УДК 004.4
DOI: https://doi.org/10.26642/ten-2019-2(84)-70-76

В.В. Болотіна, асист.
О.Г. Чижмотря, ст. викл.
О.В. Чижмотря, ст. викл.
Державний університет «Житомирська політехніка»

Перспективи використання технології Flexbox при адаптивній верстці вебсторінок

(Представлено: д.т.н., проф. Пількевич І.А.)

Правильна верстка сайту відповідає за коректне відображення контенту. Від верстальника вимагається таке відображення в браузері, яке буде найбільш близьким до попередньо створеного макета дизайну сайту, заздалегідь розробленого за допомогою графічних редакторів. Для цього існує багато інструментів: HTML, CSS, різного типу фреймворки. Розрізняють «жорстку» (фіксовану) і «гумову» (адаптивну) верстку сайту. При «жорсткій» верстці всі елементи вебсторінки мають завжди фіксовані розміри, незалежно від розміру монітора користувача й установленого дозволу екрана. Адаптивна верстка сайту дозволяє змінювати розміри елементів сторінки, підлаштовуючись під різні розміри й дозволи моніторів. Найбільш значущою і активно досліджуваною сферою веброзробки є адаптивна верстка. Зважаючи на велике різноманіття технологій, що дозволяють вирішувати ряд проблем, які постають перед розробником при верстці адаптивних вебсайтів, варто дослідити найбільш ефективні з них. Досліджено технологію адаптивної верстки вебсторінок Flexbox. Виокремлено основні особливості застосування Flexbox для вирішення основних завдань, таких як вирівнювання графічного контенту відповідно до ширини екрану, автоматичне визначення центру для точного вирівнювання контейнерів на сторінці. Вирізнено основні переваги та перспективи використання технології Flexbox при адаптивній верстці вебсторінок, які містять динамічний контент, що формує чітке уявлення про подальше впровадження цієї технології у процес веброзробки. Для коректного відображення використовують адаптивну верстку. Щоб реалізувати адаптивність сайту, прописуються додаткові стилі, що, крім більшого обсягу роботи, тягне за собою передбачення поведінки кожного з елементів.

Ключові слова: *веброзробка; адаптивна верстка; Flexbox; вебсторінка; вебсайт; адаптивність; фреймворк.*

Актуальність теми. Сучасний світ спостерігає стрімкий розвиток технологій. Щодня у людей з’являються все нові і нові гаджети. В наш час людина велику частину часу приділяє відвідуванню мережі Інтернет. Завдяки Інтернету ми спілкуємося, навчаємося, розважаємося, оплачуємо послуги та здійснюємо покупки. В суспільстві, де людина знаходиться в постійному русі, дуже важливо не витрачати час дарма, тому більшість людей для перегляду вебсторінок та здійснення різного типу операцій використовує мобільні телефони та планшети, що дозволяє знаходитися в мережі постійно, де б вони не були. Тому, вимогою сучасного світу до вебресурсів є їх адаптивність до різноманітних розширень екранів пристроїв.

Кожен звичайний сайт можна переглянути за допомогою мобільного пристрою. Однак виникає ряд нюансів при перегляді, наприклад, для читання окремих блоків тексту на відносно невеликому екрані необхідно масштабувати сторінку, змінювати розміри зображень, інколи навіть підвантажувати стиснені зображення. Як наслідок, втрачається читабельність, зручність, сайт потрібно постійно перегортати, розтягувати, наближати. Багато елементів управління сайтом незручно використовувати, оскільки сторінки не розраховані на управління за допомогою дотиків до екрану. Для вирішення цих проблем програмістами було розпочато розробку вебсайтів, які здатні зручно відобразити інформацію на будь-якому пристрої з різними дозволами екранів.

Сучасний front-end розробник має активно застосовувати на практиці різні інструменти, що дозволяють автоматизувати процес верстки макетів і програмування клієнтської складової проекту. Для цього на даному етапі розвитку вебтехнологій існує безліч фреймворків як великих, так і незначних: системи збирання, пакетні менеджери, велика кількість пакетів для розв’язання задач будь-якого рівня, препроцесори, шаблонізатори, які створені спростити і підвищити продуктивність роботи фахівця в цій області. Новачкам буває складно розібратися в такій кількості інструментів для верстки, оскільки тут потрібно постійно розвиватися і знати основи. Тому продовжують верстати вже застарілими способами. Якщо раніше верстали на таблицях, а потім перейшли на блочну верстку, то зараз масштабного застосування набирає технологія Flexbox.

