Министерство науки и высшего образования Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

“НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО”

Факультет Программной инженерии и компьютерной техники

Образовательная программа Системное и прикладное программное обеспечение

Направление подготовки (специальность) 09.04.04 Программная инженерия

ОТЧЕТ

о научно-исследовательской работе

Тема задания: «Выработка методов к анализу мультиязыковых текстов программ»

Обучающийся: Орловский М.Ю (**Фамилия И.О.**) Р4116 (**номер группы**)

Руководитель практики от университета: Маркина Т.А, доцент факультета ПИиКТ

Санкт-Петербург
2022 г.

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ

В настоящем отчете применяют следующие сокращения и обозначения:

API (англ. Application Programming Interface) – интерфейс прикладного программирования

HTML (англ. Hypertext Markup Language) – язык разметки веб-документов

CSS (англ. Cascade Style Sheets) – язык для представления стилей веб-документов

ПК – программный код

ПО – программное обеспечение

ЯП – язык программирования

ВВЕДЕНИЕ

В современной индустрии разработки программного обеспечения нередким является применение нескольких языков программирования в одном программном проекте. Такой подход порождает проблему учета согласованности компонентов, реализованных на разных языках и применяемых в составе одной программной системы. Как правило, такую несогласованность нельзя выявить до процесса отладки или тестирования.

Однако, исходя из особенностей применяемых языков программирования нередко возникает возможность применения методов статического анализа связности языковых конструкций между различными программными модулями.

Данная работа рассматривает различные методы применимые к анализу мультиязыковых программных проектов, их особенности и реализацию одного из методов в виде прототипа анализатора.

1 Эпоха

1.1 Написать техническое задание на проект

В ходе выполнения этапа было написано техническое задание для проекта по теме «Выработка методов к анализу мультиязыковых текстов программ». Были сформированы эпохи на семестр, поставлены задачи. Были определены первичные требования к проекту, а также пользовательские сценарии.

1.2 Согласовать техническое задание на проект

Задание было согласовано и утверждено.

1.3 Выбор дисциплин для изучения

В качестве дисциплин были выбраны два курса: «Язык программирования C# 10» и «Введение в искусственный интеллект»

2 Эпоха

2.1 Определить сценарии использования мультиязыковых анализаторов

Современные программные проекты зачастую содержат комбинацию ресурсов (программного кода, конфигурационных текстов и других) основанную на нескольких нотациях и/или языках программирования. Примером может служить любое веб-приложение построенное на ASP фреймворке – в таком приложении код избранного ЯП (C#, Java, Ruby или иной) используется вкупе с конфигурационными файлами (обычно в нотации XML), файлами системы сборки и веб-ресурсами (содержащими HTML, CSS, JavaScript и иные виды ресурсов).

Однако, мультиязыковые программные проекты не ограничиваются лишь веб-приложениями или фреймворками для разработки приложений в целом – современные языки программирования нередко в силу исторических особенностей или инженерных решений представляют собой сочетание нескольких самостоятельных (хоть и объединенных воедино) языков. Примером может служить, например язык C++ который содержит реализацию как минимум двух различных нотаций – язык C++ и язык препроцессора. Так как препроцессор входит в компилятор, такое разделение обычно не производят, но оно может быть полезно с точки зрения анализа программного кода.

Обычно для получения информации о синтаксической структуре достаточно анализа одиночных единиц программного кода (файлов или фрагментов), поэтому необходимость согласованного мультиязыкового анализа в отношении синтаксиса отпадает. Чаще всего программист заинтересован в выявлении логических связей между программными компонентами, что в свою очередь может быть достигнуто только посредством семантического анализа. Таким образом, в контексте анализа мультиязыковых программ имеет смысл только семантический анализ.

Существуют различные области применения мультиязыкового анализа и различные подходы к его проведению. Следует подробнее обсудить возможные сценарии использования семантической информации извлеченной посредством мультиязыкового анализа.

Сценарии использования покрывают многие этапы процесса разработки ПО. В такие сценарии входит:

1. Разработка компиляторов, в которых всегда используется по крайней мере одно промежуточное представление для извлечения информации о программном коде и чаще всего для реализации оптимизационных техник;
2. Разработка различных виртуальных машин и интерпретаторов, в которых неизбежно хранение и использование семантической информации о программном коде в реальном времени;
3. Построение обособленных средств анализа позволяющих выполнять различные предметно-ориентированные задачи – сбор статистики, верификация и другие;
4. Создание компонентов (плагинов, модулей) для IDE позволяющих упростить процесс разработки ПО.

