stat**, або увікнути автоматичний вивід при запуску режиму продакшн:

statistics: {  
enable: true,  
showonbuild: true  
}

Наразі можна отримати такі дані:

* Дата початку роботи над проєктом (перший запуск режиму розробника)
* Дата кінця роботи над проєктом (остання зупинка режиму розробника, або останній запуск режиму продакшн)
* Загальний час роботи над проєктом (час роботи режиму розробника)
* Загальна кількість редагованих файлів
* Кількість сесій (кількість запусків/зупинок режиму розробника)

Для очишення статистики проєкту ви можете скористатись командою терміналу npm run stat-clear

Робота з компонентами. Створення власних компонентів

Нове

12 квітня

Що таке компонент?

Компонент - це окремий самодостатній об'єкт зі своїм власним HTML/SCSS/JS кодом, який може використовуватись та перевикористовуватись в проєкті. Розмір компонента може бути як дуже малий, наприклад одна кнопка, так і дуже великий - таби, спойлери, модальні вікна тощо.

В стартовому шаблоні "Чертоги Фрілансера" 4 (далі ЧФ4) реалізований повний компонентний підхід до розробки а також "гаряче" підключення компонентів в проєкт.

Додатково, для тих чи інших компонентів, реалізована система плагінів, які теж оснащені "горячим" підключенням.

Компоненти розташовані в папці src/components. Та розбиті на декілька груп згідно зі своїм типом:

* custom - папка для розміщення власних компонентів
* effects - компоненти для додавання певних ефектів (анімацій) в проєкт
* forms - компоненти для роботи з HTML-формами та їхніми елементами
* layout - компоненти різних частин сайту, містить також шаблони сторінок
* wordpress - містить файли і папки для [розробки під CMS WordPress](https://www.patreon.com/posts/126493216?collection=1439491)

Використання компонентів

В ЧФ4 було досягнуто мінімальне втручання в JavaScript при роботі з шаблоном, тому для того щоб підключити той чи інший компонент в проєкт достатньо додати певний HTML-код, частіше всього це дата-атрибут який складається з **data-fls-ім'я папки компоненту.**

ЧФ4 розпізнає цей код, та автоматично підключить JS-файл та стилі з відповідної папки компоненту

Наприклад, для додавання компоненту табів в проєкт, ми використовуємо HTML-структуру яка містить дата-атрибут **data-fls-tabs:**

<div data-fls-tabs class="tabs">  
...  
</div>

А якщо там треба використати функціонал ["Динамічного адаптиву"](https://www.patreon.com/posts/126543656?collection=1439491) то користуємось відповідним дата-атрибутом data-fls-dynamic="..."тощо.

**Як правильно використовувати той чи інший компонент ЧФ4 детально описано у** [відповідній документації компоненту](https://www.patreon.com/collection/1439491?view=condensed)

Використання плагінів

Трапляється, що у певного компоненту є або можуть з'явитись додаткові невеликі JS/SCSS можливості, тому в ЧФ4 розроблена система плагінів для компонентів.

Наприклад, компонент "Header" сам по собі не містить ніякого JS-функціоналу, і використовується для виводу HTML-структури та стилізації шапки сайту.

Але, згідно задачі, буває що необхідно додати певні ефекти до шапки залежно від скролу сайту. Для цього існує плагін scroll який знаходиться в папці plugins папки компоненту.

Відповідно, для підключення даного плагіну ми маємо використати дата-атрибут data-fls-header-scroll:

<header data-fls-header data-fls-header-scroll class="header">  
...  
</header>

При викостанні плагінів, береться до уваги, що вони використовуються як доповнення до компоненту і не використовуються окремо.

Створення та використання власних компонентів та плагінів

В ЧФ4 є можливість створення власних компонентів і плагінів з подальшим використанням їх в проєкті.

**Для створення власних плагінів та компонентів, виконайте наступні дії:**

Для власного компоненту

* Створіть папку компоненту в src/component/custom/ назвавши її ім'ям вашого компоненту
* Створіть HTML/SCSS/JS файли компоненту з тим же ім'ям що й папка вашого компоненту. Якщо ваш компонент не використовує HTML чи SCSS чи JS код то відповідні файли створювати не обов'язково.
* Якщо ви створили та плануєте використовувати SCSS-файл стилів то його необхідно підключити в JS-файл (якщо він є) за допомогою import './ім'я компоненту.scss' або в HTML-файл: <link rel="stylesheet" href="@components/custom/ім'я компоненту/ім'я компоненту.scss">

**Увага!** Для швидкого створення базових файлів власного компоненту в папці src/component/custom ви можете скористатись командою npm run add ім'я компоненту.

Ім'я має бути латинецею без пробілів цифр та символів.

Після виконання в папці src/component/custom/ім'я компоненту будуть створені HTML, SCSS та JS файли компоненту. В JS-файл вже буде підключений SCSS-файл

Для власного плагіну

* Створіть папку плагіну в src/component/custom/ім'я компоненту/plugins назвавши її ім'ям вашого плагіну
* Створіть HTML/SCSS/JS файли плагіну з тим же ім'ям що й папка вашого плагіну. Якщо ваш плагіну не використовує HTML чи SCSS чи JS код то відповідні файли створювати не обов'язково.
* Якщо ви створили та плануєте використовувати SCSS-файл стилів то його необхідно підключити в JS-файл (якщо він є) за допомогою import './ім'я плагіну.scss' або в HTML-файл: <link rel="stylesheet" href="@components/custom/ім'я компоненту/plugins/ім'я плагіну/ім'я плагіну.scss">

**Для використання власних компонентів та плагінів, виконайте наступні дії:**

Для використання власного компоненту

* Якщо ваш компонент містить JS-файл, то для його автоматичного ("горячого") підключення в проєкт додайте у ваш HTML-код дата-атрибут який складається з **data-fls-ім'я папки вашого компоненту.** Наприклад, ви створили компонент "aside", тож для його підключеннядата-атрибут буде виглядати як **data-fls-aside**
* Якщо ваш компонент містить HTML-файл, який необхідно підключати в проєкт, робіть це за допомогою [<include>](https://www.patreon.com/posts/126344103?collection=1439491)
* Якщо ваш компонентмістить SCSS-файл, він підключеться разом з файлом компоненту у спосіб, відповідний до того як саме ви створили компонент

Для використання власного плагіну

* Якщо ваш плагін містить JS-файл, то для його автоматичного ("горячого") підключення в проєкт додайте у ваш HTML-код дата-атрибут який складається з **data-fls-ім'я папки вашого компоненту-ім'я папки вашого плагіну.** Наприклад, ви створили плагін "move" компоненту "aside", тож для його підключеннядата-атрибут буде виглядати як **data-fls-aside-move**
* Якщо ваш плагін містить HTML-файл, який необхідно підключати в проєкт, робіть це за допомогою [<include>](https://www.patreon.com/posts/126344103?collection=1439491)
* Якщо ваш плагін містить SCSS-файл, він підключеться разом з файлом плагіну у спосіб відповідний до того як саме ви створили плагін

Сторінка зручної розробки компонентів

Якщо ви розробляєте компоненти перед їх використанням, в ЧФ4 присутня окрема сторінка для зручного виводу компонентів та їх розробки.

Ця сторінка не потрапляє у зібраний проєкт (режим продакшн) і служить лише для зручності роботи з компонентами

Для відображення сторінки в панелі розробкика, увікніть параметр devcomponents -> enable: true в налаштуваннях ЧФ4

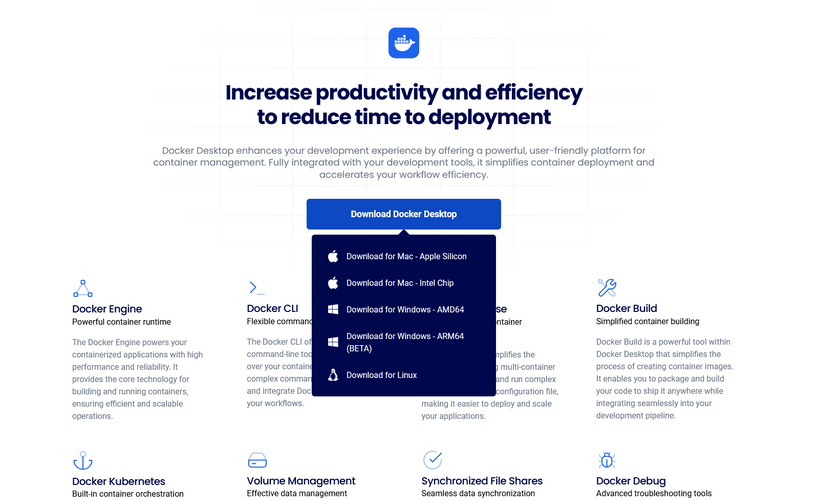
Розробка під CMS WordPress

Нове

12 квітня

Стартовий шаблон "Чертоги Фрілансера" 4 (ЧФ4) оснащений можливостями які дозволяють з легкістю розробляти інтерфейс відразу з інтеграцією на [CMS WordPress](https://wordpress.com/)

Для того щоб мати можливість працювати з повноційним сервером на своєму ПК, необхідно встановити систему [Docker](https://www.docker.com/products/docker-desktop/) (вона безкоштовна для власного використання)



Після встановлення, **Docker** **має бути просто увімкнений**, без необхидності робити з ним щось ще.

