(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第5989248号
(P5989248)
(45)発行日　平成28年9月7日(2016.9.7)　(24)登録日　平成28年8月19日(2016.8.19)

(51)Int.Cl.　F I
G06F 13/00 (2006.01)　G06F 13/00 550A

請求項の数 15 (全 53 頁)

(21)出願番号　特願2015-527458(P2015-527458)
(86)(22)出願日　平成25年6月17日(2013.6.17)
(65)公表番号　特表2015-531925(P2015-531925A)
(43)公表日　平成27年11月5日(2015.11.5)
(86)国際出願番号　PCT/US2013/046099
(87)国際公開番号　WO2014/028115
(87)国際公開日　平成26年2月20日(2014.2.20)
審査請求日　平成28年6月16日(2016.6.16)
(31)優先権主張番号　61/683,999
(32)優先日　平成24年8月16日(2012.8.16)
(33)優先権主張国　米国(US)
(31)優先権主張番号　61/684,594
(32)優先日　平成24年8月17日(2012.8.17)
(33)優先権主張国　米国(US)

(73)特許権者　507364838
クアルコム，インコーポレイテッド
アメリカ合衆国　カリフォルニア　９２１２１　サン　ディエゴ　モアハウス　ドライブ　５７７５
(74)代理人　100108453
弁理士　村山　靖彦
(74)代理人　100163522
弁理士　黒田　晋平
(72)発明者　マイケル・ウェーバー
アメリカ合衆国・カリフォルニア・９２１２１・サン・ディエゴ・モアハウス・ドライヴ・５７７５

最終頁に続く

(54)【発明の名称】ウェブブラウザにおけるスクリプトの事前処理

(57)【特許請求の範囲】
【請求項１】
HTMLドキュメントに含まれるスクリプトを準備する方法であって、
前記HTMLドキュメントをスキャンして複数のスクリプトを発見するステップと、
前記複数のスクリプトを実行に備えてスクリプトエンジンに送るステップと、
前記スクリプトエンジンが実行のために前記複数のスクリプトを準備している間に前記HTMLドキュメントを解析するステップと、
実行されるべき次のスクリプトを前記複数のスクリプトから特定するステップと、
実行されるべき前記特定された次のスクリプトに対応する情報を前記スクリプトエンジンに送るステップと、
前記HTMLドキュメントの前記解析を中断するステップと、
実行されるべき前記特定された次のスクリプトが実行されたことを示す通知を受け取るステップと、
前記通知を受け取ったことに応答して前記HTMLドキュメントの前記解析を再開するステップと
を含む、方法。
【請求項２】
実行されるべき前記特定された次のスクリプトに対応する情報を前記スクリプトエンジンに送るステップが、実行されるべき前記特定された次のスクリプトを前記スクリプトエンジンに送るステップを含む、請求項1に記載の方法。

【請求項３】
前記複数のスクリプトの各々の識別子を生成するステップをさらに含み、
前記複数のスクリプトをスクリプトエンジンに送るステップが、前記複数のスクリプトおよび識別子を前記スクリプトエンジンに送るステップを含み、
実行されるべき前記特定された次のスクリプトに対応する情報を前記スクリプトエンジンに送るステップが、実行されるべき前記次のスクリプトの前記識別子を前記スクリプトエンジンに送るステップを含み、
前記複数のスクリプトの各々の識別子を生成するステップが、少なくとも1つのスクリプトをuniform resource identifier (URI)と関連付けるステップを含むか、又は、
前記複数のスクリプトの各々の識別子を生成するステップが、少なくとも1つのスクリプトのための署名を生成するステップを含むか、又は、
前記複数のスクリプトの各々の識別子を生成するステップが、少なくとも1つのスクリプトのテキストを含む少なくとも1つの識別子を生成するステップを含む、請求項1に記載の方法。
【請求項４】
HTMLドキュメントをスキャンして複数のスクリプトを発見するステップが、第1のプロセッサにおいて前記HTMLドキュメントをスキャンするステップを含み、
前記スクリプトエンジンが実行のために前記複数のスクリプトを準備している間に前記HTMLドキュメントを解析するステップが、第2のプロセッサにおいて前記HTMLドキュメントを解析するステップを含む、請求項1に記載の方法。
【請求項５】
HTMLドキュメントをスキャンして複数のスクリプトを発見するステップが、プロセッサにおいて実行される第1のプロセスによって前記HTMLドキュメントをスキャンするステップを含み、
前記スクリプトエンジンが実行のために前記複数のスクリプトを準備している間に前記HTMLドキュメントを解析するステップが、前記プロセッサにおいて実行される第2のプロセスによって前記HTMLドキュメントを解析するステップを含み、
前記スクリプトエンジンが実行のために前記複数のスクリプトを準備している間に前記HTMLドキュメントを解析するステップが、前記スクリプトエンジンが第2のスクリプトを解析し、分析し、コンパイルするのと並列に、前記スクリプトエンジンが第1のスクリプトを解析し、分析し、コンパイルしている間に、前記HTMLドキュメントを解析するステップを含む、請求項1に記載の方法。
【請求項６】
前記スクリプトエンジンが実行のために前記複数のスクリプトを準備している間に前記HTMLドキュメントを解析するステップが、前記複数のスクリプトが実行される実行順序とは異なる準備順序で、前記スクリプトエンジンが実行のために前記複数のスクリプトを準備している間に、前記HTMLドキュメントを解析するステップを含む、請求項1に記載の方法。
【請求項７】
実行されるべき次のスクリプトを前記複数のスクリプトから特定するステップが、定義された実行順序に基づいて、実行されるべき前記次のスクリプトを特定するステップを含む、請求項1に記載の方法。
【請求項８】
HTMLドキュメントをスキャンして複数のスクリプトを発見するための手段と、
前記複数のスクリプトを実行に備えてスクリプトエンジンに送るための手段と、
前記スクリプトエンジンが実行のために前記複数のスクリプトを準備している間に前記HTMLドキュメントを解析するための手段と、
実行されるべき次のスクリプトを前記複数のスクリプトから特定するための手段と、
実行されるべき前記特定された次のスクリプトに対応する情報を前記スクリプトエンジンに送るための手段と、

前記HTMLドキュメントの前記解析を中断するための手段と、
実行されるべき前記特定された次のスクリプトが実行されたことを示す通知を受け取るための手段と、
前記通知を受け取ったことに応答して前記HTMLドキュメントの前記解析を再開するための手段と
を含む、コンピューティングデバイス。
【請求項９】
実行されるべき前記特定された次のスクリプトに対応する情報を前記スクリプトエンジンに送るための手段が、実行されるべき前記特定された次のスクリプトを前記スクリプトエンジンに送るための手段を含む、請求項8に記載のコンピューティングデバイス。
【請求項１０】
前記複数のスクリプトの各々の識別子を生成するための手段をさらに含み、
前記複数のスクリプトをスクリプトエンジンに送るための手段が、前記複数のスクリプトおよび識別子を前記スクリプトエンジンに送るための手段を含み、
実行されるべき前記特定された次のスクリプトに対応する情報を前記スクリプトエンジンに送るための手段が、実行されるべき前記次のスクリプトの前記識別子を前記スクリプトエンジンに送るための手段を含み、
前記複数のスクリプトの各々の識別子を生成するための手段が、少なくとも1つのスクリプトをuniform resource identifier (URI)と関連付けるための手段を含むか、又は、
前記複数のスクリプトの各々の識別子を生成するための手段が、少なくとも1つのスクリプトのための署名を生成するための手段を含むか、又は、
前記複数のスクリプトの各々の識別子を生成するための手段が、少なくとも1つのスクリプトのテキストを含む少なくとも1つの識別子を生成するための手段を含む、請求項8に記載のコンピューティングデバイス。
【請求項１１】
HTMLドキュメントをスキャンして複数のスクリプトを発見するための手段が、第1のプロセッサにおいて前記HTMLドキュメントをスキャンするための手段を含み、
前記スクリプトエンジンが実行のために前記複数のスクリプトを準備している間に前記HTMLドキュメントを解析するための手段が、第2のプロセッサにおいて前記HTMLドキュメントを解析するための手段を含む、請求項8に記載のコンピューティングデバイス。
【請求項１２】
HTMLドキュメントをスキャンして複数のスクリプトを発見するための手段が、プロセッサにおいて実行される第1のプロセスによって前記HTMLドキュメントをスキャンするための手段を含み、
前記スクリプトエンジンが実行のために前記複数のスクリプトを準備している間に前記HTMLドキュメントを解析するための手段が、前記プロセッサにおいて実行される第2のプロセスによって前記HTMLドキュメントを解析するための手段を含み、
前記スクリプトエンジンが実行のために前記複数のスクリプトを準備している間に前記HTMLドキュメントを解析するための手段が、前記スクリプトエンジンが第2のスクリプトを解析し、分析し、コンパイルするのと並列に、前記スクリプトエンジンが第1のスクリプトを解析し、分析し、コンパイルしている間に、前記HTMLドキュメントを解析するための手段を含む、請求項8に記載のコンピューティングデバイス。
【請求項１３】
前記スクリプトエンジンが実行のために前記複数のスクリプトを準備している間に前記HTMLドキュメントを解析するための手段が、前記複数のスクリプトが実行される実行順序とは異なる準備順序で、前記スクリプトエンジンが実行のために前記複数のスクリプトを準備している間に、前記HTMLドキュメントを解析するための手段を含むか、又は、
実行されるべき次のスクリプトを前記複数のスクリプトから特定するための手段が、定義された実行順序に基づいて、実行されるべき前記次のスクリプトを特定するための手段を含む、請求項8に記載のコンピューティングデバイス。

【請求項１４】
前記各手段がプロセッサ実行可能命令によって構成されるプロセッサで実現される、請求項8乃至13のいずれか1項に記載のコンピューティングデバイス。
【請求項１５】
請求項1乃至7のいずれか1項に記載の方法のステップをプロセッサに実行させるように構成されるプロセッサ実行可能ソフトウェア命令を記憶した非一時的コンピュータ可読記憶媒体。
【発明の詳細な説明】
【技術分野】
【０００１】
関連特許出願
本出願は、2012年8月17日に出願された「Pre-Processing of Scripts in Web Browsers」という表題の米国仮特許出願第61/684,594号および2012年8月16日に出願された「Pre-Processing of Scripts in Web Browsers」という表題の米国仮特許出願第61/683,999号の優先権の利益を主張し、上記の仮出願の両方の内容全体が参照によって本明細書に組み込まれる。
【０００２】
本発明は、ウェブブラウザにおいてHTMLドキュメントをレンダリングするための方法、システム、およびデバイスに関し、より具体的には、ウェブブラウザ動作を並列化する方法に関する。
【背景技術】
【０００３】
ワイヤレス通信技術およびモバイル電子デバイス(たとえば、携帯電話、タブレット、ラップトップなど)は、過去数年にわたって人気および使用率が上がっている。増大する消費者の要求についていくために、モバイル電子デバイスは、より機能が豊富になり、今や一般的に、複数のプロセッサ、システムオンチップ(SoC)、およびモバイルデバイスのユーザが自分のモバイルデバイスにおいて複雑で処理負荷の大きいソフトウェアアプリケーション(たとえば、ウェブブラウザ、ビデオストリーミングアプリケーションなど)を実行することを可能にする他の要素を含む。これらの改善および他の改善により、スマートフォンおよびタブレットコンピュータは人気が高まり、多くのユーザが選択するプラットフォームとして、ラップトップおよびデスクトップマシンにとって代わっている。
【発明の概要】
【発明が解決しようとする課題】
【０００４】
モバイルデバイスのユーザは今や、自分のモバイルデバイス上のブラウザアプリケーションを介してインターネットにアクセスすることによって、簡単かつ便利に、多くの毎日の作業を遂行することができる。モバイルデバイスの人気が高まり続けているので、現在のモバイルデバイスの多重処理能力をより良好に利用することが可能なウェブブラウザが、消費者にとって望ましい。
【課題を解決するための手段】
【０００５】
様々な態様は、HTMLドキュメントに含まれるスクリプトを準備する方法を含み、この方法は、HTMLドキュメントをスキャンして複数のスクリプトを発見するステップと、複数のスクリプトをスクリプトエンジンに送るステップと、スクリプトエンジンが実行のために複数のスクリプトを準備している間にHTMLドキュメントを解析するステップと、実行されるべき次のスクリプトを複数のスクリプトから特定するステップと、実行されるべき特定された次のスクリプトに対応する情報をスクリプトエンジンに送るステップと、HTMLドキュメントの解析を中断するステップと、実行されるべき特定された次のスクリプトが実行されたことを示す通知を受け取るステップと、通知を受け取ったことに応答してHTMLドキュメントの解析を再開するステップとを含み得る。ある態様では、実行されるべき特定さ

れた次のスクリプトに対応する情報をスクリプトエンジンに送るステップは、実行されるべき特定された次のスクリプトをスクリプトエンジンに送るステップを含み得る。

【０００６】

ある態様では、方法は、複数のスクリプトの各々の識別子を生成するステップを含み得る。さらなる態様では、複数のスクリプトをスクリプトエンジンに送るステップは、複数のスクリプトおよび識別子をスクリプトエンジンに送るステップを含んでよく、実行されるべき特定された次のスクリプトに対応する情報をスクリプトエンジンに送るステップは、実行されるべき次のスクリプトの識別子をスクリプトエンジンに送るステップを含んでよい。さらなる態様では、複数のスクリプトの各々の識別子を生成するステップは、少なくとも1つのスクリプトをuniform resource identifier (URI)と関連付けるステップを含み得る。さらなる態様では、複数のスクリプトの各々の識別子を生成するステップは、少なくとも1つのスクリプトのための署名を生成するステップを含み得る。さらなる態様では、複数のスクリプトの各々の識別子を生成するステップは、少なくとも1つのスクリプトのテキストを含み得る少なくとも1つの識別子を生成するステップを含み得る。

【０００７】

さらなる態様では、HTMLドキュメントをスキャンして複数のスクリプトを発見するステップは、第1のプロセッサにおいてHTMLドキュメントをスキャンするステップを含んでよく、スクリプトエンジンが実行のために複数のスクリプトを準備している間にHTMLドキュメントを解析するステップは、第2のプロセッサにおいてHTMLドキュメントを解析するステップを含んでよい。さらなる態様では、HTMLドキュメントをスキャンして複数のスクリプトを発見するステップは、プロセッサにおいて実行される第1のプロセスによってHTMLドキュメントをスキャンするステップを含んでよく、スクリプトエンジンが実行のために複数のスクリプトを準備している間にHTMLドキュメントを解析するステップは、プロセッサにおいて実行される第2のプロセスによってHTMLドキュメントを解析するステップを含んでよい。

【０００８】

さらなる態様では、スクリプトエンジンが実行のために複数のスクリプトを準備している間にHTMLドキュメントを解析するステップは、スクリプトエンジンが第2のスクリプトを解析し、分析し、コンパイルするのと並列に、スクリプトエンジンが第1のスクリプトを解析し、分析し、コンパイルしている間に、HTMLドキュメントを解析するステップを含み得る。さらなる態様では、スクリプトエンジンが実行のために複数のスクリプトを準備している間にHTMLドキュメントを解析するステップは、複数のスクリプトが実行される実行順序とは異なる準備順序で、スクリプトエンジンが実行のために複数のスクリプトを準備している間に、HTMLドキュメントを解析するステップを含み得る。さらなる態様では、実行されるべき次のスクリプトを複数のスクリプトから特定するステップは、定義された実行順序に基づいて、実行されるべき次のスクリプトを特定するステップを含み得る。

【０００９】

さらなる態様はコンピューティングデバイスを含み、このコンピューティングデバイスは、HTMLドキュメントをスキャンして複数のスクリプトを発見するための手段と、複数のスクリプトをスクリプトエンジンに送るための手段と、スクリプトエンジンが実行のために複数のスクリプトを準備している間にHTMLドキュメントを解析するための手段と、実行されるべき次のスクリプトを複数のスクリプトから特定するための手段と、実行されるべき特定された次のスクリプトに対応する情報をスクリプトエンジンに送るための手段と、HTMLドキュメントの解析を中断するための手段と、実行されるべき特定された次のスクリプトが実行されたことを示す通知を受け取るための手段と、通知を受け取ったことに応答してHTMLドキュメントの解析を再開するための手段とを含み得る。

【００１０】

ある態様では、実行されるべき特定された次のスクリプトに対応する情報をスクリプトエンジンに送るための手段は、実行されるべき特定された次のスクリプトをスクリプトエンジンに送るための手段を含み得る。さらなる態様では、コンピューティングデバイスは

、複数のスクリプトの各々の識別子を生成するための手段を含み得る。さらなる態様では、複数のスクリプトをスクリプトエンジンに送るための手段は、複数のスクリプトおよび識別子をスクリプトエンジンに送るための手段を含んでよく、実行されるべき特定された次のスクリプトに対応する情報をスクリプトエンジンに送るための手段は、実行されるべき次のスクリプトの識別子をスクリプトエンジンに送るための手段を含んでよい。さらなる態様では、複数のスクリプトの各々の識別子を生成するための手段は、少なくとも1つのスクリプトをuniform resource identifier (URI)と関連付けるための手段を含み得る。さらなる態様では、複数のスクリプトの各々の識別子を生成するための手段は、少なくとも1つのスクリプトのための署名を生成するための手段を含み得る。さらなる態様では、複数のスクリプトの各々の識別子を生成するための手段は、少なくとも1つのスクリプトのテキストを含み得る少なくとも1つの識別子を生成するための手段を含み得る。

【００１１】

さらなる態様では、HTMLドキュメントをスキャンして複数のスクリプトを発見するための手段は、第1のプロセッサにおいてHTMLドキュメントをスキャンするための手段を含んでよく、スクリプトエンジンが実行のために複数のスクリプトを準備している間にHTMLドキュメントを解析するための手段は、第2のプロセッサにおいてHTMLドキュメントを解析するための手段を含んでよい。さらなる態様では、HTMLドキュメントをスキャンして複数のスクリプトを発見するための手段は、プロセッサにおいて実行される第1のプロセスによってHTMLドキュメントをスキャンするための手段を含んでよく、スクリプトエンジンが実行のために複数のスクリプトを準備している間にHTMLドキュメントを解析するための手段は、プロセッサにおいて実行される第2のプロセスによってHTMLドキュメントを解析するための手段を含んでよい。