2.2 **Определить множество анализаторов исходных текстов программ для исследования и составить их краткие характеристики по критериям: функциональные возможности, место (этап) использования в жизненном цикле проекта**

Исходя из большого количества различных сценариев использования анализаторов, следует выбрать рассмотреть возможное множество анализаторов. В рамках данной работы производится нестрогое разделение исходя из особенностей методов анализа.

По количеству обрабатываемой информации анализаторы можно разделить на:

1. Анализаторы отдельных фрагментов ПК или файлов;
2. Анализаторы групп файлов или групп фрагментов ПК;
3. Анализаторы программных проектов, являющихся обособленными единицами программного комплекса.

В категорию 1 входят различные синтаксические анализаторы, используемые для форматирования или подсветки синтаксиса, поэтому их рассмотрение не будет производится в данной работе.

В категорию 2 и 3 входят инструменты, учитывающие семантику исходных языков и, следовательно, подходящие для рассмотрения.

По быстродействию анализаторы можно нестрого разделить на следующие категории:

4. Быстрые или моментальные (задержка в 0–3 секунды);
5. Медленные, но не прерывающие рабочий процесс (от 3 секунд до минуты);
6. Очень медленные, прерывающие рабочий процесс (от минуты до суток).

В категорию 1 входят анализаторы языка (редко языков) встроенные в IDE для обеспечения различных действий с программным кодом (переименование символа, рефакторинг методов, построение структуры классов и многие другие).

Категории 2 и 3 обычно включают в себя анализаторы программных проектов целиком, направленные на проверку определенных гарантий программного кода (линтеры, санитайзеры и другие). Обычно являются самодостаточными инструментами и запускаются системой сборки и/или системой непрерывной интеграции.

В данной работе рассматриваются анализаторы, ориентированные на высокое быстродействие и низкий отклик, при этом использующие мультиязыковую информацию из разных фрагментов и файлов. Как станет ясно далее, разделение программного кода по файлам весьма условно в рамках построения семантических моделей программ при мультиязыковом анализе.

2.3 **Исследовать возможности существующих средств анализа в части расширения наборов поддерживаемых сочетаний языков программирования**

Задача мультиязыкового анализа является довольно серьезным и сложным предприятием, затрагивающим большое количество областей программной инженерии. Текущие средства анализа являются недостаточными для программных проектов большого объема [1].

Основной проблемой мультиязыкового анализа является обработка текстов мультиязыковых программ – построение парсеров для каждого задействованного в приложении языка [2], что не всегда является возможным ввиду существования

фреймворков со смешанным или предметно-ориентированным синтаксисом (таких как MyBatis или Blazor).

Также многие мультиязыковые решения полагаются на динамическую генерацию связанных элементов (например генерация узлов DOM в HTML при исполнении кода JavaScript). Следует также отметить, что такой сценарий может быть экстраполирован на многие предметно-ориентированные языки, которые имеют прямое сходство с языками имеющими сильную систему макроподстановок и метагенерации (C++, Common Lisp, Nemerle и другие).

Анализаторы мультиязыковых программ включены в различные компоненты IDE и конвейеров сборки и интеграции, однако также существуют и самостоятельные решения.

В числе комбинированных решений с неявным использованием анализаторов можно привести в пример различные плагины для IDE. Например, некоторые IDE от JetBrains имеют поддержку мультиязыкового рефакторинга, как например плагин для AppCode [3]

Visual Studio от Microsoft имеет встроенную поддержку для рефакторинга и навигации по фрагментам кода на Razor/Blazor [4], которые сами по себе являются смесью нотации HTML, кода C# и специфичного кода разметки.

Решения от компаний обычно не предоставляют расширяемых и/или гибких инструментов для мультиязыкового анализа и по своей природе являются специфичными законченными продуктами.

Наряду с решениями от крупных компаний, есть также и самостоятельные решения направленные на универсальные сценарии использования, рассмотрим один из таких инструментов.

MLSA [5] это инструмент для осуществления мультиязыкового анализа, позволяющий:

1. Осуществлять построение графа вызовов;
2. Получать информацию о присваиваниях и использовании символов;
3. Получать зависимость символов в мультиязыковой среде.

Инструмент проводит большей частью статический анализ, однако динамические возможности тоже присутствуют в ограниченном количестве. Поддерживает ограниченный ряд языков (C++, JavaScript, Python), при этом не все возможности инструмента покрывают все языки. Для анализа используется подход анализа островных грамматик, позволяющий, по словам авторов, упрощенное расширение поддерживаемых языков. MLSA позволяет получать данные анализа в различных форматах, в основном это CSV.