Для початку роботи, після встановлення ЧФ4 та Docker, виконайте команду npm run wp

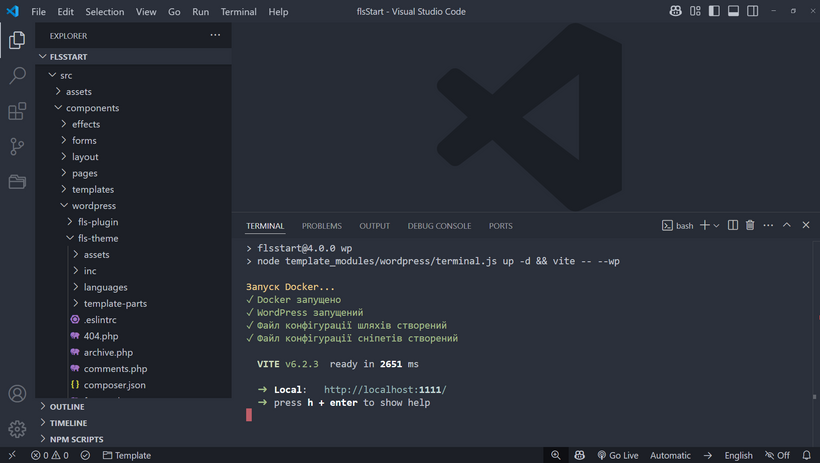
При першому запуску, система створить папку backend куди запише папку database з базою даних та папку wordpress з CMS WordPress. Не редагуйте папку backend та її вміст без чіткого розуміння своїх дій.

Дaлі система відкриє браузер з адресою <http://localhost:8080/> та запропонує встановлення CMS WordPress на ваш новостворений сервер. Після інсталяції CMS все готово для розробки.

Якщо ви запускаєте систему вже не вперше, то після команди npm run wp відкриється браузер з адресою серверу CMS WordPress <http://localhost:8080/> - головною сторінкою сайту.

**Увага!** Docker має бути запущений перед виконанням команди npm run wp

Для того щоб почати розробку, перейдіть у src/components/wordpress. Тут ви знайдете папку fls-theme. Це папка вже встановленої теми на вашу CMS WordPress і вам достатньо просто редагувати файли згідно задачі.



Редагування стилів відбувається в src/styles та і відповідних компонентах src/components

Редагування скриптів, в режимі розробника, відбувається у відповідних компонентах src/components та у файлі src/components/wordpress/fls-theme/assets/app.js, у випадку коли потрібно написати власну загальну логіку. Не видаляте підключення функціоналу import '../fls-wp-includes.js', це необхідно для коректної роботи.

Файл src/components/wordpress/fls-theme/assets/app.js, після збірки проєкту, вже не підключається. JS/CSS код збирається у відповідні файли в папці src/components/wordpress/fls-theme/assets/build

Також, після збірки, існує можливість внести правки у файли  
/wp-content/themes/fls-theme/assets/build/css/custom.css та  
/wp-content/themes/fls-theme/assets/build/js/custom.js

**Увага!** Не редагуйте інші JS-файли які знаходяться в src/components/wordpress

Також, у src/components/wordpress доступна папка fls-plugin для створення власного плагіну

Після завершення роботи на сайтом, виконайте команду npm run wpbuild, після цього, за допомогою плагіна WordPress для переносу сайту, який треба встановити самостійно, створіть дамп сайту та бази даних для подальшого переносу на віддалений сервер.

Через те що після виконання команд npm run wp та npm run wpbuild запускається Docker сервер, його варто зупиняти в момент перерви у розробці або її завершення. Для цього виконуйте команду npm run wpstop

Робота з SVG-спрайтом

Нове

12 квітня

[Завантажити "Чертоги Фрілансера" 4](https://www.patreon.com/freelancerlifestyle/shop/startovii-shablon-chertogi-frilansera-4-1444862)

В стартовому шаблоні "Чертоги Фрілансера" 4 (ЧФ4) є можливість працювати з SVG-спрайтом

Для того щоб створити SVG-спрайт виконайте наступні дії:

* В налаштуваннях ЧФ4 увімкніть створення SVG-спрайту (images->svgsprite: true)
* Розмістіть SVG-іконки в папці src/assets/svgicons
* Запустіть режим розробника npm run dev

Використання SVG-спрайту

Для виводу іконки зі спрайту в HTML в режимі розробника ви можете скористатись одним з двох способів

Вивід за допомогою тегу <use> (можна використовувати сніпет **usvg**):

<svg>  
<use xlink:href="\_\_spritemap#sprite-fb\_icon"></use>  
</svg>

Або тегу <img> (можна використовувати сніпет **usvgi**):

<img src="\_\_spritemap#sprite-fb\_icon-view" alt="Image">

Де fb\_icon це назва файлу іконки. Зверніть увагу, у варіанті виводу за допомогою тегу <img> до імені іконки додається суфікс -view

Для виводу інокни зі спрайту в SCSS в режимі розробника ви можете скористатись спеціальним міксіном:

@include sprite("fb\_icon");

Де fb\_icon це назва файлу іконки. В результаті буде виведена властивість background з url іконки, без повторення по центру.

Робота з SVG-спрайтом

Нове

12 квітня

[Завантажити "Чертоги Фрілансера" 4](https://www.patreon.com/freelancerlifestyle/shop/startovii-shablon-chertogi-frilansera-4-1444862)

В стартовому шаблоні "Чертоги Фрілансера" 4 (ЧФ4) є можливість працювати з SVG-спрайтом

Для того щоб створити SVG-спрайт виконайте наступні дії:

* В налаштуваннях ЧФ4 увімкніть створення SVG-спрайту (images->svgsprite: true)
* Розмістіть SVG-іконки в папці src/assets/svgicons
* Запустіть режим розробника npm run dev

Використання SVG-спрайту

Для виводу іконки зі спрайту в HTML в режимі розробника ви можете скористатись одним з двох способів

Вивід за допомогою тегу <use> (можна використовувати сніпет **usvg**):

<svg>  
<use xlink:href="\_\_spritemap#sprite-fb\_icon"></use>  
</svg>

Або тегу <img> (можна використовувати сніпет **usvgi**):

<img src="\_\_spritemap#sprite-fb\_icon-view" alt="Image">

Де fb\_icon це назва файлу іконки. Зверніть увагу, у варіанті виводу за допомогою тегу <img> до імені іконки додається суфікс -view

Для виводу інокни зі спрайту в SCSS в режимі розробника ви можете скористатись спеціальним міксіном:

@include sprite("fb\_icon");

Де fb\_icon це назва файлу іконки. В результаті буде виведена властивість background з url іконки, без повторення по центру.

Завантаження на сервер через FTP

Нове

12 квітня

В стартовому шаблоні "Чертоги Фрілансера" 4 (ЧФ4) є можливість автоматичного завантаження зібраного проєкту (папка dist) на сервер за допомогою FTP

Для цього вкажіть необхідні дані в налаштуваннях ЧФ4 в розділі ftp:

* **ftp** - Налаштування відправлення кінцевих файлів (з папки dist) на сервер (хостинг) по FTP в режимі продакшена
  + **host** - Ім'я FTP хосту
  + **port** - FTP Порт
  + **remoteDir** - Віддалена папка на сервері. Можна використовувати змінну projectName яка містить ім'я папки проєкту
  + **user** - Ім'я користувача FTP
  + **password** - Пароль користувача FTP

Після налаштування ви можете користуватись командою термінала **npm run deploy**, яка об'єднує в собі команду збірки проєкту (режим продакшн) а також завантаження результату на сервер за допомогою FTP

Чуйна (адаптивна) властивість

Нове

13 квітня

За допомогою SCSS-міксину “Чуйна (адаптивна) властивість” можна налаштувати (в залежності від ширини екрану) автоматичну зміну значення тієї чи іншої CSS-властивості від початкового значення, на певній ширині екрана, до кінцевого значення на іншій ширині екрану. Можна вказувати довільні проміжки ширини. Також існує кілька режимів поведінки міксину поза вказаними проміжками.

Використання міксину

Щоб працювати з міксином потрібно в SCSS селекторі викликати сніпет **av**:

*Базовий режим роботи міксину:*

@include adaptiveValue("властивість", початкове значення, кінцеве значення);

Де:

* **Властивість** – CSS-властивість, значення якого необхідно адаптувати. Можна вказати будь-яку властивість, значення якого вказується у цифрах.
* **Початкове значення** – стартове значення властивості в пікселях, пишемо число без px. Зазвичай вказується по макету.
* **Кінцеве значення** – фінальне значення властивості в пікселях, пишемо число без px. Значення, до якого ми хочемо прийти на менших ширинах екрану.

*Приклади:*

@include adaptiveValue("font-size", 50, 20);  
@include adaptiveValue("padding-top", 80, 10);  
...

Алгоритм работы миксина

Міксин працює на основі значень змінних **$minWidth**, **$maxWidth**, **$maxWidthContainer**, **$containerPadding** и **$containerWidth** розташованих в файлі [src/styles/settings.scss](https://www.patreon.com/posts/126343911?collection=1439491)

У підсумку, за замовчанням, міксин працюватиме так:

Якщо **$maxWidthContainer** більше нуля, то значення властивості змінюватимуться в проміжку ширин від **$containerWidth** до **$minWidth**. Тобто, по всій ширині контейнера, що обмежує.

При цьому, якщо ширина екрана більша ніж **$containerWidth**, то значення властивості дорівнюватиме **початковому** значенню. Якщо ширина екрана менша ніж **$minWidth**, то значення властивості дорівнюватиме **кінцевому значенню**.

Якщо **$maxWidthContainer** дорівнює нулю, то значення властивості змінюватимуться в проміжку ширин від **$maxWidth** до **$minWidth**.

Додаткові налаштування та режими роботи

**Режими роботи**

Поведінку міксину ми можемо налаштувати вказавши режим роботи.

