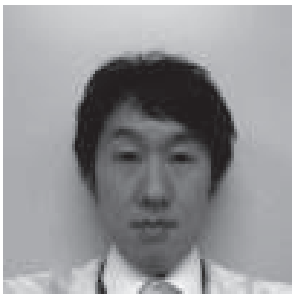
國元 祐二

株式会社ミガロ.

RAD事業部 技術支援課

[SmartPad4i] Web & ハイブリッドアプリ開発で役立つ IBM i & ブラウザデバッグテクニック



略歴
1979 年 3 月 27 日生まれ
2002 年 追手門学院大学文学部アジア文化学科卒業
2010 年 10 月 株式会社ミガロ. 入社
2010 年 10 月 RAD 事業部配属

現在の仕事内容
JC/400、SmartPad4i、Business4 Mobile の製品試験やサポート業務、導入支援などを行っている。

1.はじめに

プログラム開発において、デバッグ作業は非常に重要である。

デバッグとは、作成したプログラムにバグがないかを確認するテストや、障害が発生した際にプログラムを動かしながら原因を調査する作業を意味する。つまりデバッグに精通していれば、開発時にバグを減らし、障害発生時に問題を早急に解決できる。

SmartPad4i のプログラム開発では、RPG、COBOL などの IBM i プログラムがビジネスロジックの中心となるため、プログラム開発時のデバッグ作業は 5250 エミュレータ上で行える。使い慣れた IBM i プログラム言語を使ってデバッグできるので、バッチジョブのデバッグ手順を知っていれば、開発時に困ることはない。

しかしアプリケーションの運用中に不定期に発生するエラーなど、再現できない障害は、デバッグ作業で調査するのが難しい。そういう場合は、調査のための知識と工夫が必要である。

また SmartPad4i の画面は HTML や CSS で作成するので、JavaScript で機能をカスタマイズすることも多い。そうした JavaScript でのカスタマイズ部分は、IBM i 側ではデバッグできないので、Web ブラウザ側のデバッグ機能を知っていると、開発や調査で非常に役立つ。

このようにデバッグ作業をいろいろな角度で行うほど、開発時のテストや障害解決の精度を上げられる。そのためには、デバッグや使用ツールについて知識を深める必要がある。

本稿では、IBM i 側でのデバッグと Web ブラウザ側のデバッグについて、知っておくと役立つ情報・テクニックを説明する。

2. IBM iでのデバッグ手法

2-1. IBM i プログラムでのデバッグ

SmartPad4i は画面には HTML を、ビジネスロジックには IBM i プログラム（RPG、COBOL など）を使って開発する。デバッグ作業は IBM i 上で行えるが、5250 画面の対話型ジョブではなく、バッチジョブとして動作している。対話型ジョブとは手順の若干違う部分があるので、注意が必要である。

まず、基本的な対話型ジョブのデバッグについて確認する（IBM i プログラムのデバッグではこれが基本となる）。

デバッグ手順は、次のとおりである。

・プログラム実行前にデバッグオプションを設定
・コンパイル
・STRDBG コマンドを実行
・ソースにブレークポイントを設定
・プログラムを動作させてデバッグ作業

これらの手順でポイントになる点を、いくつか補足する。

図1

デバッグオプション

RPG

CRTRPGPGM PGM(ライブラリ名/プログラム名)
SRCFILE(ライブラリ名/ソースファイル名)
SRCMBR(ソースファイルメンバー名)
OPTION(*SRCDBG)

ILERPG

CRTBNDRPG PGM(ライブラリ名/プログラム名)
SRCFILE(ライブラリ名/ソースファイル名)
SRCMBR(ソースファイルメンバー名)
DBGVIEW(*SOURCE)

図2

デバッグオプション

COBOL

CRTCBLPGM PGM(ライブラリ名/プログラム名)
SRCFILE(ライブラリ名/ソースファイル名)
SRCMBR(ソースファイルメンバー名)
OPTION(*SRCDBG)

図3

STRDBGコマンド

STRDBG PGM(ライブラリ名/プログラム名)
UPDPROD(*YES)
OPMSRC(*YES)

コンパイル時のデバッグオプション

RPG/400 のプログラム作成の場合、CRTRPGPGM コマンドでコンパイルを実行時に、ソース・リスト・オプションへ「*SRCDBG」を設定する。ILERPG の場合は、CRTBNDRPG コマンドでコンパイルを実行時、デバッグ用ビューに「*SOURCE」を設定する。【図 1】

COBOL の場合、CRTCBLPGM コマンドでコンパイルを実行時に、ソース・リスト・オプションに *SRCDBG を設定する。【図 2】

STRDBG コマンド

デバッグオプションを設定してコンパイルしたプログラムに対して、STRDBG コマンドを実行する。【図 3】

STRDBG コマンドを実行すると、5250 エミュレータ上でソースが表示されるので、ブレークポイントを設定する行を選択して、F6 キーを押下する。【図 4】

プログラムを実行すると、設定したブレークポイントでプログラムが停止してデバッグ調査が行える。対話型ジョブの RPG や COBOL であれば、この手順だけでデバッグが可能である。

しかし SmartPad4i のアプリケーションは前述したとおり、バッチジョブとして動作するため、次にそのポイントを説明する。

2-2. バッチジョブのデバッグポイント

バッチジョブのデバッグでは、前述したデバッグ手法に加えて、IBM i プログラム実行前に、サービス・ジョブ開始（STRSRVJOB）コマンドを使ってバッチジョブを指定する必要がある。

プログラム実行前には、通常の IBM i プログラムと同様に、デバッグオプションを設定してコンパイルを実行する。

次に、SmartPad4i アプリケーションを実行することで作成されるジョブの「ジョブ」「ユーザー」「番号」を、WRKACTJOB コマンドから取得する（番号はブラウザのタイトルバーに表示される）。【図 5】

ジョブの情報を取得したあと、5250 エミュレータ上で STRSRVJOB コマンド（サービスジョブ開始）を実行する。引数には確認したジョブ、ユーザー、番号を指定する。【図 6】

あとは対話型ジョブのデバッグと同じように、STRDBG コマンド（デバッグ開始）を実行する。【図 3】

5250 エミュレータ上でソースが表示されるので、ブレークポイントを設定する行を選択し、F6 キーを押下する。

ブラウザで SmartPad4i アプリケーションを操作すると、IBM i 側のプログラム処理で停止してデバッグ調査が行える。【図 7】

こうしたデバッグ手法を知っていれば、開発時のプログラム確認で非常に役立つ。ただし問題となる動作を確実に再現・実行できなければ、有効ではない。

たとえばアプリケーションの運用上は稀に発生するが、テストでは再現できない障害の場合は、IBM i プログラム側で定様式ダンプを出力する手法が有効である。

2-3. 定様式ダンプの活用

定様式ダンプとは、IBM i プログラムのフィールドの内容、データ構造の内容、配列やテーブルの内容、ファイル情報のデータ構造、およびプログラム状況のデータ構造を含むファイルである。

IBM i ではあらかじめプログラムに設定しておくと、エラーが発生したときに、定様式ダンプを自動で出力できる。この機能を利用すると、エラーが発生したあとに出力された情報から原因を調査できる。

通常、IBM i プログラムでエラーが発生した場合には、「ダンプを出力する」「終了する」などのメッセージ応答を行える。そのためこの応答を、自動的に「ダンプを出力する」で返すように設定しておく必要がある。

応答の設定は、IBM i のシステム応答リスト項目が有効である。システム応答リストを利用すると、IBM i 側のプログラムでエラーが発生した際に、自動的に応答できる。

応答リストは、【図 8】のコマンドで追加できる。使用する言語によって、設定するコマンドが異なるので、注意が必要である。

ADDRPYLE はシステム応答リスト項目を追加するコマンドで、MSGID に定義されたエラーが発生した際に、RPY で設定した応答メッセージを SEQNBR 順に返す。

これだけでシステム応答リスト項目の設定は完了である。ただし応答するプログラム側にも、システム応答リストを利用するように設定する。

SmartPad4i プログラムが起動時に実行する、SETENV の CL プログラムに自動応答を追加するとわかりやすい。【ソース 1】

以上で、定様式ダンプを自動出力する設定は完了である。

SmartPad4i のプログラムを実行して、IBM i プログラム側でエラーが発生した場合には、エラー発生時のダンプ内容がアウトキューの QEZDEBUG に QPPGMDMP のファイルとして出力される。【図 9】

出力されたダンプファイルを確認することで、再現が難しい現象でも、あとから発生原因を解析できる。特殊ではあるが、デバッグの手法としては、非常に有効なテクニックである。【図 10】

