УДК 62-526

Є. О. Звуздецький
А.С. Васюра

Дослідження роботи «Garbage collector» у мові програмування JavaScript

Вінницький національний технічний університет

Анотація
В роботі розглянуто фукнцію «Garbage collector» в мові програмування JavaScript, досліджено один з головних принципів його роботи та визначено недоліки.upp
Ключові слова: *JavaScript, Garbage collector, Front-end, TypeScript, React, Angular, Node*

Abstract
The article proposes description how work «Garbage collector» in a programming language JavaScript. One of the main principles of his work and his disadvantages were described.
Keywords: *JavaScript, Garbage collector, Front-end, TypeScript, React, Angular, Node*

Вступ

Головна мета «Garbage collector» - запропонувати операційній системі, в разі необхідності, розподілену динамічну пам'ять, та після завершення використання, звільнити її від об'єктів, які вже не використовуються. Відомо, що такі мови, як C, C ++, мають достатньо примітивний функціонал для розподілу пам'яті, як malloc(), де певні комп'ютерні архітектури високого рівня (наприклад, JavaScript) містять «garbage collector», який виконує таку роботу. Він відстежує розподіл пам’яті та ідентифікує, якщо виділена пам’ять більше не використовується, а потім звільняється автоматично. Але такі алгоритми не можуть повністю вирішити, потрібна пам'ять чи ні. Тому для програміста вкрай важливо зрозуміти та визначити, а чи потрібен певний фрагмент коду пам'яті, чи ні.

Результати дослідження

Колекціонер сміття JavaScript Engine шукає недоступні об'єкти, вилучені з пам'яті. Існує два алгоритми збору сміття:

- Довідковий збір сміття
- Алгоритм розмітки та зачистки

В роботі досліджується алгоритм «Довідковий збір сміття»

Довідковий збір сміття - це наївний алгоритм збору сміття. Він розглядає тільки ті об’єкти, на яких не залишилося посилань. Об'єкт стає фактором для видалення, якщо на нього немає посилань.

```
var obj1 = {

    property1: {

         subproperty1: 20

     }
};
```

Створимо об’єкт, як показано у наведеному вище прикладі, задля того, щоб зрозуміти цей алгоритм. Тут obj1 має об'єкт, у якому його property1 містить ще один об'єкт. Оскільки obj1 має посилання на об'єкт, то ніщо не підходить для видалення.

```
var obj2 = obj1;

obj1 = "some random text"
```

Тепер, obj2 також має посилання на той самий об'єкт, на який посилався obj1, але згодом obj1 було оновлено "деяким випадковим текстом", який призводить до того, що obj2 має унікальне посилання на цей об'єкт.

```
var obj_property1 = obj2.property1;
```

Тепер obj_property1 відноситься до obj2.property1, який також містить об'єкт. Відповідно, на цей об'єкт вже є два посилання, а саме:

1. Як властивість obj2
2. У змінній obj_property1

```
obj2 = "some random text"
```

Obj2 було ”відв’язано”, оновивши "якийсь випадковий текст". Тому може здатися, що об'єкт, який він тримав раніше, не має до нього посилання, і його можна видалити. Але це може бути не вірним, оскільки obj_property1 має посилання на obj2.property1. Тому він і не видаляється.

```
obj_property1 = null;
```

Тепер на об'єкт, який був спочатку в obj1, не залишилося посилань, оскільки було видалено посилання з obj_property1. Тож, тепер його можна видалити.

Де цей алгоритм виходить з ладу?

```
function example() {

     var obj1 = {

         property1 : {

              subproperty1: 20

         }

     };

     var obj2 = obj1.property1;

     obj2.property1 = obj1;

     return 'some random text'

}

example();
```

Таким чином, алгоритм підрахунку посилань не видаляє obj1 та obj2 з пам'яті після виклику функції, оскільки обидва об'єкта посилаються один на одного.

Висновок

У високорівневій мові програмування JavaScript є система, що контролює обсяг виділеного місця для програми і видаляє непотрібні об’єкти, тим самим чистить пам’ять від зайвих змінних. В роботі було розглянуто один із методів роботи “garbage collector” та знайдені його недоліки, на які важливо звертати увагу розробнику, що використовує мову програмування JavaScript.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. https://learn.javascript.ru/memory-managment
2. https://javascript.info/garbage-collection
3. https://developer.mozilla.org/en-US/docs/Web/JavaScript/Memory_Management

Звуздецький Єгор Олегович – студент групи 1ІСТ-17б, факультету комп’ютерних систем та автоматики, Вінницький національний технічний університет, м. Вінниця, egorzvuzdetskyi@gmail.com,

Науковий керівник: ***Васюра Анатолій Степанович*** – професор кафедри атоматизації та інтелектуальних інформаційних систем, Вінницький національний технічний університет, м. Вінниця,

Zvuzdetskyi Egor Olegovych? – student of 1IST-17b, Faculty of Computer Systems and Automation, Vinnytsia National Technical University, Vinnytsia, egorzvuzdetskyi@gmail.com,

Supervisor: ***Vasyura Anatoly Stepanovich*** –professor at the Department of Atomization and Intelligent Information Systems, Vinnytsia National Technical University, Vinnytsia, vasyura.a.s@vntu.edu