Однією із вимог сучасності постає проблема обробки та відображення інформації на вебсторінках, що подається за допомогою пристроїв IoT, тобто таких, що входять до системи інтернету речей і представляють будь-які автономні пристрої, підключені до мережі Інтернет, якими можна керувати дистанційно. Нині IoT належить до мільярдів фізичних пристроїв по всьому світі, які тепер підключені до Інтернету, аналізують і обробляють величезну кількість даних. Передбачається, що в майбутньому Інтернет-речі стануть активними учасниками бізнесу, інформаційних і соціальних процесів, де зможуть взаємодіяти між собою, обмінюючись інформацією про навколишнє середовище, не потребуючи при цьому втручання людини. Завдяки процесорам і бездротовим мережам в частину IoT можна перетворити все що завгодно – від пігулки до літака. Це додає рівень цифрового інтелекту пристроям, які в іншому випадку були б неактивними, дозволяючи їм спілкуватися без участі людини і поєднання цифрових і фізичних світів.

Аналіз останніх досліджень та публікації, на які спирається автор. Враховуючи те, що на даному етапі інструментів для створення адаптивних вебсторінок існує велика кількість, молодим спеціалістам дуже важко обрати правильну технологію з ряду існуючих. Адже раніше верстку виконували використовуючи таблиці, згодом популярності набрала верстка за допомогою блоків, а на сьогодні популярності серед розробників набирає використання Flexbox у поєднанні з сітками, медіа-запитами та вже готовими css-фреймворками.

Дослідження в сфері використання різноманітних інструментів для реалізації найбільш швидким та сучасним методом адаптивної верстки не зупиняються і до сьогодні. Філімоненкова Т.М., Дунаєвский А.С. [1] аналізують найкращі підходи до реалізації адаптивного сайту для екранів з різним розширенням, а також ними проаналізовано найпопулярніші технології адаптивної верстки, були описані можливості сервісів і переваги кожного з них. Фадєєва А.С. [2] описує стратегії адаптації веб-сторінок до мобільних пристроїв, також наведено їх переваги та недоліки. У статті Дмитрієвої Ю.С. [3] на основі статистики, що показує кількість користувачів, які віддають перевагу користуванню мережею Інтернет за допомогою мобільних пристроїв, була обґрунтована необхідність використання адаптивних інтерфейсів. Метою статті було створення методики, що дозволяла б привести до адаптивної версії застарілі сайти, при цьому покращити їх показники у швидкості завантаження. Проведено аналіз декількох підходів до розробки адаптивних вебсторінок, роз'яснено відмову від використання окремої мобільної версії як альтернативи адаптивному дизайну. В дослідженнях Р.Р. Биктимиров, А.Б. Джемалетдинов [4] пропонують варіанти використання HTML і CSS для front-end розробки при побудові вебсайтів, також описані вимоги до використання інструментів, які допоможуть, в першу чергу, забезпечити конкурентоспроможність front-end розробки вебсайту.

Постановка проблеми у загальному вигляді та її зв'язок з важливими практичними завданнями. Для регулювання дорожнього руху використовуються IoT-техгології, зокрема на автошляху в потенційно-небезпечних місцях, які будуть обиратися, виходячи з наданих даних про аварійно-небезпечні частини шляхів і місця концентрації ДТП управліннями патрульної поліції. Сучасний автошлях являє собою складну систему, важливою складовою якої для безпеки руху є камери. Вони розташовані по всій довжині шляху, що становить приблизно на відстані від Житомира до Києва на 120 км 250 камер, з яких щосекунди надходить фотозйомка стану дорожнього руху. В будь-якому випадку всі ці фотографії передаються через INTERNET по каналам 3G, CPRS на сервер і зберігаються там необхідну кількість часу. Виникає проблема: під час розгляду і аналізу дорожньо-транспортних пригод співробітники поліції змушені розглядати непристосованими для цього засобами велику кількість фотографій. Тому виникає питання створення швидкого інструментарію для швидкого перегляду відображення цих фотографій. На сьогоднішній день для швидкої та адаптивної верстки сторінок з динамічним контентом, що складається з графічних зображень, які надходять з пристроїв IoT-системи, існують такі вебтехнології, як Flexbox та CSS Grid.