【００１２】

さらなる態様では、スクリプトエンジンが実行のために複数のスクリプトを準備している間にHTMLドキュメントを解析するための手段は、スクリプトエンジンが第2のスクリプトを解析し、分析し、コンパイルするのと並列に、スクリプトエンジンが第1のスクリプトを解析し、分析し、コンパイルしている間に、HTMLドキュメントを解析するための手段を含み得る。さらなる態様では、スクリプトエンジンが実行のために複数のスクリプトを準備している間にHTMLドキュメントを解析するための手段は、複数のスクリプトが実行される実行順序とは異なる準備順序で、スクリプトエンジンが実行のために複数のスクリプトを準備している間に、HTMLドキュメントを解析するための手段を含み得る。さらなる態様では、実行されるべき次のスクリプトを複数のスクリプトから特定するための手段は、定義された実行順序に基づいて、実行されるべき次のスクリプトを特定するための手段を含み得る。

【００１３】

さらなる態様は、動作を実行するようにプロセッサ実行可能命令によって構成されるプロセッサを含み得るコンピューティングデバイスを含み、この動作は、HTMLドキュメントをスキャンして複数のスクリプトを発見するステップと、複数のスクリプトをスクリプトエンジンに送るステップと、スクリプトエンジンが実行のために複数のスクリプトを準備している間にHTMLドキュメントを解析するステップと、実行されるべき次のスクリプトを複数のスクリプトから特定するステップと、実行されるべき特定された次のスクリプトに対応する情報をスクリプトエンジンに送るステップと、HTMLドキュメントの解析を中断するステップと、実行されるべき特定された次のスクリプトが実行されたことを示す通知を受け取るステップと、通知を受け取ったことに応答してHTMLドキュメントの解析を再開するステップとを含み得る。

【００１４】

ある態様では、実行されるべき特定された次のスクリプトに対応する情報をスクリプトエンジンに送るステップが、実行されるべき特定された次のスクリプトをスクリプトエンジンに送るステップを含み得るように、動作を実行するように、プロセッサがプロセッサ実行可能命令によって構成され得る。さらなる態様では、プロセッサは、複数のスクリプ

トの各々の識別子を生成するステップをさらに含む動作を実行するようにプロセッサ実行可能命令によって構成されてよく、プロセッサは、複数のスクリプトをスクリプトエンジンに送るステップが、複数のスクリプトおよび識別子をスクリプトエンジンに送るステップを含み得るように、かつ、実行されるべき特定された次のスクリプトに対応する情報をスクリプトエンジンに送るステップが、実行されるべき次のスクリプトの識別子をスクリプトエンジンに送るステップを含み得るように、動作を実行するように、プロセッサ実行可能命令によって構成され得る。

【００１５】

さらなる態様では、複数のスクリプトの各々の識別子を生成するステップが、少なくとも1つのスクリプトをuniform resource identifier (URI)と関連付けるステップを含み得るように、動作を実行するように、プロセッサがプロセッサ実行可能命令によって構成され得る。さらなる態様では、複数のスクリプトの各々の識別子を生成するステップが、少なくとも1つのスクリプトのための署名を生成するステップを含み得るように、動作を実行するように、プロセッサがプロセッサ実行可能命令によって構成され得る。さらなる態様では、複数のスクリプトの各々の識別子を生成するステップが、少なくとも1つのスクリプトのテキストを含み得る少なくとも1つの識別子を生成するステップを含み得るように、動作を実行するように、プロセッサがプロセッサ実行可能命令によって構成され得る。

【００１６】

さらなる態様では、HTMLドキュメントをスキャンして複数のスクリプトを発見するステップが、プロセッサにおいて実行される第1のプロセスによってHTMLドキュメントをスキャンするステップを含み得るように、かつ、スクリプトエンジンが実行のために複数のスクリプトを準備している間にHTMLドキュメントを解析するステップが、プロセッサにおいて実行される第2のプロセスによってHTMLドキュメントを解析するステップを含み得るように、動作を実行するように、プロセッサがプロセッサ実行可能命令によって構成され得る。さらなる態様では、実行のために複数のスクリプトを準備するステップが、第2のスクリプトを解析し、分析し、コンパイルするのと並列に、第2のプロセスが第1のスクリプトを解析し、分析し、コンパイルするステップを含み得るように、動作を実行するように、プロセッサがプロセッサ実行可能命令によって構成され得る。

【００１７】

さらなる態様では、スクリプトエンジンが実行のために並列に複数のスクリプトを準備している間にHTMLドキュメントを解析するステップが、複数のスクリプトが実行される実行順序とは異なる準備順序で、スクリプトエンジンが実行のために複数のスクリプトを準備している間に、HTMLドキュメントを解析するステップを含み得るように、動作を実行するように、プロセッサがプロセッサ実行可能命令によって構成され得る。さらなる態様では、実行されるべき次のスクリプトを複数のスクリプトから特定するステップが、定義された実行順序に基づいて、実行されるべき次のスクリプトを特定するステップを含み得るように、動作を実行するように、プロセッサがプロセッサ実行可能命令によって構成され得る。

【００１８】

さらなる態様は、HTMLドキュメントに含まれるスクリプトを準備するための動作をプロセッサに実行させるように構成されるプロセッサ実行可能ソフトウェア命令を記憶した、非一時的コンピュータ可読記憶媒体を含み、この動作は、HTMLドキュメントをスキャンして複数のスクリプトを発見するステップと、複数のスクリプトをスクリプトエンジンに送るステップと、スクリプトエンジンが実行のために複数のスクリプトを準備している間にHTMLドキュメントを解析するステップと、実行されるべき次のスクリプトを複数のスクリプトから特定するステップと、実行されるべき特定された次のスクリプトに対応する情報をスクリプトエンジンに送るステップと、HTMLドキュメントの解析を中断するステップと、実行されるべき特定された次のスクリプトが実行されたことを示す通知を受け取るステップと、通知を受け取ったことに応答してHTMLドキュメントの解析を再開するステップと

を含み得る。ある態様では、記憶されたプロセッサ実行可能ソフトウェア命令は、実行されるべき特定された次のスクリプトに対応する情報をスクリプトエンジンに送るステップが、実行されるべき特定された次のスクリプトをスクリプトエンジンに送るステップを含み得るように、プロセッサに動作を実行させるように構成され得る。

【００１９】

さらなる態様では、記憶されたプロセッサ実行可能ソフトウェア命令は、複数のスクリプトの各々の識別子を生成するステップを含む動作をプロセッサに実行させるように構成されてよく、記憶されたプロセッサ実行可能ソフトウェア命令は、複数のスクリプトをスクリプトエンジンに送るステップが、複数のスクリプトおよび識別子をスクリプトエンジンに送るステップを含み得るように、かつ、実行されるべき特定された次のスクリプトに対応する情報をスクリプトエンジンに送るステップが、実行されるべき次のスクリプトの識別子をスクリプトエンジンに送るステップを含み得るように、プロセッサに動作を実行させるように構成され得る。さらなる態様では、記憶されたプロセッサ実行可能ソフトウェア命令は、複数のスクリプトの各々の識別子を生成するステップが、少なくとも1つのスクリプトをuniform resource identifier (URI)と関連付けるステップを含み得るように、プロセッサに動作を実行させるように構成され得る。

【００２０】

さらなる態様では、記憶されたプロセッサ実行可能ソフトウェア命令は、複数のスクリプトの各々の識別子を生成するステップが、少なくとも1つのスクリプトのための署名を生成するステップを含み得るように、プロセッサに動作を実行させるように構成され得る。さらなる態様では、記憶されたプロセッサ実行可能ソフトウェア命令は、複数のスクリプトの各々の識別子を生成するステップが、少なくとも1つのスクリプトのテキストを含み得る少なくとも1つの識別子を生成するステップを含み得るように、プロセッサに動作を実行させるように構成され得る。

【００２１】

さらなる態様では、記憶されたプロセッサ実行可能ソフトウェア命令は、HTMLドキュメントをスキャンして複数のスクリプトを発見するステップが、プロセッサにおいて実行される第1のプロセスによってHTMLドキュメントをスキャンするステップを含み得るように、かつ、スクリプトエンジンが実行のために複数のスクリプトを準備している間にHTMLドキュメントを解析するステップが、プロセッサにおいて実行される第2のプロセスによってHTMLドキュメントを解析するステップを含み得るように、プロセッサに動作を実行させるように構成され得る。さらなる態様では、記憶されたプロセッサ実行可能ソフトウェア命令は、実行のために複数のスクリプトを準備するステップが、第2のスクリプトを解析し、分析し、コンパイルするのと並列に、第2のプロセスが第1のスクリプトを解析し、分析し、コンパイルするステップを含み得るように、プロセッサに動作を実行させるように構成され得る。

【００２２】

さらなる態様では、記憶されたプロセッサ実行可能ソフトウェア命令は、スクリプトエンジンが実行のために並列に複数のスクリプトを準備している間にHTMLドキュメントを解析するステップが、複数のスクリプトが実行される実行順序とは異なる準備順序で、スクリプトエンジンが実行のために複数のスクリプトを準備している間に、HTMLドキュメントを解析するステップを含み得るように、プロセッサに動作を実行させるように構成され得る。さらなる態様では、記憶されたプロセッサ実行可能ソフトウェア命令は、実行されるべき次のスクリプトを複数のスクリプトから特定するステップが、定義された実行順序に基づいて、実行されるべき次のスクリプトを特定するステップを含み得るように、プロセッサに動作を実行させるように構成され得る。

【００２３】

本明細書に組み込まれ、本明細書の一部を構成する添付の図面は、本発明の例示的な態様を示す。上記の概略的な説明および下記の発明を実施するための形態とともに、図面は、開示される態様を限定するのではなく、本発明の特徴を説明するのに役立つ。

【図面の簡単な説明】
【００２４】
【図１】様々な態様を実装するコンピューティングデバイスで使用され得る例示的なシステムオンチップ(SOC)アーキテクチャを示すコンポーネントブロック図である。
【図２】様々な態様を実装するために使用され得る例示的なマルチコアプロセッサアーキテクチャを示す機能ブロック図である。
【図３Ａ】HTMLドキュメントをレンダリングするためのある態様のブラウザ方法を示すプロセスフロー図である。
【図３Ｂ】ある態様のブラウザシステムにおける、例示的な論理コンポーネント、情報の流れ、動作、および変換を示す、機能およびプロセスフロー図である。
【図４】ある態様のブラウザシステムにおける、例示的な論理コンポーネント、機能コンポーネント、情報の流れ、およびサブシステムを示す、機能ブロック図である。
【図５】ある態様による、並列のブラウザインフラストラクチャを実装する態様のブラウザシステムを示す、機能ブロック図である。
【図６】ページのロード/レンダリングの動作より前にリソースを発見しプリフェッチするためにHTMLドキュメントを処理するある態様のブラウザ方法を示す、プロセスフロー図である。
【図７Ａ】推測技法および経験則を使用してドキュメントリソースの使用率を予測するある態様のブラウザ方法を示す、プロセスフロー図である。
【図７Ｂ】並列に推測的にリソースをプリフェッチするある態様のブラウザ方法を示すプロセスフロー図である。
【図７Ｃ】並列にスクリプトを事前処理するある態様のブラウザ方法を示すプロセスフロー図である。
【図８】プリフェッチされたリソースを処理するある態様のブラウザ方法を示すプロセスフロー図である。
【図９】様々な態様とともに使用するのに適した、CSSエンジン中の例示的な機能コンポーネントを示す機能ブロック図でわる。
【図１０】いくつかのノード上でルールマッチングとカスケード化の動作を並列に実行するためのある態様のスタイリング方法を示すプロセスフロー図である。
【図１１Ａ】様々な態様で使用するのに適した例示的なドキュメントオブジェクトモデル(DOM)ツリーの図である。
【図１１Ｂ】図11Aに示されるDOMツリーに対応するタスク有向非巡回グラフ(DAG)の図である。
【図１２】様々な態様とともに使用するのに適した例示的なモバイルデバイスのコンポーネントブロック図である。
【図１３】様々な態様とともに使用するのに適した例示的なサーバのコンポーネントブロック図である。
【図１４】様々な態様を実装するのに適したラップトップコンピュータのコンポーネントブロック図である。
【発明を実施するための形態】
【００２５】
様々な態様が添付の図面を参照して詳細に説明される。可能な場合は常に、同じ参照番号は、図面全体を通して同じまたは同様の部分を指すために使用される。特定の例および実装形態へと行われる言及は、説明を目的とし、本発明の範囲または特許請求の範囲を限定するものではない。
【００２６】
ウェブブラウザは、複数の規格を実装し、レガシーの挙動をサポートする必要がある、複雑なソフトウェアアプリケーションであり、高度に動的かつ双方向型である。ウェブブラウザの設計者は一般に、ページロードのための高速な応答時間(長いネットワークレイテンシがある場合でも)と、高い性能(たとえば、ウェブアプリケーションの双方向性を可

能にするための)と、良好なユーザ体験を提供するためのユーザインターフェースの高い応答性との最適な混合を達成することを目指す。

【００２７】

様々な態様は、現在のマルチプロセッサモバイルデバイスアーキテクチャにより可能にされる同時実行性/並列性を利用する技法を介して、高速な応答時間、高い性能、およびユーザインターフェースの高い応答性を達成するように構成される、ウェブブラウザ、ブラウザ方法、およびブラウザシステムを提供する。

【００２８】

ハイパーテキストマークアップ言語(HTML)コードは、JavaScript(登録商標)コード(「インラインスクリプト」と呼ばれる)を埋め込むことと、JavaScript(登録商標)コード(「外部スクリプト」と呼ばれる)へのリンクを含めることの両方を行い得る。HTMLドキュメントを同時に処理するために、インラインスクリプトと外部スクリプトの両方が通常、HTML規格によって定義される特定の順序で実行される。すなわち、スクリプトの最終的な実行順序が維持されることを規格は要求する。

【００２９】

様々な態様の方法およびブラウザは、並列に、かつ/または順序通りではなく、スクリプトをダウンロードし、解析し、分析し、コンパイルして、規格によって要求される最終的な実行順序でスクリプトを実行するように構成され得る。

【００３０】

一般に、HTMLドキュメントに含まれる(すなわち、HTMLドキュメントに埋め込まれる、またはHTMLドキュメントへリンクされる)スクリプトのすべてが実際に実行されるとは限らず、実行のためにすべてのスクリプトを事前に準備することは、電力および処理リソースを無駄にし得る。様々な態様は、実行のために準備されるべきスクリプトをインテリジェントに選択する。

【００３１】

複数のスクリプトが並列にダウンロードされ、解析され、分析され、コンパイルされるとき、スクリプトの実行の準備ができる順序は、HTML規格によって定義される特定の実行順序とは異なることがある。あるスクリプトの実行の準備ができていないが、そのスクリプトがHTML規格によって定義される特定の実行順序における次のスクリプトである場合、ブラウザは、HTMLドキュメントの任意の追加の処理を実行する前に、スクリプトの実行の準備ができるまで待つことが必要とされ得る。様々な態様は、この待ち時間を利用して、実行のために他のスクリプトまたはリソース(HTML規格によって制御されない)を準備する。複数のスクリプトおよびリソースは、並列に、かつ/または他のスクリプトの実行の間に準備され得る。

【００３２】

「例示的」という語は、本明細書では「例、実例、または例示として機能する」ことを意味するように使用される。本明細書で「例示的」として説明されるいかなる実装形態も、必ずしも他の実装形態よりも好ましいか、または有利であると解釈されるべきではない。

【００３３】

「モバイルデバイス」および「コンピューティングデバイス」という用語は、携帯電話、スマートフォン、パーソナルまたはモバイルマルチメディアプレーヤー、携帯情報端末(PDA)、ラップトップコンピュータ、タブレットコンピュータ、スマートブック、パームトップコンピュータ、ワイヤレス電子メール受信機、マルチメディアインターネット対応携帯電話、ワイヤレスゲームコントローラ、ならびに、プログラム可能プロセッサおよびメモリを含む同様の個人用電子デバイスのうちの、任意の1つまたはすべてを指すように、本明細書では交換可能に使用される。様々な態様は、限られた処理電力しか有し得ない携帯電話のようなモバイルデバイスにおいて特に有用であるが、態様は、一般に、動的言語、スクリプト言語および/またはマークアップ言語で書かれたスクリプトおよび/またはアプリケーションを実行する任意のコンピューティングデバイスにおいて有用である。

【００３４】
「システムオンチップ」(SOC)という用語は、単一の基板に統合された複数のリソースおよびプロセッサを含む単一の集積回路(IC)チップを指すように本明細書で使用される。単一のSOCは、デジタル、アナログ、混合信号、および高周波機能のための回路を含み得る。単一のSOCは、任意の数の汎用および/または専用のプロセッサ(デジタルシグナルプロセッサ、モデムプロセッサ、ビデオプロセッサなど)、メモリブロック(たとえば、ROM、RAM、Flashなど)、およびリソース(タイマー、電圧調整器、発振器など)も含み得る。SOCは、統合されたリソースおよびプロセッサを制御するための、また周辺デバイスを制御するためのソフトウェアも含み得る。
【００３５】
「マルチコアプロセッサ」という用語は、プログラム命令を読み取り実行するように構成された2つ以上の独立した処理コア(たとえば、CPUコア)を含む単一の集積回路(IC)チップまたはチップパッケージを指すように本明細書で使用される。SOCは、複数のマルチコアプロセッサを含んでよく、SOCにおける各プロセッサは、コアと呼ばれ得る。「マルチプロセッサ」という用語は、本明細書では、プログラム命令を読み取り実行するように構成された2つ以上の処理ユニットを含むシステムまたはデバイスを指すために使用される。
【００３６】
本出願で使用されるように、「コンポーネント」、「モジュール」、「システム」、「エンジン」、「マネージャ」などの用語は、限定はされないが、特定の動作または機能を実行するように構成された、ハードウェア、ファームウェア、ハードウェアとソフトウェアの組合せ、ソフトウェア、または実行中のソフトウェアなどの、コンピュータ関連のエンティティを含むものとする。たとえば、コンポーネントは、限定はされないが、プロセッサ上で実行されているプロセス、プロセッサ、オブジェクト、実行ファイル、実行スレッド、プログラム、および/またはコンピュータであり得る。例を挙げると、コンピューティングデバイス上で実行されているアプリケーションとコンピューティングデバイスの両方が、コンポーネントと呼ばれてよい。1つまたは複数のコンポーネントは、プロセスおよび/または実行スレッドの中に存在してよく、1つのコンポーネントは、1つのプロセッサまたはコアに局在してよく、かつ/または2つ以上のプロセッサまたはコアの間に分散してよい。加えて、これらのコンポーネントは、様々な命令および/またはデータ構造を記憶している様々な非一時的コンピュータ可読媒体から実行することができる。コンポーネントは、ローカルおよび/またはリモートプロセス、関数呼出しまたはプロシージャ呼出し、電子信号、データパケット、メモリ読出し/書込み、ならびに他の知られているコンピュータ、プロセッサ、および/または通信方法に関するプロセスによって通信できる。
【００３７】
「アプリケーションプログラミングインターフェース」という用語およびその頭字語「API」は全般に、本出願では、第2のソフトウェアコンポーネントと通信するために第1のソフトウェアコンポーネントにより使用され得る任意のソフトウェアインターフェースを指すために使用される。APIは、ルーチン、プロシージャ、関数、メソッド、データ構造、オブジェクトクラス、および変数に対する仕様を含み得る。APIはまた、別の高水準プログラミング言語の機能(シンタックス的またはセマンティクス的な)にAPIをマッピングするための機構を含み得る。そのような機構および/またはマッピングは、それら自体がAPIであることがあり、「言語束縛」または「束縛」として知られている。
【００３８】
「マークアップ言語」という用語は全般に、本明細書では、プロセッサがシンタックス的にアノテーションをテキストと区別できるように、テキストにアノテーションを行うための任意のプログラミング言語および/またはシステムを指すために使用される。マークアップ言語の例には、Scribe、Standard Generalized Markup Language (SGML)、Hyper-Text Markup Language (HTML)、Extensible Markup Language (XML)、およびExtensible H

yper-Text Markup Language (XHTML)がある。

【００３９】

「動的言語」および「スクリプト言語」という用語は、本出願では、任意の動的言語、スクリプト言語、または実行時に解釈および/もしくはコンパイルされるプログラムを(本明細書では「スクリプト」として)書くために使用される任意の言語を指すために、全般に、かつ互換的に使用される。これらの用語はまた、管理された実行時に実行され動的にコンパイルされる、任意の言語も指し得る。したがって、本明細書では、様々な態様の説明における「動的言語」および「スクリプト言語」という用語の使用は、特許請求の範囲を、ソースコードもしくはバイトコードから解釈される言語、または従来はネイティブマシンコードにコンパイルされるプログラムとともに実行する言語に限定するものとして解釈されるべきではない。本明細書の範囲内の動的言語およびスクリプト言語の例は、たとえば、JavaScript(登録商標)、Perl、PythonおよびRuby、ならびに今後開発される可能性がある他の同様の言語を含む。