@include adaptiveValue("властивість", початкове значення, кінцеве значення, режим роботи);

**Режим роботи** – може приймати числові значення **0 / не вказано** , **1**, **2** або **3**:

* **0 / не вказано** – Якщо ширина екрану більше ніж **ширина від**, то значення властивості дорівнює **початковому значенню**. Якщо ширина екрану менше ніж **ширина до**, то значення властивості дорівнюватиме **кінцевому значенню**.
* **1** – Міксин буде працювати **тільки** між **мінімальною** та **максимальною** шириною екрану, або **тільки у зазначених проміжках** якщо вони задані (про проміжки дивись далі). За цих ширин, значення властивості будуть за замовчюванням або успадковуватися від предків.
* **2** – Якщо ширина екрана більше ніж **ширина від**, то значення властивості дорівнює **початковому значенню**. Якщо ширина екрану менше ніж **ширина до**, значення властивості буде за замовчуванням або успадковуватися від предків.
* **3** – Якщо ширина екрана більше ніж **ширина від**, то значення властивості буде за замовчуванням або успадковуватися від предків. Якщо ширина екрана менше ніж **ширина до**, то значення властивості дорівнюватиме **кінцевому значенню**.

*Приклад:*

@include adaptiveValue("font-size", 50, 20, 1);

**Власні проміжки**

Міксин дозволяє вказати свій проміжок ширини всередині якого адаптуватиметься значення властивості.

@include adaptiveValue("властивість", початкове значення, кінцеве значення, режим роботи, ширина від, ширина до);

* **ширина від** – стартова ширина менше за яку почнеться адаптація, пишемо число без px.
* **ширина до** – кінцева ширина до якої буде адаптуватися значення властивості, пишемо число без px.

**Увага!**  
При вказанні проміжків необхідно обов’язково вказати режим роботи

*Приклад:*

@include adaptiveValue("font-size", 50, 20, 0, 800, 480);

Також ми можемо використовувати кілька міксинових викликів з різними проміжками:

@include adaptiveValue("font-size", 50, 20, 2, 800, 480);  
@include adaptiveValue("font-size", 20, 10, 3, 480);

У прикладі станеться таке: у проміжку ширин екрану від 800px до 480px значення властивості font-size буде чуйно адаптуватися від 50px до 20px, а в проміжку від 480px до 320px чуйно адаптуватися від 20px до 10px. Ширше 800px значення властивості буде 50px, вже 320px значення властивості буде 20px.

**Увага!** Не використовуйте додаткові медіа-запити для роботи з чуйною властивістю. Все, що вам потрібно, можна налаштувати за допомогою комбінацій кількості адаптивних властивостей, режимів і проміжків.

Розташування

Міксін **adaptiveValue** знаходиться у файлі **src/styles/includes/mixins.scss**

**Компонент "Динамічний адаптив"**

**Нове**

**13 квітня**

**Під час роботи над адаптивом сайту, нам постійно доводиться змінювати зовнішній вигляд об’єкта. У деяких випадках нам необхідно змінити порядок елементів, коли при певній ширині екрану деякий блок повинен перебувати в іншому місці HTML-структури.**

**Функціонал динамічного адаптива переміщає необхідний блок (на певному розширенні екрану) до іншого блоку. Переміщення відображається у розмітці HTML. Коли ширина екрану змінюється в більшу сторону, об'єкт повертається на своє місце.**

**Використання функціоналу**

**До блоку який потрібно перемістити додаємо атрибут data-fls-dynamic. Можна використовувати сніпет da. Підключення JS функціоналу відбудеться автоматично  
  
Ось можливі значення атрибуту:**

**data-fls-dynamic="селектор блоку, ширина екрану, позиція, селектор батьківського блоку"**

**Де:**

* **селектор блоку - (обов'язково) селектор об'єкту куди треба перемістити**
* **ширина екрану - (обов'язково, за замовченням 767,98) ширина екрану при якій спрацює магія переміщення**
* **позиція - (за замовченням 0, тобто перший) позиція, яку займе переміщений об'єкт серед вже присутніх**
* **селектор батьківського блоку - селектор об'єкту в якому знаходиться селектор блоку. Використовується для уточнення місця куди треба перемістити. Якщо вказано, то компонент буде шукати селектор блоку який вкладений у селектор батьківського блоку**

**Компонент "Адаптивне меню" (бургер)**

* **Нове**
* **13 квітня**
* **Функціонал реагує на клік на кнопку меню “бургера” (елемент з класом icon-menu). При цьому до тега html додається атрибут data-fls-menu-open, а також спрацьовує блокування прокручування сторінки. При повторному натисканні відбуваються зворотні дії.**
* **Також, компонент містить базові стилі іконки меню та анимації відкриття.**
* **Використання**
* **Для побудови меню та підключення функціоналу компоненту можна використовувати сніпет menu або побудувати структуру вручну або підключити готовий приклад з папки компоненту.**
* **Код меню:**
* **<div class="menu">  
  <button type="button" data-fls-menu class="menu\_\_icon icon-menu"><span></span></button>  
  <nav class="menu\_\_body">  
  <ul class="menu\_\_list">  
  <li class="menu\_\_item">  
  <a href="#" class="menu\_\_link">Home</a>  
  </li>  
  ...  
  </ul>  
  </nav>  
  </div>**
* **Або підключення HTML-коду меню з компоненту:**
* **<include src="@components/layout/menu/menu.html"></include>**
* **Важливо щоб у кнопки відкриття меню був атрибут data-fls-menu**
* **HTML та базові стилі знаходиться в папці компоненту src/components/layout/menu/**

**Компонент "Модальне вікно" (popup)**

**Нове**

**13 квітня**

**Цей функціонал додає можливість використовувати модульні (випливаючі) вікна. Робота модуля полягає в наступному: користувач натискає на вказаний елемент (за замовчюванням це атрибут із вказаним ім'ям попапу data-fls-popup-link=’popup`s name’). При цьому до тега body додається атрибут data-fls-popup-open. Також блокується прокручування сторінки (можна вимкнути), фокусування елементів “перелітає” на popup, із запам’ятовуванням попереднього сфокусованого елемента на сторінці. Закриття popup відбувається при натисканні кнопки закриття (за замовчуванням елемент з атрибутом data-fls-popup-close), по кліку на “порожньому місці” (не на popup), за натисканням кнопки ESC.**

**HTML/SCSS/JS файли компоненту знаходяться в папці src/components/layout/popup/**

**Використання компоненту**

**За допомогою include підключаємо HTML-файл компоненту до нашої HTML-сторнки (можна скористатись сніпетом pop):**

**<include src="@components/layout/popup/popup.html"></include>**

**В файлі src/components/layout/popup/popup.html вже є базовий HTML-код одного попапу, важливо що об'єкт має атрибут data-fls-popup зі значенням popup. Це значення і є ім'ям попапу.**

**Ви можете створювати ще модальні вікна надаючі їм унікальні імена**

**Для того щоб виклакати відкриття того чи іншого вікна додайте об'єкту (посилання, кнопка тощо) атрибут data-fls-popup-link та в заченні вкажіть ім'я попапу який треба відкрити (можна скористатись сніпетом pl).**

**Наприклад по кліку на це посилання відкриється вікно з ім'ям popup:**

**<a href="#" data-fls-popup-link="popup">Popup</a>**

**Додаткові можливості**

**Компонент "Модальне вікно" вміє відкривати popup по хешу. Наприклад за цією адресою відкриється вікно з ім'ям popup:**

[**http://localhost:1111/**](http://localhost:1111/)**#popup**

**Також, є можливість відкривати YouTube-відео у модальному вікні з автоматичним запуском програвання. Для цього додайте атрибут data-fls-popup-youtube, до об'єкту який відкриває попап, зі значенням ID YouTube-відео.**

**Наприклад, якщо YouTube-відео має ось таке посилання** [**https://www.youtube.com/watch?v=TzqNA7FoEjY**](https://www.youtube.com/watch?v=TzqNA7FoEjY) **то його ID це TzqNA7FoEjY. Саме цей код і вказується в атрибуті:**

**<a href="#" data-fls-popup-link="popup" data-fls-popup-youtube="TzqNA7FoEjY">Popup</a>**

**Якщо ваше модальне вікно відкривається, наприклад по хешу або після відправки форми, то атрибут data-fls-popup-youtube з ID відео додайте до головного тегу попапу в файлі src/components/layout/popup/popup.html:**

**<div data-fls-popup="popup" data-fls-popup-youtube="TzqNA7FoEjY" aria-hidden="true" class="popup">  
.....  
</div>**

**В обох випадках відео буде інтегроване в об'єкт з атрибутом data-fls-popup-content з заміною усього вмісту, якщо виникає потреба вивести відео в якомусь іншому місці модального вікна, створіть об'єкт і додайте йому атрибут data-fls-popup-youtube-place**

**Методи та події**

**Методи**

**Для того щоб викликати той чи інший метод компонента звертайтесь до нього через глобальну змінну window.flsPopup.**

**Наприклад, наступне звернення відкриє попап з ім'ям popup:**

**window.flsPopup.open('popup')**

**Події**

**У класі попапів існує ряд подій:**

* **beforePopupOpen – спрацює перед відкриттям попапа**
* **afterPopupOpen – спрацює після відкриття попапа**
* **beforePopupClose – спрацює перед відкриттям попапа**
* **afterPopupClose – спрацює після відкриття попапа**

**Щоб працювати з подією вішаємо прослуховування на document**

**document.addEventListener("afterPopupOpen", function (e) {  
// Попап  
const currentPopup = e.detail.popup;  
})**

**Компонент "Спостерігач"**

**Нове**

**13 квітня**

**Компонент “Спостерігач” можна використовувати для вирішення різних завдань: анімація елементів при скролі, підсвічування активного пункту меню (використовується в компоненті “СкролДо”) та багатьох інших.**

