УДК 004.774
DOI: 10.31673/2412-9070.2023.062327

В. В. ЗАЛИВА[1], ст. викладач;
О. В. БОКІЙ[2], наук. співробітник;
І. І. БЕРДНИК[1], аспірантка,
[1] Державний університет інформаційно-комунікаційних технологій, Київ
[2] Військовий інститут телекомунікацій та інформатизації імені Героїв Крут, Київ

ТРАДИЦІЙНЕ ЯДРО DOM (Document Object Model) ТА ЙОГО ФОРМАЛЬНЕ ПОДАННЯ fDOM

Розглянуто відмінність між двома моделями вебдокументів: традиційним DOM (Document Object Model) та fDOM (формальним DOM). Описано Core DOM, який є структурою вебдокументів як ієрархічне дерево вузлів. Кожен вузол відображає елементи, атрибути та текстовий вміст, що дає змогу навігувати, модифікувати та видаляти компоненти документа. На прикладі HTML-коду проілюстровано візуальну структуру DOM. Далі розкрито використання Isabelle/HOL, системи логіки вищого порядку, для роботи з fDOM, що уможливлює формалізацію математичних властивостей fDOM з переконанням у їхній коректності та стійкості до вразливостей. Приклад коду на Isabelle/HOL продемонстрував, як можна визначати та перевіряти операції над елементами вебдокументів. Здійснено порівняння між DOM та fDOM, зазначаючи відмінності в їх підходах до подання та взаємодії з вебдокументами. Наголошено, що традиційний DOM забезпечує гнучкий інтерфейс для вебдокументів, тоді як fDOM пропонує більш строге математичне подання, зосереджене на точності. Сформульовано переваги та недоліки fDOM. Серед переваг виокремлено формальну верифікацію, сильну типізацію, послідовність у поведінці, чіткість визначень та підвищену безпеку. Проте звернено увагу на збільшення складності та часу на розроблення, меншу гнучкість, вищі витрати на ресурси та круту криву навчання, які є недоліками fDOM. Висновок акцентує на тому, що fDOM пропонує надійну, перевірену основу для вебзастосунків, але також вносить додаткові вимоги до процесу розроблення, що робить його особливо корисним для застосунків, де важливі правильність і безпека.

Ключові слова: fDOM; Isabelle/HOL; вебкомпоненти; інформаційні системи; вебтехнології; масштабування вебзастосунків; оптимізація.

Вступ

Основна модель даних DOM та її розширення через Isabelle/HOL у контексті fDOM відкривають нові горизонти в розумінні та взаємодії з вебдокументами. З одного боку, Core DOM визначає структуру і взаємозв'язки елементів у вебдокументі, подаючи їх як ієрархічне дерево, що дає змогу ефективно навігувати, модифікувати та керувати вмістом. З другого боку, Isabelle/HOL вносить новий рівень строгості та логіки в роботу з fDOM, даючи можливість формалізувати математичні властивості та забезпечити більшу передбачуваність та безпеку вебзастосунків. Через детальний аналіз обох підходів відкривається більш глибоке розуміння важливості кожної моделі та їхнього впливу на сучасне веброзроблення.

Аналіз дослідження. Аналіз дослідження зосереджено на значущості DOM як фундаментальної моделі для репрезентації структури вебдокументів. DOM подає елементи сторінки як ієрархічне дерево, що надає зручні засоби для навігації та модифікації контенту. Розглядаючи DOM через призму вебінженерії, модель є інтуїтивно зрозумілою і широко використовується для динамічного керування вебсторінками.

Розвиток fDOM, який уводить у вжиток формальні методи через Isabelle/HOL, забезпечує додаткову користь у вигляді формальної верифікації та математичної коректності. Це забезпечує стійкість до помилок та вразливостей, що є невід'ємною частиною традиційного DOM. Різниця між двома моделями полягає в підвищеній строгості та логічній консистентності fDOM, що зумовлює детерміновані та надійні операції з вебдокументами.

Проте додавання (включення) fDOM у веброзроблення висуває додаткові вимоги до знань розробників і може призвести до збільшення витрат часу та ресурсів. Формальні методи можуть зробити розроблення вебзастосунків складнішим і менш гнучким, особливо коли потрібні швидкі вирішення за умов, що стрімко змінюються. З огляду на ресурсоємність та складність формальних інструментів верифікації розробники можуть зіткнутися з необхідністю знайти баланс між формальною коректністю та оперативністю розвитку проєктів.

Отже, хоча fDOM відкриває шлях для створення вебзастосунків із забезпеченою правильністю та безпекою, він вносить додаткові складнощі в процес розроблення. Вибір між використанням DOM і fDOM залежатиме від конкретних потреб проєкту та рівня спеціалізації команди. З одного боку, DOM залишається швидким і гнучким рішенням для веброзроблення, а з другого — fDOM пропонує більш строгу

та математично обґрунтовану альтернативу, що може бути найбільш доцільною для проєктів із високими вимогами до надійності.