Как видно из описания, MLSA хоть и является достаточно универсальным инструментом, он предназначен для решения относительно узкого круга задач анализа и может быть несовместим с некоторыми грамматиками избранных языков.

Таким образом, существующие средства достаточно просты в отношении мультиязыкового анализа и чаще всего не отвечают требованиям современной индустрии.

2.4 Определить показатели эффективности как для самих средств, так и для процесса расширения их функциональности

Одной из основных проблем при реализации текстовых редакторов для языков программирования является задача интеграции анализаторов, результаты работы которых редактор должен отображать. Интеграция заключается в передаче информации об исходных текстах между внутренними представлениями данных текстового редактора и анализатора.

Следовательно, типовым решением данной проблемы является конвертация внутренних представлений между собой для обеспечения отображения и анализа. Существующие решения (такие как наиболее популярные IDE и плагины для текстовых редакторов, использующие LSP) справляются достаточно хорошо с задачей анализа программ, написанных на одном языке. Однако, касаемо анализа мультиязыковых программных проектов многие решения проигрывают в скорости отклика и/или скорости анализа.

Одним из примеров можно привести обработку ошибок в коде Razor в IDE Visual Studio. Задержка при очередном цикле анализа порой может достигать от трех до пяти секунд, что сильно продуктивность программиста.

3 Эпоха

3.1 Выбрать сочетание языков программирования и сценарии использования анализатора для пробной программной реализации

Исходя из вышесказанных ограничений и сложностей связанных с мультиязыковым анализом было принято решение разработать минималистичное средство, способное анализировать различные ЯП в реальном времени с возможностью его интеграции в другие программные решения.

Для разработки прототипа было рассмотрено несколько множеств языков для анализа:

1. C и Python;
2. Blazor (являющийся смесью HTML и C#);
3. XML и SQL;
4. Markdown и Go;
5. HTML и JavaScript.

Первый вариант был отвергнут по причине сложностей реализации анализаторов AST таких семантически сложных языков как C и Python. Вариант два имеет достаточно специфический синтаксис (по причине того, что является DSL ASP.Net framework), поэтому написание парсера без спецификации грамматики являлось бы достаточно трудоемким. Варианты три и четыре были отринуты в основном по причине специфичности грамматик взаимодействующих языков (в первую очередь SQL и Markdown), а также из-за редкости такого языкового сочетания.

Таким образом, была выбрана комбинация HTML и JavaScript. Она отвечает следующим требованиям:

1. Такое сочетание является одним из наиболее популярных в контексте мультиязыковых программных проектов;
2. Парсер и анализатор HTML является достаточно простым в реализации;

3. Парсер и анализатор JavaScript тоже является относительно простым в реализации, грамматика языка довольно хорошо проработана;
4. Семантические связи между HTML и JavaScript легко описываются без необходимости разбираться в предметно-ориентированных особенностях этих языков.

3.2 Определить критерии для выполнения анализа мультиязыковых программ

Следует немного подробнее остановиться на семантических ролях JavaScript и HTML.

HTML служит структурным представлением веб-страницы и, следовательно, содержит в себе всё её содержимое в виде структурированных элементов – узлов дерева. Такое дерево обычно называется DOM.

JavaScript является скриптовым языком, в первую очередь используемым для взаимодействия с веб-страницей – её преобразование или обеспечение взаимодействия с пользователем после загрузки веб-страницы (реакция на нажатия кнопок, ввод текста и иное)

Исходя из семантики взаимодействия JavaScript и HTML было решено избрать следующие аспекты межязыкового взаимодействия:

1. Анализ взаимодействия JavaScript с HTML посредством запросов к DOM;
2. Анализ взаимодействия HTML с JavaScript посредством использования интерактивных элементов.

3.3 Сформулировать требования к программной реализации прототипа анализатора

Основной особенностью прототипа должно быть использование унифицированного способа представления информации о языковых элементах/символах.

Такой подход подразумевает единое онтологическое представление информации, что позволяет избежать зависимости от используемых техник парсинга и предоставляет возможности для унифицированного анализа.

В целях реализации прототипа, решено что такая онтология будет определяться негласным соглашением общего именования символов в разных парсерах. В дальнейшем, конечно, необходимо будет рассмотреть возможность принципиального синтеза такой онтологии для некоторых предметных областей.

Основное разделение символов было произведено по признаку задействованности одного символа другим. Таким образом были синтезированы две категории: категория Want и категория Give.