**Суть роботи спостерігача полягає в додаванні класу --watcher-view елементу в момент його появи у в’юпорті (екрані) при скролі та при відкритті сторінки. І, відповідно, прибраний при відході об’єкта з в’юпорту.**

**Використання компонента**

**Для того щоб "Спостерігач" почав стежити за об'єктом, додайте йому атрибут data-fls-watcher**

**<div data-fls-watcher class="block"> ... </div>**

**Додаткові налаштування:**

* **data-fls-watcher-root=’селектор’ – селектор батька всередині якого спостерігатиме за об’єктом. За замовчуванням <body>**
* **data-fls-watcher-margin=’значення’ – відступ від батька. Вказуємо значення у PX або у %**
* **data-fls-watcher-threshold=’значення’ – відсоток показу об’єкта для спрацьовування. Де 1 = 100% показ об’єкта. Вказуємо лише цілі чи десяткові числа, за замовчуванням 0. Може містити масив значень через кому.**
* **data-fls-watcher-once – спостерігати лише один раз. Тобто клас до об’єкта додасться тільки один раз і не буде прибраний при виході з в’юпорту.**

**Приклад – клас додасться лише один раз, з появою об’єкта на 50% його висоти:**

**<div data-fls-watcher-threshold="0.5" data-fls-watcher-once data-fls-watcher class="block"> ... </div>**

**Події**

**Після кожного спрацювання спостерігача виникає подія watcherСallback, його можна відловити в будь-якій частині коду:**

**document.addEventListener("watcherCallback", function (e) {  
// Повна інформація від спостерігача  
const entry = e.detail.entry  
// Спостерігаємий об'єкт  
const targetElement = entry.target  
})**

**Компонент "СкролДо". Плавна навігація по сторінці.**

**Нове**

**13 квітня**

**У цьому компоненті реалізований плавний скрол після натискання на посилання (кнопці) до потрібного блоку на сторінці. Якщо у адресі сторінки буде передано хеш і блок з таким селектором буде знайдено на сторінці, прокручування до блоку відбудеться автоматично. Також модуль дозволяє побудувати навігацію по сторінці з додаванням класу поточному пункту навігації (пункту меню) при скроллі до певного блоку.**

**Використання компоненту**

**До елементів навігації (пунктів меню), або до довільного об’єкта, додаємо HTML-атрибут data-fls-scrollto, а в якості значення вказуємо CSS селектор блоку до якого потрібно прокрутити:**

**<a href="#" data-fls-scrollto=".ім'я класу блоку" class="link">Пункт навігаціі</a>  
<a href="#" data-fls-scrollto="#id блоку" class="link">Пункт навігаціі</a>**

**Якщо потрібно, щоб скролл враховував шапку (не докручував на висоту шапки, використовується при фіксованих шапках) потрібно додати до об’єкта навігації атрибут data-fls-scrollto-header:**

**<a href="#" data-fls-scrollto-header data-fls-scrollto=".ім'я класу блоку" class="link">Пункт навігаціі</a>**

**Якщо потрібно, щоб скрол не докручував до блоку на вказану висоту, необхідно додати до об’єкта навігації атрибут data-fls-scrollto-top, а в якості значення вказати число – необхідну висоту:**

**<a href="#" data-fls-scrollto-top="30" data-fls-scrollto="#id блоку" class="link">Пункт навігаціі</a>**

**Додавання класу до поточного пункту навігації**

**Для включення функціоналу додавання класу необхідно задіяти** [**компонент "спостерігач"**](https://www.patreon.com/posts/126546933?collection=1439491)**.**

**Тож для секцій, при появі яких буде додаватись клас до відповідного пункту навігації, додаємо атрибут data-fls-watcher зі значенням navigator:**

**<a href="#" data-fls-scrollto=".some-section" class="link">Пункт навігаціі</a>  
...  
<section data-fls-watcher="navigator" class="some-section"></section>**

**Після цього, при прокручуванні до об’єкту навігації, до відповідного пункту навігації буде додано клас --navigator-active. А коли об'єкт зникає з в'юпорта, клас прибиреться.**

**Прокручування до потрібного блоку по хешу (при відкритті сторінки)**

**Для того щоб прокрутити сторінку до потрібного блоку при відкритті сторінки необхідно додати до адреси хеш, що містить ім’я класу потрібного блоку.**

***Приклад адресного рядка та потрібного блоку:***

[**https://fls.guru/index.html**](https://fls.guru/index.html)**#some-section**

**<section class="some-section"> ... </section>**

**Компонент “Показати ще”**

**Нове**

**15 квітня**

**Компонент “Показати ще” дозволяє спочатку приховати частину тексту або елементів списку, показуючи тільки вказану висоту або кількість елементів. Є можливість включати функціонал на певній ширині екрану (брейкпоінті).**

**Розташування компоненту**

**Файли компоненту знаходяться в папці src/components/layout/showmore, тут ви знайдете JS-файл компоненту, HTML-файл з прикладом коду, а також SCSS-файл для стилізації компоненту.**

**Використання компоненту**

**У потрібному місці викликати сніпет showmore (класи замінити на потрібні). Або вручну створити структуру де для оболонки доданий атрибут data-fls-showmore, для дочірнього елемента data-fls-showmore-content і кнопки data-fls-showmore-button. Кнопку спочатку потрібно приховати додавши атрибут hidden і додати два теги <span> з текстом показу та приховування контенту:**

**<div data-fls-showmore class="block">  
<div data-fls-showmore-content class="block\_\_content"></div>  
<button hidden data-fls-showmore-button type="button" class="block\_\_more"> <span>Показати ще</span> <span>Приховати</span> </button>  
</div>**

**До елемента з атрибутом data-fls-showmore-content додаємо текст та інший контент, або якщо це список (UL/OL) елементи списку (LI).**

**Залежно від того, який контент використовується (текст або елементи списку), вказуємо значення для атрибута data-fls-showmore:**

* **size – обмеження по висоті блоку (за замовчуванням)**
* **items – обмеження кількості виведених елементів списку**

**<div data-fls-showmore="items" class="block">  
<ul data-fls-showmore-content class="block\_\_content">  
<li>Пункт №1</li>  
<li>Пункт №2</li>  
<li>Пункт №3</li>  
<li>Пункт №4</li>  
<li>Пункт №5</li>  
</ul>  
<button hidden data-fls-showmore-button type="button" class="block\_\_more"> <span>Показати ще</span> <span>Приховати</span> </button>  
</div>**

**Залежно від того, який тип обрано, вказуємо значення для атрибуту data-fls-showmore-content :**

* **Висота блоку у пікселях (число без px, за замовчуванням 150)**
* **Кількість виведених елементів списку (число, за замовчуванням 3 )**

**<div data-fls-showmore class="block">  
<div data-fls-showmore-content="200" class="block\_\_content"> Lorem ipsum dolor sit amet consectetur, adipisicing elit. Blanditiis explicabo voluptates magni culpa, perferendis vel quam consequuntur possimus, vero placeat quo enim obcaecati quas, veritatis magnam non. Architecto, porro voluptatum?  
</div>  
<button hidden data-fls-showmore-button type="button" class="block\_\_more"> <span>Показати ще</span> <span>Приховати</span> </button>  
</div>**

**Якщо контент буде меншим, ніж зазначене обмеження, кнопка “Показати ще” не буде показана. В іншому випадку, контент обмежиться за висотою або за кількістю елементів і при натисканні на кнопку буде показаний повністю, також, до елемента з атрибутом data-fls-showmore додасться клас --showmore-active (перший спан в кнопці буде прихований, а другий показаний). Повторний клік поверне обмеження.**

**Є можливість керувати швидкістю розгортання контенту, для цього слід вказати значення атрибуту data-fls-showmore-button у мілісекундах (за замовчуванням 500):**

**<div data-fls-showmore class="block">  
<div data-fls-showmore-content="200" class="block\_\_content"> Lorem ipsum dolor sit amet consectetur, adipisicing elit. Blanditiis explicabo voluptates magni culpa, perferendis vel quam consequuntur possimus, vero placeat quo enim obcaecati quas, veritatis magnam non. Architecto, porro voluptatum?  
</div>  
<button hidden data-fls-showmore-button="1000" type="button" class="block\_\_more"> <span>Показати ще</span> <span>Приховати</span> </button>  
</div>**

**Увімкнення функціоналу на певній ширині екрану**

**Для того щоб використовувати функціонал на певній ширині екрана, до об’єкта з атрибутом data-fls-showmore додаємо атрибут data-fls-showmore-media де, через кому, вказуємо потрібну ширину, а також тип:**

* **max (за замовчуванням) – функціонал увімкнеться на ширині меншій ніж зазначена**
* **min – функціонал увімкнеться на ширині більшій ніж зазначена**

**<div data-fls-showmore data-fls-showmore-media="768,min" class="block">  
<div data-fls-showmore-content="200" class="block\_\_content"> Lorem ipsum dolor sit amet consectetur, adipisicing elit. Blanditiis explicabo voluptates magni culpa, perferendis vel quam consequuntur possimus, vero placeat quo enim obcaecati quas, veritatis magnam non. Architecto, porro voluptatum?  
</div>  
<button hidden data-fls-showmore-button="1000" type="button" class="block\_\_more"> <span>Показати ще</span> <span>Приховати</span> </button>  
</div>**

**Компонент "Таби"**

**Нове**

**15 квітня**

**Таби – це заголовки та відповідні їм блоки. Як правило, за замовчуванням відкритий лише один блок, інші приховані. При натисканні на заголовок показується відповідний блок.**