【００４０】

「スタイルシート言語」および「スタイル言語」という用語は全般に、本明細書では、ドキュメントの表示スタイルがドキュメントの内容に対して分離され得るように、構造化されたドキュメントの表示を表現する、任意のコンピュータ言語を指すために使用される。スタイルシート言語の例はCascading Style Sheets (CSS)であり、これは通常、マークアップ言語で書かれたドキュメントの表示セマンティクスを記述するために使用される。

【００４１】

言及を容易にするために、本明細書全体で、HTMLが例示的なマークアップ言語として使用され、CSSが例示的なスタイルシート言語として使用され、JavaScript(登録商標)が例示的な動的スクリプト言語として使用される。しかしながら、本明細書におけるHTML、CSS、およびJavaScript(登録商標)の使用は例示を目的としたものにすぎず、特許請求の範囲によって明示的に述べられていない限り、特許請求の範囲を特定の言語に限定するものと解釈されるべきではない。

【００４２】

HTMLは、ISO/IEC15445規格を実装するマークアップ言語である。HTMLは、ウェブブラウザなどのソフトウェアアプリケーションによって表示され得るようにウェブページを記述するために使用される、マークアップタグ(たとえば、アノテーション)のセットとして特徴付けられ得る。HTMLは、見出し、段落、リスト、リンク、引用、および他の項目などのテキスト用の構造的なセマンティクスを表記することによって、構造化されたドキュメントの作成を可能にする。

【００４３】

JavaScript(登録商標)は、ECMAScript言語規格(ECMA-262仕様においてECMA International によって規格化された)および/またはISO/IDC 16262規格を実装する、動的で、弱い型付けの、オブジェクト指向スクリプト言語である。JavaScript(登録商標)は、モバイルデバイスプロセッサ上で実行されるウェブブラウザのような、ホスト環境内の計算オブジェクトへのプログラム的なアクセスを可能にする。

【００４４】

Cascading Style Sheets (CSS)は、ウェブサイトの外観およびフォーマットを記述するために使用されるスタイル言語であり、ドキュメントの表示をドキュメントの内容に対して分離するために使用されることが意図される。各スタイルシートは、セレクタ{プロパティ1:値;...プロパティn:値; }というフォーマットの、順序付けられたルールの集合体を含み得る。例として、p > cite { color: white; background-color: red; }というCSSコードは、赤の背景の上に白の前景を使用して、直接の先祖が<p>要素であるすべての<cite>要素をレンダリングするように、ブラウザに伝える。ウェブサイトが数万個のそのようなルールを含むことは一般的ではない。

【００４５】

HTMLは、親のHTMLページの挙動および/または表示に影響を及ぼすことが可能なJavaScr

ipt(登録商標)コードに対するリンクを埋め込み、かつ/または含んでよい。埋め込まれた/リンクされたJavaScript(登録商標)コードはまた、親のHTMLページ(JavaScript(登録商標)が埋め込まれたHTMLコード)に挿入され得る、さらなるHTMLコードを生成することができる。JavaScript(登録商標)は、機能がHTMLページのドキュメントオブジェクトモデル(DOM)と対話し、それを操作するように、機能をHTMLコードに埋め込むために使用され得る。DOMは、HTML内のオブジェクトを表示しそれと対話するための、言語に依存しない規約であり、JavaScript(登録商標)コードが親のHTMLページにアクセスし、それを操作することを可能にする。DOMツリーは通常、ページを定義するそれぞれのコンポーネントのコンポーネント、相対的な構造、関係、および挙動を特定するために、ウェブページをレンダリングすることの一部として生成される。

【００４６】

HTMLはCSSコードを含み得る(たとえば、CSSコードを埋め込み、かつ/またはCSSコードにリンクし得る)。別個のファイルとして規定されるCSSコードは、リモートサーバに記憶され得る。従来のCSS処理エンジン(たとえば、WebkitまたはFirefox)は、メインブラウザスレッドにおいて順番にCSSを解析し、高水準の並列性または同時実行性をサポートしない。たとえば、CSSコードがHTMLドキュメントに埋め込まれると、HTMLパーサは、CSSエンジンがHTMLドキュメントのヘッダ中のスタイル要素を解析するまで、HTMLドキュメントの残りの部分を解析できない。HTMLドキュメントがいくつかのCSSファイルへのリンクを含む場合、従来のCSS処理エンジンは、すべてのリンクされたCSSファイルを順番に解析する。これらおよび他の理由で、従来のCSS処理エンジンは、大きなCSSファイルの場合には特に(これはよくあることである)、大きな速度低下を引き起こし得る。

【００４７】

様々な態様の方法およびブラウザは、ページロード、ウェブアプリケーション、およびネットワーク通信の効率性と速度を向上させるために、現在のモバイルデバイスにおいて利用可能な並列性を利用する。

【００４８】

様々な態様は、推測/予測技法を使用してウェブドキュメント(HTMLページ)を事前処理して、必要とされる可能性が高いリソースを情報の不完全なセットから特定し、ウェブドキュメントの適切なレンダリングのために必要とされる確率が高いと判定されたリソースを要求/プリフェッチすることによって、ウェブページをロード/レンダリングするブラウザ方法を含み得る。これらのリソースのプリフェッチは、ウェブブラウザ(およびしたがってモバイルデバイス)が、利用可能な帯域幅をより活用し、転送レイテンシを重複させ、ドキュメントのロード時間を改善することを可能にし得る。

【００４９】

近年では、モバイル電子デバイス(たとえば、携帯電話、タブレット、ラップトップなど)は、より機能が豊富になり、今や一般的に、マルチプロセッサ、システムオンチップ(SoC)、複数のメモリ、および、モバイルデバイスのユーザが自分のモバイルデバイスにおいて複雑で処理負荷の大きいソフトウェアアプリケーション(たとえば、ウェブブラウザ、ビデオストリーミングアプリケーションなど)を実行することを可能にする他のリソースを含む。これらの改善および他の改善により、スマートフォンおよびタブレットコンピュータは人気が高まり、多くのユーザが選択するプラットフォームとして、ラップトップおよびデスクトップマシンを置き換えている。モバイルデバイスのユーザは今や、自分のモバイルデバイスのウェブブラウザを介してインターネットにアクセスすることによって、簡単かつ便利に、多くの毎日の作業を遂行することができる。

【００５０】

様々な態様は、高速なプロセッサおよびマルチプロセッサモバイルデバイスアーキテクチャ、ならびにリソースの推測的な処理およびプリフェッチの使用によって可能にされる、同時実行性/並列性を利用することによって、高速な応答時間、高い性能、およびユーザインターフェースの高い応答性を達成し、これによってネットワークレイテンシを見えなくして全体のユーザ体験を改善するように構成される、ブラウザ方法および/またはウ

ェブブラウザを提供する。

【００５１】

ウェブブラウザは、複数の規格を実装し、レガシーの挙動をサポートする必要がある、複雑なアプリケーションであり、高度に動的かつ双方向型である。ウェブブラウザの設計者は一般に、ページロードのための高速な応答時間(長いネットワークレイテンシがある場合でも)と、高い性能(たとえば、ウェブアプリケーションの双方向性を可能にするための)と、ユーザインターフェースの高い応答性(たとえば、良好なユーザ体験を提供するための)との最適な混合を達成することを目指す。

【００５２】

ウェブブラウザの同時実行性を利用することは、比較的新しい手法である。大半の既存のブラウザ(たとえば、FirefoxおよびWebkitベースのChromeおよびSafariブラウザ)は、双方向性を支援するためにイベントドリブンモデルを使用する、順次的エンジンとして基本的に設計されている。モバイルデバイスおよび/またはブラウザサブシステムの間の多数の依存関係により(および、多くの既存のデータ構造はスレッドセーフではないので)、これらの既存の解決法は、高水準の並列性または同時実行性をサポートしない。

【００５３】

ChromeおよびWebKit2は、各ブラウザタブに対して別個のプロセスを生成し、このことは、異なるウェブサイトの間のある程度の分離を実現するが、複数のコアを使用することの責務をオペレーティングシステムに委ねる。加えて、これらのプロセスは、メモリとスタートアップオーバーヘッドの両方に関して、負荷が大きい。したがって、これらの解決法は、個々のページロードを高速化せず、または、ネットワーク通信の効率性を向上させず、単に、同じアプリケーションの複数のインスタンスを実行することに関する並列性をサポートするものである。そのようなタブレベルの並列性は、単一タブの性能が不十分であることが多くユーザは一度に多くのタブを開かないという、モバイルブラウザのニーズに対処するものではない。

【００５４】

OPおよびOP2ブラウザは、ウェブページごとにプロセスの新たな集合体(「ウェブインスタンス」と呼ばれる)を生成することができ、ブラウザコンポーネント(たとえば、ネットワーキング)は異なるプロセスで実行され得る。しかしながら、これらの解決法は、すべての他の既存のブラウザ解決法のように、依然として本質的には順次的である。たとえば、ネットワーク動作は解析動作として別個のプロセスで実行され得るが、各動作が互いに依存しているので、ネットワークプロセスは依然として解析処理を待たなければならない(そしてその逆も当てはまる)。すなわち、OPおよびOP2ブラウザは、複数のプロセスまたはスレッドの使用を可能にするが、これらの解決法は、ウェブページをダウンロードし、処理し、レンダリングするためのブラウザ処理アルゴリズムの直列的/順次的な性質に対処するものではないので、ウェブページをレンダリングする際に高水準の並列性を実現しない。

【００５５】

様々な態様は、既存のブラウザ処理アルゴリズムの直列的/順次的な性質を克服し、高速プロセッサおよびマルチプロセッサモバイルデバイスアーキテクチャのマルチスレッド実行および並列処理能力を利用し、広範に並列性を利用してブラウザ性能を向上させ、ネットワークレイテンシを低減し、モバイルデバイスのユーザに対するユーザ体験を向上させるように構成される、高性能ウェブブラウザを含む。

【００５６】

様々な態様は、システムオンチップ(SOC)を含む、多数の単一プロセッサおよびマルチプロセッサのコンピュータシステムで実装され得る。図1は、様々な態様を実施するコンピューティングデバイス内で使用され得る例示的なシステムオンチップ(SOC)100アーキテクチャを示す。SOC 100は、デジタルシグナルプロセッサ(DSP)102、モデムプロセッサ104、グラフィクスプロセッサ106、およびアプリケーションプロセッサ108のような、いくつかの異質のプロセッサを含み得る。SOC 100はまた、異質のプロセッサ102、104、106、10

8のうちの1つまたは複数に接続された1つまたは複数のコプロセッサ110(たとえば、ベクトルコプロセッサ)を含み得る。各プロセッサ102、104、106、108、110は、1つまたは複数のコアを含んでよく、各プロセッサ/コアは、他のプロセッサ/コアとは無関係に動作を実行できる。たとえば、SOC 100は、第1のタイプのオペレーティングシステム(たとえば、FreeBSD、LINUX(登録商標)、OS Xなど)を実行するプロセッサ、および第2のタイプのオペレーティングシステム(たとえば、Microsoft Windows(登録商標) 8)を実行するプロセッサを含み得る。

【００５７】

SOC 100はまた、センサーデータ、アナログデジタル変換、ワイヤレスデータ送信を管理し、ウェブブラウザにおいてレンダリングするための符号化されたオーディオ信号およびビデオ信号の処理のような、他の特殊な動作を実行するための、アナログ回路およびカスタム回路114を含み得る。SOC 100は、電圧調整器、発振器、位相ロックループ、周辺ブリッジ、データコントローラ、メモリコントローラ、システムコントローラ、アクセスポート、タイマー、ならびにコンピューティングデバイス上で実行されているプロセッサおよびソフトウェアクライアント(たとえば、ウェブブラウザ)をサポートするために使用される他の同様のコンポーネントのような、システムコンポーネントおよびリソース116をさらに含み得る。

【００５８】

システムコンポーネントおよびリソース116および/またはカスタム回路114は、カメラ、電子ディスプレイ、ワイヤレス通信デバイス、外部メモリチップなどの周辺デバイスとインターフェースするための回路を含み得る。プロセッサ102、104、106、108は、一連の再構成可能な論理ゲートを含み、かつ/またはバスアーキテクチャ(たとえば、CoreConnect、AMBAなど)を実装し得る、相互接続/バスモジュール124を介して、1つまたは複数のメモリ要素112、システムコンポーネントおよびリソース116、ならびにカスタム回路114に相互接続され得る。通信は、高性能ネットワークオンチップ(NoC)のような、高度な相互接続によって提供され得る。

【００５９】

SOC 100は、クロック118および電圧調整器120のような、SOCの外部のリソースと通信するための入力/出力モジュール(図示せず)をさらに含み得る。SOCの外部のリソース(たとえば、クロック118、電圧調整器120)は、内部SOCプロセッサ/コア(たとえば、DSP 102、モデムプロセッサ104、グラフィクスプロセッサ106、アプリケーションプロセッサ108など)のうちの2つ以上によって共有され得る。

【００６０】

上で論じられたSOC 100に加えて、様々な態様が、単一のプロセッサ、複数のプロセッサ、マルチコアプロセッサ、またはこれらの任意の組合せを含み得る、多種多様なコンピューティングシステムにおいて実装され得る。

【００６１】

図2は、様々な態様を実施するために使用され得る例示的なマルチコアプロセッサアーキテクチャを示す。マルチコアプロセッサ202は、(たとえば、単一の基板、ダイ、集積チップ上などで)極めて近接した2つ以上の独立した処理コア204、206、230、232を含み得る。処理コア204、206、230、232が近接していることで、メモリは、信号がチップから離れて進まなければならない場合に可能な周波数/クロック速度よりもはるかに高い周波数/クロック速度で動作することが可能になる。その上、処理コア204、206、230、232が近接していることで、オンチップメモリおよびリソース(たとえば、電圧レール)の共有、ならびにコア間のより協調した連携が可能になる。

【００６２】

マルチコアプロセッサ202は、レベル1(L1)キャッシュ212、214、238、240、およびレベル2(L2)キャッシュ216、226、242を含むマルチレベルキャッシュを含み得る。マルチコアプロセッサ202はまた、バス/相互接続インターフェース218、メインメモリ220、および入力/出力モジュール222を含み得る。L2キャッシュ216、226、242は、L1キャッシュ212、21

4、238、240よりも大きい(かつ遅い)が、メインメモリユニット220よりも小さい(かつはるかに速い)ことがある。各処理コア204、206、230、232は、L1キャッシュ212、214、238、240へのプライベートアクセスを有する処理ユニット208、210、234、236を含み得る。処理コア204、206、230、232は、L2キャッシュ(たとえば、L2キャッシュ242)へのアクセス権を共有してよく、または、独立したL2キャッシュ(たとえば、L2キャッシュ216、226)へのアクセス権を有してよい。

【００６３】

L1およびL2キャッシュは、処理ユニットによって頻繁にアクセスされるデータを記憶するために使用され得る一方で、メインメモリ220は、処理コア204、206、230、232によってアクセスされる、より大きいファイルおよびデータユニットを記憶するために使用され得る。マルチコアプロセッサ202は、処理コア204、206、230、232がメモリから順番にデータを探し、最初にL1キャッシュに、次いでL2キャッシュに、次いで情報がキャッシュに記憶されていない場合にメインメモリに問い合わせるように構成され得る。情報がキャッシュにもメインメモリ220にも記憶されていない場合、マルチコアプロセッサ202は、外部メモリおよび/またはハードディスクメモリ224から情報を探すことができる。

【００６４】

処理コア204、206、230、232は、バス/相互接続インターフェース218を介して互いに通信することができる。各処理コア204、206、230、232は、いくつかのリソースに対して排他的制御権を有し、他のコアと共有することができる。

【００６５】

処理コア204、206、230、232は、互いに同等であっても、異質であってもよく、かつ/または様々な専用の機能を実装してよい。したがって、処理コア204、206、230、232は、オペレーティングシステムの観点から対称的である必要はなく(たとえば、異なるオペレーティングシステムを実行できる)、またはハードウェアの観点から対称的である必要もない(たとえば、異なる命令セット/アーキテクチャを実装できる)。

【００６６】

図1および図2を参照して上で論じられたようなマルチプロセッサハードウェア設計は、同じパッケージの内側の、しばしば同じシリコン片上に、異なる機能の複数の処理コアを含み得る。対称な多重処理ハードウェアは、単一のオペレーティングシステムによって制御される単一の共有されたメインメモリに接続された2つ以上の同一のプロセッサを含む。非対称のまたは「緩く結合された」多重処理ハードウェアは、独立したオペレーティングシステムによって各々制御され1つまたは複数の共有されたメモリ/リソースに接続され得る、2つ以上の異質のプロセッサ/コアを含み得る。

