(19)日本国特許庁(JP)　(12)公開特許公報(A)　(11)特許出願公開番号
特開2016-18552
(P2016-18552A)
(43)公開日　平成28年2月1日(2016.2.1)

(51)Int.Cl.			F I			テーマコード（参考）
G06F	***9/45***	***(2006.01)***	G06F	9/44	320A	5B081
G06F	***9/50***	***(2006.01)***	G06F	9/06	640H	5B376

審査請求 未請求　請求項の数 4　ＯＬ　（全 9 頁）

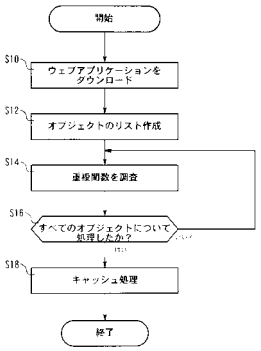
(21)出願番号　特願2014-168917 (P2014-168917)
(22)出願日　平成26年8月22日 (2014.8.22)
(62)分割の表示　特願2014-138438 (P2014-138438)の分割
原出願日　平成26年7月4日 (2014.7.4)

(71)出願人　599115217
株式会社　ディー・エヌ・エー
東京都渋谷区渋谷二丁目２１番１号
(74)代理人　110001210
特許業務法人ＹＫＩ国際特許事務所
(72)発明者　坊野　博典
東京都渋谷区渋谷二丁目２１番１号　株式会社　ディー・エヌ・エー内

Ｆターム(参考)　5B081 AA09
5B376 AC13　BC57　DA14　EA21　FA10

(54)【発明の名称】スクリプトのキャッシュ方法及びそれを適用した情報処理装置

(57)【要約】
【課題】スクリプトにより記述されたウェブアプリケーションを処理する際の負担を軽減する。
【解決手段】ウェブアプリケーションにおける所定の処理単位毎に処理されるオブジェクトを記述するスクリプトの１つを着目スクリプトとして、着目スクリプトを内部コードに変換した場合に含まれる関数名と、残りのオブジェクトを記述するスクリプトを内部コードに変換した場合に含まれる関数名と、が共通でない場合には着目スクリプトの内部コードをキャッシュする。
【選択図】図４



【特許請求の範囲】
【請求項１】
少なくとも一部の機能がスクリプトにより記述されたウェブアプリケーションを対象として、前記ウェブアプリケーションにおける所定の処理単位毎に、当該処理単位において処理されるオブジェクトを記述するスクリプトの１つを着目スクリプトとして、前記着目スクリプトを内部コードに変換した場合に当該内部コードに含まれる関数名と、当該処理単位において処理される残りのオブジェクトを記述するスクリプトを内部コードに変換した場合に当該内部コードに含まれる関数名と、が共通でない場合には前記着目スクリプトの内部コードをキャッシュすることを特徴とするスクリプトのキャッシュ方法。
【請求項２】
請求項１に記載のスクリプトのキャッシュ方法であって、
前記処理単位は、ブラウザに表示されるウェブページ単位であることを特徴とするスクリプトのキャッシュ方法。
【請求項３】
請求項１又は２に記載のスクリプトのキャッシュ方法であって、
前記処理単位毎に、当該単位に含まれるオブジェクトのリストを生成し、
前記リストを参照して、当該処理単位において処理されるオブジェクトを記述するスクリプトの１つを着目スクリプトとして、前記着目スクリプトを内部コードに変換した場合に当該内部コードに含まれる関数名と、当該処理単位において処理される残りのオブジェクトを記述するスクリプトを内部コードに変換した場合に当該内部コードに含まれる関数名と、が共通でない場合には前記着目スクリプトの内部コードをキャッシュすることを特徴とするスクリプトのキャッシュ方法。
【請求項４】
少なくとも一部の機能がスクリプトにより記述されたウェブアプリケーションを対象として、前記ウェブアプリケーションにおける所定の処理単位毎に、当該処理単位において処理されるオブジェクトを記述するスクリプトの１つを着目スクリプトとして、前記着目スクリプトを内部コードに変換した場合に当該内部コードに含まれる関数名と、当該処理単位において処理される残りのオブジェクトを記述するスクリプトを内部コードに変換した場合に当該内部コードに含まれる関数名と、が共通でない場合には前記着目スクリプトの内部コードをキャッシュすることを特徴とする情報処理装置。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、ウェブアプリケーションにおけるスクリプトのキャッシュ方法及びそれを適用した情報処理装置に関する。
【背景技術】
【０００２】
サーバからスクリプトにより記述されたウェブアプリケーションをダウンロードし、クライアントがスクリプトを解析及び実行するシステムが用いられている。このようなシステムでは、クライアントをインターネット等のネットワークに接続してアプリケーションを記述するスクリプトをダウンロードする必要があり、ネットワークの通信負荷が増加するおそれがある。また、クライアントではダウンロードされたスクリプトをその都度解析する必要があり、クライアントでの処理負担が増加するおそれがある。
【０００３】
そこで、ウェブページのｈｔｍｌファイルに管理スクリプトを埋め込み、ウェブページの読み込みの際にクライアントの記憶領域に実行対象となるスクリプトが存在する場合には当該記憶領域から読み出して実行し、そうでない場合にはサーバからスクリプトを取得して実行するスクリプトのキャッシュ技術が開示されている（特許文献１）。
【先行技術文献】
【特許文献】