Категория Want присваивается символу если он запрашивает определенную информацию о другом символе. Категория Give присваивается символу если он предоставляет информацию о себе для использования.

Символ может находиться одновременно и в категории Want и в категории Give. Решение о присвоении символу категории и различных свойств определяется анализатором конкретного языка.

В результате обособленного анализа из полученных данных о символах затем составляются пары Want-Give которые таким образом отражают различные семантические связи между символами.

Данные об анализе имеют динамическую структуру, поэтому прототип обеспечивает извлечение данных об анализе в обобщенной форме для возможной последующей интеграции.

Для реализации прототипа был избран язык C# по нескольким причинам:

1. Простота прототипирования из-за обилия удобных языковых средств;
2. Относительно высокая производительность полученного приложения;
3. Большое количество программных библиотек, доступных из единого репозитория;
4. Возможность простой интеграции с другими программными проектами, запускаемыми на CLR.

Для анализа кода JavaScript был выбран инструмент Hime [6] – это относительно простой генератор парсеров, использующий RNGLR парсер для обработки неоднозначных грамматик. Он хорошо интегрируется с

инфраструктурой CLR и имеет относительно много возможностей (включая действия на свертке правил и обработку ошибок).

Также, для обработки HTML DOM была использована библиотека HtmlAgilityPack, которая позволяет обойти дерево узлов за достаточно быстрое время. Также, в ней имеется множество API для извлечения информации об узле.

4 Эпоха

4.1 Выполнить программную реализацию прототипа анализатора

В ходе выполнения этапа была выполнена реализация прототипа анализатора в соответствии с заданными требованиями. Реализация была успешно спроектирована и отлажена, проведено тестирование.

4.2 Выполнить тестирование анализатора и анализ показателей эффективности, определить область применимости полученного решения

Тестирование прототипа проводилось на одном из примеров [7] кода с использованием JavaScript, HTML, CSS. Программный код соответствующих файлов представлен в Приложении А. Приложение представляет собой простую интерактивную веб-страницу с многоэтапным процессом регистрации. При этом, в нем задействованы обе стороны взаимодействия HTML с JavaScript и в противоположную сторону.

В анализатор JavaScript были заложены правила по извлечению запросов к DOM (getElementBy*), а также извлечение объявлений и использований функций. В анализатор HTML были заложены правила по извлечению информации о тегах и используемом в них коде JavaScript.

Результаты анализа приведены в Приложении Б. Структура результатов анализа имеет следующую форму:

1. Информация о символах, запрашивающих информацию (категория Want);
2. Информация о символах, предоставляющих информацию (категория Give);
3. Информация о сопоставленных парах Want-Give.

При анализе результатов следует обратить внимание на следующие аспекты:

1. Текущая версия анализатора не учитывала кратность отношения Want-Give и формально оно является многие-ко-многим, хотя семантически это не всегда корректно;
2. Несмотря на упор на мультиязыковой анализ, были также найдены зависимости между символами в файле скрипта JavaScript;
3. Предоставляемая информация достаточно полна чтобы обеспечить графическое обозначение символов в редакторе;
4. Скорость анализа в первую очередь зависит от быстродействия парсеров и анализаторов исходных языков, составление пар Want-Give является относительно быстрым процессом.

Таким образом, можно заключить что избранный метод анализа и прототип анализатора пригодны для использования в реальном времени на ограниченном наборе данных, при этом правила анализа обладают достаточной гибкостью для установления связей различной семантической природы между элементами HTML и символами JavaScript. В дальнейшем возможно расширение анализатора для поддержки иных языков (например CSS).

ЗАКЛЮЧЕНИЕ

В ходе выполнения научно-исследовательской работы были проведены анализ и оценка методов к осуществлению мультиязыкового анализа, выявлены их достоинства и недостатки, а также проведен анализ существующих анализаторов мультиязыковых проектов.

Были выявлены ключевые характеристики мультиязыкового анализа и выстроена универсальная семантическая структура для представления данных анализатора. Выполнена разработка прототипа мультиязыкового анализатора, ориентированного на использование в реальном времени с учетом последующей интеграции в IDE. Проведено его тестирование и оценка результатов анализа.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. T. van der Storm and J. J. Vinju, ‘‘Towards multilingual programming environments,’ Sci. Comput. Program. - стр. 143–149 - 2015.