【００６７】

図3Aは、HTMLドキュメントをロードしてレンダリングするある態様のブラウザ方法300を示す。ブロック302において、ウェブブラウザコンポーネントは、特定のuniform resource locator (URL)に位置するHTMLドキュメントのロードを要求する、ユーザ入力を受け取り得る。ブロック304において、ウェブブラウザコンポーネントは、インターネットを介して通信されるよく知られているハイパーテキスト転送プロトコル(HTTP)メッセージを介して、そのURLに位置するウェブサーバからのHTMLドキュメントを要求し得る。ブロック306において、ウェブブラウザコンポーネントは、そのURLに位置するウェブサーバからHTMLドキュメントを受信し得る。ブロック308において、ウェブブラウザコンポーネントは、受信されたHTMLドキュメントを解析して、HTMLファイルにおいて参照される外部リソース(画像、オーディオ、CSSなど)を特定/発見し得る。

【００６８】

ブロック310において、ウェブブラウザコンポーネントは、リソースが保持されているネットワークサーバから、特定された外部リソースを要求することができ、このネットワークサーバは、HTMLドキュメントを提供したサーバ、またはインターネットを介してアクセス可能な任意の他のサーバを含み得る。ブロック312において、ウェブブラウザコンポーネントは、ネットワークサーバから、要求された外部リソースを受信し得る。判定ブロ

ック314において、ウェブブラウザコンポーネントは、受信されたリソースのいずれかが他の外部リソースを参照しているかどうかを判定し得る。

【００６９】

受信されたリソースが他の外部リソースを参照しているとウェブブラウザコンポーネントが判定すると(すなわち、判定ブロック314=「Yes」)、ウェブブラウザは、ブロック310～314で新たに受信されたリソースによって参照される、他の/追加の外部リソースを要求/受信することができる。これらの動作は、すべての参照された外部リソースがダウンロードされるまで、繰り返し実行され得る。

【００７０】

受信されたリソースがいずれの追加の外部リソースも参照しないとウェブブラウザが判定すると(すなわち、判定ブロック314=「No」)、ブロック316において、ウェブブラウザは、受信された外部リソースを分析して、ウェブページを適切にレンダリングするために必要とされるリソースを判定することができる。ブロック318において、ウェブブラウザは、要求されたダウンロードリソースを使用してウェブページをレンダリングすることができる。

【００７１】

図3Bは、ある態様のブラウザシステム350における、例示的な論理コンポーネント、情報の流れ、動作、および変換を示す。ブラウザシステム350は、インターネットから情報および/またはリソースを取り出し、コンピューティングデバイス(たとえば、モバイルデバイス)の電子ディスプレイにウェブページをレンダリングするための様々な動作を、プロセッサに実行させるように構成される、ソフトウェアアプリケーション/モジュールであり得る。

【００７２】

ブラウザシステム350は、様々な段階で、かつ/または様々な動作の間に(たとえば、ページロード動作の間および後などに)ウェブページと対話して、外部モジュール380との双方向性を提供するように構成される、スクリプトコンポーネント362を含み得る。外部モジュール380は、I/Oモジュール(たとえば、マウス、キーボードなど)および/またはアプリケーションモジュール(たとえば、プラグイン、GPSなど)を含み得る。ある態様では、スクリプトコンポーネント362は、JavaScript(登録商標)コードをコンパイルおよび/または実行するように構成される、JavaScript(登録商標)エンジンを含み得る。

【００７３】

ブロック354において、ブラウザシステム350は、フェッチ動作を実行して、ウェブ352中のサーバから(たとえば、HTTPを介して)プログラミング命令356を要求/受信することができる。ブロック358において、ブラウザシステム350は、受信されたプログラミング命令356を変換/復号して、HTMLコード360を生成することができる。生成されたHTMLコード360は、JavaScript(登録商標)コードを含んでよく(すなわち、JavaScript(登録商標)コードを埋め込んでよく、またはJavaScript(登録商標)コードに対する参照を含んでよく)、JavaScript(登録商標)コードの実行により、親のHTMLページへと挿入するための追加のHTMLコード(たとえば、JavaScript(登録商標)が含まれるHTMLコード)が生成され得る。そのような生成されたHTMLコードは、HTMLページの挙動および/または表示に影響を及ぼし得る。生成されたHTMLコード360はまた、スタイルシートおよび/またはCSSコードを含み得る。

【００７４】

ブロック364において、ブラウザシステム350は、HTMLコード360(および埋め込まれた/参照されたJavaScript(登録商標)コード)を解析して、HTMLドキュメントのドキュメントオブジェクトモデル(DOM)366を生成することができる。DOM 366は、HTMLコードにおける様々なオブジェクトのコンテンツ、関係、スタイル、および位置を表し得る。ブラウザ「パス」とコンポーネントとの間の通信が、DOM 366を介して行われ得る。「ブラウザパス」は、HTMLドキュメントの関連のある部分の単一の繰返しと関連付けられる、スレッド、プロセス、またはアプリケーションであり得る。ある実施形態では、ブラウザパスは「作

業項目」であり得る。

【００７５】

上で言及されたように、JavaScript(登録商標)コードはHTMLコードに埋め込まれてよく、同時に、親のHTMLページに挿入されるさらなるHTMLコードを生成することができる。コードの挿入を可能にするために(かつ、適切な順序を確保するために)、2つの異なるプロセスが、JavaScript(登録商標)コードおよび親のHTMLコードを解釈し、解析し、実行するために必要とされ得る。したがって、ある態様では、ブロック364の解析動作は、複数のプロセスまたはアプリケーションによって実行され得る。

【００７６】

ブロック368において、ブラウザシステム350は、スタイル動作を実行して、たとえば、1つまたは複数のスタイルシート(たとえば、CSS)をHTMLドキュメントおよび/または生成されたDOM366ツリーに適用することによって、修正されたDOMツリー370を生成することができる。

【００７７】

ブロック372において、ブラウザシステム350は、レイアウト動作を実行することによって、ページレイアウト374を「解決する」ことができる。ある態様では、ページを表示するのに必要な追加のコンテンツが利用可能になる(たとえば、ダウンロードされる、処理される、かつ/またはDOMに追加される)につれてページレイアウトが付加的に解決されるように、レイアウト動作が実行され得る。

【００７８】

ブロック376において、ブラウザシステム350は、レンダリング動作を実行して、コンピューティングデバイスの電子ディスプレイにHTMLドキュメントのコンテンツ378を表示することができる。

【００７９】

様々な態様は、既存のブラウザ処理アルゴリズムの根本的な直列の性質を変えることができる。様々な態様は、高水準の並列性および/または同時実行性をサポートする、動的で同時実行可能なブラウザシステムを含み得る。様々な態様は、複数のレベルの同時実行性を利用し得る。様々な態様は、個々のブラウザパスに対して並列のアルゴリズムを実行して、様々なブラウザコンポーネントおよび/または動作の処理時間および/または実行時間を高速化することができる。様々な態様は、ブラウザパスを重複させて、総実行時間を高速化し得る。

【００８０】

図4および図5は、様々な態様による、複数のレベルの同時実行性を利用するのに適した、ある態様のブラウザシステム500における、例示的なコンポーネント、情報の流れ、およびサブシステムを示す。

【００８１】

図4は、フェッチマネージャコンポーネント502と、DOMディスパッチャコンポーネント504と、HTMLパーサコンポーネント506と、HTMLプレスキャナコンポーネント508と、画像デコーダコンポーネント510と、CSSエンジンコンポーネント512と、JavaScript(登録商標)エンジンコンポーネント514と、レイアウトおよびレンダリングエンジンコンポーネント516と、ユーザインターフェースコンポーネント518とを含む、ブラウザシステム500を示す。ある態様では、ブラウザシステム500はまた、サンドボックス化されたJavaScript(登録商標)エンジンコンポーネント530を含み得る。これらのコンポーネント502～530の各々はソフトウェアモジュール(たとえば、プロセッサ上で実行されるプロセス、実行スレッド、スレッドプール、プログラムなど)であり得る。様々な態様において、コンポーネント502～530のいずれかまたはすべてが、同時実行性をサポートするために、スレッドライブラリ(たとえば、Pthreadsなど)または並列タスクライブラリ(たとえば、Intel Thread Building Blocks、Cilkなど)を利用し得る。

【００８２】

ある態様では、ブラウザシステム500のコンポーネント502～518、530は、緩やかに結合

され、同時実行性をサポートするように構成され得る。

【００８３】

フェッチマネージャコンポーネント502は、ネットワークからリソースをフェッチし、フェッチされたリソースに対するキャッシュ管理を実行し、ネットワークからのデータの到着の通知を他のブラウザコンポーネントに提供するように構成され得る。ある態様では、フェッチマネージャコンポーネント502は、リソースがHTMLドキュメントにおいて現れる順序で(すなわち、何ら優先順位を付けることなく)、リソースをフェッチするように構成され得る。別の態様では、フェッチマネージャコンポーネント502は、優先順位を割り当て、かつ/または、事前に割り当てられた優先順位に基づいてリソースをフェッチするように構成され得る。

【００８４】

DOMディスパッチャコンポーネント504は、DOMの更新をスケジューリングし、DOMツリーへのアクセスを直列化し、様々なブラウザコンポーネント間の対話を管理するように構成され得る。他のサブシステム(すなわち、ブラウザインフラストラクチャの残り)は、作業項目(「DOMディスパッチャ作業項目」とも呼ばれる)を、同時実行のDOMディスパッチャキューへと割り振ることができる。DOMディスパッチャコンポーネント504は、DOMディスパッチャキューからそれらの作業項目を引き抜いて、それらの作業項目を1つずつ処理するように構成され得る。様々な態様では、作業項目は、ブラウザパスおよび/またはイベント(たとえば、タイマーイベント、ユーザインターフェースからのイベントなど)を含み得る。

【００８５】

HTMLパーサコンポーネント506は、HTMLドキュメントの入来する(たとえば、部分的な、などの)データチャンクを(たとえば、DOMディスパッチャ作業項目などを介して)受け取り、HTML解析アルゴリズム(たとえば、HTML5解析アルゴリズムなど)を実行することによってDOMツリーを構築するように構成され得る。HTMLパーサコンポーネント506は、フェッチマネージャコンポーネント502にアクセス可能なフェッチマネージャキューに、HTMLドキュメントにおいて参照される外部リソースを追加することができる。HTMLパーサコンポーネント506はまた、解析動作の間の適切な時点でJavaScript(登録商標)エンジンコンポーネント514を呼び出すことによって、JavaScript(登録商標)コードの実行を開始することができる。

【００８６】

HTMLプレスキャナコンポーネント508は、HTMLドキュメントをスキャンして、HTMLドキュメントによって要求される/必要とされる外部リソースを迅速に判定するように構成され得る。HTMLプレスキャナコンポーネント508は、外部リソースのダウンロードおよび/または外部リソースに基づくさらなる処理の実行を開始するように、フェッチマネージャコンポーネント502に(たとえば、通知、メモリ書込み動作などを介して)タスクを課すことができる。

【００８７】

画像デコーダコンポーネント510は、画像を復号するように構成され得る。たとえば、フェッチマネージャコンポーネント502が画像の完全なデータを受信した場合、フェッチマネージャコンポーネント502は画像を画像デコーダコンポーネント510に渡すことができ、次いで画像デコーダコンポーネント510が後で使用するために画像を復号することができる。

【００８８】

CSSエンジンコンポーネント512は、より後の段階(たとえば、レイアウトおよびレンダリングの段階)で使用するために、DOM要素のルックアンドフィールを計算するように構成され得る。上で論じられた画像復号動作と同様に、フェッチマネージャコンポーネント502は、解析のために、かつ、要求されるべき新たなリソースの発見のために、CSSスタイルシートをCSSエンジンに渡すことができる。

【００８９】

ある態様では、CSSエンジンコンポーネント512は、CSSリソースプリフェッチャコンポーネント520と、CSSパーサコンポーネント522と、DOMスタイラコンポーネント524とを含み得る。CSSリソースプリフェッチャコンポーネント520は、CSSスキャンおよび/またはプリフェッチ動作を実行することができ、これらの動作は、どの外部リソースがCSSドキュメントによって要求されている/必要とされているかを迅速に判定するために、CSSドキュメントをスキャンすることを含み得る。ある態様では、CSSリソースプリフェッチャコンポーネント520は、外部リソースのダウンロードおよび/または外部リソースに基づくさらなる処理の実行を開始するように、フェッチマネージャコンポーネント502にタスクを課すことができる。

【００９０】

CSSパーサコンポーネント522は、CSSコードを読み取り、メモリ中にデータ構造の集合体(たとえば、CSSルール)を作成するように構成され得る。DOMスタイラコンポーネント524は、CSSパーサコンポーネント522によって作成されたデータ構造を使用して、DOMツリー中のノードのスタイルを決定するように構成され得る。各ノードに対して、CSSエンジンコンポーネント512は、ルールマッチング動作を実行して、セレクタがノードと一致するルールを見つけることができる。そのようなルールマッチング動作は、ノードごとに多数の(かつ、場合によっては競合する)ルールを返し得る。様々な態様において、CSSエンジン512は、カスケード化動作を使用して、重みをルールに割り当て、最大の重みを有するルールを選ぶように構成され得る。

【００９１】

JavaScript(登録商標)エンジンコンポーネント514は、JavaScript(登録商標)コードをコンパイルして実行するように構成され得る。フェッチマネージャ502は、JavaScript(登録商標)スクリプトをダウンロードし、それらを、コンパイルされるようにJavaScript(登録商標)エンジンコンポーネント514に送ることができる。HTMLパーサ506および/またはDOMディスパッチャ504は、JavaScript(登録商標)エンジンコンポーネント514がスクリプトを実行することを要求し得る。

【００９２】

JavaScript(登録商標)エンジンコンポーネント514は、コンパイルのタスク/動作のためのスレッドプールを含んでよく、複数のスクリプト(JavaScript(登録商標)コード)を並列にコンパイルするように構成され得る。JavaScript(登録商標)のセマンティクスにより、ある態様では、スクリプトの実行は、メインエンジンスレッドにおいて順番に実行され得る。ある態様では、HTMLパーサ506またはDOMディスパッチャ504が(たとえば、ユーザインターフェースイベントのために)、コンパイルされていないスクリプトを実行するようにJavaScript(登録商標)エンジンコンポーネント514に要求するときに、JavaScript(登録商標)エンジンコンポーネント514が自動的にスクリプトのコンパイルを開始し、要求されたスクリプトの実行を試みる前にコンパイルの結果を待つように、JavaScript(登録商標)エンジンコンポーネント514は構成され得る。

【００９３】

様々な態様において、JavaScript(登録商標)エンジンコンポーネント514は、(たとえば、JavaScript(登録商標)コードの適応的なコンパイルおよび実行をサポートするために)ライトコンパイラ526およびフルコンパイラ528を含み得る。ライトコンパイラ526は、稀に再使用されるJavaScript(登録商標)コードのための、かつ/またはページロードに最適化された、実行可能コードを生成するように構成され得る。フルコンパイラ528は、頻繁に再使用されるJavaScript(登録商標)コードのための、かつ/または双方向性およびウェブアプリケーションのために最適化された、より高品質のコードを生成するように構成され得る。様々な態様において、フルコンパイラ528のより低速なコード生成は、再使用されるコードの複数の実行の間で清算され得る。ライトコンパイラ526と比較して、フルコンパイラ528は、反復的なウェブアプリケーションに対する大きな高速化を達成し得る。たとえば、フルコンパイラ528を使用すると、N体シミュレーションのウェブアプリケーションを6倍高速に実行できる。

【００９４】
サンドボックス化されたJavaScript(登録商標)エンジンコンポーネント530は、主要なJavaScript(登録商標)エンジンコンポーネント514とは別個である単独のJavaScript(登録商標)エンジンであり得る。サンドボックス化されたJavaScript(登録商標)エンジンコンポーネント530は、JavaScript(登録商標)エンジンコンポーネント514のすべてのコンポーネント、特徴、および機能を含み得る。
【００９５】
レイアウトおよびレンダリングエンジンコンポーネント516は、スタイリングされたDOMツリーを、見ることができるウェブページへと変換するように構成され得る。ある態様では、レイアウトおよびレンダリングエンジンコンポーネント516は、ユーザが更新されたHTMLドキュメントを見てそれと対話できるように、モバイルデバイスの電子ディスプレイ上で、DOMおよび/またはCSSスタイルシートに対する変更を反映するように構成され得る。DOMおよび/またはCSSに対する変更は、フェッチマネージャコンポーネント502が新たなリソースを届けたことによるもの、HTMLパーサコンポーネント506がDOMを更新したことによるもの、JavaScript(登録商標)エンジンコンポーネント514の計算の結果などであってよい。
【００９６】
ある態様では、レイアウトおよびレンダリングエンジン516は、DOM情報のスナップショットをとり、かつ/または非同期的にレイアウトおよび/またはレンダリング動作を実行するように構成され得る。別の態様では、レイアウトおよびレンダリングエンジン516は、(たとえば、JavaScript(登録商標)がレイアウト情報を問い合わせるAPIを利用する場合)同期的にレイアウトおよび/またはレンダリング動作を呼び出すように構成され得る。
【００９７】
ユーザインターフェースコンポーネント518は、ブラウザシステム500とモバイルデバイスのユーザとの対話を管理するように構成され得る。ユーザインターフェースコンポーネント518は、ユーザの対話(たとえば、モバイルデバイスの電子ディスプレイ上のリンクをタッチすること)を、DOMディスパッチャキューへ配置するための作業項目を生み出す関数/メソッドの呼出し(たとえば、Java(登録商標) Native Interfaceまたは「JNI」メソッド呼出し)へと変換することができる。
【００９８】
ある態様では、すべての上述のコンポーネント502～518、530が、各ウェブページに対して1回インスタンス化され得る。別の態様では、フェッチマネージャコンポーネント502とレイアウトおよびレンダリングエンジンコンポーネント516はグローバルであってよく、一方、他のコンポーネント(たとえば、504、506、508、510、512、514、および518)は、各ウェブページまたはHTMLドキュメントに対して1回インスタンス化され得る。
【００９９】
図5は、上で論じられた態様のブラウザシステム500における、例示的なサブシステムおよび情報の流れを示す。具体的には、図5は、ブラウザシステム500が、ユーザインターフェースサブシステム552と、リソースマネージャサブシステム554と、ページ毎DOMエンジンサブシステム556と、ページ毎JavaScript(登録商標)エンジンサブシステム558と、レンダリングエンジンサブシステム560とを含み得ることを示す。
【０１００】
サブシステム555～560の各々は、緩やかに結合され、同時実行性をサポートするように構成され得る。サブシステム552～560はソフトウェアモジュール(たとえば、プロセッサ上で実行されるプロセス、実行スレッド、プログラムなど)として実装され得る。サブシステム552～560の動作は、図4を参照して上で論じられたコンポーネントの1つまたは複数によって、かつ/または、任意のシングルプロセッサもしくはマルチプロセッサコンピューティングシステム上で実行され得る。
【０１０１】
ある態様では、リソースマネージャサブシステム554およびレンダリングエンジンサブ