Метою статті є дослідження сучасних технологій верстки для швидкого і якісного відображення великих обсягів графічних даних, що надходять з пристроїв IoT-системи.

Викладення основного матеріалу. При розробці вебсайтів сучасний front-end розробник активно має використовувати на практиці різноманітні інструменти, що дозволять йому автоматизувати процес верстки макетів і програмування клієнтської частини проекту. На цьому етапі розвитку вебтехнологій існує величезна кількість фреймворків, бібліотек, систем збірки, препроцесорів, шаблонізаторів, що дозволяють значно підвищити продуктивність роботи спеціалістів в цій сфері.

На жаль, початківцям у цій сфері складно розібратися в такій кількості інструментів, оскільки веброзробка вимагає постійного розвитку та активного і стабільного вивчення все нових та нових технологій.

Розробники протягом довгого часу використовували для верстки таблиці, float-елементи, inline-block й інші CSS властивості, щоб надати блокам потрібне розташування. Прості речі, такі як вертикальне центрування, здійснювалися досить складно. Створення ж макета на основі гнучких сіток вважається вдалою практикою, саме тому широкого вживання набули CSS-фреймворки, що організовані на основі сіток.

Flexbox – це сучасна та зручна технологія верстки, яка значно спрощує процес адаптивної верстки вебсторінок. Головною ідеєю верстки за допомогою Flexbox є надання контейнеру здатності змінювати ширину, висоту, порядок своїх елементів для найкращого та повного заповнення простору (в більшості випадків – для підтримки всіх видів дисплеїв і розмірів екранів), що дає безсумнівні переваги при вирішенні такої проблеми, як розміщення різних блоків зображень на вебсторінці, які надходять з пристроїв IoT. Потік зображень є неперервним і кількість зображень постійно змінюється, а використання Flexbox робить контейнер резиновим і рівномірно розміщає зображення.

Історія розвитку технології Flexbox розпочалася ще з 2008 року, коли CSS Working Group провели огляд «Flexible Box Model», який ґрунтувався на XUL (XML User Interface Language – мова розмітки в додатках Mozilla) і XAML (Extensible Application Markup Language – мова розмітки для додатків Microsoft).

А вже в 2009 році було опубліковано чорновий варіант «Flexible Box Layout Module». Chrome і Safari додають часткову підтримку, в той час, коли Mozilla починає підтримувати XUL-подібний синтаксис, відомий як «Flexbox 2009».

У 2011 році – Tab Atkins працюють над розвитком Flexbox і опубліковують два варіанти демо-версій. У цих чернетках синтаксис значно змінюється. Chrome, Opera і IE 10 впроваджують підтримку даного синтаксису, відомого під назвою «Flexbox 2011».

У 2012 році – синтаксис знову змінюється і уточнюється. Специфікація отримує статус Candidate Recommendation і стає відома під назвою «Flexbox 2012». Браузери Opera впроваджує безпрефіксну підтримку, а Google Chrome підтримує поточну специфікацію з префіксами, а Mozilla без них, IE10 додає підтримку застарілого синтаксису «Flexbox 2011».

Як це не дивно, але застарілий синтаксис зразка 2009 року досить непогано підтримується більшістю браузерів, підтримка реалізована в: браузерах Chrome, Firefox 2+, Safari 3.1+ і ін. Вона є практично скрізь, за винятком IE 9 і ранніх версій IE і Opera. Але, на жаль, реалізація цієї технології в браузерах була неповною і частково неузгодженою, що і стало причиною перегляду специфікації.