3.ブラウザでのデバッグ手法

3-1. Web やハイブリッドアプリケーションのデバッグ

一般的に Web やハイブリッドアプリケーションの開発では HTML、CSS、JavaScript を利用する。

SmartPad4i でも、ビジネスロジックは IBM i 側のプログラムで動作するが、こうした Web 側のカスタマイズ開発も可能である。

開発した HTML や CSS、JavaScript がどのように動作・表示されるかを確認するには、ブラウザで実際に実行するしかない。ブラウザでの実行は簡単だが、前述した IBM i プログラムのようにブレークポイントを設定して、プログラムコードを追うようなデバッグ調査は行えない。

これまで、Web アプリケーション開発ではこうした点が非常に面倒であったが、最近のブラウザではデバッグ専用機能が実装され、便利になっている。次に、このブラウザ自体のデバッグ機能について説明する。

3-2. ブラウザのデベロッパー・ツール

現在使われているブラウザには HTML や CSS、JavaScript を簡単にデバッグできるツールが搭載されている。

図4

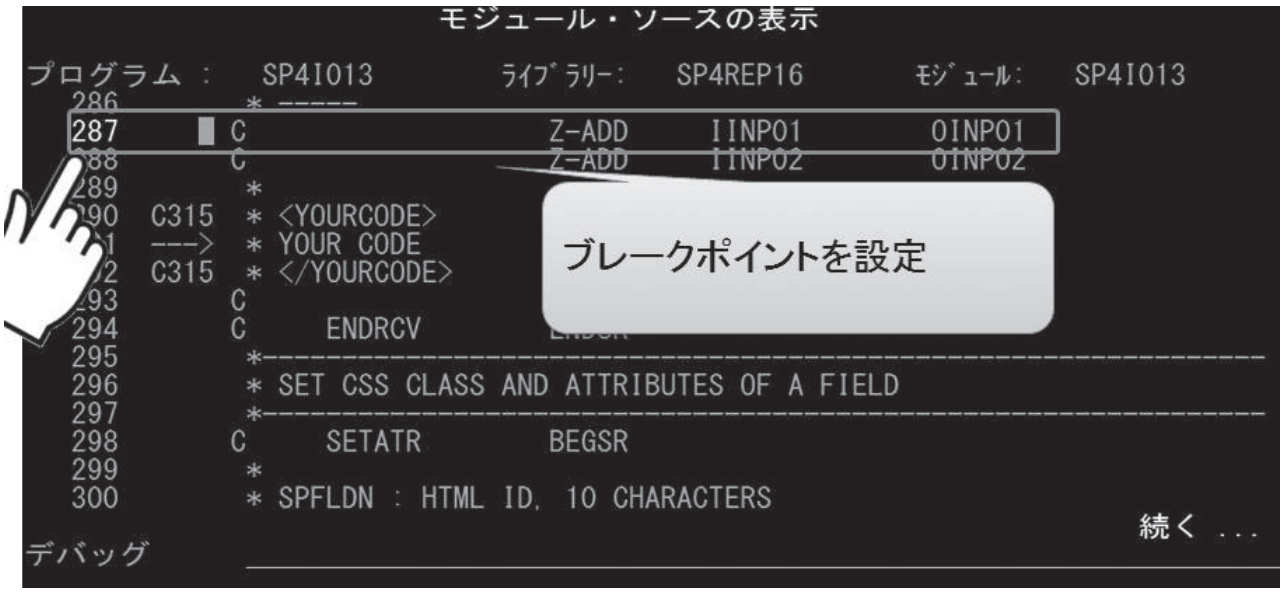
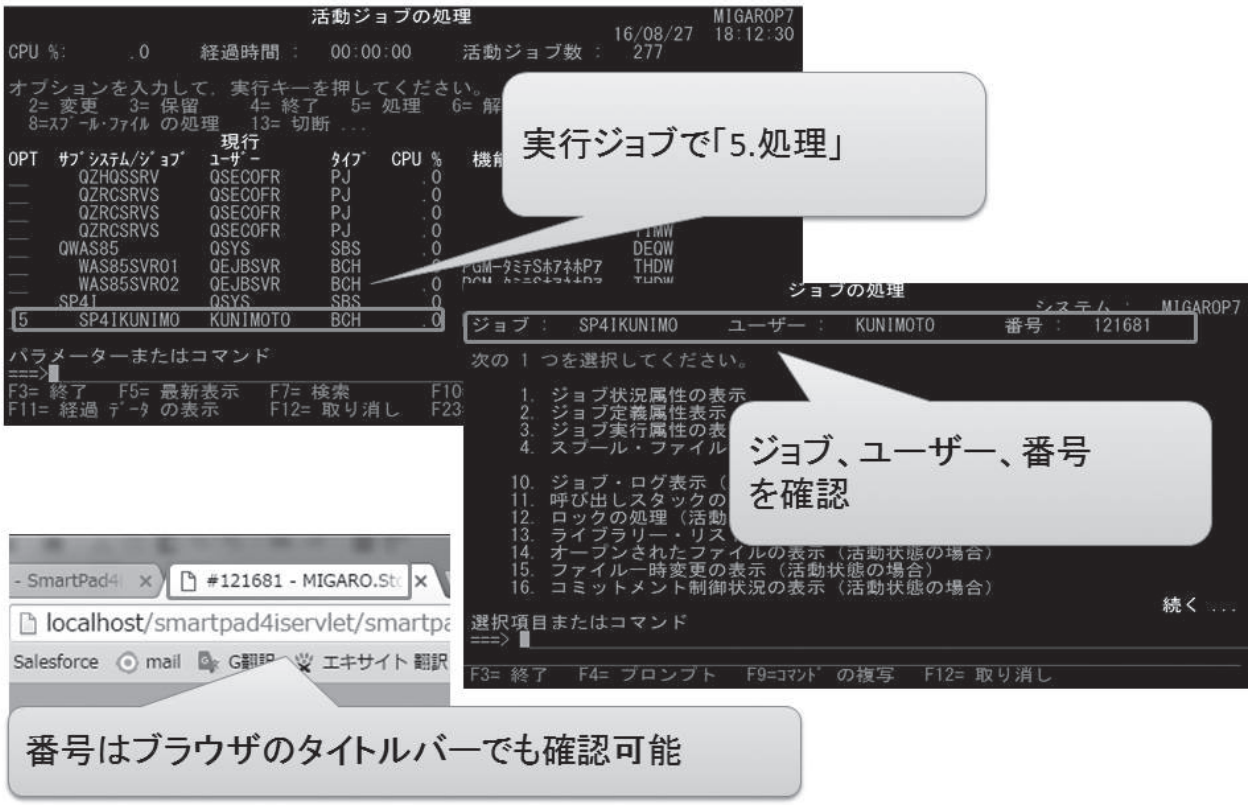

モジュール・ソースの表示
プログラム : SP4I013 ﾗｲﾌﾞﾗﾘｰ: SP4REP16 ﾓｼﾞｭｰﾙ: SP4I013
287 C Z-ADD IINP01 OINP01
C Z-ADD IINP02 OINP02
C315 * <YOURCODE>
---> * YOUR CODE
C315 * </YOURCODE>
C ENDRCV
* SET CSS CLASS AND ATTRIBUTES OF A FIELD
C SETATR BEGSR
* SPFLDN : HTML ID, 10 CHARACTERS
続く ...
デバッグ
ブレークポイントを設定

図5


活動ジョブの処理
MIGAROP7
16/08/27 18:12:30
CPU %: .0 経過時間 : 00:00:00 活動ジョブ数 : 277
オプションを入力して、実行キーを押してください。
2= 変更 3= 保留 4= 終了 5= 処理 6= 解
8=ｽﾌﾟｰﾙ･ﾌｧｲﾙ の処理 13= 切断 ...
現行
OPT サブシステム/ジョブ ユーザー タイプ CPU %
QZHQSSRV QSECOFR PJ .0
QZRCSRVS QSECOFR PJ .0
QZRCSRVS QSECOFR PJ .0
QZRCSRVS QSECOFR PJ .0
QWAS85 QSYS SBS .0
WAS85SVR01 QEJBSVR BCH .0
WAS85SVR02 QEJBSVR BCH .0
SP4I QSYS SBS .0
5 SP4IKUNIMO KUNIMOTO BCH .0
パラメーターまたはコマンド
===>
F3= 終了 F5= 最新表示 F7= 検索
F11= 経過 ﾃﾞｰﾀ の表示 F12= 取り消し
実行ジョブで「5.処理」
ジョブの処理
システム : MIGAROP7
ジョブ : SP4IKUNIMO ユーザー : KUNIMOTO 番号 : 121681
次の 1 つを選択してください。
1. ジョブ状況属性の表示
2. ジョブ定義属性表示
3. ジョブ実行属性の表
4. スプール・ファイル
10. ジョブ・ログ表示 (
11. 呼び出しスタックの
12. ロックの処理（活動
13. ライブラリー・リス
14. オープンされたファイルの表示（活動状態の場合）
15. ファイル一時変更の表示（活動状態の場合）
16. コミットメント制御状況の表示（活動状態の場合）
続く ...
選択項目またはコマンド
===>
F3= 終了 F4= プロンプト F9=ｺﾏﾝﾄﾞ の複写 F12= 取り消し
ジョブ、ユーザー、番号を確認
- SmartPad4
#121681 - MIGARO.St
localhost/smartpad4iservlet/smartpa
Salesforce mail G翻訳 エキサイト 翻訳
番号はブラウザのタイトルバーでも確認可能