**Основні можливості**

* **Використання безлічі блоків з табами**
* **Відкриття потрібного таба з хешу**
* **Перетворення табів на спойлери на вказаній ширині екрана (зручно для адаптива)**
* **Можливість анімованого відкриття табів**
* **Семантика**

**Розташування компоненту**

**Файли компоненту знаходяться в папці src/components/layout/tabs, тут ви знайдете JS-файл компоненту, HTML-файл з прикладом коду, а також SCSS-файл для стилізації компоненту.**

**Використання компоненту**

**У потрібному місці викликати сніпет tabs (класи можна замінити на потрібні). Або вручну створити структуру із відповідними дата-атрибутами. Зверніть увагу, що додавання класу --tab-active для заголовка таба зробить таб активним (відкритим)**

***Приклад блоку з трьома табами:***

**<div data-fls-tabs class="tabs">  
<nav data-fls-tabs-titles class="tabs\_\_navigation">  
<button type="button" class="tabs\_\_title --tab-active">Таб №1</button>  
<button type="button" class="tabs\_\_title">Таб №2</button>  
<button type="button" class="tabs\_\_title">Таб №3</button>  
</nav>  
<div data-fls-tabs-body class="tabs\_\_content">  
<div class="tabs\_\_body">Вміст першого таба</div>  
<div class="tabs\_\_body">Вміст другого таба</div>  
<div class="tabs\_\_body">Вміст третього таба</div>  
</div>  
</div>**

**Перетворення табів на спойлери**

**Для того, щоб таби перетворювалися на спойлери, необхідно для елемента з атрибутом data-fls-tabs вказати значення ширини екрана нижче якої відбудеться перетворення:**

**<div data-fls-tabs="768" class="tabs"> ... </div>**

**У момент перетворення, до об’єкта з атрибутом data-tabs додасться клас --tab-spoller, яким можна змінити стилі для нового представлення табов-спойлеров.**

**Відкриття потрібного таба по хешу**

**Якщо є необхідність відкривати конкретний таб у конкретному блоці табів при відкритті сторінки по хешу, необхідно для елемента з атрибутом data-fls-tabs додати атрибут data-fls-tabs-hash:**

**<div data-fls-tabs data-fls-tabs-hash class="tabs"> ... </div>**

**Тепер, при натисканні на заголовки табів, до адреси сторінки додаватиметься хеш виду: #tab-0-1, де 0 – це ідентифікатор блоку з табами, а 1 – ідентифікатор таба в цьому блоці.**

**Відповідно, перейшовши на сторінку з хешем #tab-0-1, відкриється другий таб у першому блоці з табами. При #tab-2-0 відкриється перший таб у третьому блоці з табами і так далі.**

**Анімація при відкритті таба**

**Для того, щоб таби відкривалися плавно, необхідно об’єкту з атрибутом data-fls-tabs, додати атрибут data-fls-tabs-animate, а в якості значення вказати кількість мілісекунд, за які відкриється таб (за замовчуванням 500).**

**<div data-fls-tabs data-fls-tabs-animate="1000" class="tabs"> ... </div>**

**Компонент "Спойлери"**

**Нове**

**15 квітня**

**Спойлер – це заголовок, при натисканні на який під ним розгортається певний контент.**

**Основні можливості**

* **Використання безлічі блоків зі спойлерами**
* **Вимкнення/увімкнення функціоналу на певній ширині екрану**
* **Функція “акордеон”, коли в блоці може бути відкритий лише один спойлер**
* **Можливість закриття по кліку за межами спойлера**
* **Можливість анімованого відкриття**
* **Семантика та доступність**

**Розташування компоненту**

**Файли компоненту знаходяться в папці src/components/layout/spollers, тут ви знайдете JS-файл компоненту, HTML-файл з прикладом коду, а також SCSS-файл для стилізації компоненту.**

**Використання компоненту**

**У потрібному місці викликати сніпет spollers (класи можна замінити на потрібні). Або вручну створити структуру із відповідними дата-атрибутами. Зауважте, що додавання атрибуту data-fls-spollers-open для елемента details зробить спойлер активним (відкритим).**

***Приклад блоку з двома спойлерами:***

**<div data-fls-spollers class="spollers">  
<details class="spollers\_\_item">  
<summary class="spollers\_\_title">Заголовок спойлера</summary>  
<div class="spollers\_\_body">Контент спойлера #1</div>  
</details>  
<details class="spollers\_\_item">  
<summary class="spollers\_\_title">Заголовок спойлера #2</summary>  
<div class="spollers\_\_body">Контент спойлера</div>  
</details>  
</div>**

**У момент ініціалізації (увімкнення) функціонала спойлера контент буде прихований, а до елемента з атрибутом data-fls-spollers буде додано клас --spoller-init**

**Вимкнення/увімкнення функціоналу на певній ширині екрану**

**Для того, щоб вимкнути/включити функціонал спойлера на певній ширині екрана, необхідно для атрибуту data-fls-spollers через кому вказати потрібну ширину екрану а також тип:**

* **max (за замовчуванням) — функціонал увімкнеться на ширині меншій за вказану**
* **min — функціонал увімкнеться на ширині більшій ніж зазначена**

**<div data-fls-spollers="768,min" class="spollers"> ... </div>**

**Увімкнення режиму “акордеон”**

**Для того, щоб увімкнути режим акордеон, необхідно для елемента з атрибутом data-fls-spollers додати атрибут data-fls-spollers-one**

**<div data-fls-spollers data-fls-spollers-one class="spollers"> ... </div>**

**Тепер, при відкритті спойлера, інший відкритий спойлер у блоці закриватиметься**

**Управління швидкістю анімації**

**Для того, щоб керувати часом анімації відкриття/закриття спойлера, необхідно для елемента з атрибутом data-fls-spollers додати атрибут data-fls-spollers-speed, а в якості значення вказати час анімації в мілісекундах (за замовчуванням 500).**

**<div data-fls-spollers data-fls-spollers-speed="1000" class="spollers"> ... </div>**

**Закриття при кліку поза спойлером**

**Якщо необхідно закривати спойлер(и) при кліку поза блоком (“на порожньому місці”), слід додати потрібному заголовку(кам) атрибут data-fls-spollers-close**

**<div data-fls-spollers class="spollers">  
<details class="spollers\_\_item">  
<summary data-fls-spollers-close class="spollers\_\_title">Заголовок спойлера</summary>  
<div class="spollers\_\_body">Контент спойлера</div>  
</details>  
</div>**

**Прокрутка до заголовку**

**Якщо необхідно прокрутити скрол до заголовку відкритого спойлеру, слід додати потрібному елементу (тег details) атрибут data-fls-spollers-scroll**

**<div data-fls-spollers class="spollers">  
<details data-fls-spollers-scroll class="spollers\_\_item">  
<summary class="spollers\_\_title">Заголовок спойлера</summary>  
<div class="spollers\_\_body">Контент спойлера</div>  
</details>  
</div>**

**Є можливість більш тонких налаштувань. Наприклад, якщо треба щоб скрол враховув шапку сайту (коли вона фіксована), то варто додати атрибут data-fls-spollers-scroll-noheader.**

**Також, в самому атрибуті data-fls-spoller-scroll можна дописати числове значення яке буде враховане при скролі:**

**<div data-fls-spollers class="spollers">  
<details data-fls-spollers-scroll="20" data-fls-spollers-scroll-noheader class="spollers\_\_item">  
<summary class="spollers\_\_title">Заголовок спойлера</summary>  
<div class="spollers\_\_body">Контент спойлера</div>  
</details>  
</div>**

**Компонент “До/Після”**

**Нове**

**15 квітня**

**Компонент “До/Після” – це вивід 2х зображень, з можливістю, перетаскуючи роздиляючу, лінію відобразити одне або друге зображення.**

**Розташування компоненту**

**Файли компоненту знаходяться в папці src/components/layout/beforeafter, тут ви знайдете JS-файл компоненту, HTML-файл з прикладом коду, а також SCSS-файл для стилізації компоненту.**

**Використання компоненту**

**В HTML, у потрібному місці, викликати сніпет ba (класи можна замінити на потрібні). Або вручну створити структуру із відповідними дата-атрибутами.**

**<div data-fls-beforeafter class="before-after">  
<div data-fls-beforeafter-before class="before-after\_\_item">...</div>  
<div data-fls-beforeafter-after class="before-after\_\_item">...</div>  
<div data-fls-beforeafter-arrow class="before-after\_\_arrow"></div>  
</div>**

**За допомогою тегу IMG додайте забраження у відповідні HTML-елементи, додайте необхідну стилізацію для розділяючої лінії data-fls-beforeafter-arrow**

**<div data-fls-beforeafter class="before-after">  
<div data-fls-beforeafter-before class="before-after\_\_item">  
<img src="@img/image-1.jpg" alt="Image">  
</div>  
<div data-fls-beforeafter-after class="before-after\_\_item">  
<img src="@img/image-2.jpg" alt="Image">  
</div>  
<div data-fls-beforeafter-arrow class="before-after\_\_arrow"></div>  
</div>**

**Компонент “Галерея”**

**Нове**

**15 квітня**

**Функціонал підключає галерею (**[**LightGallery**](https://www.lightgalleryjs.com/)**). При натисканні на зображення картинки повне зображення відкривається з можливістю слайд-шоу. Можна створювати як вбудовані, так і лайтбокс-галереї (в правому куті розташовується ряд кнопок (налаштовується) для розширених функцій. Наприклад: “Закрити слайд-шоу”, “Завантаження зображення” та ін. Галерея, підтримує як прості зображення, так і відео чи фрейми.**

**Для роботи галереї необхідні зображення (відеофайли) у двох форматах: мініатюри – зображення невеликого розміру, які використовуються для попереднього перегляду та зображення повного розміру – необхідні для детального перегляду.**