2. B. Hugo, J. Cabot, F. Jouault, and F. Madiot, ‘‘MoDisco: A generic and extensible framework for model driven reverse engineering,’’ in Proc. IEEE/ACM Int. Conf. Autom. Softw. Eng. – 2010 - стр. 173–174

3. Плагин для мультиязыкового рефакторинга в IDE AppCode – URL: https://blog.jetbrains.com/kotlin/2021/10/kmm-change-signature/

4. Razor pages – URL: https://learn.microsoft.com/en-us/aspnet/core/razor-pages/

5. Главный репозиторий Multilingual Static Analysis tool – URL: https://github.com/MultilingualStaticAnalysis/MLSA

6. Официальная страница инструмента Hime – URL: https://cenotelie.fr/projects/hime/

7. Пример веб-приложения многоэтапной регистрации – URL: https://www.w3schools.com/howto/howto_js_form_steps.asp

ПРИЛОЖЕНИЕ А

Структура тестового приложения

1. Содержимое файла index.html

```
<form id="regForm" action="">

<h1>Register:</h1>

<!-- One "tab" for each step in the form: -->
<div class="tab">Name:
  <p><input placeholder="First name..." oninput="this.className = ''"></p>
  <p><input placeholder="Last name..." oninput="this.className = ''"></p>
</div>

<div class="tab">Contact Info:
  <p><input placeholder="E-mail..." oninput="this.className = ''"></p>
  <p><input placeholder="Phone..." oninput="this.className = ''"></p>
</div>

<div class="tab">Birthday:
  <p><input placeholder="dd" oninput="this.className = ''"></p>
  <p><input placeholder="mm" oninput="this.className = ''"></p>
  <p><input placeholder="yyyy" oninput="this.className = ''"></p>
</div>

<div class="tab">Login Info:
  <p><input placeholder="Username..." oninput="this.className = ''"></p>
  <p><input placeholder="Password..." oninput="this.className = ''"></p>
</div>

<div style="overflow:auto;">
  <div style="float:right;">
    <button type="button" id="prevBtn" onclick="nextPrev(-1)">Previous</button>
    <button type="button" id="nextBtn" onclick="nextPrev(1)">Next</button>
  </div>
</div>

<!-- Circles which indicates the steps of the form: -->
<div style="text-align:center;margin-top:40px;">
  <span class="step"></span>
  <span class="step"></span>
  <span class="step"></span>
  <span class="step"></span>
</div>

</form>
```

2. Содержимое файла script.js

```
var currentTab = 0; // Current tab is set to be the first tab (0)
showTab(currentTab); // Display the current tab

function showTab(n) {
  // This function will display the specified tab of the form ...
  var x = document.getElementsByClassName("tab");
  x[n].style.display = "block";
  // ... and fix the Previous/Next buttons:
  if (n == 0) {
    document.getElementById("prevBtn").style.display = "none";
  } else {
```

```
    document.getElementById("prevBtn").style.display = "inline";
  }
  if (n == (x.length - 1)) {
    document.getElementById("nextBtn").innerHTML = "Submit";
  } else {
    document.getElementById("nextBtn").innerHTML = "Next";
  }
  // ... and run a function that displays the correct step indicator:
  fixStepIndicator(n)
}

function nextPrev(n) {
  // This function will figure out which tab to display
  var x = document.getElementsByClassName("tab");
  // Exit the function if any field in the current tab is invalid:
  if (n == 1 && !validateForm()) return false;
  // Hide the current tab:
  x[currentTab].style.display = "none";
  // Increase or decrease the current tab by 1:
  currentTab = currentTab + n;
  // if you have reached the end of the form... :
  if (currentTab >= x.length) {
    //...the form gets submitted:
    document.getElementById("regForm").submit();
    return false;
  }
  // Otherwise, display the correct tab:
  showTab(currentTab);
}

function validateForm() {
  // This function deals with validation of the form fields
  var x, y, i, valid = true;
  x = document.getElementsByClassName("tab");
  y = x[currentTab].getElementsByTagName("input");
  // A loop that checks every input field in the current tab:
  for (i = 0; i < y.length; i++) {
    // If a field is empty...
    if (y[i].value == "") {
      // add an "invalid" class to the field:
      y[i].className += " invalid";
      // and set the current valid status to false:
      valid = false;
    }
  }
  // If the valid status is true, mark the step as finished and valid:
  if (valid) {
    document.getElementsByClassName("step")[currentTab].className += " finish";
  }
  return valid; // return the valid status
}

function fixStepIndicator(n) {
  // This function removes the "active" class of all steps...
  var i, x = document.getElementsByClassName("step");
  for (i = 0; i < x.length; i++) {
    x[i].className = x[i].className.replace(" active", "");
  }
  //... and adds the "active" class to the current step:
  x[n].className += " active";
}
```