最近のブラウザでは、Chrome が機能や動作速度で優れており、使用しているユーザーが最も多い。

そこで数種あるブラウザのなかから、本稿では Chrome ブラウザに標準搭載されている「デベロッパー・ツール」を題材に説明する。デベロッパー・ツールは Chrome ブラウザを導入していれば、無償で利用できる。

デベロッパー・ツールの実行方法は簡単である。Chrome ブラウザを選択した状態で F12 キーを押下、または「ブラウザのメニュー」→「その他のツール」→「デベロッパー・ツール」から起動できる。【図 11】

デベロッパー・ツールは、デフォルトではブラウザにドッキングした状態で表示される。ドッキングされた状態では使いづらい場合、デベロッパー・ツールのメニューから［Dock side］を選ぶことで、別ウインドウの表示に変更できる。【図 12】

3-3. JavaScript のデバッグ手法

ブラウザのデベロッパー・ツールでは、開発ツールのように JavaScript のソースへブレークポイントを設定し、ステップ実行や変数の内容をチェックしながら JavaScript を実行できる。これによって IBM i プログラムと同様に、JavaScript などのデバッグ作業が可能となる。

ここからは、実際に JavaScript のデバッグ方法について説明していく。

まず SmartPad4i アプリケーションを実行後、デベロッパー・ツールを起動する。【図 13】

メニューの「Sources」タブを選択後、ツリーに表示されるファイルを選ぶと、実行中の JavaScript ソースが表示される。【図 14】

表示されたソースの行番号をクリックすることで、ソースにブレークポイントを設定できる。ブレークポイントを設定しておくと、画面を操作して JavaScript が該当行に進んだ時点で停止する。

またブレークポイントを設定する別の方法として、JavaScript のソースに、「debugger;」と記述する方法もある。

debugger; が呼び出されると、ブレークポイントと同様に JavaScript を一時停止させられる。【図 15】

JavaScript の処理がブレークポイントに到達すると、ブラウザの画面側は停止状態になるので操作はできない。【図 16】

停止後は、右上のメニューで実行、停止、ステップ実行が可能となり、プログラムの実行内容を細かくチェックできる。【図 17】

また JavaScript のデバッグ時には、コンソールから任意の JavaScript コードを実行できる。コンソールはソース表示の下部にあり、Console タブを選んで利用する。

たとえば、コンソールで計算結果位置に“TEST”の文字列を出力する JavaScript を記述して実行すると、画面上に“TEST”が表示される。【図 18】

とくに特殊データや実行条件を必要とする場合、そうしたテスト環境を作らなくても簡単に指定できるので、調査時に便利である。

またデベロッパー・ツールでは、表示されたソースを直接編集することも可能である。この機能を使うと、デバッグをしながら JavaScript を修正でき、作業効率が非常によい。【図 19】

3-4. HTML のデバッグ手法

デベロッパー・ツールを利用すると、JavaScript だけではなく、HTML、CSS についても値を変更しながら表示確認できる。

使い方は、開発者ツールの「Elements」タブを選び、一番左上のアイコンを選択後、ブラウザに表示されている画面で確認したい項目をクリックするだけである。【図 20】【図 21】

項目を選択するとソース上の該当箇所が反転し、CSS で定義されている設定、画面上のサイズ、イベント処理などさまざまな情報を確認できる。

さらに表示された設定は、デベロッパー・ツールで変更すると、ブラウザ上の画面にも直接反映される。画面が思いどおりに調整できない場合は、画面を見ながらソースを変更できる。【図 22】

もちろんこの設定は一時的な変更なので、最終的には HTML や CSS の設定を再定義する必要はあるが、レイアウト調整はかなり効率化できる。

3-5. 通信内容のチェック

デベロッパー・ツールには、Web サーバーとブラウザ間の通信内容の詳細を確認する機能も搭載されている。

デベロッパー・ツールの［Network］タブを選択後に、SmartPad4i プログラムからサーバーにリクエストを送信すると、HTML、CSS、JavaScript ファイル、画像ファイルなどサーバーから受信するファイルのリストが表示される。

これは、画像や外部定義のファイルが読み込めない場合の確認に有効である。パス記述の誤りや、ファイルがサーバーに存在しないなどの誤りを即座にチェックできる。【図 23】

画像ファイルが存在しない場合などは、ブラウザ画面に表示されないので、比較的簡単に特定できる。しかし外部定義の CSS や JavaScript ファイルが読み込まれていない場合には、気づかないこともあるので、ネットワーク監視は有用である。

また画面表示の過程で必要とされる時間も確認でき、パフォーマンスの指標としても利用できる。

3-6. 他のブラウザツール

本稿では Chrome に搭載されているデベロッパー・ツールについて紹介してきたが、Internet Explorer、Microsoft Edge、FireFox にも開発者ツールは搭載されている。

それぞれにインターフェースは異なるが、ここで紹介したような機能は Chrome のデベロッパー・ツールと同じく標準搭載されているので、実際に利用しているブラウザを使うのがよい【図 24】。もちろんこれらのツールも、Chrome の「デベロッパー・ツール」と同じく、ブラウザに標準で搭載されている。

4.まとめ

以上、SmartPad4i を使った Web やハイブリッドアプリケーション開発で有効なデバッグテクニックを説明した。

デバッグでは IBM i 側とブラウザ側の両方で、さまざまな角度から調査するためのツールがすべて標準で用意されている。これらのツールは非常に便利で、優れた機能を備えている。

図6

STRSRVJOBコマンド

STRSRVJOB

STRSRVJOB JOB(番号/ユーザー/ジョブ)

図7

こうした機能を調査・検証するなかで、IBM i やブラウザの機能が日々進化していることをあらためて実感した。

プログラム開発やアプリケーション障害時の調査では、いかにデバッグテクニックを駆使できるかが、作業効率の観点では重要になる。開発者の方々には、こうした機能を作業効率アップにぜひ活用していただきたい。

M

図8

自動応答リストの追加コマンド

RPG

```
ADDRPYLE SEQNBR(9700) MSGID(RPG0000) RPY('D')
```

ILERPG

```
ADDRPYLE SEQNBR(9800) MSGID(RNQ0000) RPY('D')
```

COBOL

```
ADDRPYLE SEQNBR(9900) MSGID(LBE0000) RPY('D')
```

ソース1

自動応答を返答するための処理

```
0001.00 PGM
0002.00 CHGJOB INQMSGRPY(*SYSRPYL)
0003.00 CHGLIBL LIBL(SMPLIB  SP4I QTEMP QGPL)
0004.00 ENDPGM
```