**Розташування компоненту**

**Файли компоненту знаходяться в папці src/components/layout/gallery, тут ви знайдете JS-файл компоненту з базовоми налаштуваннями, HTML-файл з прикладом коду, а також папку assets з CSS-файлами компоненту.**

**Використання компоненту**

**В HTML, До тегу батька об’єктів галереї, додаємо дата-атрибут data-fls-gallery.**

**<div data-fls-gallery class="gallery"> .... </div>**

**Сам об’єкт галереї складаються з такої конструкції (тег для повного розміру зображення і всередині тег для превью). Наприклад, можна використовувати таку розмітку:**

**<div data-fls-gallery class="gallery">  
<a href="img/img-full-1.jpg" class="gallery\_\_image">  
<img alt="Превью" src="img/img-thumb-1.jpg" class="gallery\_\_preview">  
</a>  
<a href="img/img-full-2.jpg" class="gallery\_\_image">  
<img alt="Превью" src="img/img-thumb-2.jpg" class="gallery\_\_preview">  
</a>  
...  
</div>**

**Для більш широкого використання можливостей компоненту галереї, дивіться** [**документацію LiteGallery**](https://www.lightgalleryjs.com/docs/getting-started/)

**Компонент "Масонрі сітка"**

**Нове**

**15 квітня**

**Масонрі сітка - це специфічний вивід елементів таким чином, що вони заповнюють усе місце в структурних колонках. Приклад:** [**вивід зображень на Pinterest**](https://www.pinterest.com/)

**В стартовому шаблоні "Чертоги Фрілансера" 4 реалізована можливість побудови масонрі сітки на основі** [**Isotope**](https://isotope.metafizzy.co/)**. Що дозволяє не тільки виводити елементи у вигляді масонрі сітки, а й додавати функціонал фільтрації, сортування тощо**

**Розташування компоненту**

**Файли компоненту знаходяться в папці src/components/layout/masonry, тут ви знайдете JS-файл компоненту з базовоми налаштуваннями та функціоналом фільтрації, HTML-файл з прикладом коду, а також SCSS-файл для стилізації компоненту**

**Використання компоненту**

**У потрібному місці HTML-файту, ви можете скористатись сніпетом masonry, або вручну створити структуру із відповідними дата-атрибутами:**

**<div data-fls-masonry class="masonry">  
<div data-fls-masonry-item class="masonry\_\_item">A</div>  
<div data-fls-masonry-item class="masonry\_\_item">B</div>  
<div data-fls-masonry-item class="masonry\_\_item">C</div>  
</div>**

**Ось приклад масонрі сітки з можливістю фільтрації:**

**<div class="masonry">  
<ul class="masonry\_\_filter filter-masonry">  
<li class="filter-masonry\_\_item"><a href="#" data-fls-masonry-filter-link="\*" class="filter-masonry\_\_link">Усі</a></li>  
<li class="filter-masonry\_\_item"><a href="#" data-fls-masonry-filter-link="a" class="filter-masonry\_\_link">Фільтр А</a></li>  
<li class="filter-masonry\_\_item"><a href="#" data-fls-masonry-filter-link="b" class="filter-masonry\_\_link">Фільтр B</a></li>  
<li class="filter-masonry\_\_item"><a href="#" data-fls-masonry-filter-link="c" class="filter-masonry\_\_link">Фільтр C</a></li>  
</ul>  
<div data-fls-masonry class="masonry\_\_items">  
<div data-fls-masonry-item data-fls-masonry-filter="a" class="masonry\_\_item">A</div>  
<div data-fls-masonry-item data-fls-masonry-filter="b" class="masonry\_\_item">B</div>  
<div data-fls-masonry-item data-fls-masonry-filter="c" class="masonry\_\_item">C</div>  
</div>  
</div>**

**Для більш широкого використання можливостей компоненту, дивіться** [**документацію Isotop**](https://isotope.metafizzy.co/)

**Компонент "Слайдер"**

**Нове**

**17 квітня**

**У шаблоні “Чертоги Фрілансера” 4, далі ЧФ 4, є підготовка до роботи з** [**слайдером Swiper**](https://swiperjs.com/)**.**

**Розташування компоненту**

**Файли компоненту знаходяться в папці src/components/layout/slider, тут ви знайдете JS-файл компоненту з ініціалізацією та базовими налаштуваннями, HTML-файл з прикладом коду, а також SCSS-файл для стилізації компоненту.**

**Використання компоненту**

**Існує два сніпети для побудови HTML-структури слайдера:**

* **swiper – будує мінімальну структуру з уже доданими класами слайдера (для досвідченіших)**
* **swiperfull – будує повну структуру слайдера з додаванням всіх можливих елементів керування (кнопки “ліворуч/праворуч”, скролл, пагінація (булети)) з уже доданими класами слайдера. Весь код супроводжується коментарями (для новачків).**

**У файлі src/components/layout/slider/slider.js виконується підключення самого слайдера “Swiper” з пакета NPM (підключено за замовчуванням)**

**import Swiper, { Navigation } from 'swiper';**

**При необхідності можна підключити більше потрібних модулів:**

**import Swiper, { Navigation, Pagination, Lazy, Autoplay } from 'swiper';**

**Повний перелік модулів –** [**тут**](https://swiperjs.com/swiper-api#using-js-modules)

**Також, нижче за кодом, є приклад-підготовка для створення конкретного слайдера (функція initSliders();) Тут ми створюємо та налаштовуємо потрібні нам слайдери, не забуваємо вказувати модулі для конкретного слайдера**

**new Swiper('.swiper', { modules: [Navigation, Autoplay], ... }**

**Інформацію з налаштування дивись у** [**документаціі**](https://swiperjs.com/swiper-api) **на сайті слайдера.**

**Ініціалізація слайдера(ів) initSliders(); відбувається після повного завантаження сторінки нижче за кодом:**

**window.addEventListener("load", initSliders)**

**За замовчуванням, у файлі src/components/layout/slider/slider.js підключені базові, мінімально необхідні для роботи стилі слайдера import "./slider.scss"; (Для більш досвідчених). Також є можливість підключити (розкоментувавши рядок) повні стилі слайдера з npm пакету import ‘swiper/css’; (для початківців)**

**Для більш широкого використання можливостей компоненту галереї, дивіться** [**документацію Swiper**](https://swiperjs.com/swiper-api)

**Компонент "ФулПейдж"**

**Нове**

**17 квітня**

**У шаблоні вбудований функціонал поекранного прокручування (скрола) сторінки. За замовченням кожна секція сторінки займає мінімум всю висоту екрану.**

**Розташування компоненту**

**Файли компоненту знаходяться в папці src/components/layout/fullpage, тут ви знайдете JS-файл компоненту, HTML-файл з прикладом коду, а також SCSS-файл для стилізації компоненту.**

**Використання компоненту**

**В HTML, ви можете використовувати сніпет fullpage або побудувати HTML-структуру самостійно:**

**<div data-fls-fullpage class="some-class">  
<div data-fls-fullpage-section class="some-class\_\_section"> ... </div>  
...  
</div>**

**Буллети**

**Існує можливість додати кнопки перемикання секцій (буллети). Для цього до об’єкта з дата атрибутом data-fls-fullpage додати дата атрибут data-fls-fullpage-bullets**

**<div data-fls-fullpage data-fls-fullpage-bullets class="some-class">  
<div data-fls-fullpage-section class="some-class\_\_section"> ... </div>  
...  
</div>**

**Ефекти перемикання**

**Для зміни ефекту перемикання, об’єкту з дата атрибутом data-fls-fullpage додати дата атрибут data-fls-fullpage-effect і як значення додати один із наступних параметрів fade, cards, slider (за замовчуванням)**

**<div data-fls-fullpage data-fls-fullpage-effect="fade" class="some-class">  
<div data-fp-section class="some-class\_\_section"> ... </div>  
...  
</div>**

**Компонент "Цифровий лічильник"**

**Нове**

**18 квітня**

**Компонент виводить та анімує цтфрову статистику, що є дуже поширеним елементом в дизайні**

**Розташування компоненту**

**Файли компоненту знаходяться в папці src/components/layout/digcounter, тут ви знайдете JS-файл компоненту, HTML-файл з прикладом коду, а також SCSS-файл для стилізації компоненту.**

**Використання компоненту**

**В HTML, ви можете використовувати сніпет digcounter або побудувати HTML-структуру самостійно:**

**<div data-fls-watcher class="block">  
<div data-fls-digcounter class="block\_\_item">1500</div>  
<div data-fls-digcounter class="block\_\_item">500</div>  
<div data-fls-digcounter class="block\_\_item">800</div>  
...  
</div>**

**Зверніть увагу, для запуску анімації використовується [компонент "Спостерігач"](https://www.patreon.com/posts/126546933?collection=1439491), тому до батьківського елементу лічильника доданий дата-атрибут data-fls-watcher**

**Додаткові можливості**

**Зміна швидкості**

**Для зміни швидкості анімації необхідно додати атрибут data-fls-digcounter-speed до елементу лічильника та вказати швидкість у мілісекундах (1000 за замовченням)**

**<div data-fls-watcher class="block">  
<div data-fls-digcounter data-fls-digcounter-speed="500" class="block\_\_item">1500</div>  
<div data-fls-digcounter data-fls-digcounter-speed="200" class="block\_\_item">500</div>  
<div data-fls-digcounter data-fls-digcounter-speed="1700" class="block\_\_item">800</div>  
...  
</div>**