【０００４】
【特許文献１】特許第５２３１５００号公報
【発明の概要】
【発明が解決しようとする課題】
【０００５】
ところで、スクリプトをキャッシュする技術によって、ダウンロードに伴うネットワークの通信負荷及びクライアントにおける処理負荷を低減することができるが、クライアントにおけるスクリプトの解析処理の負荷を低減することができない。
【０００６】
本発明は、スクリプトにより記述されたウェブアプリケーションを処理する際の負担を軽減することを可能とするスクリプトのキャッシュ方法及びそれを適用した情報処理装置を提供することを目的とする。
【課題を解決するための手段】
【０００７】
本発明の１つの態様は、少なくとも一部の機能がスクリプトにより記述されたウェブアプリケーションを対象として、前記ウェブアプリケーションにおける所定の処理単位毎に、当該処理単位において処理されるオブジェクトを記述するスクリプトの１つを着目スクリプトとして、前記着目スクリプトを内部コードに変換した場合に当該内部コードに含まれる関数名と、当該処理単位において処理される残りのオブジェクトを記述するスクリプトを内部コードに変換した場合に当該内部コードに含まれる関数名と、が共通でない場合には前記着目スクリプトの内部コードをキャッシュすることを特徴とするスクリプトのキャッシュ方法である。
【０００８】
本発明の別の態様は、少なくとも一部の機能がスクリプトにより記述されたウェブアプリケーションを対象として、前記ウェブアプリケーションにおける所定の処理単位毎に、当該処理単位において処理されるオブジェクトを記述するスクリプトの１つを着目スクリプトとして、前記着目スクリプトを内部コードに変換した場合に当該内部コードに含まれる関数名と、当該処理単位において処理される残りのオブジェクトを記述するスクリプトを内部コードに変換した場合に当該内部コードに含まれる関数名と、が共通でない場合には前記着目スクリプトの内部コードをキャッシュすることを特徴とする情報処理装置である。
【０００９】
ここで、前記処理単位は、ブラウザに表示されるウェブページ単位であることが好適である。
【００１０】
また、前記処理単位毎に、当該単位に含まれるオブジェクトのリストを生成し、前記リストを参照して、当該処理単位において処理されるオブジェクトを記述するスクリプトの１つを着目スクリプトとして、前記着目スクリプトを内部コードに変換した場合に当該内部コードに含まれる関数名と、当該処理単位において処理される残りのオブジェクトを記述するスクリプトを内部コードに変換した場合に当該内部コードに含まれる関数名と、が共通でない場合には前記着目スクリプトの内部コードをキャッシュすることが好適である。
【発明の効果】
【００１１】
本発明によれば、スクリプトにより記述されたウェブアプリケーションを処理する際の負担を軽減することができる。
【図面の簡単な説明】
【００１２】
【図１】本発明の実施の形態における情報処理システムの構成を示す図である。
【図２】本発明の実施の形態における情報端末の構成を示す図である。

