**“Vasile Alecsandri” University of Bacău**
**Faculty of Sciences**
**Scientific Studies and Research**
**Series Mathematics and Informatics**
**Vol. 35 (2025), No. 1, 41 - 52**

# NEW TOOLS FOR CODE CONVERSION AND OPTIMIZATION

ALEXANDRU PINTEA

**Abstract.** Optimization aims to make code either faster or smaller in size. The applications presented in this paper accomplish mostly storage-related optimizations. Most CSS frameworks provide users with classes, that require a longer syntax to incorporate than CSS attributes. As such, a tool was proposed to enable attribute usage over class usage in CSS. Within the same context of web development, it can be stated that repetitive code inspections hinder development, as they needlessly increase development time. As such, a tool was made to show all the properties of an HTML element when the developer hovers over it. The tool is only required in development, and can be removed seamlessly when the application is released. An HTML generator is also introduced in this paper, which aims to provide a starting point for a web application, using a custom lightweight (attribute-based) CSS framework. This HTML generator takes paragraphed text as input and outputs an HTML page, leaving the possibility for conversion to other formats as well (RTF/PDF/...). Given that AI tries to enable natural language programming, such a tool can be adjacent to its endeavors. A JSON-HTML converter is also introduced. Since JSON is more lightweight than XML, a JSON-HTML converter can help compress static HTML pages, to use less network. The converter enables conversion to and from HTML. Besides the JSON-HTML converter, a separate tool introduces the possibility of shortening function/variable names to optimize large code files. Another aspect of this tool is making code contents less accessible to parties that might handle with it.

**Keywords and phrases:** code optimization tools, CSS framework, JSON-HTML converter.
**(2020) Mathematics Subject Classification:** 68M12.

## 1. Introduction

Usually, code optimization refers to improvements to algorithms or file formats. Such optimizations tend to mostly lead to storage and execution time reductions. Since the essential factor for the effectiveness of the program is the optimization method involved [1], it can be stated that the sole source of efficiency improvement is optimization.

Optimizations tools are usually packaged within libraries. Yet, the usage of such libraries needs should be backed by "good practice" programming, to ensure its benefits are not shadowed by suboptimal code.

CSS libraries are not optimized to offer the user the least HTML file size. Frontend web developers need to inspect the code constantly and the default webpage styling cannot be considered modern [2].

Non-developer users face problems when converting between complex file formats (PDF/HTML/Markdown). Most such conversions are done server-side (inefficient) and inaccurately reflect the expectations on formatting that users have (conversion is "messy", destroys intended formatting).

To address the specified problem, this paper propose new open-source C++/ JS/CSS conversion/optimization tools, including an encoding free paragraph based formatting application [3]. At first, the new `Lng` optimizer can replace large parts of code, therefore reducing filesize. Such an optimization helps developing a more robust/structured application, minimizing the file's time spent on traversing devices/network. Many times, the same text needs to be written to multiple formats. Conversions between HTML/PDF often alter formatting beyond anything that can be considered acceptable [4]. As such, a new tool (JS `LIB`) converter was proposed to provided users with a easy encoding that can produce both HTML/PDF files. As a result, the output has the same content to multiple formats.

There are many CSS libraries that use classes to help styling complex HTML elements [5]. However, using classes is not always necessary since most of the HTML elements do not have any attributes, which are faster to type and easier to see in a text editor. Since many elements use only one class, the code reduction when using the new CSS Atibs tool can reach 50%, for short identifiers (`class= "name"` is twice the size of just `name`, an attribute).

Secondly, the discussion focuses on new tools that enhance code readability and optimize development processes. `oftile, script` and `cnvrt` [3], The `cnvrt` software compresses the XML-like structure of HTML into JSON, which accomplishes significant size reductions. Theoretically, a maximum reduction of half is achievable, as JSON does not require closing tags like HTML does. ”Thus, theoretically, a converted file transmits across the network in less than half the time; however, there are additional delays introduced by the packaging processes involved.” Once the JSON reaches the user, it is converted into HTML. `cnvrt` also helps developers save time when coding, since JSON takes less to type than HTML.