ПРИЛОЖЕНИЕ Б

Отчет о результатах анализа

```
Analysis results:
Want js-function at tests/form.html.css.js/script.js:2:1
       with Name = showTab
Give js-function at tests/form.html.css.js/script.js:4:1
       with Name = showTab
Want html-element at tests/form.html.css.js/script.js:6:20
       with class = tab
Want html-element at tests/form.html.css.js/script.js:10:14
       with id = prevBtn
Want html-element at tests/form.html.css.js/script.js:12:14
       with id = prevBtn
Want html-element at tests/form.html.css.js/script.js:15:14
       with id = nextBtn
Want html-element at tests/form.html.css.js/script.js:17:14
       with id = nextBtn
Want js-function at tests/form.html.css.js/script.js:20:3
       with Name = fixStepIndicator
Give js-function at tests/form.html.css.js/script.js:23:1
       with Name = nextPrev
Want html-element at tests/form.html.css.js/script.js:25:20
       with class = tab
Want js-function at tests/form.html.css.js/script.js:27:18
       with Name = validateForm
Want html-element at tests/form.html.css.js/script.js:35:14
       with id = regForm
Want js-function at tests/form.html.css.js/script.js:35:40
       with Name = submit
Want js-function at tests/form.html.css.js/script.js:39:3
       with Name = showTab
Give js-function at tests/form.html.css.js/script.js:42:1
       with Name = validateForm
Want html-element at tests/form.html.css.js/script.js:45:16
       with class = tab
Want html-element at tests/form.html.css.js/script.js:46:21
       with tag = input
Want html-element at tests/form.html.css.js/script.js:59:14
       with class = step
Give js-function at tests/form.html.css.js/script.js:64:1
       with Name = fixStepIndicator
Want html-element at tests/form.html.css.js/script.js:66:23
       with class = step
Want js-function at tests/form.html.css.js/script.js:68:37
       with Name = replace
Give html-element at tests/form.html.css.js/index.html:10:0
       with tag = body
Give html-element at tests/form.html.css.js/index.html:11:2
       with tag = form
       with id = regForm
       with action =
Give html-element at tests/form.html.css.js/index.html:13:4
       with tag = h1
Give html-element at tests/form.html.css.js/index.html:16:4
       with tag = div
       with class = tab
Give html-element at tests/form.html.css.js/index.html:17:6
       with tag = p
Give html-element at tests/form.html.css.js/index.html:17:9
```

```
       with tag = input
       with placeholder = First name...
       with oninput = this.className = ''
Give html-element at tests/form.html.css.js/index.html:18:6
       with tag = p
Give html-element at tests/form.html.css.js/index.html:18:9
       with tag = input
       with placeholder = Last name...
       with oninput = this.className = ''
Give html-element at tests/form.html.css.js/index.html:21:4
       with tag = div
       with class = tab
Give html-element at tests/form.html.css.js/index.html:22:6
       with tag = p
Give html-element at tests/form.html.css.js/index.html:22:9
       with tag = input
       with placeholder = E-mail...
       with oninput = this.className = ''
Give html-element at tests/form.html.css.js/index.html:23:6
       with tag = p
Give html-element at tests/form.html.css.js/index.html:23:9
       with tag = input
       with placeholder = Phone...
       with oninput = this.className = ''
Give html-element at tests/form.html.css.js/index.html:26:4
       with tag = div
       with class = tab
Give html-element at tests/form.html.css.js/index.html:27:6
       with tag = p
Give html-element at tests/form.html.css.js/index.html:27:9
       with tag = input
       with placeholder = dd
       with oninput = this.className = ''
Give html-element at tests/form.html.css.js/index.html:28:6
       with tag = p
Give html-element at tests/form.html.css.js/index.html:28:9
       with tag = input
       with placeholder = mm
       with oninput = this.className = ''
Give html-element at tests/form.html.css.js/index.html:29:6
       with tag = p
Give html-element at tests/form.html.css.js/index.html:29:9
       with tag = input
       with placeholder = yyyy
       with oninput = this.className = ''
Give html-element at tests/form.html.css.js/index.html:32:4
       with tag = div
       with class = tab
Give html-element at tests/form.html.css.js/index.html:33:6
       with tag = p
Give html-element at tests/form.html.css.js/index.html:33:9
       with tag = input
       with placeholder = Username...
       with oninput = this.className = ''
Give html-element at tests/form.html.css.js/index.html:34:6
       with tag = p
Give html-element at tests/form.html.css.js/index.html:34:9
       with tag = input
       with placeholder = Password...
       with oninput = this.className = ''
Give html-element at tests/form.html.css.js/index.html:37:4
```