【図３】本発明の実施の形態におけるサーバの構成を示す図である。
【図４】本発明の実施の形態におけるスクリプトのキャッシュ方法を説明するフローチャートである。
【図５】本発明の実施の形態におけるオブジェクトのリストを作成するためのプログラムの例を示す図である。
【図６】本発明の実施の形態におけるオブジェクトのリストの例を示す図である。
【図７】本発明の実施の形態におけるスクリプトにより記述されたオブジェクト（リスト）の例を示す図である。
【図８】本発明の実施の形態におけるオブジェクト（リスト）を内部コードに変換した例を示す図である。
【発明を実施するための形態】
【００１３】
＜情報処理システムの構成＞
本発明の実施の形態における情報処理システムは、図１に示すように、情報端末１００及びサーバ１０２を含んで構成される。サーバ１０２は、インターネット等の情報通信網１０４を介して、情報端末１００と通信可能に接続される。サーバ１０２は、情報端末１００からのアクセスを受けて、ウェブアプリケーションを情報端末１００に送信する。情報端末１００は、サーバ１０２からウェブアプリケーションを受信し、当該ウェブアプリケーションを解析及び実行する。
【００１４】
情報端末１００は、図２に示すように、処理部１０、記憶部１２、入力部１４、出力部１６及び通信部１８を含んで構成される。情報端末１００は、携帯電話、スマートフォン、タブレット端末等の通信可能な携帯端末の基本構成を備えている。
【００１５】
処理部１０は、ＣＰＵ等の演算処理を行う手段を含む。処理部１０は、ウェブアプリケーションプログラムを実行する。ウェブアプリケーションは、サーバ１０２からダウンロードされる。記憶部１２は、半導体メモリやメモリカード等の記憶手段を含む。記憶部１２は、処理部１０とアクセス可能に接続され、ウェブアプリケーション、その処理に必要なデータ等の情報を記憶する。入力部１４は、情報端末１００に情報を入力する手段を含む。入力部１４は、例えば、ユーザからの入力を受けるタッチパネルやボタン等を備える。出力部１６は、ユーザから入力情報を受け付けるためのユーザインターフェース画面（ＵＩ）等や情報端末１００での処理結果を出力する手段を含む。出力部１６は、例えば、ユーザに対して画像を呈示するディスプレイを備える。通信部１８は、情報通信網１０４を介して、他の情報通信機器と情報をやり取りするためのインターフェースを含んで構成される。通信部１８による通信は有線及び無線を問わない。
【００１６】
サーバ１０２は、図３に示すように、処理部２０、記憶部２２、入力部２４、出力部２６及び通信部２８を含んで構成される。サーバ１０２は、通信機能を備えたコンピュータの基本構成を備えている。
【００１７】
処理部２０は、ＣＰＵ等の演算処理を行う手段を含む。処理部２０は、情報端末１００からのアクセスを受けて、記憶部２２に記憶されているウェブアプリケーションを情報端末１００へ提供する処理を行う。記憶部２２は、半導体メモリ、ハードディスク等の記憶手段を含む。記憶部２２は、処理部２０とアクセス可能に接続され、ウェブアプリケーション、その処理に必要なデータ等の情報を記憶する。入力部２４は、サーバ１０２に情報を入力する手段を含む。入力部２４は、例えば、ユーザからの入力を受けるキーボード等を備える。出力部２６は、ユーザから入力情報を受け付けるためのユーザインターフェース画面（ＵＩ）等やサーバ１０２での処理結果を出力する手段を含む。出力部２６は、例えば、ユーザに対して画像を呈示するディスプレイを備える。通信部２８は、情報通信網１０４を介して、情報端末１００と情報をやり取りするためのインターフェースを含んで