Мета дослідження. Метою дослідження є порівняльний аналіз двох основних моделей керування вебдокументами: традиційної моделі DOM та її формалізованого варіанта fDOM. Це дослідження прагне розкрити як практичність і гнучкість DOM, так і надійність fDOM, використовуючи формальні методи, зокрема Isabelle/HOL, для забезпечення математичної перевірки і верифікації вебоперацій. Основну увагу приділено можливості fDOM знижувати кількість помилок та вразливостей, що можуть виникати під час роботи з DOM, та визначенню сфер їх найкращого застосування в контексті сучасного веброзроблення.

Основна частина

Модель даних Core DOM окреслює ієрархічну структуру та взаємозв'язки елементів вебдокумента. По суті, вона подає структуровані документи як дерева, де кожен вузол відповідає частинам документа, зокрема елементам, атрибутам і текстовому вмісту. Таке деревоподібне подання забезпечує надійні засоби для навігації, модифікації і навіть видалення компонентів вебдокумента.

Щоб простіше зрозуміти структуру DOM, візуалізуємо невелику частину коду HTML:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>DOM візуалізатор</title>
  </head>
  <body>
    <h1 id="title">DOM візуалізатор</h1>
    <div class="contact">
      <!-- Це коментар -->
      <h2>Віталій Залива</h2>
      <p id="twitter">Twitter : @usernickname</p>
    </div>
  </body>
</html>
```

Із наведеного коду можемо дістати візуальну структуру DOM, зображену на рисунку.

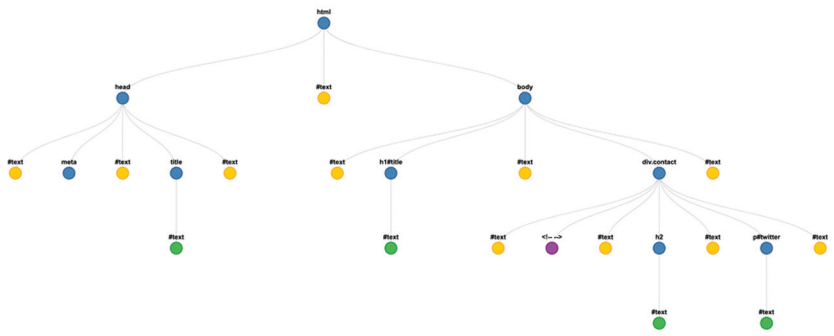

Рис. 1. Візуальна структура DOM

Для fDOM Isabelle/HOL слугує основою для логічного обґрунтування, гарантуючи, що поведінка моделі є послідовною та передбачуваною. За допомогою Isabelle/HOL ми можемо формалізувати математичні властивості fDOM, переконатися в їх коректності та встановити їх стійкість до певних класів уразливостей. Ця інтеграція поєднує строгість формальних методів із гнучкістю та надійністю DOM.

Зразковий сегмент fDOM можна подати так:

datatype Element = create(string name)

datatype Attr = assign(Element e, string property, string value)

У цьому спрощеному прикладі тип даних Element створює елемент із заданим іменем, а Attr призначає конкретному елементу атрибут із властивістю і значенням.

Розглянемо припущення, що кожному створеному елементу Element можна присвоїти атрибут. Формально це можна записати у вигляді

$\forall e \in Element, \exists a \in Attr\ where\ assign(e, property, value)$.

За допомогою Isabelle/HOL ми можемо перевірити цю пропозицію, визначивши універсальність присвоєння атрибутів елементам у fDOM.

Як уже зазначалося, DOM подає документ у вигляді дерева вузлів. Кожен вузол відповідає частинам сторінки, таким як елементи та атрибути. Тоді як fDOM підкреслює більш строге, математичне і формальне подання структури вебдокумента і його елементів. Поняття «класи як типи даних» уточнює абстрактні визначення, що зумовлює сильнішу систему типів і чіткіші ієрархічні зв'язки.

Наведемо операційну узгодженість двох вебдокументів:

♦ **DOM.** Хоча операції DOM детерміновані, їх результати іноді можуть бути несподіваними через неоднозначні визначення або необроблені граничні випадки, особливо коли з DOM взаємодіють складні операції або скрипти.

♦ **fDOM.** Використовуючи формальні методи та інструменти, зокрема Isabelle/HOL, fDOM гарантує, що операції над моделлю не тільки детерміновані, а й строго дотримуються формально перевіреної поведінки, що гарантує відсутність сюрпризів.

Далі порівняємо DOM і fDOM на практичному прикладі для більшого зрозуміння основних принципів роботи обох компонентів. Уявімо, що в нас є проста операція DOM: додавання дочірнього вузла до батьківського.

У традиційному DOM (за допомогою JavaScript) матимемо:

```
let parent = document.createElement("div")
let child = document.createElement("span");
parent.appendChild(child);
```

Щоб перекласти цю операцію на мову fDOM, ми подамо елементи та їхні зв'язки у формальному вигляді:

```
Isabelle/HOL:
datatype Element = Div | Span
datatype Operation = AppendChild (parent: Element, child: Element)
definition appendChildOp :: "Element  Element  Operation" where
"appendChildOp p c = AppendChild p c"
```

Тепер формально доведемо, що після операції додавання дочірній елемент дійсно є частиною батьківського елемента. Це можна записати мовою:

```
Isabelle/HOL у вигляді леми:
лема child_is_appended: "appendChildOp Div Span = AppendChild Div Span"
apply(simple add: appendChildOp_def)
done
```

Наведений код Isabelle/HOL є спрощеним поданням і не відображає всіх складнощів. Але він ілюструє ідею: ми використовуємо формальну мову для визначення операцій над вебдокументами, а потім перевіряємо їх з математичною строгістю.

По суті, тоді як традиційний DOM полягає в тому, щоб дозволити операції над вебдокументами, fDOM має строго визначити і довести правильність цих операцій. Використання Isabelle/HOL у поєднанні з fDOM здатне зумовити створення вебзастосунків, які є не лише функціонально обширними, а й математично перевіреними на коректність, забезпечуючи надійність і цілісність вебоперацій.

Результати проведеного дослідження відкривають дискусію щодо практичного застосування та теоретичної значущості моделей DOM та fDOM у веброзробленні. Зазначається, що традиційний DOM залишається переважним інструментом для більшості розробників, завдяки його гнучкості, широкому підтриманню в браузерах та легкості інтеграції з наявними технологічними стеками. Його інтуїтивно зрозуміла структура та прямий вплив на вебсторінки роблять DOM незамінним для швидкого розроблення та динамічної модифікації контенту.

Водночас fDOM, який уводиться за допомогою формальних методів перевірки, показує обіцяючі результати стосовно безпеки та надійності. Використання Isabelle/HOL для формалізації властивостей fDOM демонструє, що такий підхід може значно знизити ризик помилок та вразливостей. Проте це також вносить додаткові витрати у вигляді часу розроблення, складності та потреби в глибоких знаннях щодо формальних методів.

Подальше обговорення виявляє, що використання fDOM може бути надмірним для проєктів, які потребують швидкої ітерації та адаптації. Водночас для застосунків, де важлива максимальна коректність та безпека, наприклад, у фінансових системах, медичних застосунках чи критичному програмному забезпеченні, переваги fDOM можуть перевищувати його обмеження.

Ключовим аспектом обговорення є також ідентифікація сфер, де компроміс між швидкістю розроблення та формальною коректністю може бути досягнутий. Можливе вирішення полягає у гібридному

підході, де основні компоненти системи будуються з використанням fDOM, а менш критичні частини — з використанням гнучкого DOM.

Отже, дослідження fDOM треба розділити на переваги та недоліки, котрі були виявлені в процесі аналізу. Як переваги можна розглядати наведені далі функції.

• *Формальна верифікація.* Здатність fDOM проходити формальну перевірку за допомогою таких інструментів, як Isabelle/HOL. Це гарантує, що операції над fDOM є математично доведено коректними, усуваючи цілі класи помилок і вразливостей.

• *Типізація.* Подаючи елементи та їх зв'язки як типи даних, fDOM забезпечує сильнішу систему типів. Це може зумовити меншу кількість помилок під час виконання та передбачуванішу поведінку, оскільки невідповідності типів можуть бути виявлені на ранній стадії розроблення.

• *Послідовність.* fDOM сприяє детермінованій поведінці, строго дотримуючись формально перевіреної поведінки, гарантуючи, що операції над моделлю дають очікувані результати.

• *Чіткість у визначеннях.* Завдяки своїй строгій природі, fDOM може надавати більш чіткі та менш неоднозначні визначення структури вебдокумента та його елементів.

• *Підвищена безпека.* Формальна перевірка також може зумовити створення більш безпечних застосунків. Уразливості, що виникають через непослідовну поведінку або невизначені стани в DOM, можна мінімізувати або усунути.

Далі виокремимо деякі недоліки, виявлені під час роботи з fDOM.

• *Складність.* Упровадження формальних методів і перевірок у веброзроблення може ускладнити процес. Щоб ефективно працювати з fDOM, розробники мають бути обізнані як щодо веброзроблення, так і стосовно формальних методів.

• *Витрати на розроблення.* Формальна перевірка здебільшого передбачає написання додаткового перевірного коду або специфікацій. Це може збільшити час, потрібний для розроблення вебзастосунку, особливо якщо команда не має досвіду роботи з такими методами.