The remainder of the article is organised as follows.
In Section 2 related Literature is discussed, highlighting existing similar research. Section 3 describes the proposed applications, introducing the newly proposed code optimization/ conversion tools for C$^{++}$, JS and CSS. Further, Section 3 discusses new conversion tools that enable code readability. Finally, in Section 4, the conclusion provides a summary of the tools presented, as well as future directions.

## 2. Related literature

In Programming by optimization, Hoos [6] used ”relatively simple” code conversions and robust design optimization methods to create a programming language extension. It proposes several tools for code conversion and optimization, specifically designed to accomplish particular tasks determined by the software created with them.

In 2020, Agner [7] in the Optimizing software in C$^{++}$ book described several optimization approaches. Beside compiler optimization, optimizing memory access, and specific optimization topics are included. For example, conversions between floating point numbers and integers is discussed broadly, along with several optimization-related mathematical functions.

Optimized programming is also required in mobile development, as showcased in [8], The authors use the following optimization criteria: memory optimization, object-oriented optimization and coding style optimization. For mobile applications are also implemented converters, for example an native mobile application code converter based on trans-compilation [9]. Another converter for mobile applications, from swift for iOS to Java for Android [10] shows promising results when compared with 2023 OpenAI’s ChatGPT [11] results.

To migrate source code from one programming language to another, pioneers Flex/Bison stepped in to provide the first ever Scanner/-Parser pair [12, 13], Since then, universal file converters were developed [14], many of which are accessible online, free of charge [15, 16], Yet, for lengthy code conversions, converters (AI or not) need to be assisted by developers to successfully accomplish compilation/interpretation.

Web-related optimization techniques are as numerous as web applications. Studies as [17] reflect the impact such optimizations can have on general-purpose web applications. Automatic CSS refactoring for size reduction [18, 19], as well as CSS selector reduction [20] are just some of the many web optimization techniques. More similar methods, with appropriate guidelines to facilitate the implementation are outlined in [21].

Based on the study of existing research and the previous work [22] with the aim of making applications accessible to both users and developers, the present paper proposes several tools for code conversion and optimization.

## 3. Proposed applications

### 3.1. **Conversions Tools for Code Optimization.**

#### (a) `Lng` **optimizer**

A source code can be vastly reduced by replacing commonly used, repetitive parts by using the new tool called `Lng` optimizer [3], For a given list of $C^{++}$ specific symbols and their replacements [7, 23], the application converts code written in the language that uses the replacements to $C^{++}$, then compiles it.
`Lng` specifications include:

- [leftmargin=0.5cm]
- the code can exist outside any other functions (`main` function is not required);
- it uses separate files to copy and paste text into the code (for text that is not user editable);
- it has its own type of inclusions (it uses both $C^{++}$ libraries and libraries in its own language);
- in order to add its own inclusions, the application needs to copy and paste the library code in its own file, to then convert the whole code into $C^{++}$ and compile it;
- it makes sure not to move comments when converting to $C^{++}$.

Specific error messages are shown when the user fails to properly include a library. The function also identifies the line of code with the mistake. Another error message is shown when the user misses for example an ending '}' for functions. Other standard error messages are left to the C++ compiler [7], The position of characters is used to determine if they are commented or not, to determine if they should be moved: `int is_commented (string file, int pos);`

The code that is outside of any functions will be moved into the main function in order to compile it:

```
if (have_at_pos(file, i, function) and ! is_commented(file, i))
for (int l = i; l < size; l = l + 1)
...
// keep functions above the main function
functions = functions + "\n\n" + get_string(file, i, l); ...
remain = remain + file[i]; // for the main function
```

Validations are included. To convert the file to C++, each character is checked to determine if it is the first character of a replaceable string (`have_at_pos(string file, int position, string replaceable_user_string)`).

### (b) LIB convert

Conversion between multiple data types, by using `string` as a conversion middle-point is made with the new tool, `LIB convert` [3, 24], There are functions for every data type that convert to and from `string`, Conversion for both unitary data structures and arrays is available for `int`, `long`, `double`, `float`, `char`, `bool` and `string` data types.

Validations for every data type (variables and arrays) are available; it determines if a string can be converted to the requested data-type, before attempting to convert.

### (c) CSS Atibs converter

CSS `Atibs` optimize HTML programming by converting classes, attributes and attributes values amongst each other. It does not remove user-given files. A numerically identified menu option with three choices for input, is shown until the user exits the application, or until a valid option is chosen.