**Форматування**

**Існує можливість форматування виводу цифр відділяючи тисячі сотрі тисяч і т.д. Для цього необхідно додати атрибут data-fls-digcounter-format до елементу лічильника. Для налаштування розмежувача можно додати символ у якості значення атрибуту (за замовченням пробіл)**

**<div data-fls-watcher class="block">  
<div data-fls-digcounter data-fls-digcounter-format="." class="block\_\_item">1500</div>  
<div data-fls-digcounter data-fls-digcounter-format class="block\_\_item">500</div>  
<div data-fls-digcounter data-fls-digcounter-format="," class="block\_\_item">800</div>  
...  
</div>**

**Компонент "Прелоадер"**

**Нове**

**20 квітня**

**Компонент "Прелоадер" здатен відображати певну анімаю під час завантаження сторінки. Підходить для сторінок з великою кількістю зображень.**

**Розташування компоненту**

**Файли компоненту знаходяться в папці src/components/effects/preloader, тут ви знайдете JS-файл компоненту, HTML-файл який автоматично підключається в <head>, а також SCSS-файл для стилізації компоненту.**

**Використання компоненту**

**В HTML-файлах сторінок, де потрібен прелоадер, налаштуйте змінні які передаються через locals в шаблон сторінки:**

**...  
"preloader": {  
"enable":"true",  
"once":"false"  
},  
...**

**Де preloader->enable це вмикач компоненту, а preloader->once це вмикач лише одно показу анімації підвантаження.**

**Компонент "Курсор"**

**Нове**

**20 квітня**

**В стартовому шаблоні "Чертоги Фрілансера" 4 (ЧФ4) є можливість використовувати власну стилізацію та анімацію курсору миші**

**Розташування компоненту**

**Файли компоненту знаходяться в папці src/components/effects/cursor, тут ви знайдете JS-файл компоненту а також SCSS-файл для стилізації компоненту.**

**Використання компоненту**

**В HTML до об'єкту, всередені якого треба виводити стилізований курсор, додайте дата-атрибут data-fls-cursor**

**<html data-fls-cursor>**

**Для додавання анімаційного елементу курсору, так званої тіні, додайте ще один атрибут data-fls-cursor-shadow**

**<html data-fls-cursor data-fls-cursor-shadow>**

**Стилізуйте компонент за бажанням.**

**Компонент "Рядок що біжить"**

**Нове**

**20 квітня**

**Дуже корисний компонент з анімацією рядка з елементами**

**Розташування компоненту**

**Файли компоненту знаходяться в папці src/components/effects/marquee, тут ви знайдете JS-файл компоненту, HTML-файл з прикладом коду, а також SCSS-файл для стилізації компоненту.**

**Використання компоненту**

**В HTML скористайтесь сніпетом mar, або побудуйте структуру з використання дата-атрибуту data-fls-marquee**

**<div data-fls-marquee>  
<div>елемент один</div>  
<div>елемент два</div>  
...  
</div>**

**Додаткові можливості**

**Для коригування відступів між елементами використовуйте дата-атрибут data-fls-marquee-space зі значенням (за замовченням значення 30px):**

**<div data-fls-marquee data-fls-marquee-space="50">  
<div>елемент один</div>  
<div>елемент два</div>  
...  
</div>**

**Для коригування швидкості анімації використовуйте дата-атрибут data-fls-marquee-speed зі значенням в мілісекундах (за замовченням значення 1000ms):**

**<div data-fls-marquee data-fls-marquee-speed="2000">  
<div>елемент один</div>  
<div>елемент два</div>  
...  
</div>**

**Для зупинки анімації при наведенні миші використовуйте дата-атрибут data-fls-marquee-pause:**

**<div data-fls-marquee data-fls-marquee-pause>  
<div>елемент один</div>  
<div>елемент два</div>  
...  
</div>**

**Для зміни напрямку руху використовуйте дата-атрибут data-fls-marquee-direction та значення запрмку - left, right, top, bottom (значення за замовченням left)**

**<div data-fls-marquee data-fls-marquee-direction="right">  
<div>елемент один</div>  
<div>елемент два</div>  
...  
</div>**

**При використанні data-fls-marquee-direction 'top' або 'bottom', у об'єкта має бути фіксована висота та overflow: hidden;**

**Компонент "Хедер"**

**Нове**

**21 квітня**

**В стартовому шаблоні "Чертоги Фрілансера" 4 (ЧФ4) реалізований окремий компонент шапки сайту та його плагини**

**Розташування компоненту**

**Файли компоненту знаходяться в папці src/components/layout/header, тут ви знайдете JS-файл для написання логіки компоненту, HTML-файл з базовим кодом, SCSS-файл для стилізації компоненту.**

**Також в папці src/components/layout/header/plugins присутні плагіни компоненту**

**Використання компоненту**

**Підключіть HTML-файл компоненту на свою сторінку за допомогою <include> (підключено за замовченням).**

**Використання плагінів**

**Плагін "Scroll"**

**Дозволяє керувати класами які додаються <header> під час скролу. Для підключення додайте дата-атрибут data-fls-header-scroll до тега <header>:**

**<header data-fls-header data-fls-header-scroll class="header">  
...  
</header>**

**Тепер, при скролі більше одного пікселя до шапки буде додаватись клас --header-scroll**

**Для керування кількістю прокручених пікселів перед додаванням класу, додайте дата-атрибуту data-fls-header-scroll значення:**

**<header data-fls-header data-fls-header-scroll="500" class="header">  
...  
</header>**

**Також можна підключити функціонал "зникаючої" шапки для покращення юзабіліті сайту, для цього додайте дата-атрибут data-fls-header-scroll-show до тега <header>:**

**<header data-fls-header data-fls-header-scroll data-fls-header-scroll-show class="header">  
...  
</header>**

**Тепер шапка буде зникати при скролі вниз та з'являтись через 500ms паузи. Для керування паузою додайте значення дата-атрибуту data-fls-header-scroll-show:**

**<header data-fls-header data-fls-header-scroll data-fls-header-scroll-show="1000" class="header">  
...  
</header>**

**При необхідності змінити стилізацію плагіну "Scroll" використовуйте файл src/components/layout/header/plugins/scroll/scroll.scss**

**Компонент "День/Ніч"**

**Нове**

**22 квітня**

**В стартовому шаблоні "Чертоги Фрілансера" 4 (ЧФ4) реалізований функціонал зміни стилю сайту темний / світлий, опираючись на налаштування системи користувача або за часом доби або за допомогою кнопки.**

**Розташування компоненту**

**Файли компоненту знаходяться в папці src/components/effects/darklite, тут ви знайдете JS-файл компоненту та SCSS-файл для стилізації**

**Використання компоненту**

**Додайте на свою HTML-сторінку дата-атрибут data-fls-darklite, наприклад до тегу <html>:**

**<html data-fls-darklite>**

**Цього достатньо щоб компонент розпізнав налаштування системи користувача і додав відповідний дата-атрибут до тегу <html>: data-fls-darklite-light або data-fls-darklite-dark**

**Якщо ви хочете щоб атрибути стилю додавались за часом доби, додайте дата атрибут data-fls-darklite-time:**

**<html data-fls-darklite data-fls-darklite-time>**

**Тепер атрибут data-fls-darklite-dark буде додаватиз з 18 години вечора до 5 години ранку, весь інший час буде додаватись атрибут data-fls-darklite-light**

**Для налаштування власного проміжку в якому буде додаватись атрибут data-fls-darklite-dark, вкажіть для атрибуту data-fls-darklite-time значення через кому:**

**<html data-fls-darklite data-fls-darklite-time="13,23">**

**Тепер атрибут data-fls-darklite-dark буде додаватись у проміжку з 13 годин до 23 години**

**Якщо ви хочете надати користувачу можливість самостійно вмикати ту чи іншу тему сайту, додайте у ваш інтерфейс кнопку з дата-атрибутом data-fls-darklite-set:**

**<button data-fls-darklite-set>...</button>**

**При натискані кнопки поточна тема змінеться на протилежну, і в браузер користувача буде збережений його вибір.**

**Щоб надати користувачу можливість видаляти збережений вибір теми, додайте у ваш інтерфейс кнопку з дата-атрибутом data-fls-darklite-reset:**

**<button data-fls-darklite-reset>...</button>**

**Компонент "Кошик"**

**Нове**

**23 квітня**

**В стартовому шаблоні "Чертоги Фрілансера" 4 (ЧФ4) реалізований функціонал додавання товарів в кошик, з можливістю анімації зображення товару**

**Розташування компоненту**

**Файли компоненту знаходяться в папці src/components/forms/addtocart, тут ви знайдете JS-файл компоненту, HTML-файл з прикладом коду, а також SCSS-файл для стилізації компоненту.**

**Використання компоненту**

**В HTML беспосередньо до об'єкту який виводить кількість товарів в кошику додайте дата-атрибут data-fls-addtocart:**

**<div class="cart">  
<span class="cart\_\_quantity" data-fls-addtocart>0</span>  
</div>**

**Для кнопки "Купити", "Додати у кошик" тощо додайте дата-атрибут data-fls-addtocart-button:**

**<button data-fls-addtocart-button>Додати у кошик</button>**

**Для використання додаткового функціоналу, додаємо обгортку товару що додається в кошик, з дата-атрибутом data-fls-addtocart-product:**

**<div data-fls-addtocart-product class="product">  
...  
<button data-fls-addtocart-button>Додати в кошик</button>  
</div>**

**Тепер ми маємо можливість додати керування кількістю товарів які будуть додані у кошик. Для цього необхідно додати дата-атрибут data-fls-addtocart-quantity для тегу <input> який керує кількістю, або використати компонент "Кількість" ЧФ4 теж з додаванням data-fls-addtocart-quantity:**