Незважаючи на те, що застарілий синтаксис підтримується браузерами, його використання не рекомендується через те, що попередня версія специфікації вже є неактуальною. В майбутньому через певний проміжок часу браузери неодмінно зовсім припинять її підтримку. Оскільки новий синтаксис значно простіший у вивченні і застосуванні, то ймовірніше за все, саме його реалізація буде більш масштабною [5]. Ті браузери, які ще до цього часу не підтримували модулі Flexbox, почнуть це використовувати вже в новій формі, передбаченій останньою специфікацією, що отримала статус CR (Candidate Recommendation). У 2019 році підтримка Flexbox браузерами за версією сайту CANIUSE [6] сягає 89,81 %.

Практично в кожному макеті, що адаптивно верстається, виникає потреба розмістити блок елементів на різних відстанях та позиціях. Для того щоб розв'язати цю задачу, раніше використовувалася властивість float, проте вона мала безліч недоліків. Цю саму задачу ми можемо розв'язати, задавши основному контейнеру властивість display:flex і всі блоки, що входять до нього, стануть flex-елементами, які автоматично розміщуються по горизонталі [7]. Для того щоб змінювати розташування елементів, варто використовувати властивість flex-direction, що приймає такі параметри:

1. Row – розміщення блоків по-горизонталі.
2. Row-reverse – розміщення блоків по-горизонталі, але в напрямку з права на ліво.
3. Column – розміщення блоків по-вертикалі.
4. Column-reverse – розміщення блоків по-вертикалі, але блоки розміщуються знизу вверх.

Не менш важливою властивістю Flexbox є властивість flex-wrap, яка реалізує адаптивність блоків. flex-wrap визначає, чи буде flex-контейнер однорядковим або багаторядковим. Використовуючи параметр flex-wrap:wrap, реалізуємо переміщення блоків залежно від заповнення ними ширини екрану (рис. 1). Оскільки пристрій IoT з великою періодичністю подає зображення на сторінку, в рядку зображень велика кількість і вони не поміщаються по-ширині екрана, Flexbox переносить їх автоматично на новий рядок, що є зручним при зміні розміру вікна.

Для визначення переваг Flexbox варто виконати порівняння технології з CSS Grid. Для більш точного результату виконаємо порівняння на прикладі HTML-сторінки, яку ми зверстаємо за допомого Flexbox та CSS Grid. Макет сторінки складається з карточного інтерфейсу (рис 1). Основні функції, які потрібно створити, зберігаючи CSS та HTML, полягають у:

1. Розміщенні основних функцій на макеті.
2. Створенні адаптивної сторінки.

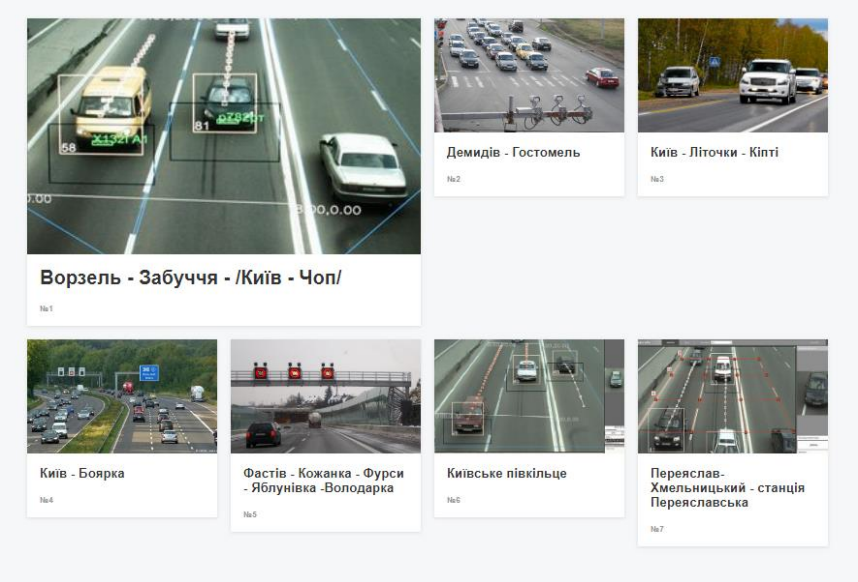

Рис. 1. Приклад сторінки для реалізація основних задач верстки вебсторінки

Для детального порівняння можливостей Flexbox та CSS Grid розв'яжемо задачі, що є найпопулярнішими серед тих, що постають перед розробником при верстці адаптивних вебсторінок.