図9

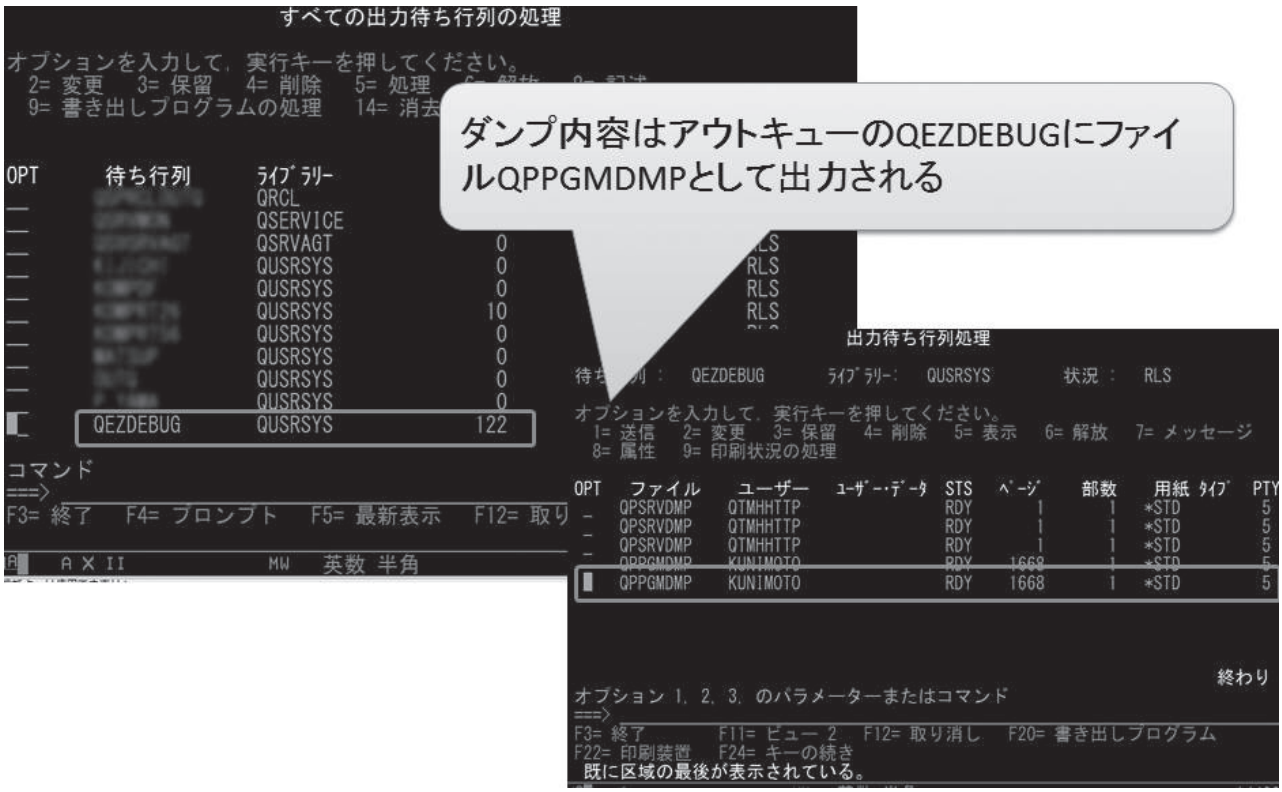

すべての出力待ち行列の処理
オプションを入力して、実行キーを押してください。
2= 変更 3= 保留 4= 削除 5= 処理
9= 書き出しプログラムの処理 14= 消去
OPT 待ち行列 ﾗｲﾌﾞﾗﾘｰ
QRCL
QSERVICE
QSRVAGT 0
QUSRSYS 0
QUSRSYS 0
QUSRSYS 10
QUSRSYS 0
QUSRSYS 0
QUSRSYS 0
QUSRSYS 0
QEZDEBUG QUSRSYS 122
コマンド
===>
F3= 終了 F4= プロンプト F5= 最新表示 F12= 取り
A X II MW 英数 半角
ダンプ内容はアウトキューのQEZDEBUGにファイルQPPGMDMPとして出力される
出力待ち行列処理
待ち行列 : QEZDEBUG ﾗｲﾌﾞﾗﾘｰ: QUSRSYS 状況 : RLS
オプションを入力して、実行キーを押してください。
1= 送信 2= 変更 3= 保留 4= 削除 5= 表示 6= 解放 7= メッセージ
8= 属性 9= 印刷状況の処理
OPT ファイル ユーザー ﾕｰｻﾞｰ･ﾃﾞｰﾀ STS ﾍﾟｰｼﾞ 部数 用紙 ﾀｲﾌﾟ PTY
QPSRVDMP QTMHHTTP RDY 1 1 *STD 5
QPSRVDMP QTMHHTTP RDY 1 1 *STD 5
QPSRVDMP QTMHHTTP RDY 1 1 *STD 5
QPPGMDMP KUNIMOTO RDY 1668 1 *STD 5
終わり
オプション 1, 2, 3, のパラメーターまたはコマンド
===>
F3= 終了 F11= ビュー 2 F12= 取り消し F20= 書き出しプログラム
F22= 印刷装置 F24= キーの続き
既に区域の最後が表示されている。

図10

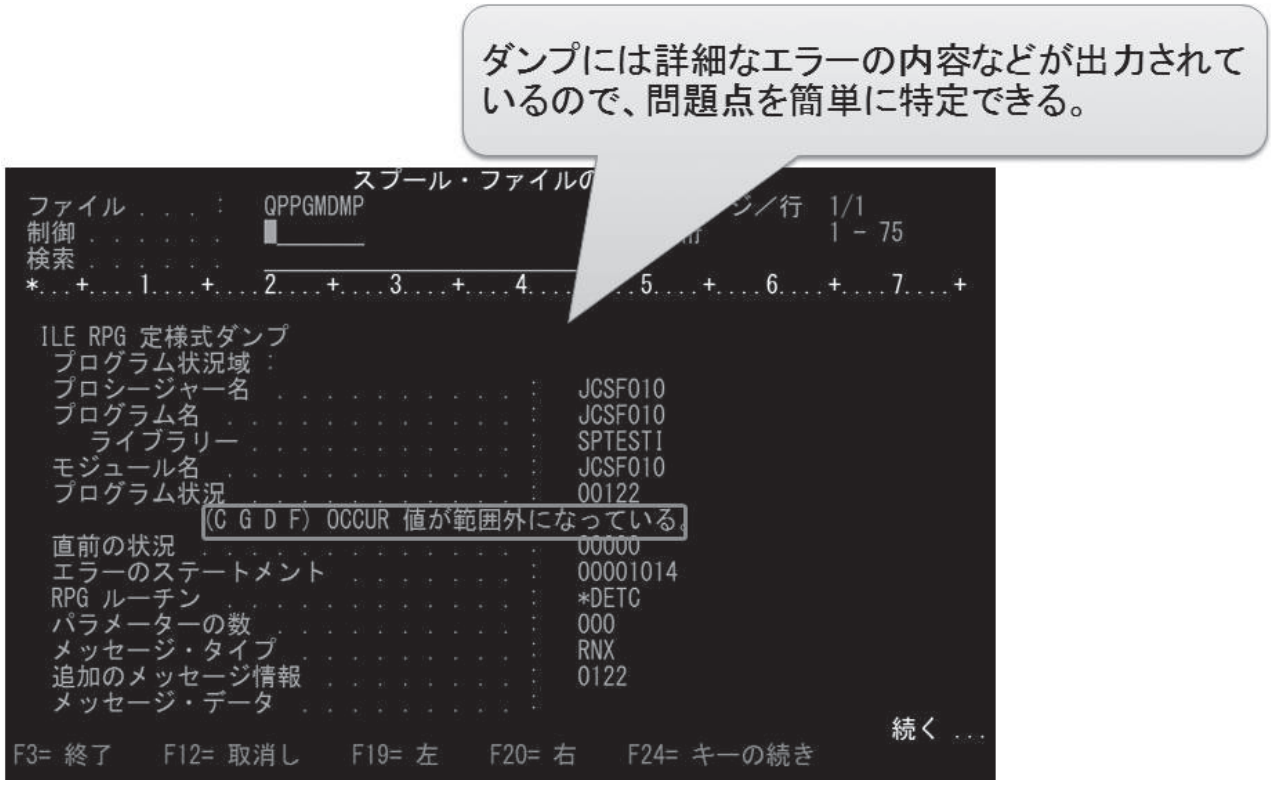

ダンプには詳細なエラーの内容などが出力されているので、問題点を簡単に特定できる。
スプール・ファイルの
ファイル . . . : QPPGMDMP
制御 . . . . . _____
検索 . . . . . _____
*...+...1...+...2...+...3...+...4...+...5...+...6...+...7...+
ILE RPG 定様式ダンプ
プログラム状況域 :
プロシージャー名 . . . . . : JCSF010
プログラム名 . . . . . : JCSF010
ライブラリー . . . . . : SPTESTI
モジュール名 . . . . . : JCSF010
プログラム状況 . . . . . : 00122
(C G D F) OCCUR 値が範囲外になっている。
直前の状況 . . . . . : 00000
エラーのステートメント . . . . . : 00001014
RPG ルーチン . . . . . : *DETC
パラメーターの数 . . . . . : 000
メッセージ・タイプ . . . . . : RNX
追加のメッセージ情報 . . . . . : 0122
メッセージ・データ . . . . . :
続く ...
F3= 終了 F12= 取消し F19= 左 F20= 右 F24= キーの続き

図11


1. メニュー
2. その他のツール
3. デベロッパーツール
新しいタブ
新しいタブ(T) Ctrl+T
新しいウィンドウ(N) Ctrl+N
シークレット ウィンドウを開く(I) Ctrl+Shift+N
履歴(H)
ダウンロード(D) Ctrl+J
その他のツール(L)
編集 切り取り(T) コピー(C) 貼り付け(P)
設定(S)
ヘルプ(H)
終了(X) Ctrl+Shift+Q
閲覧履歴を消 Ctrl+Shift+Delete
拡張機能(E)
タスク マネー Shift+Esc
エンコード(E)
デベロッパー ツール(D) Ctrl+Shift+I

図12


Dock sideをUnlock into separate window
にして別ウインドウにすると使いやすい
別ウインドウ表示
Dock side
Hide console Esc
Search all files Ctrl + Shift + F
More tools
Shortcuts
Settings F1
Help