**<div data-fls-addtocart-product class="product">  
...  
<input data-fls-addtocart-quantity autocomplete="off" type="text" name="form[]" value="1">  
<button data-fls-addtocart-button>Додати у кошик</button>  
</div>**

**Або**

**<div data-fls-addtocart-product class="product">  
...  
<div data-fls-quantity class="quantity">  
<button data-fls-quantity-minus type="button" class="quantity\_\_button quantity\_\_button--minus"></button>  
<div class="quantity\_\_input">  
<input data-fls-addtocart-quantity data-fls-quantity-value autocomplete="off" type="text" name="form[]" value="1">  
</div>  
<button data-fls-quantity-plus type="button" class="quantity\_\_button quantity\_\_button--plus"></button>  
</div>  
<button data-fls-addtocart-button>Додати у кошик</button>  
</div>**

**Також, є можливість анімувати зображення товару створюючи "політ у кошик". Для цього тегу <img> товару додайте дата-трибут data-fls-addtocart-image:**

**<div data-fls-addtocart-product class="product">  
...  
<img src="..." data-fls-addtocart-image alt="Image">  
<button data-fls-addtocart-button>Додати у кошик</button>  
</div>**

**Для керуванням швидкістю анімації додайте значення атрибуту data-fls-addtocart-image у мілісекундах (за замовченням 500ms):**

**<div data-fls-addtocart-product class="product">  
...  
<img src="..." data-fls-addtocart-image="1000" alt="Image">  
<button data-fls-addtocart-button>Додати у кошик</button>  
</div>**

**Компонент "Google Maps"**

**Нове**

**16 травня**

**В стартовому шаблоні "Чертоги Фрілансера" 4 (ЧФ4) реалізована можливість додавання функціоналу** [**Google Maps**](https://www.google.com/maps)

**Розташування компоненту**

**Файли компоненту знаходяться в папці src/components/layout/map, тут ви знайдете JS-файли компоненту та налаштувань, HTML-файл з прикладом коду, а також SCSS-файл для стилізації компоненту.**

**Використання компоненту**

**Для початку ви можете отримати власний KEY для повноцінного використання функціоналу Google Maps та додати його в константу MAP\_KEY в файлі \_settings.js в папці компоненту. Або відкласти цей крок і працювати з мапаки в режимі розробника.**

**Для додавання мапи в потрібному місці HTML скористайтесь сніпетом map, або побудуйте структуру використовуючі певні дата-атрибути.**

**Цей приклад виведе мапу з одним маркером:**

**<div data-fls-map class="map">  
<div data-fls-map-marker class="map\_\_marker" data-fls-map-lat="50.449688" data-fls-map-lng="30.52348" data-fls-map-title="Title 1"></div>  
<div data-fls-map-body class="map\_\_body" data-fls-map-lat="50.449688" data-fls-map-lng="30.52348"></div>  
</div>**

**Налаштування компоненту**

* **data-fls-map - додається до оболонки конструкції мапи**
* **data-fls-map-marker - додається до елементу який є маркером на мапі**
* **data-fls-map-body - додається до елементу який виводить мапу**
* **data-fls-map-lat та data-fls-map-lng - встановлють координати центру мапи та\або виводу маркеру**
* **data-fls-map-title - виводить підказку для маркеру**
* **data-fls-map-icon - виводить власне зображення маркеру**
* **data-fls-map-marker-zoom - при наявності атрибуту для маркера, робить зум на маркер при кліку по ньому**
* **data-fls-map-marker-popup - при наявності атрибуту для маркера, відкриває модальне вікно (popup), яке вказане в значенні атрибуту, при кліку по маркеру. Дивись** [**компонент "Модальне вікно"**](https://www.patreon.com/posts/126545546)

**Додатково можна вказати власний** [**стиль Google Maps**](https://snazzymaps.com/) **в файлі \_settings.js в папці компоненту.**

**Компонент "Форма"**

**У шаблоні “Чертоги Фрілансера” 4 (ЧФ4) реалізований функціонал, що допомагає вирішувати різні рутинні завдання при роботі з формами.**

**У цьому документі описано загальний функціонал роботи з полями, плейсхолдерами, валідацією, варіантами відправлення форм тощо. В окремих розділах буде описано функціонал конкретних елементів форм, таких як:**

* **Надсилання листів на пошту (Робота з PHPMailer)**
* **Компонент "Селект"**
* **Компонент "Чекбокс та радіо"**
* **Компонент "Інпут"**
* **Компонент “Зоряний рейтинг”**
* **Компонент “Кількість”**
* **Компонент “Повзунок”**
* **Список доповнюється...**

**Розташування компоненту**

**Файли компоненту знаходяться в папці src/components/layout/map, тут ви знайдете JS-файли компоненту та налаштувань, HTML-файл з прикладом коду, а також SCSS-файл для стилізації компоненту.**

**Використання компоненту**

**В HTML або побудуйте структуру форми з використанням дата-атрибуту data-fls-form:**

**<form data-fls-form method="POST" action="..." >  
<input required type="email" name="form[email]" data-fls-form-errtext="Error" placeholder="E-mail" class="input">  
<button type="submit" class="button">Send</button>  
</form>**

**Загальна робота з полями форм**

**Робота з полями форм має на увазі наступний функціонал:**

* **Додавання класів полю та його батькові при фокусі**
* **Можливість моментальної валідації поля при втраті фокусу**

**Додавання класів**

**За замовчанням, при виникненні фокусу, до поля і до його безпосереднього батьківського елементу, додасться клас --form-focus. Якщо нам потрібно вимкнути додавання класів для конкретного поля, йому слід додати атрибут data-fls-form-nofocus**

**Валідація поля при втраті фокусу**

**Для того, щоб скрипт почав викликати функціонал валідації поля в момент втрати ним фокусу, слід додати полю атрибут data-fls-form-validatenow.**

**Варто додати, що функціонал роботи з полями форм також видалить помилку додану до поля валідатор при отриманні фокусу. Докладніше про функціонал валідації полів дивись далі в цьому документі.**

**Валідація елементів форм**

**Щоб елемент форми почав проходити валідацію, йому слід додати атрибут required. Тепер при надсиланні форми, а також якщо елементу доданий атрибут data-fls-form-validatenow, він перевірятиметься на предмет правильного заповнення.**

**При доданому атрибуті data-fls-form-validatenow валідація буде відбуватись миттєво  
при втраті полем фокусу**

**Щоб включити особливі правила валідації поля, можна змінити значення його отрибуту type:**

* **email – увімкне валідацію на введення коретного E-mail**
* **йде робота над новими пресетами**

***Приклад:***

**<input required type="email" name="form[]">**

**Якщо елемент заповнений неправильно, до нього, а також до його батька додасться клас --form-error. Якщо ми хочемо додатково вивести довільний текст помилки, слід вказати його в атрибуті data-fls-form-errtext і додати до поля:**

**<input data-fls-form-errtext="Помилка E-mail" required type="email" name="form[]">**

**Тепер, при виникненні помилки валідації, під елементом додасться об’єкт із класом --form-error, який містить ваш текст помилки.**

**Нагадаю, що функціонал роботи з полями видалить класи помилок та об’єкт із класом --form-error при отриманні полем фокусу.**

**Для відключення валідації елементів конкретної форми їй слід додати атрибут data-fls-form-novalidate**

**Відправлення форм**

**Валідація елементів форми**

**За замовчанням, при відправленні форми поля відмічені required будуть проходити валідацію. Для відключення валідації елементів конкретної форми їй слід додати атрибут data-fls-form-novalidate**

**Режими відправлення форми**

**У ЧФ4 існує кілька режимів контролю надсилання форм:**

* **Стандартне HTML відправлення форми (за замовчуванням) – якщо валідація пройдена (була включена), форма відправиться на вказану в атрибуті action адресу (з переходом сторінки), методом вказаним в атрибуті метод.**
* **AJAX відправка форми – якщо валідацію пройдено (було включено), форма відправиться AJAX запитом на вказану в атрибуті action адресу, методом вказаним в атрибуті method. Сторінка не перезавантажиться, всі елементи форми повернуться до вихідних значень (очищення форми).**
* **Режим імітації відправки форми – якщо валідацію пройдено (було включено), форма нікуди не відправиться, сторінка не перезавантажиться, всі елементи форми повернуться до вихідних значень (очищення форми). Використовується для демонстрації додаткових можливостей форм, таких, як показ попапа про успішну відправку.**

**Для включення режиму AJAX надсилання достатньо додати значення ajax атрибуту data-fls-form, а якщо потрібний режим імітації надсилання додаємо значення dev.**

**<form data-fls-form="ajax" class="form" method="POST" action="...">  
...  
</form>**

**Прокручування до елемента з помилкою**

**Буває, що форма дуже велика і, при виникненні помилки валідації, добре показати елемент з помилкою користувачеві. Для цього є функціонал “прокрутка до елементу з помилкою”. Для включення достатньо додати формі атрибут data-fls-form-gotoerr**

**<form data-fls-form-gotoerr class="form" action="...">  
...  
</form>**

**Показ попапа після надсилання форми**

**Якщо необхідно, після відправки форми, показати попап додаємо до форми атрибут data-fls-form-popup і як значення вказуємо ім'я попапа**

**Тільки для режимів ajax або dev.** [**Компонент попапів**](https://www.patreon.com/posts/126545546?collection=1439491) **також має бути підключений**

**<form data-fls-form="ajax" data-fls-form-popup="form-message" class="form" action="...">  
...  
</form>**

**Події**

**Після кожної відправки форми спрацьовує подію у formSent, її можна відловити в будь-якій частині коду:**

**document.addEventListener("formSent", function (e) {  
// Форма  
const currentForm= e.detail.form  
})**