```
       with tag = div
       with style = overflow:auto;
Give html-element at tests/form.html.css.js/index.html:38:6
       with tag = div
       with style = float:right;
Give html-element at tests/form.html.css.js/index.html:39:8
       with tag = button
       with type = button
       with id = prevBtn
       with onclick = nextPrev(-1)
Give html-element at tests/form.html.css.js/index.html:40:8
       with tag = button
       with type = button
       with id = nextBtn
       with onclick = nextPrev(1)
Give html-element at tests/form.html.css.js/index.html:45:4
       with tag = div
       with style = text-align:center;margin-top:40px;
Give html-element at tests/form.html.css.js/index.html:46:6
       with tag = span
       with class = step
Give html-element at tests/form.html.css.js/index.html:47:6
       with tag = span
       with class = step
Give html-element at tests/form.html.css.js/index.html:48:6
       with tag = span
       with class = step
Give html-element at tests/form.html.css.js/index.html:49:6
       with tag = span
       with class = step
Want js-function at tests/form.html.css.js/index.html:1:1
       with Name = nextPrev
Want js-function at tests/form.html.css.js/index.html:1:1
       with Name = nextPrev

Resulted pairs:
Want js-function at tests/form.html.css.js/script.js:2:1
       with Name = showTab
Give js-function at tests/form.html.css.js/script.js:4:1
       with Name = showTab

Want html-element at tests/form.html.css.js/script.js:6:20
       with class = tab
Give html-element at tests/form.html.css.js/index.html:16:4
       with tag = div
       with class = tab

Want html-element at tests/form.html.css.js/script.js:6:20
       with class = tab
Give html-element at tests/form.html.css.js/index.html:21:4
       with tag = div
       with class = tab

Want html-element at tests/form.html.css.js/script.js:6:20
       with class = tab
Give html-element at tests/form.html.css.js/index.html:26:4
       with tag = div
       with class = tab

Want html-element at tests/form.html.css.js/script.js:6:20
       with class = tab
```

```
Give html-element at tests/form.html.css.js/index.html:32:4
       with tag = div
       with class = tab

Want html-element at tests/form.html.css.js/script.js:10:14
       with id = prevBtn
Give html-element at tests/form.html.css.js/index.html:39:8
       with tag = button
       with type = button
       with id = prevBtn
       with onclick = nextPrev(-1)

Want html-element at tests/form.html.css.js/script.js:12:14
       with id = prevBtn
Give html-element at tests/form.html.css.js/index.html:39:8
       with tag = button
       with type = button
       with id = prevBtn
       with onclick = nextPrev(-1)

Want html-element at tests/form.html.css.js/script.js:15:14
       with id = nextBtn
Give html-element at tests/form.html.css.js/index.html:40:8
       with tag = button
       with type = button
       with id = nextBtn
       with onclick = nextPrev(1)

Want html-element at tests/form.html.css.js/script.js:17:14
       with id = nextBtn
Give html-element at tests/form.html.css.js/index.html:40:8
       with tag = button
       with type = button
       with id = nextBtn
       with onclick = nextPrev(1)

Want js-function at tests/form.html.css.js/script.js:20:3
       with Name = fixStepIndicator
Give js-function at tests/form.html.css.js/script.js:64:1
       with Name = fixStepIndicator

Want html-element at tests/form.html.css.js/script.js:25:20
       with class = tab
Give html-element at tests/form.html.css.js/index.html:16:4
       with tag = div
       with class = tab

Want html-element at tests/form.html.css.js/script.js:25:20
       with class = tab
Give html-element at tests/form.html.css.js/index.html:21:4
       with tag = div
       with class = tab

Want html-element at tests/form.html.css.js/script.js:25:20
       with class = tab
Give html-element at tests/form.html.css.js/index.html:26:4
       with tag = div
       with class = tab

Want html-element at tests/form.html.css.js/script.js:25:20
       with class = tab
```

```
Give html-element at tests/form.html.css.js/index.html:32:4
       with tag = div
       with class = tab

Want js-function at tests/form.html.css.js/script.js:27:18
       with Name = validateForm
Give js-function at tests/form.html.css.js/script.js:42:1
       with Name = validateForm

Want html-element at tests/form.html.css.js/script.js:35:14
       with id = regForm
Give html-element at tests/form.html.css.js/index.html:11:2
       with tag = form
       with id = regForm
       with action =

Want js-function at tests/form.html.css.js/script.js:39:3
       with Name = showTab
Give js-function at tests/form.html.css.js/script.js:4:1
       with Name = showTab

Want html-element at tests/form.html.css.js/script.js:45:16