• *Менша гнучкість.* Використання fDOM може призвести до меншої гнучкості в деяких сценаріях, особливо коли шукаються швидкі і брудні рішення для стрімко змінюваних вимог.

• *Споживання ресурсів.* Формальні інструменти верифікації можуть бути ресурсоємними, потребувати дужчої обчислювальної потужності і, можливо, збільшувати витрати на розроблення.

• *Крива навчання.* Традиційні веброзробники можуть зіткнутися з крутою кривою навчання, щоб зрозуміти і застосувати концепції формальних методів у контексті веброзроблення.

Висновки

Класичний DOM продовжує бути незамінним інструментом завдяки своєму широкому підтриманню та простоті використання, що дає змогу розробникам швидко реагувати на зміни у вимогах до проєктів. Водночас інтеграція формальних методів через fDOM відкриває нові перспективи для підвищення безпеки та надійності вебзастосунків, хоча й потребує додаткових ресурсів і спеціалізованих знань.

У дослідженні обгрунтовано можливість упровадження гібридних підходів, які поєднують гнучкість DOM із формальною перевіркою fDOM, для балансування між швидкістю розвитку та стабільністю систем. Важливою є рекомендація розробникам розглядати обидві моделі не як взаємовиключні, а як доповнювальні стратегії, кожна з яких може бути вибрана залежно від специфічних потреб проєкту. Вибрана модель має відображати пріоритети проєкту: швидкість та адаптивність для DOM проти безпеки та надійності для fDOM. Такий вибір вимагатиме від розробників знань та навичок в обох сферах, що стимулюватиме подальший розвиток індустрії веброзроблення.

Список використаної літератури

1. ***Haverbeke M.*** *Eloquent JavaScript: A Modern Introduction to Programming. 3rd Edition. No Starch Press, 2018.*

2. ***Pilgrim M., Russell A.*** *Dive Into HTML5 and the DOM. O'Reilly Media, 2019.*

3. ***Lowe D.*** *Web Development All-in-One For Dummies. John Wiley & Sons, 2019.*

4. ***Keith J.*** *DOM Enlightenment: Exploring JavaScript and the Modern DOM. O'Reilly Media, 2019.*

5. ***Gasston P.*** *The Modern Web: Multi-Device Web Development with HTML5, CSS3, and JavaScript. No Starch Press, 2020.*

6. ***Freeman E., Robson E.*** *Head First HTML and CSS: A Learner's Guide to Creating Standards-Based Web Pages. 3rd Edition. O'Reilly Media, 2020.*

7. ***Clark K.*** *HTML5 and CSS3 All-in-One For Dummies. 4th Edition. John Wiley & Sons, 2021.*

8. ***Palermo J.*** *Progressive Web Apps with Angular: Create Responsive, Fast and Scalable Web Applications. Apress, 2019.*

V. V. Zalyva, O. V. Bokii, I. I. Berdnyk

THE TRADITIONAL CORE OF DOM (DOCUMENT OBJECT MODEL) AND ITS FORMAL REPRESENTATION IN fDOM

This article discusses the differences between two models of web documents: the traditional DOM (Document Object Model) and fDOM (formal DOM). The discussion begins with a description of Core DOM, which represents the structure of web documents as a hierarchical tree of nodes. Each node represents elements, attributes, and textual content, allowing for navigation, modification, and deletion of document components. The visual structure of DOM is illustrated using HTML code. The use of Isabelle/HOL, a higher-order logic system, for working with fDOM is then considered. This allows for the formalization of the mathematical properties of fDOM, ensuring their correctness and resistance to vulnerabilities. An example of code in Isabelle/HOL demonstrates how operations on web document elements can be defined and verified. The article also compares DOM and fDOM, highlighting differences in their approaches to representation and interaction with web documents. It is emphasized that while the traditional DOM provides a flexible interface for web documents, fDOM offers a more rigorous, mathematical representation, focused on precision and strictness. The discussion concludes with the advantages and disadvantages of fDOM. Among the advantages are formal verification, strong typing, consistent behavior, clear definitions, and enhanced security. However, increased complexity and development time, less flexibility, higher resource costs, and a steep learning curve are noted as disadvantages of fDOM. The conclusion emphasizes that fDOM offers a reliable, verified foundation for web applications but also introduces additional development requirements, making it particularly useful for applications where correctness and security are important.

Keywords: fDOM; Isabelle/HOL; web components; information systems; web technologies; scaling of web applications; optimization.