図13

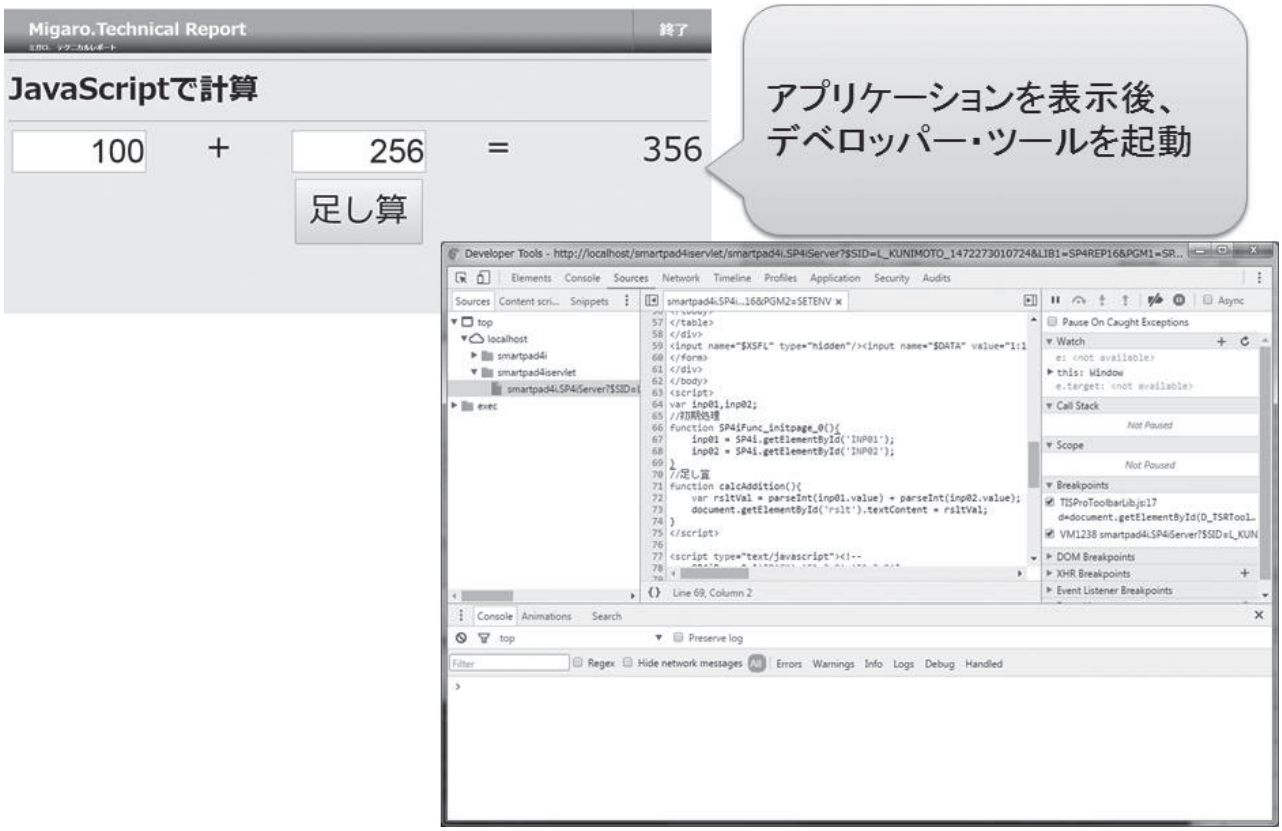

Migaro.Technical Report
終了
JavaScriptで計算
100
+
256
=
356
足し算
アプリケーションを表示後、
デベロッパー・ツールを起動

図14

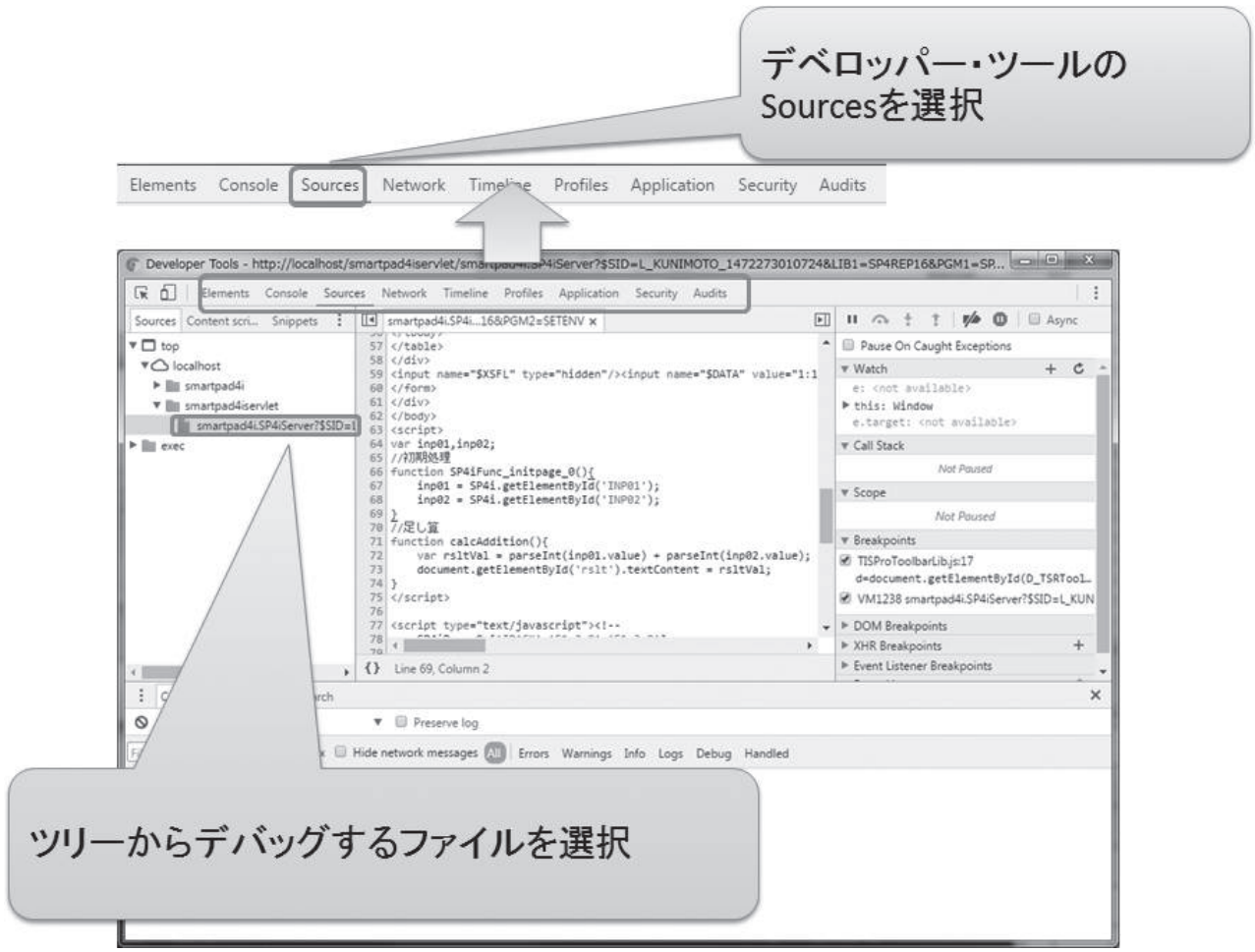

デベロッパー・ツールの
Sourcesを選択
Elements
Console
Sources
Network
Timeline
Profiles
Application
Security
Audits
ツリーからデバッグするファイルを選択

図15


行番号をクリックすることでブレークポイントが設定可能
ソースにdebugger;を記述することでもブレーク可能
63 <script>
64 var inp01,inp02;
65 //初期処理
71 function calcAddition(){
72 var rsltVal = parseInt(inp01.value) + parseInt(inp02.valu
73 document.getElementById('rslt').textContent = rsltVal;
74 }
75 </script>
71 function calcAddition(){
72 debugger;
73 var rsltVal = parseInt(inp01.value) + parseInt(inp02.valu
74 document.getElementById('rslt').textContent = rsltVal;
75 }
76 </script>

図16


localhost/smartpad4iservlet/smartpad4i.SP4iServer?$SID=L_KUNIMOTO_1472273010724&LIB1=SP4REP1
Migaro.Technical Report
Paused in debugger
JavaScriptで計算
0
+
0
=
足し算
ブレークポイントで停止
画面も停止中
59 <input name="$XSFL" type="hidden"/ ="$DATA" value=
60 </form>
61
62 </body>
63 <script>
64 var inp01,inp02;
65 //初期処理
66 function SP4iFunc_initpage_0(){
67 inp01 = SP4i.getElementById('I 1');
68 inp02 = SP4i.getElementById('I 02');
69 }
70 //足し算
71 function calcAddition(){
72 debugger;
73 var rsltVal = parseInt(inp01.value) + parseInt(inp02.valu
74 document.getElementById('rslt').textContent = rsltVal;
75 }
76 </script>