構成される。通信部２８による通信は有線及び無線を問わない。
【００１８】
本実施の形態の情報処理システムで扱われるウェブアプリケーションは、その少なくても一部がスクリプトによって記述されている。スクリプトは、コンピュータプログラムの記述言語の１つであり、機械語への変換や実行可能ファイルの作成などの過程を省略または自動化し、ソースコードを解釈することによって実行できるようなプログラム記述言語を意味する。スクリプトとしては、例えば、Ｊａｖａ　Ｓｃｒｉｐｔ（登録商標）等が挙げられる。
【００１９】
例えば、ＨＴＭＬ等のマークアップ言語で記述されたアプリケーション内に内部コードとしてスクリプトで記述されたプログラムを直接記述することができる。また、ｔｙｐｅ属性及びｓｃｒ属性を用いることによって、外部ファイルからスクリプトで記述されたプログラムを読み込んで実行させることもできる。
【００２０】
スクリプトには、様々な命令が記述できる。記述できる命令は、例えば、情報端末１００の記憶部１２に保持されているデータやスクリプトに記述されているデータを保存する命令、情報端末１００の記憶部１２に記憶されているデータを消去する命令、情報端末１００の記憶部１２に記憶されているデータを表示する命令、データをサーバ１０２等に送信する命令、データに対して四則演算や文字列操作などを実行する命令、実行すべきスクリプトの取得先を指定してそのスクリプトの実行を要求する命令、等が挙げられる。
【００２１】
情報端末１００は、次に実行すべきスクリプトがアプリケーションに記述されていると、そのスクリプトを取得してその中に書いてある命令を解釈して内部コードに変換して実行する。このように、ウェブアプリケーションでは、順に次のスクリプトを取得して、その中に記述されている命令を順次解釈及び実行することによって動作する。サービスの提供者は、スクリプトによって記述されているウェブアプリケーションをサーバ１０２に保持させ、ユーザに対してそのウェブアプリケーションを提供するアドレス（ＵＲＬ）を知らせることによってユーザにサービスを提供する。
【００２２】
＜スクリプトのキャッシュ方法＞
本実施の形態では、情報端末１００からサーバ１０２にアクセスし、サーバ１０２からウェブアプリケーションをダウンロードして情報端末１００において実行する処理を例に説明を行う。以下、情報端末１００でのスクリプトのキャッシュ処理について、図４のフローチャートを参照して説明する。
【００２３】
ステップＳ１０では、ウェブアプリケーションのダウンロードが行われる。このステップでの処理によって、情報端末１００はアプリケーション取得手段として機能し、サーバ１０２はアプリケーション提供手段として機能する。ユーザは、情報端末１００においてウェブブラウザを用いてウェブアプリケーションを取得するためのアドレス（ＵＲＬ）に基づいてウェブアプリケーションを提供するサーバ１０２にアクセスする。これにより、サーバ１０２から情報端末１００へウェブアプリケーションが送信される。情報端末１００は、ウェブアプリケーションを受信し、受信したアプリケーションを記憶部１２に記憶させる。
【００２４】
なお、ブラウザ上において実行される電子ゲーム等のウェブアプリケーションではウェブのページを処理単位として処理が行われることが多い。このような場合、ウェブページ単位でアプリケーションを送信して、以下の処理を適用するようにしてもよい。
【００２５】
ステップＳ１２では、ウェブアプリケーションに含まれるオブジェクトのリストを作成する。このステップでの処理によって、情報端末１００はリスト作成手段として機能する

。情報端末１００は、ステップＳ１０において取得したウェブアプリケーションに含まれるオブジェクトのリストを生成する処理を行う。

【００２６】

通常、Ｊａｖａ　Ｓｃｒｉｐｔ（登録商標）等のスクリプトではクラスはオブジェクトの１つして扱われ、すべてのグローバルオブジェクトをツリー構造（木構造）として管理している。そして、図５に示すようなプログラムを実行することによって、図６に示すように、既知のシステムオブジェクトを除くすべてのオブジェクトをリストとして出力することができる。リストには、オブジェクトのキー（Ｋｅｙ）及び各オブジェクトを情報端末１００において実行される内部コードに変換したときの値（Ｖａｌｕｅ）が含まれる。