システム560は1回インスタンス化されてよく(たとえば、グローバルであってよく)、ページ毎DOMエンジンサブシステム556およびページ毎JavaScript(登録商標)エンジンサブシステム558は各ウェブページまたはHTMLドキュメントに対して1回インスタンス化されてよい。

【０１０２】

ユーザインターフェースサブシステム552は、ブラウザシステム550とのユーザの対話を管理するための様々な動作を実行するように構成されてよく、これらの動作は、ユーザの対話(たとえば、モバイルデバイスの電子ディスプレイ上のリンクをタッチすること)を、DOMディスパッチャキューへ配置するための作業項目を生み出す関数/メソッドの呼出しへと変換するステップ、イベントを検出し、かつ/もしくはそれをページ毎JavaScript(登録商標)エンジンサブシステム558の正しいインスタンスに送るステップ、および/または、uniform resource locator (URL)/uniform resource identifier (URI)情報をリソースマネージャサブシステム554に(たとえば、メモリ書込み動作、関数呼出しなどを介して)送るステップを含む。

【０１０３】

リソースマネージャサブシステム554は、プリフェッチ動作562と、HTMLプレスキャン動作563と、画像復号動作564と、CSSスキャン/プリフェッチ動作566と、JavaScript(登録商標)スキャン/プリフェッチ動作567とを実行するように構成され得る。例として、これらの動作は、フェッチマネージャコンポーネント502、HTMLプレスキャナコンポーネント508、画像デコーダコンポーネント510、CSSエンジンコンポーネント512、および/またはJavaScrip(登録商標)tエンジンコンポーネント514、530、または、図4を参照して上で論じられたコンポーネントの任意の組合せによって実行され得る。

【０１０４】

プリフェッチ動作562は、URL/URIに対応するウェブサーバからのリソースおよび/またはプログラミング命令を要求/受信するステップと、受信されたプログラミング命令を変換または復号してHTMLを生成するステップと、生成されたHTMLコードをページ毎JavaScript(登録商標)エンジンサブシステム558の正しいインスタンスに(たとえば、メモリ書込み動作などを介して)送るステップとを含み得る。

【０１０５】

生成されたHTMLコードは、JavaScript(登録商標)コード、CSSコード、画像、および様々な他のリソースを埋め込み、かつ/または参照することができる。HTMLドキュメントにおいて最もよく参照されるリソースは、画像、CSSスタイルシート、およびJavaScript(登録商標)ソースである。スタイルシートおよびJavaScript(登録商標)ソースはまた、さらなる外部リソースを参照し得る。ある態様では、生成されたHTMLコードは、HTMLドキュメントによって特定されるすべての参照(埋め込まれたまたは参照されたスタイルシートおよびJavaScript(登録商標)ソースを含む)が事前にフェッチされ得る(たとえば、プリフェッチ動作562の一部として)ように、スキャンされ得る。

【０１０６】

HTMLプレスキャナ動作563は、生成されたHTMLコードをスキャンして、要求される/必要とされる外部リソースを迅速に発見するステップと、外部リソースのダウンロードおよび/または発見された外部リソースに基づくさらなる処理の実行を開始できることを、フェッチマネージャおよび/またはプリフェッチャに知らせるステップとを含み得る。ある態様では、外部リソースのダウンロードは、上で論じられるプリフェッチ動作562の一部として実行され得る。ある態様では、HTMLプレスキャナ動作508およびプリフェッチ動作562は(たとえば、別個のスレッド/プロセスで)同時に実行され得る。

【０１０７】

画像復号動作564は、レンダリングエンジンサブシステム560が後で使用するための画像を復号するステップを含み得る。画像復号動作564は、画像の完全なデータセットがダウンロードされた(たとえば、プリフェッチ動作562の一部として実行されるメモリ書込み動作などを介して)と判定したことに応答して、かつ/または、通知を(たとえば、フェッチ

マネージャコンポーネント502から)受信したことに応答して、実行され得る。ある態様では、画像復号動作564は、HTMLプレスキャナ動作563およびプリフェッチ動作562と同時に実行され得る。

【０１０８】

CSSスキャン/プリフェッチ動作566は、生成されたHTMLコードに埋め込まれる(またはそれによって参照される)CSSスタイルシートをスキャンして、CSSスタイルシートによって要求された、要求される/必要とされる外部リソースを迅速に発見するステップを含み得る。ある態様では、CSSスキャン/プリフェッチ動作566は、発見された外部リソースをダウンロードし始め得ることをフェッチマネージャおよび/またはプリフェッチャに知らせるステップを含み得る。ある態様では、CSSスキャン/プリフェッチ動作566は、発見された外部リソースのダウンロードを開始するステップを含み得る。ある態様では、CSSスキャン/プリフェッチ動作566は、フェッチマネージャコンポーネント502が1つまたは複数のCSSスタイルシートをCSSエンジンコンポーネント512に送ったことに応答して、CSSエンジンコンポーネント512において(たとえば、CSSリソースプリフェッチャ520によって)実行され得る。ある態様では、CSSスキャン/プリフェッチ動作566は、画像復号動作564、HTMLプレスキャナ動作563、およびプリフェッチ動作562と同時に実行され得る。

【０１０９】

ページ毎DOMエンジンサブシステム556は、HTML解析動作568と、CSS解析動作570と、タイマー動作572と、スタイリング動作574と、画像イベントに対する動作576とを実行するように構成され得る。ある態様では、ページ毎DOMエンジンサブシステム556の動作は、他のサブシステム552、554、558、560の動作と同時に実行され得る。

【０１１０】

HTML解析動作568は、受信されたHTMLコードを解析するステップ、大量のコンテンツからHTMLマークアップタグを分離するステップ、および/または、受信されたHTMLコードのDOMを生成するステップを含み得る。HTML解析動作568はまた、特定された外部リソースが、フェッチマネージャ502によって、かつ/または、プリフェッチ動作562の一部としてダウンロードされ得るように、HTMLドキュメントにおいて参照される外部リソースを特定するステップを含み得る。HTML解析動作568はさらに、HTMLコードの解析の間(たとえば、JavaScript(登録商標)が発見されるときなど)に、JavaScript(登録商標)コードの実行を(たとえば、実行動作578を呼び出すことによって)開始するステップを含み得る。

【０１１１】

CSS解析動作570およびスタイリング動作574は、1つまたは複数のCSSスタイルシートを生成されたDOMツリーに適用するステップ(または、CSSスタイルシートに基づいて修正されたDOMツリーを生成するステップ)を含み得る。様々な態様において、HTML解析動作568、CSS解析動作570、およびスタイリング動作574のいずれかまたはすべてが、同時に実行され得る。

【０１１２】

タイマー動作572は、タイマーおよび/またはタイマークラス(たとえば、System.Timers)に関するイベントおよび/または条件を管理し、またはそれに応答するステップを含み得る。

【０１１３】

イベント動作576は、タイマーイベントおよびユーザインターフェースイベント(たとえば、ユーザがモバイルデバイスの電子ディスプレイ上のリンクをタッチしたことに応答して生成されるイベント)のような、様々なイベントを管理するステップを含み得る。

【０１１４】

ページ毎JavaScript(登録商標)エンジンサブシステム558は、JavaScript(登録商標)実行動作578およびJavaScript(登録商標)コンパイル動作580を実行するように構成され得る。

【０１１５】

様々な態様では、ページ毎DOMエンジンサブシステム556および/またはリソースマネー

ジャサブシステム554は、HTMLコードに埋め込まれる(またはそれにより参照される)JavaScript(登録商標)コードを、コンパイルおよび/または実行のために(すなわち、実行動作578およびコンパイル動作580を介して)ページ毎JavaScript(登録商標)エンジン558の正しいインスタンスに送るように構成され得る。ある態様では、JavaScript(登録商標)エンジン558は、JavaScript(登録商標)コンパイル動作および/または実行動作578、580の結果に基づいて、生成されたDOMツリーを更新/修正することができる。

【０１１６】

レンダリングエンジンサブシステム560は、レイアウト動作582およびレンダリング動作584を実行するように構成され得る。たとえば、レンダリングエンジンサブシステム560は、(たとえば、メモリ書込み、呼出し、通知などを介して)ページ毎DOMエンジンサブシステム556からDOMツリーおよび/またはレイアウトツリーを受け取り、(レイアウト動作582を介して)ページレイアウトを解決し、(レンダリング動作584を介して)コンピューティングデバイスの電子ディスプレイにコンテンツを表示することができる。ある態様では、レイアウト動作582を実行するステップは、レンダリングエンジンサブシステム560が追加のコンテンツを利用可能になるにつれて(たとえば、追加のコンテンツがダウンロードされ、処理され、かつ/またはDOMツリーに追加されるにつれて)、付加的にページレイアウトを解決するステップを含み得る。様々な態様において、レイアウト動作582および/またはレンダリング動作584のいずれかまたはすべてが、同時に実行され得る。

【０１１７】

図4および図5を参照して上で論じられたように、HTMLパーサ506および/またはCSSパーサ522は、HTMLドキュメントをレンダリングするために要求される/必要とされる外部リソース(画像、オーディオ、CSS、JavaScript(登録商標)など)を発見し、たとえば、フェッチマネージャ502を介して、かつ/またはプリフェッチ動作の一部として、発見されたリソースがダウンロードされることを要求することができる。

【０１１８】

モバイルデバイスは、HTMLおよびCSSのコード/コンテンツにおいて発見されるリソースをダウンロードするときに、レイテンシの時間が長くなり得る。たとえば、HTML5規格の特性により、HTMLパーサは、HTMLドキュメントの残りの部分の解析を続けられるようになる前に、スクリプト要素(たとえば、<script>ブロック)が実行を終了するのを待たなければならない。したがって、ウェブページがスクリプト要素の後に外部リソースを参照する場合、そのリソースをフェッチする動作は、スクリプト要素が実行を終了するのを待つ動作と重複してはならない。このことは、しばしば、ウェブページをダウンロードして表示するのに必要とされる時間を増やす。

【０１１９】

様々な態様では、ブラウザシステム500は、スクリプト要素に先立って推測的に解析して、スクリプト要素が実行を終了するのを待つことなく新たなリソースを発見するように構成され得る。これらの態様では、ブラウザシステム500は、(たとえば、JavaScript(登録商標)がdocument.write APIを介して新たなコンテンツをDOMツリーに挿入するときなどに)推測的な解析の結果の一部を廃棄することを強いられ得る。

【０１２０】

ある態様では、ブラウザシステム500は、積極的なリソースプリフェッチ動作を実行して、可能な限り早く要求される/必要とされるリソースを発見し、複数のリソースが並列にフェッチ/ダウンロードされることを要求するように構成され得る。このようにして、様々な態様は、ブラウザシステム500が推測的な解析の結果の一部を廃棄することが強いられるのを防ぐことができ、ネットワークレイテンシを見えなくして、利用可能な帯域幅のより多くを利用し、リソースが到達するのを待つのに費やされる全体的な時間を減らすことができる。

【０１２１】

ブラウザシステム500は、サンドボックス化された実行を介した推測的なリソースのプリフェッチを含み得る、積極的なリソースプリフェッチ動作を実行するように構成され得

る。様々な態様では、これらの積極的なリソースプリフェッチ動作は、HTMLプリスキャン動作563、CSSプリフェッチ動作566、またはこれら両方の一部として実行され得る。

【０１２２】

図4～図5を参照すると、積極的なリソースプリフェッチ動作の促進において実行されるHTMLプリスキャン動作563は、HTMLドキュメント中のすべての「id」、「class」、および/または「style」属性を取得するステップと、HTMLドキュメントにおいて参照される外部リソースを迅速に発見するステップと、ネットワークからの発見されたリソースのダウンロードをトリガするステップとを含み得る。HTMLプレスキャナ508は、HTMLパーサ506から要求される大量の、または計算量の多い処理のいずれ(たとえば、DOMツリーの構築)をも実行することなく、リソースを発見するために、HTMLを「概略的に解析する」ことができる。これらの複雑な解析動作を行わずに済ますために、HTMLプレスキャン動作563は、HTML解析動作568と同時に実行されてよく(かつその前に実行されてよく)、スクリプト要素が実行を終了するのを待つ必要がない。

【０１２３】

ある態様では、ネットワークパケットは、到達するにつれて、HTMLプレスキャナ508およびHTMLパーサ506に独立に送信され得る。ある態様では、リソースが到達するのを待つのに費やされる時間はさらに、(非推測的な)HTML解析動作570と並列にHTMLプレスキャン動作563を実行することによって減らされ得る。

【０１２４】

上で論じられたように、ウェブブラウザシステム500は、CSSドキュメントを迅速にスキャンするように構成されるCSSパーサ522と、CSSプリフェッチ動作を実行するように構成されるCSSリソースプリフェッチャ520とを含み得る。ある態様では、CSSスタイルシートは、CSSを同時に解析することを担うスレッドプールに割り振られ得る。CSSルールがさらなる外部リソースを含む場合、CSSリソースパーサは、HTMLドキュメントにおいて実際に参照される確率に基づいて、さらなる外部リソースのためのプリフェッチを開始するかどうかに関する判断を行うことができる。ある態様では、CSSリソースプリフェッチャ520は、具体的な範囲/数の参照されるリソースをダウンロードする(またはそのダウンロードを開始する)ように構成され得る(ダウンロードするリソースが少なすぎることは、DOMツリーを後でスタイリングするときにより多くの新たなリソースがDOMスタイラ524によって発見されることを意味することがあり、これは追加のレイテンシをもたらし得る)。

【０１２５】

任意の所与のドキュメントのために実際に必要なリソースよりも多くのリソースを、たとえばサイト全体で共通のスタイルファイルを使用することによって参照することは、ウェブサイトにおいては一般的である。すべての含まれるリソースをダウンロードすることで、余計な帯域幅が消費され、ページのローディングが遅くなり得る。様々な態様では、CSSパーサ522は、HTMLプレスキャナ508によって発見される「id」および「class」の属性を利用して、CSSルールが一致する可能性が高いかどうかを判定するように構成され得る。CSSルールセレクタにおいて参照される属性の値のすべてがHTMLプレスキャナ508によって確認/評価されると、少なくとも1つのDOMツリー要素においてルールが一致する可能性が高いと判定されてよく、ブラウザシステム500は、CSSルールに対応するリソースのダウンロードを開始することができる。この「CSSルール」の経験則は非常に有効であり、誤った判断が、ブラウザシステム500の動作に対して大きな悪影響を与えない。リソースをダウンロードするために必要とされるレイテンシと引き換えに、失われたリソースがDOMスタイリングの段階で(DOMスタイラコンポーネント524を介して)発見され得る。

【０１２６】

ある態様では、HTMLプレスキャナ508は、JavaScript(登録商標)を実行する必要なく発見され得る、リソースを特定および/または発見するように構成され得る。

【０１２７】

上で論じられたように、モバイルデバイスは、HTML5規格の特性、たとえば、HTMLパーサが解析を続けられるようになる前にスクリプト要素(たとえば、<script>ブロック)の実

行が終了するのを待つことが要求されることが原因で、HTMLおよびCSSのコード/コンテンツにおいて発見されるリソースをダウンロードするときのレイテンシの時間が長くなり得る。加えて、現在のウェブドキュメント(たとえば、HTMLページ、HTMLドキュメントなど)は、大量の外部リソースを参照することができ、各々の外部リソースは、他の外部リソースに対する参照を含み得る。たとえば、HTMLドキュメントは通常、画像、オーディオ、Cascading Style Sheets (CSS)、およびJavaScript(登録商標)のような、様々な外部リソースに対する参照を含み、参照されるリソース(たとえば、CSS、JavaScript(登録商標))はさらに、追加の外部リソース(たとえば、画像、オーディオなど)に対する参照を含み得る。

【０１２８】

ドキュメントのロード時間(すなわち、ドキュメントを要求してからドキュメントがスクリーンに表示される準備ができるまでの時間)は、入力/出力のコスト(たとえば、必要とされるリソースのネットワーク転送)によって支配される。すべての必要とされるリソースをロードするのに必要な最短のドキュメントのロード時間は、リソース記憶デバイスとコンピューティングデバイスとの間の接続の帯域幅によって制限される。また、ドキュメントリソースを表示デバイスに転送することは、レイテンシのコストを招く。様々な態様は、可能な限り早くリソース転送を開始し、利用可能な帯域幅をよりよく利用し、転送のレイテンシを重複させ、ドキュメントのロード時間を改善するように構成され得る。

【０１２９】

上で言及されたように、参照された外部リソースのすべてが所与のウェブページをレンダリングするために必要とされる(またはさらには使用される)とは限らないので、参照されるリソースのすべての反復的なダウンロードは、大量の帯域幅および電力を無駄にし得る。加えて、リソースのいずれかが直ちに利用可能にならない場合、ブラウザは、ページが適切にレンダリングされ得る前に、それらのリソースを受信して分析するのを待たなければならない。このことは、ウェブページをロードおよび/またはレンダリングするために必要とされる時間の長さ(たとえば、ドキュメントのロード時間)を増やし、ユーザ体験を劣化させる。

【０１３０】

従来の解決法は、ページがアクセスされる次のときにダウンロードされなければならない情報を減らすためにメモリ中のウェブページの部分をキャッシュすることなどの技法を使用して、ウェブページのレンダリングを高速化しようとする。しかしながら、従来の解決法を使用すると、ウェブブラウザは、ドキュメント(すなわち、ウェブページ)全体を最初に分析することと、ドキュメントおよびサブドキュメント中で参照されるリソースの(すべてではなくても)大半を要求して受信することと、受信されたリソースを分析することとを伴わないと、ウェブページを初めてレンダリングするために必要とされる外部リソースを特定できない。したがって、従来の解決法を使用すると、ドキュメントによって必要とされるリソースの正確なセットは、ドキュメント全体が完全に分析された後まで決定できない。

【０１３１】

既存の解決法に対するこれらの制約を克服するために、様々な態様は、推測/予測技法を利用して、ドキュメント全体が分析される前にウェブページまたはドキュメントをレンダリングするのに必要とされるリソースを特定することができる。