図17

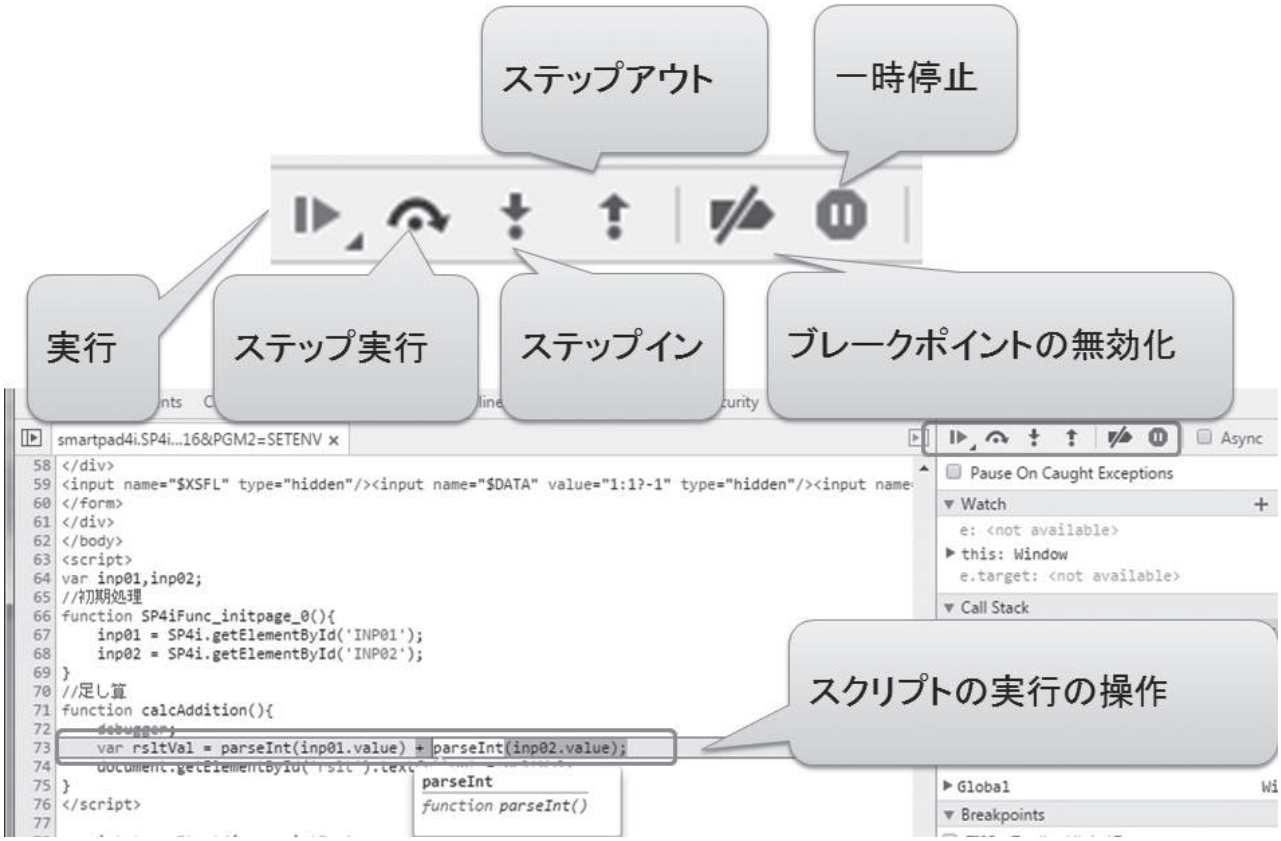

ステップアウト
一時停止
実行
ステップ実行
ステップイン
ブレークポイントの無効化
スクリプトの実行の操作
smartpad4i.SP4i...16&PGM2=SETENV
Async
Pause On Caught Exceptions
Watch
this: Window
Call Stack
Global
Breakpoints
parseInt
function parseInt()

図18


Migaro.Technical Report
JavaScriptで計算
0 + 0 =
足し算
コンソールを使うと
スクリプトが動的に実行可能
Console
Animations
Search
chrome-extension://oh...hhiiaoi
Preserve log
Filter
Regex
Hide network messages
All
Errors
> document.getElementById('rslt').textContent = "TEST"
< "TEST"
Migaro.Technical Report
JavaScriptで計算
0 + 0 = TEST
足し算
計算結果位置にTESTを出力

図19

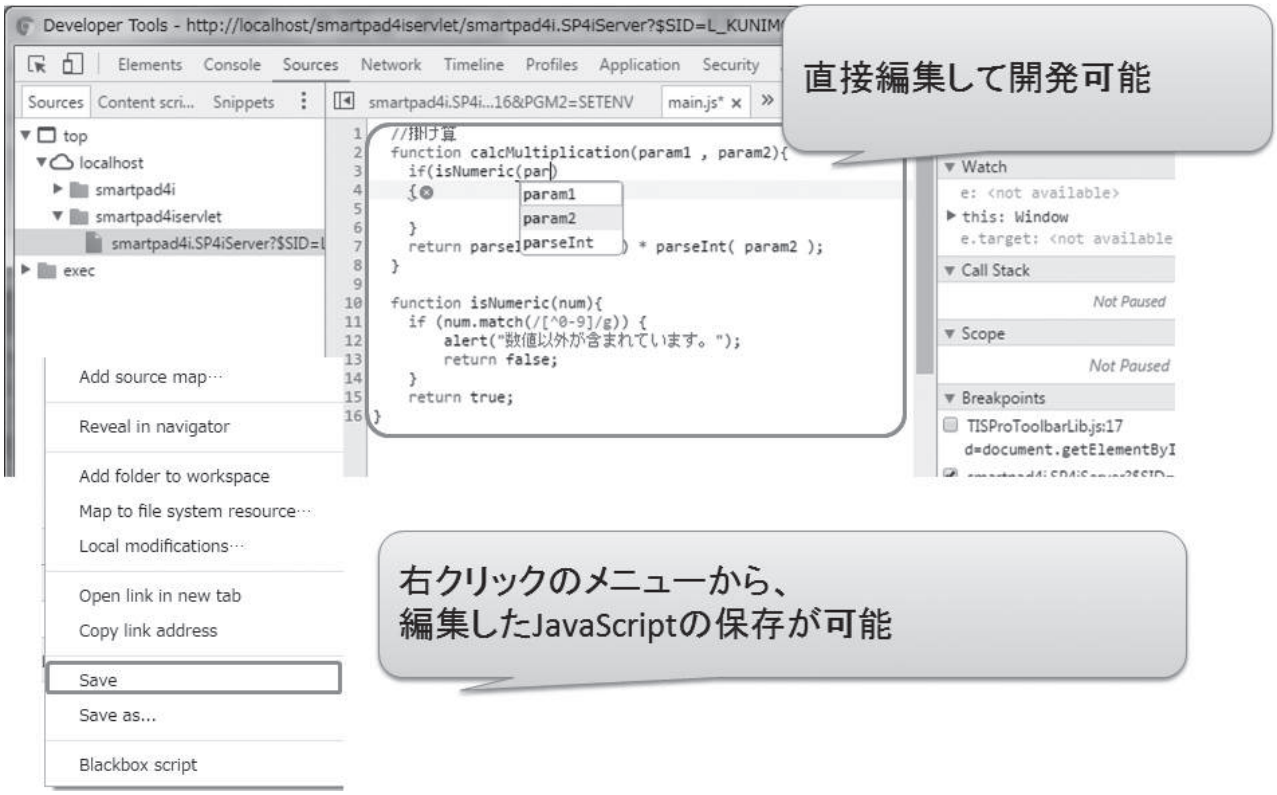

直接編集して開発可能
右クリックのメニューから、
編集したJavaScriptの保存が可能
Add source map…
Reveal in navigator
Add folder to workspace
Map to file system resource…
Local modifications…
Open link in new tab
Copy link address
Save
Save as...
Blackbox script