The application needs to identify the characters that mark either a class or an attribute, and then skip whitespace and find the end of that definition [3]. The string that defines the name of the attribute/class is extracted and converted to what is needed.

Attributes are also optimized, through reformatting. It is shorter for the user to code `attr-val` instead of `attr = "val"`, and the code highlight color stays the same for the whole statement, improving visibility. The rest of the options convert CSS classes, which are easier to identify and explain (the file-size reduction is evident).

## 3.2. **Conversions for Code Readability. (a) JS oftile optimizer**

Inspecting HTML code is one of the most frequently actions of a frontend web developer. With the new application oftile, the developer will just hover over an element to get all the data needed without the classical inspection of the entire page [3, 20], `oftile` also provides styling for its generated alerts (Figure 1.)

It converts attributes, inner text and tag name to tooltips. After getting every HTML element (either all, or just those that have a specific `tag name`/ `attribute`/ `class`/ `attribute value` or `id`, `oftile` is able to go through every attribute that the element has and convert it into string, to show it `onhover` ( with '`title` = "..."', as a tooltip).

The `oftile` tool keeps certain `tag names`/ `attributes`/ `classes`/ `attributes values` or `ids` unaltered, if the user specifies.

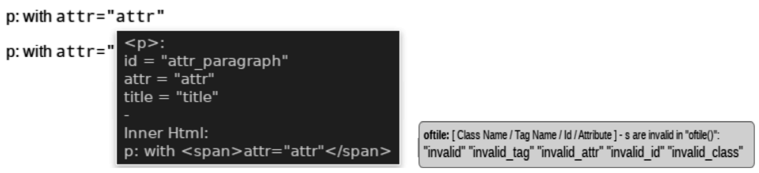
```
function oftile_prevent (tag_name) {
 oftls = get_oftiles(tag_name);
 for (var i = 0; i < oftls.length; i++)
 prevent_oftile.push(oftls[i]);
}
```

The application prevents processing style and script tags by default.

```
(oftile_prevent ("script"); oftile_prevent ("style");)
```

A demonstration oftile usage in a project is presented.

```
<script src="oftile/oftile.js" ></script>
<script>
oftile ( "h2" ); // For all with Tag Name: "h2"
oftile_prevent ( "h5" ); // Preventing oftile for Tag Name: "h5"
     ( could be Id, Attribute / Class Name )
oftile ( "h5" ); // Since the tag was prevented, oftile is not
     going to alter it
oftile ( "paragraph" ); // For all with Class / Id / Attribute /
     Attribute value: "paragraph"
// Invalid Tag Names / Classes / Id-s will be alerted
oftile ( "invalid" );
// For all: oftile ();
</script>
```



```
p: with attr="attr"

p: with attr="
<p>:
id = "attr_paragraph"
attr = "attr"
title = "title"
-
Inner Html:
p: with <span>attr="attr"</span>
```

```
oftile: [ Class Name / Tag Name / Id / Attribute ] - s are invalid in "oftile()":
"invalid" "invalid_tag" "invalid_attr" "invalid_id" "invalid_class"
```

Figure 1. Generated Tooltip (left) and an error message for invalid user-provided arguments (right).

## (b) JS Script converter

JS Script converts title attributes to styled text that glides into the page [3, 20],

It overrides the JS `alert` () function by using CSS.

```
// Alerts b)
document.write( " <div title=\"Click to Dismiss\" id=\"alert\"
onclick=\"alert_hidden()\"> " );
...
// Tips ( styled tooltips ) a)
document.write( "<div id=\"tooltip\"> " );
```

It converts the nav to a scalable, mobile-ready and styled nav that can be toggled.

```
<nav> <!-Source code needed to generate the styled nav (the plain
      HTML nav code gets converted) -->
<a href=""> Nav1 </a>
...
</nav>
```

In this context, only simple JS and CSS were used. Media queries were used (@media screen and (max-width: 1040px)) to make the site scalable. To add styled tooltips to every paragraph element set_tips(”p”) can be used, discarding the default title attribute that would’ve taken precedence onhover,