【０１３２】

一般に、リソースが必要とされるかどうかを(情報の不完全なセットに基づいて)推測的に予測することは、正しい肯定的結果、正しい否定的結果、誤った肯定的結果、および誤った否定的結果という、4つのあり得る結果のうちの1つをもたらす。正しい肯定的結果は、リソースが推測的にダウンロードされ実際に後で必要とされた場合である。正しい否定的結果は、リソースが推測的にダウンロードされず必要とされなかった場合である。誤った肯定的結果は、必要とされないリソースが推測的にダウンロードされる場合(これは帯域幅およびエネルギーを無駄にする)であり、誤った否定的結果は、リソースが推測的に

ダウンロードされなかったが必要とされる場合(したがって、推測的な事前処理からこのリソースに関して得られるものは何もない)である。

【０１３３】

正しい肯定的結果および正しい否定的結果は有益であり望ましく、それは、そのような判断がページのロード時間を減らすことによってユーザ体験を改善するからである。しかしながら、誤った肯定的結果および誤った否定的結果は不利である。たとえば、誤った否定的結果は、ドキュメント(たとえば、HTMLドキュメント)のレンダリングの間にリソースが要求されることをもたらし、これは、リソースが利用可能になるまでのドキュメントのロード時間を延ばし得る。そのリソースは、ブラウザがドキュメントを適切にレンダリングするのに必要とされないので、これは、コンピューティングリソースおよびネットワークリソース(帯域幅、処理など)の無駄である。

【０１３４】

様々な態様は、誤った肯定的なダウンロード判断および誤った否定的なダウンロード判断の数を最小にしつつ、正しい肯定的結果および正しい否定的結果の数を最大にするための経験則に基づいて、推測的なリソースのダウンロード動作を実行するように構成される、ウェブブラウザシステムを含む。

【０１３５】

図6は、ページのローディング/レンダリングの動作の前に、ウェブページの適切なレンダリングおよび発見されたリソースのプリフェッチのために必要とされる外部リソース(画像、オーディオ、CSS、JavaScript(登録商標)など)を発見するために、HTMLドキュメントを処理するある態様のブラウザ方法600を示す。方法600の動作は、適切に構成されるウェブブラウザを実行する、シングルプロセッサコンピューティングシステムまたはマルチプロセッサコンピューティングシステムのプロセッサによって実行され得る。

【０１３６】

図6を参照すると、ブロック602において、ウェブブラウザは、スキャン動作を開始しまたは呼び出して(たとえば、HTMLプレスキャナ508、CSSエンジン512などを介して)、構造的な情報のために、かつ/またはリソースを発見するために、HTMLドキュメントおよび/またはCSSドキュメントをスキャンすることができる。ある態様では、スキャン動作は、HTMLプレスキャン動作563の一部として実行され得る。ある態様では、スキャン動作は、CSSスキャン動作566の一部として実行され得る。様々な態様では、スキャン動作は、HTML解析動作568およびCSS解析動作570と同時に、かつそれらと独立に実行され得る。様々な態様において、スキャン動作は、プロセス、スレッド、アプリケーション、作業項目、および/またはブラウザパスによって実行され得る。

【０１３７】

ブロック604において、スキャン動作(たとえば、HTMLスキャン動作563および/またはCSSスキャン動作566)は、発見されたリソースのいずれが必要とされる可能性が高いかを判定(すなわち、予測、推測)することができる。ブロック606において、スキャン動作は、リソース要求を(たとえば、メモリ書込み動作などを介して)ブラウザフェッチコンポーネントに(たとえば、フェッチマネージャ502に)出して、必要とされる確率が高いと判定されたリソースのダウンロードを開始することができる。ある態様では、ブロック606の一部として、2つ以上のリソース要求は、並列にまたは同時に出され得る(または送られ得る)。ある態様では、各リソース要求は、新たなプロセスをスポーンし、かつ/または、異なる実行スレッドによって処理され得る。ブロック608において、スキャン動作は、追加の必要とされるリソースを発見するために、HTMLドキュメントおよび/またはCSSドキュメントをスキャンし続けることができる。ブロック604～608の動作は、すべての外部リソースが発見されるまで、かつ/またはHTMLドキュメント全体がスキャンされるまで、繰り返され得る。

【０１３８】

ブロック610において、ウェブブラウザは、フェッチ動作を(たとえば、フェッチマネージャ502を介して)開始し、または呼び出して、リソース要求(たとえば、ブロック606のス

キャン動作によって出されるリソース要求)によって特定される1つまたは複数のリソースをダウンロードすることができる。

【０１３９】

ブロック612において、ウェブブラウザは、ダウンロードされたリソースをスキャンして、外部リソースに対する追加の参照を発見することができる。ブロック612の一部として、ウェブブラウザは、新たなプロセスまたは実行スレッドを開始し、または呼び出して、スキャン動作を実行することができる。ある態様では、ブロック612の一部として、ウェブブラウザは、CSSスキャン動作566を開始し、または呼び出すことができる。ある態様では、ブロック612の一部として、ウェブブラウザは、HTMLスキャン動作563を開始し、または呼び出すことができる。

【０１４０】

ブロック614において、ウェブブラウザは、ダウンロードされたリソースのスキャンに基づいて、必要とされる可能性が高い発見されたリソースを判定(すなわち、予測、推測)することができる。ブロック616において、ウェブブラウザは、追加のリソース要求を(たとえば、メモリ書込み動作などを介して)ブラウザフェッチコンポーネントに(たとえば、フェッチマネージャ502に)出して、必要とされる確率が高いと判定されたリソースのダウンロードを開始することができる。ある態様では、これらの追加のリソース要求の各々は、他のプロセスをスポーンすることがあり、かつ/または、異なるプロセスまたは実行スレッドによって処理されることがある。ブロック610～616の動作は、すべての外部リソースが発見されるまで、かつ/またはダウンロードされるまで、繰り返され得る。ある態様では、ブロック602～608の動作は、ブロック610～616の動作と並列に実行され得る。

【０１４１】

従来のHTMLパーサとは異なり、図6を参照して上で論じられたスキャン動作は、スキャンされたHTMLドキュメントに対するエラー訂正を行わず、または遭遇したJavaScript(登録商標)コードを実行しない。これにより、スキャン動作が迅速に実行されることが可能になる。また、従来のHTMLパーサとは異なり、上で論じられるスキャン動作は、並列にまたは同時に(たとえば、独立のスレッドまたはプロセスなどにおいて)実行されてよく、これにより、様々な態様が、現在のコンピューティングデバイスにおいて普及しているマルチプロセッサアーキテクチャをより完全に利用することが可能になる。加えて、上で論じられるスキャンプロセスは、HTMLドキュメント(たとえば、CSSドキュメント)において参照されるリソースをスキャンすることができ、これも、従来のHTMLパーサでは実行されない。

【０１４２】

一般に、スキャン動作(たとえば、HTMLプレスキャン動作563、CSSスキャン動作566など)がHTMLドキュメントの構造のみをスキャンする場合、たとえば、ドキュメントが解析されるときにドキュメントを変更させる、ドキュメント中の構造的エラー(スキャナがエラー訂正を実行しないので)またはドキュメント中の埋め込まれたJavaScript(登録商標)コード(スキャナがJavaScript(登録商標)を実行しないので)がない限り、必要とされるリソースについて正しく推測する(すなわち、正しい肯定的結果を生む)可能性が高い。

【０１４３】

ある態様では、正しい肯定的結果および正しい否定的結果の数を最大限にするために、スキャン動作(たとえば、HTMLプレスキャン動作563、CSSスキャン動作566など)は、HTMLドキュメントの初期スキャンの間に取得される情報を使用して、必要とされる可能性が高いリソースを特定することができる。

【０１４４】

図7Aは、推測的なダウンロードのためのドキュメントリソースを発見するための、推測技法および経験則を使用するある態様のブラウザ方法700を示す。ドキュメントリソースは、画像、CSSファイル、JavaScript(登録商標)スクリプトなどを含み得る。ブラウザ方法700は、HTMLドキュメントスキャナおよび複数のCSSドキュメントスキャナが並列に実行されることを可能にし、必要とされる可能性が高いリソースをインテリジェントに特定し

、推測的なリソース要求および/またはプリフェッチ動作に起因する誤った否定的結果の数を減らす。ある態様では、ブラウザ方法700は、誤った肯定的結果の数を最小限にするための経験則(たとえば、「CSSルール」の経験則)を利用することができる。

【０１４５】

ブラウザ方法700のブロック702において、HTMLドキュメントスキャナ(たとえば、HTMLプレスキャナ508)は、HTMLドキュメントのスキャンを開始して、リソースを発見し、HTMLドキュメントに含まれる、すべてのURL/URIと、HTML要素と関連付けられる(またはそれによって記述される)HTMLの「id」、「class」、および/または「style」の属性とを取得することができる。HTMLドキュメントスキャナは、HTMLパーサとは独立であってよく、かつ/またはそれと並列に実行され得る。

【０１４６】

ブロック704において、HTMLドキュメントスキャナは、HTMLドキュメントに含まれる、URL/URIおよび/またはHTML要素によって参照される外部リソースに遭遇し得る。ブロック706において、HTMLドキュメントスキャナは、HTMLドキュメントにおいて参照される遭遇したリソースをダウンロードするための要求を(たとえば、フェッチマネージャに)出すことができる。ある態様では、HTMLドキュメントスキャナは、(たとえば、スキャナが外部リソースに遭遇するときなどに)各々の遭遇した外部CSSリソースのダウンロードおよび/または解析を呼び出すように構成され得る。ある態様では、外部CSSリソースのダウンロードは、CSSドキュメントスキャナ(たとえば、CSSエンジン512など)にCSSドキュメントのスキャンを開始させ得る。

【０１４７】

ブロック708において、HTMLドキュメントスキャナは、HTMLのid、class、およびstyleの属性に遭遇し、かつ/またはそれらを収集し得る。ブロック710において、HTMLドキュメントスキャナは、遭遇した/収集された情報(すなわち、収集されたid、class、およびstyleの属性に関する情報)をCSSドキュメントスキャナに送ることができる。ある態様では、収集された情報を送るステップは、すべての遭遇した、かつ/または特定されたHTMLのid、class、およびstyleの属性をCSSドキュメントスキャナに送るステップを含み得る。

【０１４８】

ブロック712において、HTMLドキュメントスキャナは、追加のリソースを発見するために、HTMLドキュメントのスキャンを続けることができる。判定ブロック714において、HTMLドキュメントスキャナは、HTMLドキュメントのスキャンを終えたかどうかを判定することができる。HTMLドキュメントのスキャンを終えたとHTMLドキュメントスキャナが判定すると(すなわち、判定ブロック714=「Yes」)、ブロック716において、HTMLドキュメントスキャナは、HTMLドキュメントのスキャンを終えたことを、(たとえば、メモリ書込み動作、メソッドの呼出し、通知などを介して)CSSドキュメントスキャナ(たとえば、CSSエンジン512、CSSスキャン動作566を実行するプロセスなど)に通知することができる。HTMLドキュメントのスキャンをまだ終えていないとHTMLドキュメントスキャナが判定すると(すなわち、判定ブロック714=「No」)、ブロック702において、HTMLドキュメントスキャナは、追加のリソースを発見するためにHTMLドキュメントのスキャンを続けることができる。

【０１４９】

ブラウザ方法700のブロック719において、CSSドキュメントスキャナは、外部リソースのためにCSSドキュメントのスキャンを開始することができる。ブロック719におけるCSSドキュメントスキャナの開始は、フェッチマネージャによって取得されるCSSドキュメントが利用可能であることによって(たとえば、ブロック706の一部として実行される動作などに応答して)トリガされ得る。ある態様では、CSSドキュメントのスキャンは、HTMLドキュメントのスキャン(たとえば、ブロック702～716の動作)と並列に実行され得る。したがって、HTMLドキュメントスキャナがHTMLドキュメントをスキャンし続けながら、CSSドキュメントスキャナは、受信されたCSSドキュメントをスキャンして、それらのドキュメントにおいて参照される外部リソースを特定する(たとえば、ダウンロードするための追加のCSSドキュメントを特定するなど)ことができる。さらに、(たとえば、複数のCSSドキュ

メントがダウンロードされるとき)並列に実行される複数のCSSドキュメントスキャナがあり得る。

【０１５０】

ブロック720において、CSSドキュメントスキャナは、HTMLドキュメントスキャナから、HTMLのid、class、および/またはstyleの属性に関する情報を受信することができる。ブロック721において、CSSドキュメントスキャナは、受信された情報が、(受信されたHTMLのid、class、および/またはstyleの属性と関連付けられる)CSSルールおよび/または外部リソースを、HTMLドキュメントによって必要とされるかつ/または使用される可能性が高いものとして、標識または識別するかどうかを判定することができる。ある態様では、ブロック721の一部として、CSSドキュメントスキャナは、CSSルールと関連付けられるすべてのHTMLのid、class、および/またはstyleの属性にHTMLドキュメントスキャナがすでに遭遇しているかどうかを判定することができる。

【０１５１】

判定ブロック722において、CSSドキュメントスキャナは、(受信されたHTMLのid、class、および/またはstyleの属性と関連付けられる)CSSルールおよび/または外部リソースが、HTMLドキュメントによって必要とされるかつ/または使用される可能性が高いかどうかを判定することができる。ある態様では、判定ブロック722の一部として、CSSドキュメントスキャナは、HTMLドキュメントによって記述されるすべてのURL/URI、ならびに、HTMLのid、class、および/またはstyleの属性にすでに遭遇しているかどうかを判定することができる。

【０１５２】

CSSルールおよび/または外部リソースがHTMLドキュメントによって必要とされるかつ/または使用される可能性が高いと、CSSドキュメントスキャナが判定すると(すなわち、判定ブロック722=「Yes」)、ブロック724において、CSSドキュメントスキャナは、たとえば、メモリ書込み動作を実行し、かつ/またはフェッチマネージャ502に通知することによって、そのCSSルールによって参照されるリソースがダウンロードされるように直ちに要求することができる。

【０１５３】

ある態様では、HTMLドキュメントによって記述されるすべてのURL/URI、ならびに、HTMLのid、class、および/またはstyleの属性にすでに遭遇していると判定されたときに、CSSルールおよび/または外部リソースが必要とされる可能性が高いと、CSSドキュメントスキャナが判定することができる。

【０１５４】

CSSルールおよび/または外部リソースがHTMLドキュメントによって必要とされるかつ/または使用される可能性が高くないと、CSSドキュメントスキャナが判定すると(すなわち、判定ブロック722=「No」)、ブロック723において、CSSドキュメントスキャナは、リソース参照のリスト中のCSSルールに関する情報(たとえば、受信されたHTMLのid、class、および/またはstyleの属性)をメモリに記憶することができる。ブロック725において、CSSドキュメントスキャナは、必要であれば(たとえば、スキャン/処理されるべき追加の要素がある場合など)、CSSドキュメントのスキャンを続けることができる。

【０１５５】

ブロック726において、CSSドキュメントスキャナは、HTMLドキュメントスキャナがHTMLドキュメントのスキャンを終えたことを示す通知を、HTMLドキュメントスキャナから受信することができる。ブロック727において、CSSドキュメントスキャナは、メモリに記憶されたリソース参照のリストから、CSSルールに関する情報を取り出し、取り出された情報を評価することができる。

【０１５６】

判定ブロック728において、CSSドキュメントスキャナは、取り出された情報が、CSSルールおよび/または外部リソースを、HTMLドキュメントによって必要とされる(または必要とされる可能性が高い)ものとして、標識/識別するかどうかを判定することができる。あ

る態様では、判定ブロック728の一部として、CSSドキュメントスキャナは、取り出されたCSSルールと関連付けられるすべてのHTMLのid、class、および/またはstyleの属性にHTMLドキュメントスキャナがすでに遭遇しているかどうか、かつ/またはそれらを処理したかどうかを判定することができる。

【０１５７】

取り出された情報が、HTMLドキュメントによって必要とされかつ/または使用される可能性が高いものとしてCSSルールおよび/または外部リソースを標識/識別すると、CSSドキュメントスキャナが判定する場合(すなわち、判定ブロック728=「Yes」)、ブロック729において、CSSドキュメントスキャナは、そのCSSルールに対応するリソースのダウンロードを要求することができる。このようにして、HTMLドキュメントおよびCSSドキュメントを同時にスキャンすることによって引き起こされる、誤った否定的結果の数が最小限にされ得る。加えて、様々な態様は、データ転送コストの増加をほとんどまたはまったく伴わずに、さらには、プロセッサおよびネットワークインターフェース/無線の利用率の低下による電力消費の減少を伴って、ドキュメントのロード時間を減少させる(およびしたがって、応答性を向上させる)ことができる。

【０１５８】

図7Aに戻ると、HTMLドキュメントによって必要とされる(または必要とされる可能性が高い)ものとして取り出された情報が外部リソースを標識または識別すると、CSSドキュメントスキャナが判定する場合(すなわち、判定ブロック728=「No」)、ブロック721において、CSSドキュメントスキャナは、メモリから次のルールを取り出すことができる。ブロック720～722の動作は、HTMLドキュメントスキャナによってメモリに記憶されたすべてのCSSルールが評価されるまで繰り返され得る。

【０１５９】

様々な態様では、上で説明されたCSSルールよりも正確な経験則が、性能を向上させるために、HTMLドキュメントスキャナおよび/またはCSSドキュメントスキャナによって使用され得る。たとえば、ある態様では、HTMLドキュメントスキャナは、HTMLドキュメントを修正し得るURLおよび/または命令のための埋め込まれたJavaScript(登録商標)コードをスキャンするように構成され得る。同様に、ある態様では、CSSドキュメントスキャナは、各々の遭遇したIDと関連付けられるHTMLタグについての階層的な情報を記録するように構成されてよく、これにより、CSSドキュメントスキャナが、より多くの誤りである可能性のある肯定的結果を特定し拒絶することが可能になり得る。

【０１６０】

従来のブラウザでは、HTMLパーサは一般に、外部リソースのすべてを特定し、ネットワークを介してサーバから外部リソースを要求することを担う。上で論じられたように、これらのリソースがHTMLドキュメントで明示的に規定される場合、様々な態様は、これらのリソースをプリフェッチして、従来のブラウザよりもページロードにおいてはるかに早く、要求を出すことができる。加えて、様々な態様は、並列にリソースをプリフェッチおよび/または処理することができる。

【０１６１】

ソフトウェア開発者は、特定のアプリケーションとデバイスの組合せ(たとえば、ウェブブラウザとモバイルデバイスの組合せ)に対して必要とされるであろうリソースを動的に決定するために、ますますスクリプト(たとえば、JavaScript(登録商標)コード)を使用するようになっている。たとえば、スクリプトは、クライアント(たとえば、ブラウザ)およびコンピューティングデバイスに関する様々な因子を評価して、ダウンロードされるべきリソースを特定することができる。そのようなスクリプトは、基本的に、評価された因子に基づいて、リソース(たとえば、画像、CSS、他のJavaScript(登録商標)など)に対して動的にURLを構築することができる。したがって、HTMLドキュメントは、HTMLドキュメントにおいて明示的に特定されず、HTMLドキュメントに含まれるJavaScript(登録商標)コードを実行することのみによって決定され得る、リソースを必要とすることがある。