【００２７】

処理部１０は、処理対象となっているウェブアプリケーションから各オブジェクトのスクリプトを読み出し、スクリプトエンジンによって解析し、情報端末１００において実行する内部コードに変換する。例えば、図６に示したリストにおける”ｎ．ＭｙＣｌａｓｓ１”及び”ｎ．ＭｙＣｌａｓｓ２”というクラスが図７に示すスクリプトで記述されているとすると、情報端末１００において実行される際にはスクリプトエンジンによって解析されて図８に示すような内部コードに変換される。この内部コードからクラス名がキー（Ｋｅｙ）として抽出され、関数が値（Ｖａｌｕｅ）として抽出される。”ｎ．ＭｙＣｌａｓｓ１”及び”ｎ．ＭｙＣｌａｓｓ２”では、キー（Ｋｅｙ）は”ｎ．ＭｙＣｌａｓｓ１”及び”ｎ．ＭｙＣｌａｓｓ２”となり、値（Ｖａｌｕｅ）はいずれも”ｆｕｎｃｔｉｏｎ　Ｋｌａｓｓ”となる。

【００２８】

ステップＳ１４では、ウェブアプリケーションに内部コードの関数名が共通するスクリプトが存在するか否かが判定される。このステップでの処理によって、情報端末１００は共通関数調査手段として機能する。処理部１０は、ステップＳ１２においてリストアップされたオブジェクトの１つを着目スクリプトとする。そして、着目スクリプトの値（Ｖａｌｕｅ）が関数であるか否かを判定する。さらに、着目スクリプトの値（Ｖａｌｕｅ）が関数であった場合、リスト内の他のオブジェクトの値（Ｖａｌｕｅ）に着目スクリプトの値（Ｖａｌｕｅ）と同じ関数が存在するか否かを判定する。

【００２９】

ステップＳ１６では、リスト内のすべてのオブジェクトについて着目スクリプトとして処理されたか否かが判定される。すべてのオブジェクトについて着目スクリプトとして処理されていなければステップＳ１４に処理を戻し、既にすべてのオブジェクトについて着目スクリプトとして処理されていればステップＳ１８に処理を移行させる。

【００３０】

ステップＳ１８では、ウェブアプリケーションのキャッシュ処理が行われる。このステップの処理によって、情報端末１００はキャッシュ処理手段として機能する。処理部１０は、記憶部１２においてウェブのキャッシュ領域として確保された記憶領域に処理対象となったウェブアプリケーションをキャッシュする。このとき、ステップＳ１４において内部コードの関数名が共通するとされたオブジェクト（クラス）は内部コードに変換せず、関数名が共通しないとされたオブジェクト（クラス）は内部コードに変換してキャッシュする。

【００３１】

このように、ウェブアプリケーションをキャッシュすることによって、再度同じウェブアプリケーションを使用する際にキャッシュ領域から読み出して使用することができる。したがって、ダウンロードに伴うネットワークの通信負荷及び情報端末１００（クライアント）における処理負荷を低減することができる。

【００３２】

また、情報端末１００（クライアント）におけるスクリプトの解析処理の負荷を低減することができる。ここで、内部コードにおける関数名が複数のオブジェクト（クラス）において共通する場合には当該関数名で表わされる関数が呼び出されたときにいずれのオブ

ジェクト（クラス）の関数が呼び出されたかを文脈を解釈して実行する必要があるが、本実施の形態では、関数名が共通するオブジェクトについては内部コードに変換されずにキャッシュされるのでこのような不具合を避けることができる。

【００３３】

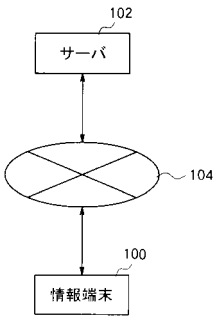
なお、情報端末１００における処理の一部をサーバ１０２において行うようにしてもよい。例えば、ステップＳ１２～Ｓ１６の処理の一部をサーバ１０２において行ってもよい。