## (c) JS cnvrt converter

JS `cnvrt` enables seamless JSON-HTML conversion for any general purpose static HTML page. Since each web browser uses the same language to decode the JSON-given webpage [21], the implementation offered by the converter can be used on any frontend host to show the user their webpage, effectively. It converts HTML to JSON and vice versa. It can be applied to any page and used to convert significant information (ending HTML tags are reduced through JSON formatting) [3], The next example shows how the JSON format can store interlacing text, and successfully convert it to HTML. It is done by separating the interlacing text with -.-, an uncommon string. To validate the conversion to JSON, the application converts its previous output to HTML, as follows:

```
[ [tag: "div", attr: {class: "container", attr0: "val0", attr1:"val1
     "}, inner: "Paragraph 1-.-Paragraph 2-.-Paragraph 3", chldn: [ [
     tag: "div", attr: ... ], [tag: "a", attr: ... ], ... ] ], [tag:
     "p", attr: {id: "paragraph", attr2: "val2", attr3:"val3"}, inner
     : "Paragraph 4-.-Paragraph 5-.-Paragraph 6", chldn: [ [tag: "
     span", attr: ... ], [tag: "div", attr: ... ], ... ] ], ...]
```

3.3. **Web development tools.**

### (a) website maker linux

The `website maker` linux application [3] helps users make a website using encoding-free (plain) text. The input file requirement is to be organized in paragraphs. The implementation of `website maker` linux uses several files with static code, bind with the user paragraphs to generate a website [21], Yet, there is some processing regarding the paragraphs the user enters, besides the binding described previously.

```
// contains the  {\footnotesize HTML} code for the page header
 fo << get\_maker\_file( "maker\_files/site\_1" );
```

An in-line output code with the data provided by the user in a text file is shown.

```
fo <<"\n <div id=\"" << get\_title(files[i]) <<
"\"style=\"position: relative; top: -45px;\"></div>";
```

Linux commands are used to create a folder for the website and its style file.

```
command("cp \"" + filepath + upper(filename) + ".html" + "\" \""
     + upper(filename) + "/index.html\"\n" );
command("cp \"maker_files/style.css\" \"" + upper(filename) + "/
     style/style.css\"\n");
```

### (b) HTML generator

HTML website tool is a C++based website generator [21, 23]. It generates a website using just the terminal. It has five themes with their own colour scheme. A website is made using plain HTML and CSS only. CSS is stored separately [3], The output code is written to a file as the text is typed into the terminal. The HTML output code, without using auxiliary files to store code is further presented.

```
file<<"<p id=\"contact\"></p>\n\n<div id=\"contact-div\">\n <div>\n\n
<p id=\"contact-title\">" << contact_title <<"</p>";
```

The web development applications show that a basic website could be made in a short amount of time, by using only specific terminal commands. Both web development applications shown that a basic website could be made in a short amount of time, by using only specific terminal commands.

## 4. Conclusion and Further work

Nowadays, it is essential to optimize code, in order to improve its execution time/readability/filesize. Therefore, applications such as CSS `Atibs`, `Lng`, and `oftile` can help developers better organise and use their code. Modernising the webpage can benefit the user both in terms of appearance and in terms of loading efficiency. As such, the newly-introduced applications can prove themselves useful for both developers and non-developers. The JS `cnvrt` application that converts HTML to JSON could be implemented in backend programming languages (NodeJS), for ease-of-use within full-stack web applications. Other JSON libraries based on [24] could be further implemented as well.

Future directions of study may include the development of applications that enable AI/ non-AI natural language programming or encoding, putting ease-of-use above the benefits of custom code.

## References

[1] P.-C. Wu and F.-J. Wang, **On efficiency and optimization of C++programs**, Software: Practice and Experience (1996) 26(4):453–465.

[2] L. Vogel, and T. Springer, **Speed Up the Web with Universal CSS Rendering**, In International Conference on Web Engineering (CWE 2023). Lecture Notes in Computer Science (2023) 13893:191–205. DOI: 10.1007/978-3-031-34444-2_14, Springer, Cham Publisher.

[3] A.Pintea, C++**libraries applications**, GitHub repository, GitHub Publisher https://github.com/AlexandruPintea2000/.

[4] P. Pathirana et al., **A Comparative Evaluation of PDF-to-HTML Conversion Tools**, In 2023 International Research Conference on Smart Computing and Systems Engineering, SCSE (2023) 6:1–7, IEEE Publisher.