図20

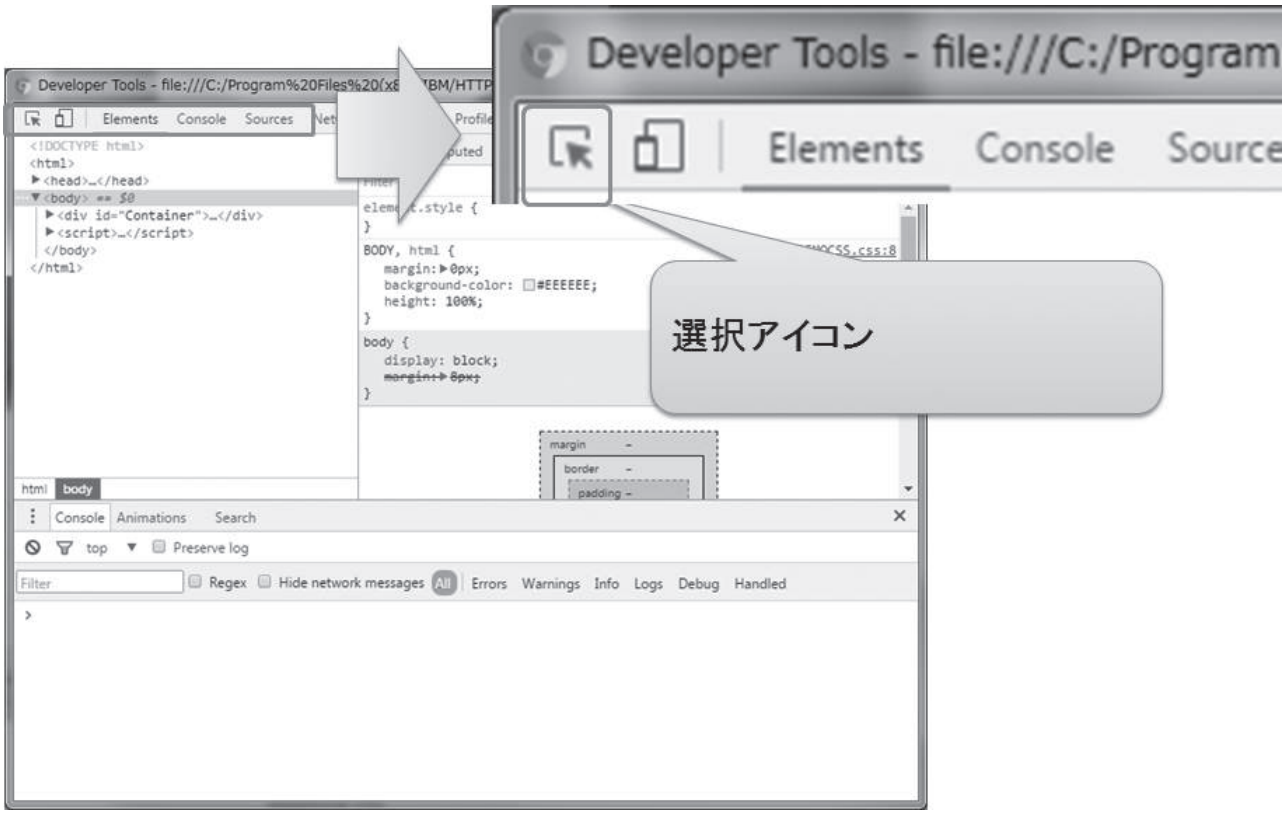

Developer Tools - file:///C:/Program
Elements
Console
Source
選択アイコン

図21

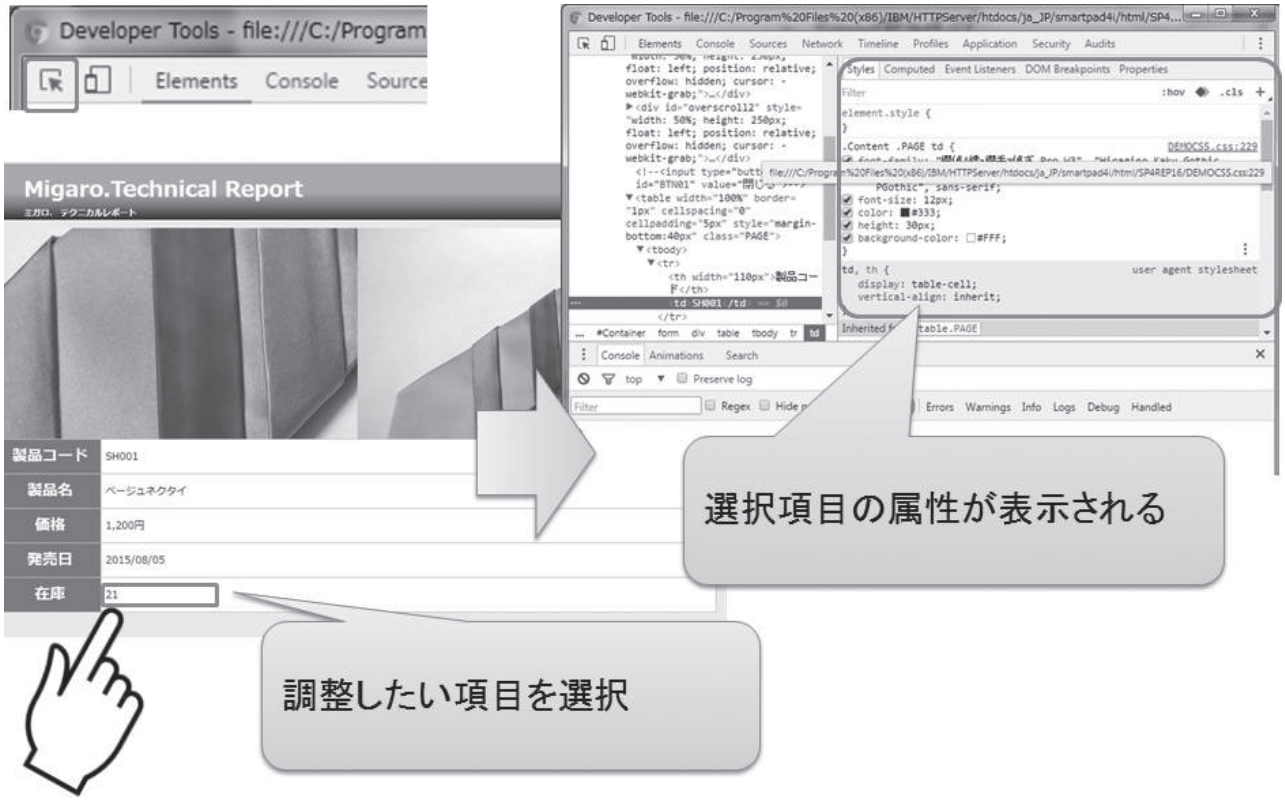

Developer Tools - file:///C:/Program
Elements
Console
Source
Migaro.Technical Report
ミガロ. テクニカルレポート
製品コード
SH001
製品名
ページュネクタイ
価格
1,200円
発売日
2015/08/05
在庫
21
選択項目の属性が表示される
調整したい項目を選択

図22

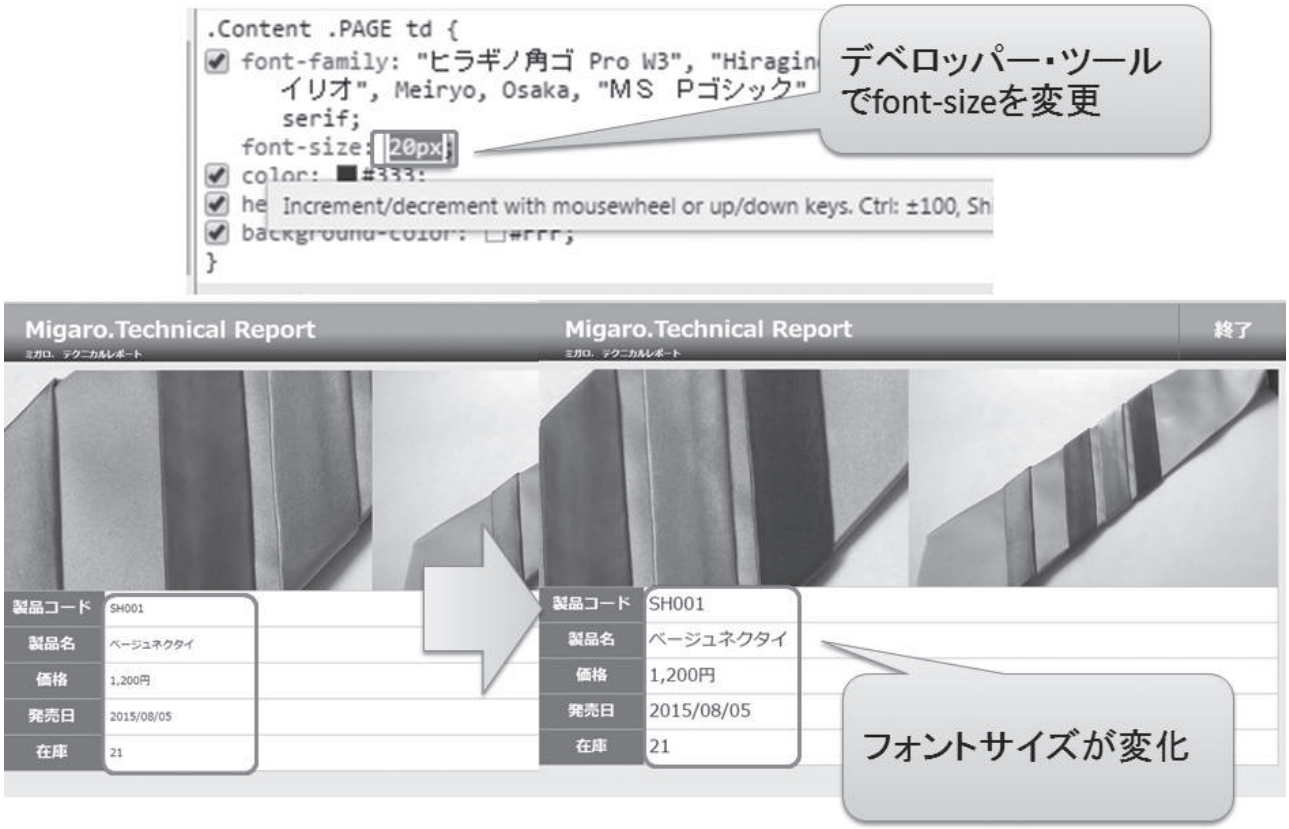

.Content .PAGE td {
font-family: "ヒラギノ角ゴ Pro W3", "Hiragin
イリオ", Meiryo, Osaka, "MS Pゴシック"
serif;
font-size: 20px;
color: #333;
background-color: #fff;
}
Increment/decrement with mousewheel or up/down keys. Ctrl: ±100, Sh
デベロッパー・ツール
でfont-sizeを変更
Migaro.Technical Report
ミガロ. テクニカルレポート
終了
製品コード
SH001
製品名
ページュネクタイ
価格
1,200円
発売日
2015/08/05
在庫
21
フォントサイズが変化

