

ソフトウェア開発 第 12 回目授業

平野 照比古

2016/12/16

jQuery ファイルの違い

- jQuery のソースファイルには jQuery-<Version No>.js と jQuery-<Version No>.min.js の 2 種類が存在
- jQuery-<Version No>.min.js は jQuery-<Version No>.js からコメントや、空白を取り除き、変数名を短いものに置き換えるという操作を行って、ファイルサイズを小さくしたもの

jQuery のソースコード (1)

```
1 /*!  
2  * jQuery JavaScript Library v3.1.1  
3  * https://jquery.com/  
4  *  
5  * Includes Sizzle.js  
6  * https://sizzlejs.com/  
7  *  
8  * Copyright jQuery Foundation and other contributors  
9  * Released under the MIT license  
10 * https://jquery.org/license  
11 *  
12 * Date: 2016-09-22T22:30Z  
13 */
```

jQuery のソースコード (2)

```
14 ( function( global, factory ) {
15     "use strict";
16     if ( typeof module === "object" && typeof module.exports === "object" ) {
17         // For CommonJS and CommonJS-like environments where a proper 'window'
18         // is present, execute the factory and get jQuery.
19         // For environments that do not have a 'window' with a 'document'
20         // (such as Node.js), expose a factory as module.exports.
21         // This accentuates the need for the creation of a real 'window'.
22         // e.g. var jQuery = require("jquery")(window);
23         // See ticket #14549 for more info.
24         module.exports = global.document ?
25             factory( global, true ) :
26             function( w ) {
27                 if ( !w.document ) {
28                     throw new Error( "jQuery requires a window with a document" );
29                 }
30                 return factory( w );
31             };
32     } else {
33         factory( global );
34     }
35 // Pass this if window is not defined yet
36 } )( typeof window !== "undefined" ? window : this, function( window, noGlobal ) {
```

jQuery のソースコード (3)

```
37 // Edge <= 12 - 13+, Firefox <=18 - 45+, IE 10 - 11, Safari 5.1 - 9+, iOS 6 - 9.1
38 // throw exceptions when non-strict code (e.g., ASP.NET 4.5) accesses strict mode
39 // arguments.callee.caller (trac-13335). But as of jQuery 3.0 (2016), strict mode should be common
40 // enough that all such attempts are guarded in a try block.
41 "use strict";
```

jQuery 3.1.1.js のソース

- このリストでは詳細なコメントが付いていて比較的読みやすい。
- コメントを見るとクロスブラウザ対策が行われていることも見て取れる。
- ソースコードの最小化のテクニックとしてグローバルなオブジェクトを短い変数名で置き換える
- 冒頭の定義された関数の引数に、関数を渡している。その関数の仮引数は window となっている。これにより、開発時はわかりやすい変数名が使えるメリットがある。
- ” strict mode ” で動作 (41 行目)

短縮化のリスト

```
1 /*! jQuery v3.1.1 | (c) jQuery Foundation | jquery.org/license */
2 !function(a,b){"use strict";"object"==typeof module&&
3     "object"==typeof module.exports?module.exports=
4     a.document?b(a,!0):
5     function(a){if(!a.document)throw
6         new Error("jQuery requires a window with a document");
7     ("undefined"!=typeof window?window:this,
8     function(a,b){"use strict";
```

短縮化の方法

- 各関数内で変数名は 1 文字から始めている。
- 関数の仮引数も a から付け直している。
- JavaScript の固有の関数は当然のことながら変換されていない。
- このライブラリーは一つの関数を定義して、その場で実行している。
14 行目の `function()` の前に (がついている。
- 短縮化されたコードではこの部分が `!function()` となっている。関数オブジェクトを演算の対象とすることで実行する。
- そのほかにもキーワード `true` の代わりに `!0` としている。
- `if(){}else{}` 構文は `?:` に置き換えている。

JavaScript ファイルの短縮化ツール

JavaScript の短縮化ツールとして、Google が提供する Closure Compiler を紹介する。このサービスは次のように説明されている。

The Closure Compiler is a tool for making JavaScript download and run faster. It is a true compiler for JavaScript. Instead of compiling from a source language to machine code, it compiles from JavaScript to better JavaScript. It parses your JavaScript, analyzes it, removes dead code and rewrites and minimizes what's left. It also checks syntax, variable references, and types, and warns about common JavaScript pitfalls.