[5] E. Meyer, and E. Weyl, CSS: **The Definitive Guide** (2023) O'Reilly Media Publisher.

[6] H. H. Hoos, **Programming by optimization**, Communications of the ACM, (2012) 55:70–80, DOI: 10.1145/2076450.2076469.

[7] F. Agner, Optimizing software in C++: **An optimization guide for Windows, Linux and Mac platforms**, Technical University of Denmark, Dbooks Publisher (2020).

[8] F. Chehimi, P. Coulton, and R. Edwards, C++**optimizations for mobile applications,** In 2006 IEEE International Symposium on Consumer Electronics (2006) pp. 1–6, DOI: 10.1109/ISCE.2006.1689506, IEEE Publisher.

[9] S. El-Kaliouby, S. Selim and A. H. Yousef, **Native Mobile Applications UI Code Conversion**, In 2021, Conference on Computer Eng. and Systems, (2021) pp. 1–5, DOI: 10.1109/ICCES54031.2021.9686093, IEEE Publisher.

[10] Mahmoud, Amira T. et al., **Compiler-based Web Services code conversion model for different languages of mobile application**, In: 2023 Intelligent Methods, Systems, and Applications (IMSA) (2023) 464–469, DOI: 10.1109/IMSA58542.2023.10217471, IEEE Publisher.

[11] OpenAI, online at: https://openai.com/.

[12] A. Latif, F. Azam, M. W. Anwar, and A. Zafar, **Comparison of Leading Language Parsers–ANTLR, JavaCC, SableCC, Tree-sitter, Yacc, Bison** , In 2023, Int. Conference on Software Tech. and Eng.(ICSTE) (2023) pp.7–13, DOI: 10.1109/ICSTE61649.2023.00009, IEEE Publisher.

[13] S. Behrens, **Prototyping interpreters using Python Lex-Yacc**, Dobbs Journal, (2004) 29:30–34.

[14] K. McHenry, et al., **Towards a universal, quantifiable, and scalable file format converter**, In 2009 Int. Conference on e-Science (2009) pp. 140–147. DOI: 10.1109/e-Science.2009.28, IEEE Publisher.

[15] Conversion-tools, (2025) online at https://conversiontools.io/.

[16] Vertopal.com Cloud platform, conversion solutions, 2024.

[17] J. van Riet, I. Malavolta, and T. A. Ghaleb, **Optimize along the way: An industrial case study on web performance**, Journal of Systems and Software (2023) 198, 111593, DOI: 10.1016/j.jss.2022.111593.

[18] M. Bosch, P. Genevès, and N. Layaïida, **Automated refactoring for size reduction of CSS style sheets**, In Proceedings of the 2014 ACM Symposium on Document engineering (2014) pp. 13–16, DOI: 10.1145/2644866.2644885, ACM Publisher.

[19] I. N. Ikhsan and M. Z. C. Candra, **Automatically: An automated refactoring method and tool for improving web accessibility**, In 2018 Fifth International Conference on Data and Software Engineering (ICoDSE), (2018) pp. 1–6, DOI: 10.1109/ICODSE.2018.8705894, IEEE Publisher.

[20] E. Uzun, **A regular expression generator based on CSS selectors for efficient extraction from HTML pages**, Turkish J. Electrical Eng. and Computer Sciences (2020) 28(6):3389–3401, DOI: 10.3906/elk-2004-67.

[21] J. Wagner, **Web Performance in Action: Building Fast Web Pages**, Manning Publisher, 2017.

[22] A. Pintea, **New Utility Libraries for Linux with Complex Data Types and System Resources**, Scientific Studies and Research. Series Mathematics and Informatics (2024) 34(1):57–62.

[23] L. Rai et al. Programming in C++. **Object Oriented Features,** (2019) volume 5, Walter de Gruyter GmbH & Co KG Publisher.

[24] A. Sarasa-Cabezuelo and J.-L. Sierra, **Grammar-driven development of JSON processing applications**, In 2023 Federated Conference on Computer Science and Information Systems (2023) pp. 1557–1564. IEEE Publisher.

Coventry University
Priory Street, Coventry, CV1 5FB, UK
e-mail: alexpintea.info@gmail.com
ORCID: 0009-0003-2158-9328