       with class = tab
Give html-element at tests/form.html.css.js/index.html:16:4
       with tag = div
       with class = tab

Want html-element at tests/form.html.css.js/script.js:45:16
       with class = tab
Give html-element at tests/form.html.css.js/index.html:21:4
       with tag = div
       with class = tab

Want html-element at tests/form.html.css.js/script.js:45:16
       with class = tab
Give html-element at tests/form.html.css.js/index.html:26:4
       with tag = div
       with class = tab

Want html-element at tests/form.html.css.js/script.js:45:16
       with class = tab
Give html-element at tests/form.html.css.js/index.html:32:4
       with tag = div
       with class = tab

Want html-element at tests/form.html.css.js/script.js:46:21
       with tag = input
Give html-element at tests/form.html.css.js/index.html:17:9
       with tag = input
       with placeholder = First name...
       with oninput = this.className = ''

Want html-element at tests/form.html.css.js/script.js:46:21
       with tag = input
Give html-element at tests/form.html.css.js/index.html:18:9
       with tag = input
       with placeholder = Last name...
       with oninput = this.className = ''

Want html-element at tests/form.html.css.js/script.js:46:21
       with tag = input
```

```
Give html-element at tests/form.html.css.js/index.html:22:9
       with tag = input
       with placeholder = E-mail...
       with oninput = this.className = ''

Want html-element at tests/form.html.css.js/script.js:46:21
       with tag = input
Give html-element at tests/form.html.css.js/index.html:23:9
       with tag = input
       with placeholder = Phone...
       with oninput = this.className = ''

Want html-element at tests/form.html.css.js/script.js:46:21
       with tag = input
Give html-element at tests/form.html.css.js/index.html:27:9
       with tag = input
       with placeholder = dd
       with oninput = this.className = ''

Want html-element at tests/form.html.css.js/script.js:46:21
       with tag = input
Give html-element at tests/form.html.css.js/index.html:28:9
       with tag = input
       with placeholder = mm
       with oninput = this.className = ''

Want html-element at tests/form.html.css.js/script.js:46:21
       with tag = input
Give html-element at tests/form.html.css.js/index.html:29:9
       with tag = input
       with placeholder = yyyy
       with oninput = this.className = ''

Want html-element at tests/form.html.css.js/script.js:46:21
       with tag = input
Give html-element at tests/form.html.css.js/index.html:33:9
       with tag = input
       with placeholder = Username...
       with oninput = this.className = ''

Want html-element at tests/form.html.css.js/script.js:46:21
       with tag = input
Give html-element at tests/form.html.css.js/index.html:34:9
       with tag = input
       with placeholder = Password...
       with oninput = this.className = ''

Want html-element at tests/form.html.css.js/script.js:59:14
       with class = step
Give html-element at tests/form.html.css.js/index.html:46:6
       with tag = span
       with class = step

Want html-element at tests/form.html.css.js/script.js:59:14
       with class = step
Give html-element at tests/form.html.css.js/index.html:47:6
       with tag = span
       with class = step

Want html-element at tests/form.html.css.js/script.js:59:14
       with class = step
```

```
Give html-element at tests/form.html.css.js/index.html:48:6
       with tag = span
       with class = step

Want html-element at tests/form.html.css.js/script.js:59:14
       with class = step
Give html-element at tests/form.html.css.js/index.html:49:6
       with tag = span
       with class = step

Want html-element at tests/form.html.css.js/script.js:66:23
       with class = step
Give html-element at tests/form.html.css.js/index.html:46:6
       with tag = span
       with class = step

Want html-element at tests/form.html.css.js/script.js:66:23
       with class = step
Give html-element at tests/form.html.css.js/index.html:47:6
       with tag = span
       with class = step

Want html-element at tests/form.html.css.js/script.js:66:23
       with class = step
Give html-element at tests/form.html.css.js/index.html:48:6
       with tag = span
       with class = step

Want html-element at tests/form.html.css.js/script.js:66:23
       with class = step
Give html-element at tests/form.html.css.js/index.html:49:6
       with tag = span
       with class = step

Want js-function at tests/form.html.css.js/index.html:1:1
       with Name = nextPrev
Give js-function at tests/form.html.css.js/script.js:23:1
       with Name = nextPrev

Want js-function at tests/form.html.css.js/index.html:1:1
       with Name = nextPrev
Give js-function at tests/form.html.css.js/script.js:23:1
       with Name = nextPrev
```