【０１６２】

JavaScript(登録商標)コードは、親のHTML(およびHTMLコード自体)の状態、挙動、および/または表示を変えることができるので、HTMLパーサは、順番に、かつ/またはHTML規格において定義される順序付けルールに従うことによって、遭遇したJavaScript(登録商標)コード(またはスクリプト)を実行することが要求される。たとえば、HTMLパーサがスクリプトタグ(すなわち、JavaScript(登録商標)スクリプトのような、クライアント側スクリプトを定義するために使用される<script>タグ)に遭遇すると、HTMLパーサは、HTMLドキュメントの残りの部分の解析を継続できるようになる前に、スクリプトがダウンロードされ実行されるのを待たなければならない。結果として、すべてのリソース要求は、JavaScript(登録商標)スクリプト(すなわち、<script>タグ内部のJavaScript(登録商標)コード)の実行の中で、直列化され得る(すなわち、順番に実行されることが要求され得る)。また、HTMLドキュメントのスキャン動作(たとえば、HTMLプレスキャン動作563など)が、ウェブページを適切にレンダリングするために必要とされるであろうリソースを統計的に予測することも、より難しくなり得る。

【０１６３】

様々な態様は、サンドボックス化されたJavaScript(登録商標)エンジン530においてリソースを推測的にプリフェッチすることによって、これらのおよび他の制約を克服することができ、これによって、ブラウザシステム500が、他のブラウザ動作(たとえば、HTML解析)および他のリソース要求と並列に、HTMLドキュメントにおいて明示的に要求されないリソースを発見しダウンロードすることが可能になる。これらの態様はまた、ブラウザシステム500が、意図しないブラウザ状態の変更を伴わずに、複数のJavaScript(登録商標)スクリプトを並列に実行することを可能にし得る。

【０１６４】

様々な態様は、スクリプト(たとえば、JavaScript(登録商標)コード)が発見されるとすぐに、他のブラウザ動作(たとえば、HTMLプレスキャン563、HTML解析568など)および/または他のスクリプトと並列に、スクリプトを実行することができる。ウェブページの通常の処理と干渉するのを防ぐために、スクリプトは、(たとえば、主要なJavaScript(登録商標)エンジンの動作に影響を及ぼさないように)他のブラウザコンポーネントから独立したかつ/または分離された、サンドボックス化されたJavaScript(登録商標)エンジン530において実行され得る。サンドボックス化されたJavaScript(登録商標)エンジン530においてスクリプトを実行することで、スクリプトの並列な実行の間にシステムがブラウザ状態を意図せずに変更するのを防ぐ。ある態様では、各スクリプトは、サンドボックス化されたJavaScript(登録商標)エンジン530の別個のインスタンス(たとえば、スレッド)において実行され得る。

【０１６５】

様々な態様は、ブラウザクライアントとJavaScript(登録商標)エンジン530との間のAPIを変更することができる。

【０１６６】

一般に、スクリプトエンジン(たとえば、JavaScript(登録商標)エンジン514、530、558)は、ブラウザAPI(すなわち、スクリプトがブラウザ動作を呼び出すことを可能にするインターフェース)に対する束縛(すなわち、言語をマッピングするためのAPI)を提供して、ブラウザ動作を呼び出す(たとえば、DOMを操作する、ネットワークにアクセスするなど)。

【０１６７】

ある態様では、JavaScript(登録商標)エンジン530は、ネットワークからのリソースを要求するブラウザAPIを監視することができる。JavaScript(登録商標)エンジン530は、束縛を修正して(または、スクリプトエンジンのための束縛の別個のセットを提供して)、リソース要求をプリフェッチャコンポーネントなどの異なるブラウザコンポーネントへとリダイレクトさせることができる。このようにして、リソース要求および/または収集された情報は、さらなる処理のためにプリフェッチャコンポーネントに直接渡され得る。

【０１６８】

サンドボックス化されたJavaScript(登録商標)エンジンは、JavaScript(登録商標)コードをスキャンし、コードの選択部分のみを実行し、かつ/または、外部リソースを発見することに最も関連のある動作のみを選択することができる。スキャン動作は、スクリプトが要求し得るリソースを発見することのみに関係するので、スキャン動作は、HTML規格のルールに縛られず、遭遇するコードのすべてを実行(run)/実行(execute)する必要はない。遭遇するコードのすべてを完全に実行しないことで、JavaScript(登録商標)スキャン動作は、サンドボックス化されたJavaScript(登録商標)エンジンによって高速に実行され得る。

【０１６９】

サンドボックス化されたJavaScript(登録商標)エンジンは、経験則を適用して、JavaScript(登録商標)スキャン動作をさらに高速化することができる。例として、そのような経験則は、総実行時間(たとえば、スクリプトまたは動作ごとに最大で10msを費やすなど)、ループの繰返しの数(たとえば、最初の10回のループの繰返しのみを処理するなど)、再帰の深さ、サポートされる機能、抽象解釈などを制限することを含み得る。

【０１７０】

様々な態様は、オブジェクトおよびデータ構造(たとえば、ハッシュテーブル、アレイなど)のサイズを制限して、JavaScript(登録商標)スキャン動作をさらに高速化することができ、それは、そのような構造は一般にリソースの依存関係に影響を与えないからである。

【０１７１】

ソフトウェア開発者は、コードの中で、一般的なパターン、フレームワーク、および/またはサービス(本明細書では総称的に「パターン」)を使用することが多い。様々な態様は、コード中のそのような一般性/パターンを検出して(たとえば、解析、分析、コンパイルなどの間に)、発見したリソースに関連のあるパターン(または、パターンによって特定されるJavaScript(登録商標)コードの部分)のみを実行することができる。ある態様では、完全な適合および保守的なコード生成の代わりに、サンドボックス化されたJavaScript(登録商標)エンジンは、(たとえば、積極的なコンパイラの最適化を介して)最も一般的なパターンを対象とするように構成され得る。パターンは、コード中のキーワードを検出すること(これは比較的単純な動作である)ならびに/またはページおよび/もしくはスクリプトの構造を分析すること(これは比較的複雑な動作である)などの、多種多様な既知のパターン認識技法を使用して検出され得る。

【０１７２】

図7Bは、サンドボックス化されたJavaScript(登録商標)エンジンにおけるスクリプトの並列処理によって、リソースを並列に推測的にプリフェッチするある態様の方法730を示す。方法730の動作は、本明細書で論じられる他のブラウザ動作と並列に実行され得る。

【０１７３】

方法730のブロック732において、HTMLドキュメントスキャナ(たとえば、HTMLプレスキャナ508)は、構造的な情報のために、かつ/またはリソースを発見するために、HTMLドキュメントのスキャンを開始することができる。ブロック734において、HTMLドキュメントスキャナは、JavaScript(登録商標)スクリプトに遭遇し、遭遇したスクリプトを(たとえば、メモリ書込み動作、リダイレクトされたリソース要求、修正された束縛などを介して)サンドボックス化されたJavaScript(登録商標)エンジンに送り、遭遇したスクリプトを直ちに実行することができる。ブロック732において、HTMLドキュメントスキャナは、構造的な情報のために、かつ/またはリソースを発見するために、HTMLドキュメントをスキャンし続けることができる。ある態様では、HTMLドキュメントスキャナは、スクリプトに遭遇したことに応答して、サンドボックス化されたJavaScript(登録商標)エンジンを生成する(またはスポーンする)ことができる。

【０１７４】

ブロック735において、サンドボックス化されたJavaScript(登録商標)エンジンは、スクリプトのスキャンを開始してリソースを発見することができる。ブロック736において

、サンドボックス化されたJavaScript(登録商標)エンジンは、スクリプト(またはスクリプトに含まれるJavaScript(登録商標)コードの部分)を推測的に実行することができる。スクリプトの推測的な実行は、発見された外部リソースに関連がある可能性が最も高い、動作および/またはコードの部分のみを実行することを含み得る。様々な態様において、推測的な実行動作は、他のブラウザ動作(たとえば、HTMLプレスキャン563、HTML解析568など)と並列に、かつ/または、他のスクリプトの実行(推測的かどうかにかかわらず)と並列に実行され得る。

【０１７５】

ある態様では、スクリプトの推測的な実行は、発見されたリソースに関連があるパターンに対応するJavaScript(登録商標)コードの部分のみを実行することを含み得る。

【０１７６】

ある態様では、ブロック736の一部として、サンドボックス化されたJavaScript(登録商標)エンジンは、(たとえば、実行時間を減らすために)経験則に基づいてJavaScript(登録商標)コードの推測的な実行を実行することができる。そのような経験則は、総実行時間、ループの繰返しの数、再帰の深さ、サポートされる機能、および/またはコードの抽象解釈を制限することを含み得る。

【０１７７】

ある態様では、ブロック736の一部として、サンドボックス化されたJavaScript(登録商標)エンジンは、スクリプトの推測的な実行から生成されたデータ構造(たとえば、ハッシュテーブル、アレイなど)のサイズを制限することができる。完全なデータ構造により、ダウンロードのためにさらなるリソースを特定することがなくなり得るので、大きなデータ構造を完全に生成する/埋めるために必要とされる処理時間が省略され得る。

【０１７８】

ブロック738において、サンドボックス化されたJavaScript(登録商標)エンジンは、HTMLドキュメントをレンダリングするために必要とされるがHTMLドキュメントにおいて明示的に要求されないリソースを発見することができる。ブロック740において、サンドボックス化されたJavaScript(登録商標)エンジンは、発見されたリソースを取り出すために、プリフェッチャに知らせる(またはプリフェッチャをスポーンする)ことができる。ブロック742において、サンドボックス化されたJavaScript(登録商標)エンジンは、ブロック736で実行される処理の結果を廃棄することができる。

【０１７９】

ブロック744において、プリフェッチャは、ブロック738においてサンドボックス化されたJavaScript(登録商標)エンジンによって発見されるリソースを見つけることができる。ブロック746において、プリフェッチャは見つけられたリソースをダウンロードすることができる。ブロック748において、プリフェッチャはダウンロードされたリソースをメモリに保存することができる。

【０１８０】

上で論じられたように、HTMLコードは、JavaScript(登録商標)コードを埋め込むこと(「インラインスクリプト」と呼ばれる)と、JavaScript(登録商標)コードへのリンクを含めること(「外部スクリプト」と呼ばれる)の両方を行うことができる。HTMLドキュメントを正しく処理するために、インラインスクリプトと外部スクリプトの両方が、HTML規格によって定義される特定の順序で実行されなければならない。

【０１８１】

複数のスクリプトが並列にダウンロードされ、解析され、分析され、コンパイルされるとき、スクリプトの実行の準備ができる順序は、HTML規格によって定義される特定の実行順序とは異なることがある。あるスクリプトの実行の準備ができていないが、そのスクリプトがHTML規格によって定義される特定の実行順序における次のスクリプトである場合、ブラウザは、HTMLドキュメントの任意の追加の処理を実行する前に、スクリプトの実行の準備ができるまで待つことが必要とされ得る。様々な態様は、この待ち時間を利用して、実行のために他のスクリプトまたはリソース(HTML規格によって制御されない)を準備する

。複数のスクリプトおよびリソースは、並列に、かつ/または他のスクリプトの実行の間に準備され得る。

【０１８２】

加えて、HTMLドキュメントに含まれる(すなわち、HTMLドキュメントに埋め込まれるまたはリンクされる)スクリプトのすべてが実際に実行されるとは限らないので、実行のためにすべてのスクリプトを事前に準備することは、電力および処理リソースを無駄にし得る。様々な態様は、実行のために準備されるべきスクリプトをインテリジェントに選択することができる。

【０１８３】

例として、HTMLプリフェッチャは、すべての参照されたスクリプトを(順序通りではなく)発見しダウンロードすることができ、HTMLパーサは後で、スクリプトの実行を、正しい順序に、かつHTMLドキュメントを処理する正しい時点において編成することができる。

【０１８４】

スクリプトの最終的な実行順序は、一般に維持されなければならない。しかしながら、スクリプトをダウンロードし、解析し、分析し、コンパイルすることと関連付けられるすべての動作は、並列に、かつ/または順序通りではなく実行されてよい。

【０１８５】

ある態様では、HTMLドキュメントに含まれるスクリプトは、(すなわち、互いに対して)並列に、かつ(すなわち、HTML規格によって定義される特定の実行順序に対して)順序通りではなく、実行のために準備され得る。これは、固有の識別子および/もしくは署名を生成すること、かつ/またはそれらを各スクリプトと関連付けることによって、達成され得る。署名は、スクリプトの内容に基づき得る。様々な態様において使用するのに適した署名およびサインプロセスの例は、ファイルオフセット(インラインスクリプトのための)、メッセージダイジェストアルゴリズム(たとえば、MD5)、セキュアハッシュアルゴリズム(SHA)、スクリプトのURL、スクリプトのURI、ブラウザキャッシュキー、および/または種々の既知のサインプロセスのいずれかを含む。

【０１８６】

図7Cは、並列実行のためにHTMLドキュメントに含まれるスクリプトをインテリジェントに準備する、ある態様のブラウザ方法750を示す。方法750の動作は、他のブラウザ動作と並列にプロセッサによって実行され得る。

【０１８７】

ブロック752において、HTMLスキャナ/プリフェッチャは、構造的な情報のために、かつ/またはリソース(画像、CSS、スクリプトなど)を発見するために、HTMLドキュメントをスキャンすることができる。ブロック754において、HTMLスキャナ/プリフェッチャは、HTMLドキュメント中の1つまたは複数のスクリプトを発見し、HTMLパーサ(HTMLスキャナと並列に実行される)に発見されたスクリプトを知らせることができる。ブロック756において、HTMLスキャナ/プリフェッチャは、外部スクリプトのダウンロードを開始することができる。

【０１８８】

ブロック758において、HTMLパーサは、各々の発見されたスクリプト(インラインスクリプトと外部スクリプトの両方)の識別子(または署名)を生成し、かつ/または各々の発見されたスクリプトを識別子と関連付けることができる。ある態様では、HTMLパーサは、発見されたスクリプトのテキストをスクリプトの識別子として設定することができる。ある態様では、HTMLパーサは、外部スクリプトのURL/URIを外部スクリプトと関連付けることができ(すなわち、それらのURL/URIをそれらの署名として設定することができ)、ダイジェストおよび/またはハッシュアルゴリズムを実行して、インラインスクリプトの署名を計算することができる。スクリプトのURL/URIが利用可能ではない、固有ではない、かつ/または別様にスクリプトを一意に特定しない場合、ブロック758の一部として、HTMLパーサは、そのスクリプトを特定するために、署名を生成し使用することができる。

【０１８９】

ブロック760において、HTMLパーサは、スクリプトおよびそれらの関連する識別子またはURL/URIを、HTMLパーサと並列に(たとえば、別個のスレッドで)実行されるJavaScript(登録商標)エンジンに送ることができる。ブロック762において、HTMLパーサは、HTMLを解析して他のスクリプトを発見することなどの、様々なHTMLパーサ動作を実行することができる。

【０１９０】

ブロック772において、JavaScript(登録商標)エンジンは、HTMLパーサから、スクリプトと、関連する識別子、署名、またはURL/URIとを受け取ることができる。ブロック774において、JavaScript(登録商標)エンジンは、実行のために、受け取られたスクリプトを準備(たとえば、解析、分析、および/またはコンパイル)することができる。準備動作は、すべての受け取られたスクリプトにわたって、順序通りではなく、かつ/または並列に実行され得る(すなわち、複数のスクリプトが一度に準備され得る)。ある態様では、ブロック774の一部として、JavaScript(登録商標)エンジンは、(たとえば、抽象解釈を介して)経験則を利用して、コードを実行することなくコールグラフを検出し、コールグラフに基づいて、実行される可能性が最も高いスクリプト(または関数)を特定し、実行される可能性が高いと判定されたスクリプトのみを実行のために準備することができる。ブロック776において、JavaScript(登録商標)エンジンは、スクリプト(たとえば、コンパイルされたコードなど)の準備の間に生成された情報を、スクリプトの識別子、署名、またはURL/URIと関連付けることができる。

【０１９１】

ブロック764において、HTMLパーサは、(たとえば、HTML規格によって定義される実行順序に基づいて)実行されるべき次のスクリプトを特定することができる。ブロック766において、HTMLパーサは、実行されるべき次のスクリプトの識別子(たとえば、スクリプトのテキスト、署名、URL/URIなど)を、JavaScript(登録商標)エンジンに送ることができる。ブロック768において、HTMLパーサは、実行の結果またはスクリプトが実行されたという通知を待機することができる。ブロック770において、HTMLパーサは、HTMLパーサ動作の実行を続けることができる。

【０１９２】

ブロック778において、JavaScript(登録商標)エンジンは、HTMLパーサから、識別子、署名、またはURL/URIを受け取ることができる。ブロック780において、JavaScript(登録商標)エンジンは、受け取られた識別子、署名、URL/URIに基づいて適切なスクリプトを特定することができる。判定ブロック782において、JavaScript(登録商標)エンジンは、たとえば、解析、分析、およびコンパイルの動作のすべてがそのスクリプトについて実行されたかどうかを判定することによって、特定されたスクリプトを直ちに実行する準備ができているかどうかを判定することができる。スクリプトを直ちに実行する準備ができているとJavaScript(登録商標)エンジンが判定すると(すなわち、判定ブロック782=「Yes」)、ブロック786において、JavaScript(登録商標)エンジンは、実行の結果、または実行が完了したことを、HTMLパーサに知らせることができる。

【０１９３】

スクリプトを直ちに実行する準備がまだできていないと判定されると(すなわち、判定ブロック782=「No」)、ブロック784において、JavaScript(登録商標)エンジンは、従来の解決法を使用して、実行のためにスクリプトを準備することができる。ブロック786において、JavaScript(登録商標)エンジンは、HTML規格によって定義される特定の実行順序に従ってスクリプトを実行することができる。このようにして、方法750は、実行のために、HTMLドキュメントに含まれるスクリプトを、並列に(すなわち、互いに対して)かつ順序通りではなく(すなわち、HTML規格によって定義される特定の実行順序に対して)準備し、スクリプトは、規格によって定義される順序で実行される。

【０１９４】

図8は、プリフェッチされたリソースを処理するある態様のブラウザの方法800を示す。ブロック802において、ウェブブラウザコンポーネントは(たとえば、フェッチマネージャ