図23

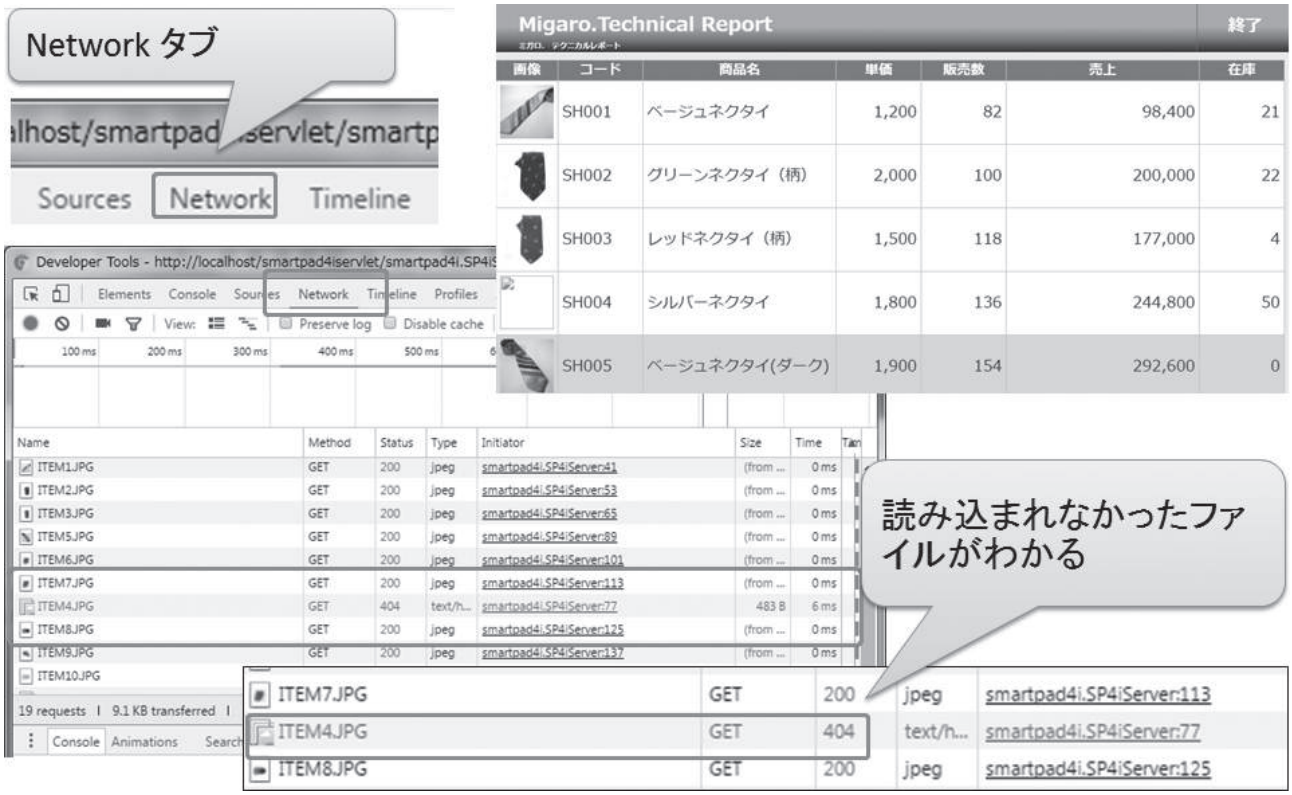
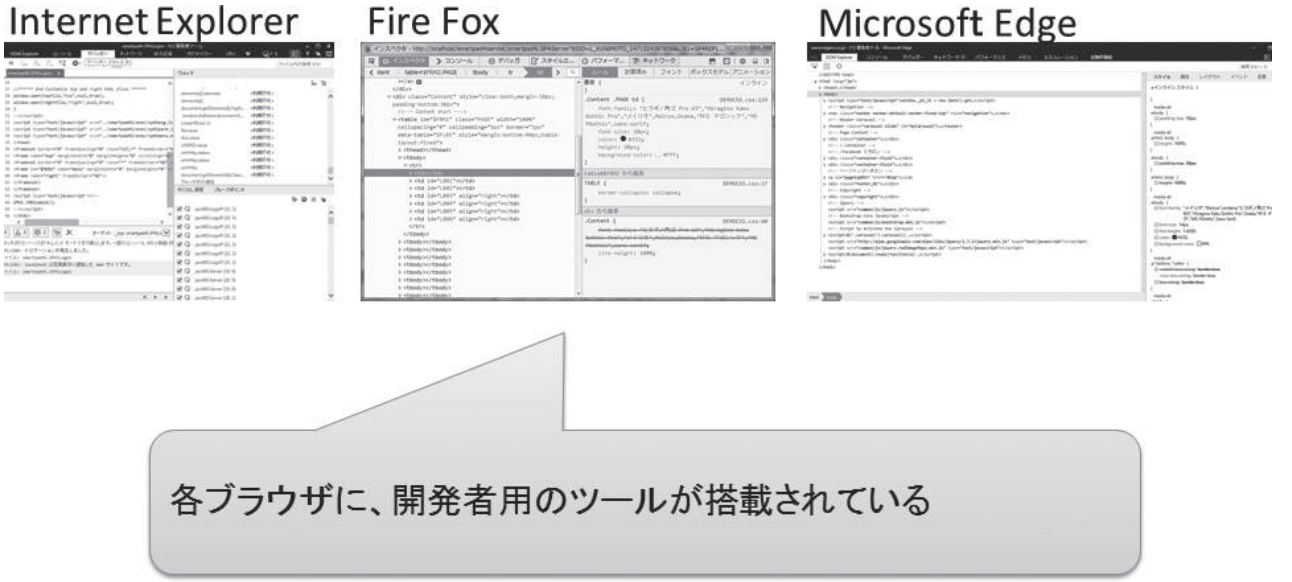

Network タブ
lhost/smartpad servlet/smartp
Sources
Network
Timeline
Migaro.Technical Report
終了
画像 コード 商品名 単価 販売数 売上 在庫
SH001 ベージュネクタイ 1,200 82 98,400 21
SH002 グリーンネクタイ（柄） 2,000 100 200,000 22
SH003 レッドネクタイ（柄） 1,500 118 177,000 4
SH004 シルバーネクタイ 1,800 136 244,800 50
SH005 ベージュネクタイ(ダーク) 1,900 154 292,600 0
Developer Tools - http://localhost/smartpad4iservlet/smartpad4i.SP4i
Elements Console Sources Network Timeline Profiles
Preserve log Disable cache
100 ms 200 ms 300 ms 400 ms 500 ms
Name Method Status Type Initiator Size Time
ITEM1JPG GET 200 jpeg smartpad4i.SP4iServer:41 (from ... 0 ms
ITEM2JPG GET 200 jpeg smartpad4i.SP4iServer:53 (from ... 0 ms
ITEM3JPG GET 200 jpeg smartpad4i.SP4iServer:65 (from ... 0 ms
ITEM5JPG GET 200 jpeg smartpad4i.SP4iServer:89 (from ... 0 ms
ITEM6JPG GET 200 jpeg smartpad4i.SP4iServer:101 (from ... 0 ms
ITEM7JPG GET 200 jpeg smartpad4i.SP4iServer:113 (from ... 0 ms
ITEM4JPG GET 404 text/h... smartpad4i.SP4iServer:77 483 B 6 ms
ITEM8JPG GET 200 jpeg smartpad4i.SP4iServer:125 (from ... 0 ms
ITEM9JPG GET 200 jpeg smartpad4i.SP4iServer:137 (from ... 0 ms
ITEM10.JPG
19 requests 9.1 KB transferred
Console Animations Search
読み込まれなかったファイルがわかる
ITEM7JPG GET 200 jpeg smartpad4i.SP4iServer:113
ITEM4JPG GET 404 text/h... smartpad4i.SP4iServer:77
ITEM8JPG GET 200 jpeg smartpad4i.SP4iServer:125

図24


Internet Explorer
Fire Fox
Microsoft Edge
各ブラウザに、開発者用のツールが搭載されている