Задамо блоку-обгортці ширину і помістимо його по центру. Трохи нижче додамо правила по сітці:

```
.band {
  width: 90%;
  max-width: 1240px;
  margin: 0 auto;

  display: grid;

  grid-template-columns: 1fr;
  grid-template-rows: auto;
  grid-gap: 20px;
}
```

Найважливіший момент – ми задали для .band властивість display: grid ;. Далі ми вказали grid-template-columns: 1fr, тобто кожна колонка буде займати одну доступну область. Поки що ми оголосили всього одну колонку, тому кожна колонка буде займати всю ширину. Наступна властивість – grid-template-rows: auto;. Це значення за замовчуванням, яке можна було б і не записувати. Воно означає, що висота ряду буде залежати тільки від контенту. Остання властивість - grid-gap: 20px. Вона додає відступи між рядами і колонками.

Тепер використаємо медіа-запити для коректного відображення сторінки на пристроях з різними екранами.

На більш широких екранах (500px взято довільно) ми будемо змінювати властивість grid-template-columns, щоб помістити дві картки в один ряд. Тепер у нас буде дві колонки, кожна буде займати свою область:

```
@media only screen and (min-width: 500px) {
  .band {
    grid-template-columns: 1fr 1fr;
  }
}
```

І для дуже великих екранів ми створимо чотири колонки:

```
@media only screen and (min-width: 850px) {
  .band {
    grid-template-columns: 1fr 1fr 1fr 1fr;
  }
}
```

Це дало нам досить хороший макет сітки. Але тепер нам необхідно використати технологію Flexbox. За допомогою використання Flexbox зробимо картки схожими на картки. І почнемо з цього:

```
.card {
  background: white;
  text-decoration: none;
  color: #444;
  box-shadow: 0 2px 5px rgba(0,0,0,0.1);
  display: flex;
  flex-direction: column;
  min-height: 100%;
}
```

Ми прописали базові стилі: білий фон, прибрали підкреслення тексту, сірий текст і невелику тінь box-shadow для більшої глибини. Далі ми перетворили нашу картку в флекс-елемент за допомогою display: flex ;. Дуже важливо, ми будемо вирівнювати контент картки по вертикалі за допомогою Flexbox. Саме тому ми задали вертикальну вісь flex-direction: column ;. Щоб картки заповнювали всю висоту батька (осередки сітки), ми вказали min-height: 100% ;.Тепер додамо position: relative; і transition, щоб ми могли рухати картку при наведенні миші:

```
position: relative;
top: 0;
transition: all .1s ease-in;
```

Номер фотографії необхідно вирівняти по нижній межі картки, незалежно від кількості контенту зверху. Ось тут нам допоможе Flexbox:

```
.card article {
  padding: 20px;
  flex: 1;

  display: flex;
  flex-direction: column;
  justify-content: space-between;
}
```

За допомогою скорочення flex: 1; ми робимо так, щоб флекс-елемент (дочірній елемент флекс-контейнера) займав весь вільний простір. Далі ми вказуємо, що тег article є окремим флекс-контейнером, і задаємо flex-direction: column; для вертикального вирівнювання. Остання властивість justify-content: space-between; рівномірно розподіляє все флекс-елементи по осі на однакових відстанях один від одного. Тепер потрібно прибрати дивні параграфи посередині карток. Для правильного вирівнювання додамо flex-grow: 1; (Або просто flex: 1;) до них, щоб ці параграфи займали вертикальний простір, який залишився, і притискалися до верхньої межі:

```
.card p {
  flex: 1; /* розтягуємо p */
  line-height: 1.4;
}
```

Тепер майже все. Однак CSS Grid дозволяє нам повністю перебудовувати макет, розміщуючи елементи в будь-якому порядку і будь-яких розмірів. Зробимо це в першому медіа–запиті:

```
@media only screen and (min-width: 500px) {
    ...
  .item-1 {
    grid-column: 1/ span 2;
  }

}
```

Крім того, що перший елемент має бути 500px, він також повинен починатися на першому рядку сітки і займати дві колонки. Інші елементи сітки розміщуються автоматично.