Closure Compiler の使い方

使い方については次のように書かれている。

You can use the Closure Compiler as:

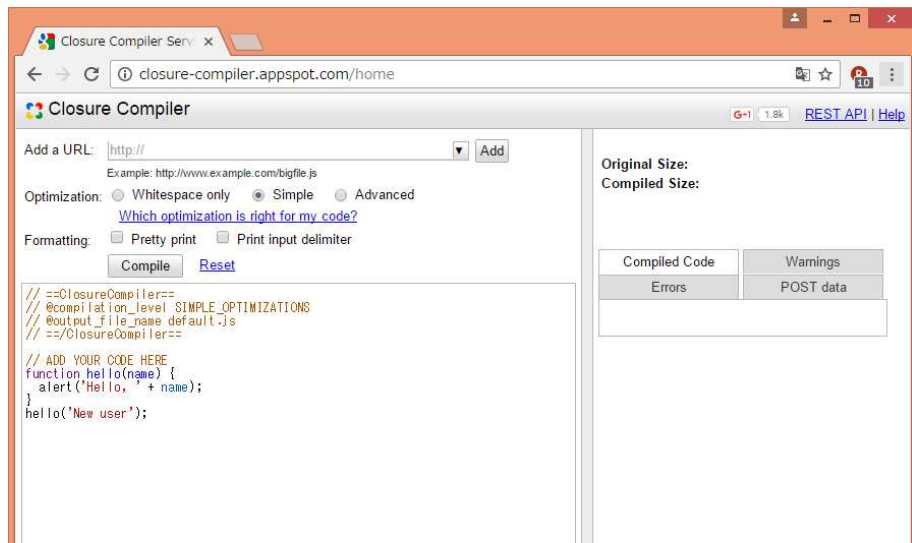
- *An open source Java application that you can run from the command line.*
- *A simple web application.*
- *A RESTful API.*

To get started with the compiler, see "How do I start" below.

ここでは 2 番目にある Web アプリケーションで行う。

Closure Compiler の Web サイト

サイトは <http://closure-compiler.appspot.com/home>



The screenshot shows the Closure Compiler web interface in a browser window. The address bar shows the URL <http://closure-compiler.appspot.com/home>. The page title is "Closure Compiler".

On the left side, there is a form with the following fields and options:

- Add a URL:** A text input field containing "http://". Below it, an example URL is shown: "Example: <http://www.example.com/bigfile.js>".
- Optimization:** Three radio buttons: "Whitespace only", "Simple" (selected), and "Advanced". Below them is a link: "[Which optimization is right for my code?](#)".
- Formatting:** Two checkboxes: "Pretty print" (checked) and "Print input delimiter".
- Buttons:** "Compile" and "Reset".

The main text area contains the following code:

```
// ==ClosureCompiler==
// @compilation_level SIMPLE_OPTIMIZATIONS
// @output_file_name default.js
// ==/ClosureCompiler==

// ADD YOUR CODE HERE
function hello(name) {
  alert('Hello, ' + name);
}
hello('New user');
```

On the right side, there is a summary section:

- Original Size:**
- Compiled Size:**

Below this, there are four tabs: "Compiled Code", "Warnings", "Errors", and "POST data". The "Compiled Code" tab is currently selected, showing the compiled output.

短縮化の例

使用するファイルは9回目の授業で用いた event.js

The screenshot shows the Closure Compiler web interface in a browser. The address bar shows the URL `closure-compiler.appspot.com/home`. The page title is "Closure Compiler".

On the left side, there is a section "Add a URL:" with a text input field containing `http://` and an "Add" button. Below it, an example URL is shown: `http://www.example.com/bigfile.js`.

Under "Optimization:", there are three radio buttons: "Whitespace only", "Simple" (which is selected), and "Advanced". Below these is a link: [Which optimization is right for my code?](#)

Under "Formatting:", there are two checkboxes: "Pretty print" and "Print input delimiter".

At the bottom of this section are two buttons: "Compile" and "Reset".

The main area displays the compiled JavaScript code. It starts with `window.onload = function() {` and contains various DOM manipulations and event listeners. The code is minified.

On the right side, there is a section for "Original Size:" and "Compiled Size:". Below this is a table with four buttons: "Compiled Code", "Warnings", "Errors", and "POST data".

短縮化の結果 (1)

画面の右のほうで、元のファイルの大きさが 1.53KB であったのに対し、短縮化の結果が 1.06KB となったことがわかる。

短縮化の詳細

```
window.onload=function(){var b=document.getElementById("Squares"),
    e=document.getElementById("select"),
    g=document.getElementById("colorName"),
    f=document.getElementById("radio"),
    h=document.getElementById("Set"),
    c=document.getElementsByClassName("click"),
    d=b.children[1];
    b.children[0].style.background="red";
    b.children[1].style.background="yellow";
    b.children[2].style.background="blue";
    e.style.fontSize="30px";
```

短縮化の結果 (2)

```
b.addEventListener("click",function(a){
  c[0].value=a.clientX;
  c[1].value=a.clientY;
  c[2].value=a.pageX;
  c[3].value=a.pageY;
  c[4].value=window.pageXOffset;
  c[5].value=window.pageYOffset;
  var b=a.target.getBoundingClientRect();
  c[6].value=a.pageX-b.left;
  c[7].value=a.pageY-b.top;
  g.value=a.target.style.background;d=a.target},!1);
```