【符号の説明】

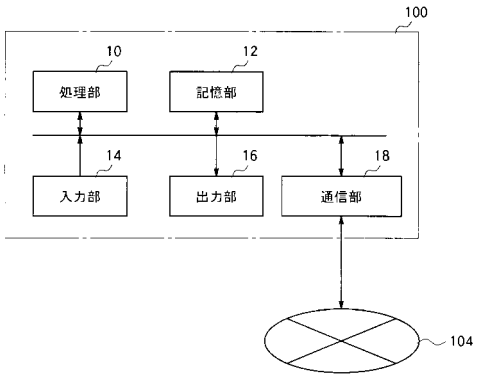
【００３４】

１０　処理部、１２　記憶部、１４　入力部、１６　出力部、１８　通信部、２０　処理部、２２　記憶部、２４　入力部、２６　出力部、２８　通信部、１００　情報端末、１０２　サーバ、１０４　情報通信網。

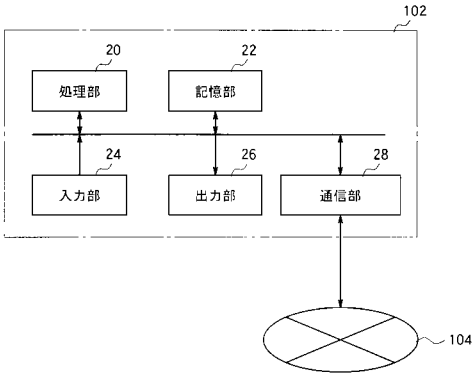
【図１】



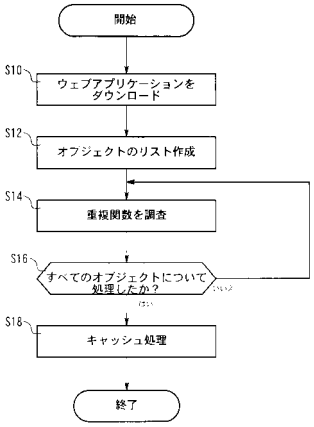
【図２】



【図３】



【図４】



【図５】

```
var n = {};
function traverse(objects, path, data) {
  var length = objects.length;
  var object = objects[length - 1];
  for (var key in object) {
    var value = object[key];
    var type = typeof value;
    var name = path + '.' + key;
    if (value == null) {
      continue;
    }
    if (type == 'object') {
      if (isSystemObject(value)) {
        continue;
      }
      if (value instanceof Array) {
        data.push({ 'key': name, 'value': value });
        continue;
      } else {
        var loop = false;
        for (var i = 0; i < length; ++i) {
          if (objects[i] === value) {
            loop = true;
            break;
          }
        }
        if (!loop) {
          objects.push(value);
          traverse(objects, name, data);
          objects.pop();
        }
      }
    } else if (type != 'undefined') {
      data.push({ 'key': name, 'value': value });
    }
  }
};

var data = [];
traverse([n], 'n' , data);
```

【図６】

```
var data0 = [];
traverse([n], 'n', data0);
// data0 = [
//   {
//     'key': 'n.MyClass1',
//     'value': function klass() {...}
//   },
//   {
//     'key': 'n.MyClass2',
//     'value': function klass() {...}
//   },
//   {
//     'key': 'n.myData.scene',
//     'value': 1
//   },
//   {
//     'key': 'n.myData.enemy',
//     'value': []
//   },
//   ...
// ];
```

【図７】

```
var n = {};
n.MyClass1 = Class.create({
  value: 0,
  init: function(value) {
    this.value = value;
  }
});

var n = {};
n.MyClass2 = Class.create({
  value: 0,
  init: function(value) {
    this.value = value;
  }
});
```

【図８】

```
var n = {};
(function() {
n.MyClass1 = function klass(value) {
  this.init.apply(this, arguments);
};
n.MyClass1.prototype.value = 0;
n.MyClass1.prototype.init =
    function(value) {
  this.value = value;
};
})();

var n = {};
(function() {
n.MyClass2 = function klass(value) {
  this.init.apply(this, arguments);
};
n.MyClass2.prototype.value = 0;
n.MyClass2.prototype.init =
    function(value) {
  this.value = value;
};
})();
```