Виходячи з порівняння, визначити кращий підхід практично не можливо, як Flexbox, так і CSS Grid мають свої переваги та недоліки і для досягнення найкращого результату їх краще використовувати в поєднанні один з одним:

– CSS Grid відмінно підходить для побудови більш широкої картини. Вона полегшує управління макетом сторінки і навіть може допомогти в створенні асиметричних проектів;

– Flexbox відмінно підходить для вирівнювання вмісту всередині елементів. Потрібно використовувати властивості flex, щоб розміщувати окремі блоки проекту;

– необхідно використовувати сітки CSS для 2D-макетів (рядків і стовпців);

– Flexbox працює краще тільки в одному вимірі (рядки або стовпці).

Для ідеального розміщення великої кількості потокових зображень є надзвичайно важливим, щоб макет був максимально гнучким і час на перебудову макета скорочувався до мінімуму. Визначальним аспектом гнучкого адаптивного макета є можливість «згинати» flex-елементи, змінюючи їх ширину чи висоту (залежно від того, який розмір знаходиться на головній осі), щоб заповнити доступний простір по головній осі. Це реалізується за допомогою властивості flex. Flex-контейнер розподіляє вільний простір між своїми дочірніми елементами (пропорційно їх коефіцієнту flex-grow) для заповнення контейнера, або стискає їх (пропорційно їх коефіцієнту flex-shrink), щоб запобігти переповненню. Flex-елемент буде повністю «негнучкий», якщо його значення flex-grow і flex-shrink дорівнюють нулю, і «гнучкий» в протилежному випадку [8].

Спираючись на всі переваги та можливості технології, можна сказати, що Flexbox є досить потужним інструментом при верстці сторінок з динамічним контентом, та таким, який постійно збільшує свої розміри. Flexbox найкраще підходить для складових частин програми та дрібномасштабних макетів, в той час як CSS Grid більше використовується для макетів великого масштабу.

Дослідивши використання технології Flexbox при верстці адаптивних сторінок, можна виокремити основні переваги цієї технології [10, с. 156–176]:

1. Flex робить блок «гумовим», завдяки чому всі блоки стають гнучкими. Всі необхідні елементи можна стиснути і розтягнути за спеціальними правилами, займаючи при цьому весь потрібний простір сторінки.
2. Базова лінія розміщення блоків легко вирівнюється по горизонталі чи вертикалі.
3. Не важливим є порядок розміщення елементів в шаблоні, при необхідності який можна легко змінити в стилях, що є особливо важливим фактором у певних аспектах адаптивної верстки.
4. Для заповнення всього простору блоки можуть автоматично розміщуватись у декілька рядків або стовпчиків.
5. Особливість Flexbox є і те, що він адаптований під реверсне розміщення блоків. Для Flexbox немає поняття ліво і право, а є початок і кінець.
6. Синтаксис правил CSS швидкий та інтуїтивно простий, його легко засвоїти початківцям.

Висновки та перспективи подальших досліджень. Впровадження технології Flexbox у процес адаптивної верстки вебсторінок з динамічним контентом надає розробнику широкий вибір гнучких властивостей, що значно спрощують процес створення блоків сторінки, які постійно змінюються та вміщують в собі контент різних розмірів, а також використання технології в поєднанні з media-запитами та CSS Grid дозволить досягнути швидких та якісних результатів при адаптивній верстці повноцінної вебсторінки.

Список використаної літератури:

1. *Филимоненкова Т.Н.* Технологии адаптивной верстки сайтов / *Т.Н. Филимоненкова, А.С. Дунаевский* // World science: problems and innovations. – 2017. – С. 78–81.
2. *Фадеева А.С.* Адаптивный дизайн или мобильная версия сайта: преимущества и недостатки / *А.С. Фадеева* // Актуальные проблемы авиации и космонавтики. – 2017. – Т. 3. – № 13. – С. 1106–1107.
3. *Дмитриева Ю.С.* Методика адаптивной модернизации веб-интерфейсов / *Ю.С. Дмитриева* // Перспективы развития науки в современном мире. – 2017. – С. 142–148.
4. *Бикитимиров Р.Р.* Современные инструменты front-end разработки / *Р.Р. Бикитимиров, А.Б. Джемалетдинов* // Информационно-компьютерные технологии в экономике, образовании и социальной сфере. – 2016. – С. 63–69.
5. *Bolagna J.* Шпаргалка по Flexbox (CSS3 Flexible Box) / *Joni Bolagna*. – 2006 [Електронний ресурс] – Режим доступу : https://habr.com/ru/post/313938/.
6. Can I Use [Електронний ресурс]. – Режим доступу : https://caniuse.com/.
7. Практическое применение FlexBox. – 2014 [Електронний ресурс]. – Режим доступу : https://habr.com/ru/post/242545/.
8. *Назарова О.* CSS flexbox / *Олена Назарова.* – 2014 [Електронний ресурс]. – Режим доступу : https://html5book.ru/css3-flexbox/#display-flex.
9. *Багрецов С.* Когда нужно использовать Flexbox / *Стас Багрецов.* – 2018 [Електронний ресурс]. – Режим доступу : https://medium.com/@stasonmars/%D0%BA%D0%BE%D0%B3%D0%B4%D0%B0%D0%-BD%D1%83%D0%B6%D0%BD%D0%BE%D0%B8%D1%81%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D1%8C-flexbox-1f02f7a1687c.
10. *Zea R.* Mastering Responsive Web Design / *Ricardo Zea.* – 2015. – 312 p.

References:

1. Filimonenkova, T.N. and Dunaevskiy, A.S. (2017), «Tekhnologii adaptivnoy verstki saytov», *World science: problems and innovations*, pp. 78–81.
2. Fadeeva, A.S. (2017), «Adaptivnyy dizayn ili mobil'naya versiya sayta: preimushchestva i nedostatki», *Aktual'nye problemy aviatsii i kosmonavtiki*, T. 3, No.13, pp. 1106–1107.
3. Dmitrieva, Yu.S. (2017), «Metodika adaptivnoy modernizatsii veb-interfeysov», *Perspektivy razvitiya nauki v sovremennom mire*, pp. 142–148.

4. Bikitimirov, R.R. and Dzhemaletdinov, A.B. (2016), «Sovremennye instrumenty front-end razrabotki», *Informatsionno-komp'yuternye tekhnologii v ekonomike, obrazovanii i sotsial'noy sfere*, pp. 63–69.
5. Bolagna, J. (2006), *Shpargalka po Flexbox (CSS3 Flexible Box)*, [Online], available at: https://habr.com/ru/post/313938/
6. *Can I Use*, [Online], available at: https://caniuse.com/
7. *Prakticheskoe primenenie FlexBox* (2014), [Online], available at: https://habr.com/ru/post/242545/
8. Nazarova, O. (2014), *CSS flexbox*, [Online], available at: https://html5book.ru/css3-flexbox/#display-flex
9. Bagretsov, S. (2018), *Kogda nuzhno ispol'zovat' Flexbox*, [Online], available at: https://medium.com/@stasonmars/%D0%BA%D0%BE%D0%B3%D0%B4%D0%B0%D0%BD%D1%83%D0%-B6%D0%BD%D0%BE%D0%B8%D1%81%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D1%8C-flexbox-1f02f7a1687c
10. Zea, R. (2015), *Mastering Responsive Web Design*, 312 p.

Болотіна Вікторія Василівна – асистент кафедри комп'ютерної інженерії та кібербезпеки Державного університету «Житомирська політехніка».

Наукові інтереси:
– комп'ютерна графіка;
– вебдизайн;
– вебпрограмування.

E-mail: kik_pvv@ztu.edu.ua.

Чижмотря Олена Геннадіївна – старший викладач кафедри інженерії програмного забезпечення Державного університету «Житомирська політехніка».

Наукові інтереси:
– вебтехнології;
– інформаційні технології в освіті.

E-mail: ch-o-g@ztu.edu.ua.

Чижмотря Олексій Володимирович – старший викладач кафедри інженерії програмного забезпечення Державного університету «Житомирська політехніка».

Наукові інтереси:
– математичне моделювання;
– бази даних.

E-mail: 4ov.ztu@gmail.com.

Стаття надійшла до редакції 03.09.2019.