短縮化の結果 (3)

```
f.addEventListener("click",function(a){
  alert(a.target.tagName);
  "DIV"===a.target.tagName&&(a.target.firstChild.checked=!0);
  console.log("----"+f.value);
  d.style.background=f.querySelector("input:checked").value,!1);
h.addEventListener("click",function(){d.style.background=g.value,!1});
```

- if 文であったものが論理式の&&で置き換えられている。
- document.getElementById などそのまま短縮化、共通化されていない。

document.getElementById を減らす

単純に関数を置き換えただけではうまくいかない。

The screenshot shows the Closure Compiler web interface. The URL bar shows the compiler's address. The main area contains a code editor with the following JavaScript code:

```
// ==ClosureCompiler==
// @output_file_name default.js
// @compilation_level SIMPLE_OPTIMIZATIONS
// ==/ClosureCompiler==

window.onload = function() {
  function getElm(N){
    return document.getElementById(N);
  }
  var Squares = getElm("Squares");
  var Select = getElm("select");
  var ColorName = getElm("colorName");
  var Radio = getElm("radio");
  var Set = getElm("Set");
  var Inputs = document.getElementsByClassName("click");
  var Squareschildren = Squares.children;
  var lastClicked = Squareschildren[1];
  function SetBackground(Pos, Color) {
    Squareschildren[Pos].style.background = Color;
  }
  var v = "value";
  SetBackground(0, "red");
  SetBackground(1, "yellow");
}
```

On the right, the compilation status is shown as a success. The original size is 605 bytes gzipped (1.57KB uncompressed), and the compiled size is 469 bytes gzipped (1.04KB uncompressed), saving 22.48% off the gzipped size (33.92% without gzip). The compiled code is shown in a box below the status:

```
window.onload=function(){var
g=document.getElementById("Squares"),e
=document.getElementById("select"),h=doc
ument.getElementById("colorName"),f=docu
ment.getElementById("radio"),k=document.
```

At the bottom, there are links for Terms of Service, Privacy Policy, and Google Home.

解説

ラップした関数は消えていて、もとの `document.getElementById` に戻っている。

これを避けるためには関数の引数に `document` と `document.getElementById` を渡すことで解決できる。

ソースコードの書き直し (1)

```
var Squares, Select, ColorName, Radio, Set;  
(function(document,getElementById){  
    Squares    = getElementById.call(document,"Squares");  
    Select     = getElementById.call(document,"select");  
    ColorName  = getElementById.call(document,"colorName");  
    Radio      = getElementById.call(document,"radio");  
    Set        = getElementById.call(document,"Set");  
})(document,document.getElementById);
```

ソースコードの書き直し (2)

The screenshot shows the Closure Compiler web interface in a browser. The URL is `closure-compiler.appspot.com/home#code%3Dwindow.onload%2520%253D%2520function()%2520{...}`. The interface includes a text area for adding a URL, optimization settings (Whitespace only, Simple, Advanced), formatting options (Pretty print, Print input delimiter), and buttons for Compile and Reset. The code being compiled is a JavaScript snippet that sets up a window.onload event to initialize a form with squares and a select element.

Compilation Status: Compilation was a success!

Original Size: 595 bytes gzipped (1.58KB uncompressed)

Compiled Size: 480 bytes gzipped (1KB uncompressed)

Saved: 19.33% off the gzipped size (36.52% without gzip)

The code may also be accessed at [default.js](#).

Compiled Code:

```

window.onload=function(){var c,d,g,e,h
(function(a,b){c=b.call(a,"Squ
ares");d=b.call(a,"select");g=b.call(a
colorName");e=b.call(a,"radio");h=ba
(a,"Set")})(document,document.getEleme
  
```

©2009 Google - [Terms of Service](#) - [Privacy Policy](#) - [Google Home](#)

ソースコードの書き直し (3)

コンパイル後の結果が 1.02KB とわずかに小さくなっていることがわかる。
短縮化を効率よくするためには元のプログラムも考えて作らないといけない。

注意

この書き直しで `document.getElementById` だけを渡せば十分と思われるかもしれないが、これは関数オブジェクトとなり、関数内での `this` が `document` にならないので、実行時にエラーが生ずる。

与えられた関数を別のオブジェクトのメソッドとして実行するために、`call` メソッドを用いる。

```
(function(document,getElementById){  
    Squares    = getElementById.call(document,"Squares");  
    Select     = getElementById.call(document,"select");  
    ColorName  = getElementById.call(document,"colorName");  
    Radio      = getElementById.call(document,"radio");  
    Set        = getElementById.call(document,"Set");  
})(document,document.getElementById);
```