502を介して)、発見されたリソース(たとえば、画像)のダウンロードを開始することができ、発見されたリソースは、他のブラウザ動作(たとえば、HTML解析など)の実行と同時に(または並列に)ダウンロード/フェッチされ得る。発見されたリソースと関連付けられるすべてのデータがダウンロードされ、かつ/または受信されると、ブロック804において、ダウンロードされたデータ(たとえば、画像データ)は、復号のためにスレッドプールに送られ得る。ある態様では、復号動作は、他のブラウザ動作と同時に実行され得る。

【０１９５】

ブロック806において、ダウンロードされたデータ(たとえば、画像データ)が復号され得る。ブロック808において、復号されたデータが、DOMディスパッチャキューに追加され得る。ブロック810において、DOMディスパッチャコンポーネント504が、DOMツリーおよびそれぞれのツリーノード(たとえば、画像データの場合は「img」ツリーノード)に対する更新を直列化することができる。ブロック812において、リソース(たとえば、画像)は、処理リスト(たとえば、未処理の画像のリスト)から除去され得る。

【０１９６】

図9は、様々な態様とともに使用するのに適したCSSエンジン512における例示的なコンポーネントを示す。CSSエンジン512は、CSSリソースプリフェッチ動作902、CSS解析動作904、およびDOMスタイリング動作906という、3つの主要なカテゴリの動作を実行するように構成され得る。

【０１９７】

CSS解析動作904は、CSSコードを読み取るステップと、メモリ中にデータ構造の集合体(たとえば、CSSルール)を作成するステップとを含み得る。CSSコードは、HTMLに埋め込まれてよく、または別個のファイルとしてリンクされてよく、異なるサーバに記憶されてよい。従来のCSSエンジン(たとえば、WebkitまたはFirefoxのCSSエンジン)は、メインブラウザスレッドにおいて順番にCSSを解析し得る。したがって、埋め込まれたCSSをページが使用する場合、HTMLパーサは、CSSエンジンがドキュメントのヘッダ中のスタイル要素を解析するまで、HTMLドキュメントの残りを解析できない。ページがいくつかのCSSファイルを使用する場合、十分に活用されていないCPUコアがある可能性があっても、それらのCSSファイルはすべて順番に解析される。そのようなCSS解析の直列化(すなわち、CSSドキュメントの直列の処理)は、サイトが大きなCSSファイルを使用する場合に、重大な速度低下を引き起こし得る。様々な態様は、非同期的なタスクを使用して、CSS解析の直列化を避けることができる。

【０１９８】

図9を参照すると、HTMLパーサ506は、ページロード動作の間に、DOMツリー中の各スタイル要素に対して、CSS解析タスク570をスポーンするように構成され得る。同様に、フェッチマネージャ502は、新たなCSSファイルが到達するときには常に、CSS解析タスク570をスポーンし得る。結果として、複数のCSS解析タスク570は、HTMLパーサ506および/またはHTML解析動作568と同時に実行され得る。

【０１９９】

スタイルシート(CSS)およびルール(CSSルール)の全体的な順序がスタイリング動作574の重要な部分であり得るので、ブラウザシステム500は、プログラマが意図した順序ですべてのスタイルシート(CSS)が解析されたかのように、全体の順序が同じであることを確実にするように構成され得る。

【０２００】

様々な態様において、解析タスク568または解析動作570の各々は、固有の順次的なパーサIDを受け取り得る。ブラウザシステム500は次いで、そのIDを使用して、ドキュメント中のスタイルシートの順序を再び決めることができる。

【０２０１】

DOMスタイリング動作906は、CSSエンジン512が、CSSパーサ522によって作成されたデータ構造を使用して、DOMツリー中のノードのスタイルを決定することを可能にし得る。各ノードに対して、CSSエンジン512は、ルールマッチング動作を実行して、そのセレクタが

ノードと一致するすべてのルールを見つけることができる。ルールマッチングは一般に、ノードごとに多数の(かつ、場合によっては競合する)ルールを返す。カスケード化を使用して、CSSエンジンは、重みをルールに割り当て、最大の重みを有するルールを選ぶことができる。

【０２０２】

ノードをスタイリングする最後のステップは、DOMスタイリング動作906が、カスケード化アルゴリズムによって選択されたルールを使用することによってスタイルデータ構造を作成するステップと、それをDOMに追加するステップとを含み得る。ルールマッチングおよびカスケード化の動作は、何らかの依存関係が強いられている限り、いくつかのノードで並列に実行され得る。

【０２０３】

様々な態様は、複数のブラウザ動作および/またはパスの同時の実行(または重複)の間、既存のHTMLおよびJavaScript(登録商標)のセマンティクスを順守/実施し得る。DOMツリーは、すべてのブラウザパスによって使用される主要なデータ構造であり得る。様々な態様では、DOMツリー(HTML5パーサによって構築され得る)へのアクセスは、HTML5規格に適合するように直列化され得る。加えて、より高度な並列性を実現するために、各パスは、(すなわち、DOMツリーに加えて)プライベートな同時のデータ構造へのアクセス権を与えられ得る。ある態様では、この追加のデータ構造はレイアウトツリーであり得る。

【０２０４】

図10は、ルールマッチングおよびカスケード化の動作がいくつかのノードで並列に実行される、ある実施形態の並列DOMスタイリング方法1000を示す。ブロック1002において、CSSエンジン512は、DOMツリーを詳細に検討して、DOMノードごとに、マッチングタスクおよびノードスタイリングタスクという2つの異なるタスクをスポーンし得る。ブロック1004において、マッチングタスクは、ルールマッチングおよびカスケード化の動作をDOMノードのために実行することができる。ブロック1006において、スタイリングタスクは、DOMノードを記述するスタイルデータ構造を作成することができる。ブロック1008において、スタイリングタスクは、DOMツリーにスタイルデータ構造を追加することができる。

【０２０５】

図11Aは、様々な態様で使用するのに適した例示的なDOMツリーを示す。図11Bは、図11Aに示される例示的なDOMツリーに対応する例示的なタスク有向非巡回グラフ(DAG)を示す。具体的には、図11Bは、マッチングタスク(三角形として表される)がどのように、互いに完全に独立であり得るか、かつスタイリングタスク(四角形として表される)と完全に独立であり得るかということと、一方で、スタイリングタスクが、互いに、かつマッチングタスクに依存していることとを示す。一般に、マッチングタスクの並列な実行は、コンピューティングシステムの処理コアの数のみによって制約される。

【０２０６】

上で言及されたように、スタイリングタスクは、互いに、かつ/またはマッチングタスクに依存し得る。各スタイリングタスクは、実行され得る前に、2つの依存関係を満たすことが必要とされ得る。第1に、スタイリングタスクは、同じノードに対するマッチングタスクの実行が完了した後にのみ、実行され得る。これは、スタイリングタスクが、マッチングタスクによって選択されるルールを使用してスタイルデータ構造を構築するからである。第2に、あるノードに対するスタイリングタスクは、そのノードの親に対するスタイリングタスクの実行が完了した後にのみ、実行され得る。これは、ノードのスタイルプロパティの一部がノードの親から受け継がれ得るからである。たとえば、CSSコードp {color: inherit}は、親と同じ前景の色を使用して<p>ノードをレンダリングするようにブラウザに命令する。

【０２０７】

マッチングタスクによって実行されるルールマッチング動作は、計算、電力、レイテンシなどの点で高価であり得る。たとえば、CSSエンジン512が、ルール「h1 p div {color: red}」が 要素Eに当てはまるかどうかを判定する必要がある場合、マッチングアルゴ

リズムは、Eの先祖のいずれかが<p>要素であるかどうか、および、<p>の先祖のいずれかが<h1>要素であるかどうかを見出す必要があり得る。これには、DOMツリーをわざわざルートまで走査することが必要であることがあり、これは高価な動作であり得る。加えて、通常のウェブサイトは、そのような400,000万個を超えるDOMツリーの走査を必要とし得る。

【０２０８】

DOMツリーの走査の数を減らすために、様々な態様は、DOMノードの先祖についての情報を記憶する、ブルームフィルタを含み得る。ブルームフィルタは、ルート(A)へのDOMツリーの走査の数を90%減らすことができ、スタイリングアルゴリズムに費やされる時間を半分にする。

【０２０９】

ブルームフィルタは、大きなデータ構造であることがあり、CSSエンジン512は、各スタイリングタスクに対してそれをコピーすることが必要とされ得る。コピーのコストは、性能の向上をはるかに上回り得るので、様々な態様は、ブルームフィルタよりも小さな構造を使用し得る。このことは、コピー動作の数を減らすことおよび/またはコピーされる要素のサイズを減らすことによって、ブラウザ性能を改善し得る。

【０２１０】

上で説明されたように、様々な態様は、要素idおよびclassの属性を使用して、CSSファイルで参照される画像がプリフェッチされるべきであるかどうかを予測することができる。ある態様では、これらの要素および属性は、それらの各々がドキュメントにおいて何回現れるかを記録するデータベースに記憶され得る。HTMLパーサはまた、情報をこのデータベースに追加することができる。

【０２１１】

ルールマッチングアルゴリズムが開始する前に、CSSエンジン512は、データベース中の項目をその頻度に従って分類することができる。ブラウザシステム500は次いで、ビットマップデータ構造(「マッチングビットマップ」と呼ばれる)中の各項目にビットを割り当てることができる。idおよびclassの数がビットマップサイズより大きい場合、単一のビットが複数の項目に割り当てられ得る。これらのビットマップは小さいので、ビットマップは、コンピューティングデバイスの性能に大きく影響を与えることなく、多くの回数コピーされ得る。

【０２１２】

ルールマッチング動作の間、各スタイリングタスクは、その親からマッチングビットマップを受け取り得る。マッチングビットマップは、その先祖のid、class、およびタグを記録することができる。スタイリングタスクは、マッチングビットマップを使用して、決して一致し得ないルールを除外することができる。その後、スタイリングタスクは、先祖のノードのid、class、およびタグをマッチングビットマップに追加し、コピーを先祖の子孫に送ることができる。平均すると、そのようなマッチングビットマップは、DOMツリーのルートへの走査の90%を防ぎ、誤った肯定的結果はわずか0.024%である。

【０２１３】

誤った肯定的結果は、マッチングビットマップが、ラベルおよびidに遭遇する順序を記録していないことが原因で発生し得る。たとえば、ルール「h2 h1 p {color: red}」があるノード<p>に当てはまるかどうか、および、<h1>と<h2>の両方が<p>の先祖であることをマッチングビットマップが示すことを判定するために、ブラウザシステム500は、DOMツリーを走査して、<h2>が<h1>の先祖であるかどうかを確認することが必要とされ得る。先祖ではない場合、それは誤った肯定的結果の状況である。そのような誤った肯定的結果により、ページが誤ってレンダリングされることはあり得ないが、CPUサイクルが無駄になり得る。

【０２１４】

ある態様では、レンダリングエンジンサブシステム560などによるレイアウトおよびレンダリング動作は、スタイリングされたDOMを、スクリーンへの表示のためにビットマッ

プ画像へと変換する、計算を実行するステップを含み得る。ビットマップ画像に適用されるDOMおよびCSSスタイルは、新たなツリー構造(レイアウトツリーと呼ばれる)を形成するために結合されてよく、この構造において、各ノードはウェブページ上の視覚的要素を表す。各DOMノードは、0個、1個、または多数のレイアウトツリーノードに変換され得る。レンダリングエンジンサブシステム560は、入力としてレイアウトツリーを扱い、各要素が占めるページの領域を計算することができる。各要素のスタイルは、レイアウトに対する制約(たとえば、インライン/ブロック表示、フロート、幅、高さなど)として見なされ得る。

【０２１５】

レンダリングエンジンサブシステム560は、レイアウトツリーを詳細に検討し、制約を解決して(たとえば、レイアウト動作582の一部として)、各要素の最終的な幅、高さ、および位置を決定することができる。レンダリングエンジンサブシステム560はまた、レイアウトツリー(これは、レイアウトエンジンの計算の結果により注釈を加えられ得る)にわたって走査して(たとえば、レンダリング動作584の一部として)、CSSのルールに従って画面上に要素を描くことができる。

【０２１６】

レイアウト動作582およびレンダリング動作584は、密接に関連しており、パイプライン方式で一緒に動作するので、ある態様では、それらは、レイアウトおよびレンダリングエンジン516のような単一のコンポーネントによって実行され得る。

【０２１７】

様々な態様では、レンダリングエンジンサブシステム560は、CSSレイアウトアルゴリズムがレイアウトツリーにわたって4つのパスで実行されるように、レイアウト動作582を実行するように構成され得る。各パスにおいて、情報は、従来の手法よりも制御された方法でツリーの中を流れることができ、レイアウトプロセスにおける並列性の可能性を見えるようにする。

【０２１８】

ある態様では、レンダリングエンジンサブシステム560は、レイアウトツリーに対して、最小の幅または好ましい幅の計算のパス、幅計算のパス、ブロック整形文脈フローのパス、および絶対位置の計算という、4つのパスを実行し得る。

【０２１９】

第1のパス(すなわち、最小の幅または好ましい幅の計算のパス)は、幅をツリーの上方向へと伝搬させて、最小の幅および好ましい幅を各要素に割り当てる、ボトムアップのパスであり得る。例として、テキストの段落を含むdiv要素に対しては、最小の幅は、各ワードの後に配置される改行としての幅であってよく、好ましい幅は、改行をまったく伴わない幅であってよい。

【０２２０】

第2のパス(すなわち、幅計算のパス)は、各要素の最終的な幅を計算するトップダウンのパスであり得る。要素のスタイルに応じて、最終的な幅は、要素の親の幅と、最小の/好ましい幅とのいずれかから導出され得る。

【０２２１】

第3のパス(すなわち、ブロック整形文脈フローのパス)の間に、各要素は既知の幅を有し、要素の内容が要素の高さを計算するために使用され得る。例として、テキストの段落を含むdiv要素に対しては、幅が決定された後で、テキストがその幅の中に配置されてよく、divの全体の高さを求めるために各行の高さが合計され得る。伝搬の方向は複雑であり得る。その内容が高さを計算するために使用される要素は、ブロック整形文脈(BFC)と呼ばれ得る。要素がブロック整形文脈かどうかは、そのCSSスタイルによって判定され得る。

【０２２２】

DOMツリー中のブロック整形文脈要素は、DOMに重畳され得る論理ツリーを形成し得る。ブロック整形文脈の重畳ツリーは、ボトムアップに走査されてよく、ブラウザシステムが

DOMツリーのルートに到達するときまでには、ブラウザシステムはウェブページ全体をレイアウトしている。この段階の最後において、ブラウザシステム500は、すべての要素の高さとともに、要素を含むブロック整形文脈内でのブロックの相対的な位置を知らされる。

【０２２３】

第4のパス(すなわち、絶対位置計算のパス)は、以前のパスからの各ブロック整形文脈内での相対的な位置を使用して、ページ上での各要素の絶対的な位置を計算する、トップダウンのパスであり得る。

【０２２４】

ある態様では、レンダリングは、前景要素よりも前に背景要素に到達するように、レイアウトツリーを走査することによって達成され得る。様々な態様は、要素のスタイルと適合する方式で各要素をグラフィクスバッファへと描き、バッファの内容をスクリーンに(たとえば、GUIを介して)表示することができる。これらのレンダリング動作は、合成ステップによって使用されるメモリの帯域幅が原因で、計算的に高価であり得る。様々な態様が、並列性または様々なコンポーネント/サブシステムの同時実行を介して、各合成ステップによって必要とされるメモリ帯域幅を減らすように構成され得る。

【０２２５】

一般に、レイアウトおよびレンダリング動作の性能は、ページロード時間からユーザインターフェースの応答性に至るまで、すべてのことに影響があるので重要である。加えて、レイアウトおよびレンダリング動作は、JavaScript(登録商標)の実行のような、他の重要なタスクとCPUサイクルを争う。

【０２２６】

順次的な最適化とともに、様々な態様は、レイアウトおよびレンダリングエンジンの性能を向上させるために、粗い並行性と精密な並行性の両方を含み得る。これらの2つの手法は相補的であり得る。粗いレベルでは、ある態様のブラウザは、クリティカルパスから、かつワーカースレッドへと、可能な限り多くの作業を移動させることができる。精密なレベルでは、その態様のブラウザは、レイアウトおよびレンダリングアルゴリズム/メソッドを並列化することができる。

【０２２７】

従来のウェブブラウザでは、DOMを操作するタスク(たとえば、解析またはJavaScript(登録商標))は、レイアウトおよびレンダリングのタスクとは決して同時には実行されず、これにより、これら2つが互いに干渉しないことを確実にしている。対照的に、様々な態様は、これらの2つのタイプのタスクを重複させる。したがって、様々な態様において、レイアウトツリーは、DOMが変化するたびに更新されなくてよい。

【０２２８】

様々な態様は、レイアウトツリーとDOMを分離する(または分離された状態に保つ)ことができる。レイアウトツリーへの更新は、レイアウトおよびレンダリング動作が通常行われるであろうときに、バッチ動作として実行されてよく、これは、解析またはJavaScript(登録商標)実行のタスクが完了した後であることが多い。この方式で更新をグループ化することは、ブラウザシステム500が、変化したDOMの部分を特定するために追加の状態情報を保持することが必要とされ得るが、レイアウトツリーがDOMの各々の中間状態に対して更新されないので不必要な作業の実行を回避することを、意味し得る。

【０２２９】

様々な態様は、有用な作業の準備ができると、レイアウトツリーを結果によって更新することができる。レイアウトツリーは、DOMとは別個のエンティティであってよい。すべてのDOMの変更は、レイアウトツリーに影響を与えることなく実行され得る。逆に、レンダリングエンジンサブシステム560は、レイアウトツリーが更新されると、何らDOMにアクセスする必要がない。このことは並列性を可能にし、また、従来はDOMのみに記憶されたであろうある情報をレイアウトツリーが複製しなければならないことを意味する。具体的には、レイアウトツリーは、テキスト、画像、CSSスタイル、およびHTMLキャンバス要素

に対する直接的な参照を含み得る。

【０２３０】

テキストおよび画像は、変更不可能であり、安全にDOMと共有され得る。CSSスタイルは論理的には変更不可能であり得るが、CSSスタイルオブジェクト中のデータの量は、オブジェクト全体を毎回コピーするには大きすぎることがある(かつ/または、更新が頻繁すぎることがある)。したがって、ある態様では、各スタイルオブジェクトは、多くのより小さなサブスタイルオブジェクトへと内部的に分割され得る。共有されるサブスタイルは、コピーオンライト手法を使用して更新され得る。共有されないサブスタイルは、適当なときに更新され得る。したがって、スタイルオブジェクトをコピーするには、同じサブスタイルを共有する新たなスタイルオブジェクトを作成することだけが必要であってよく、これははるかに安価であり得る。加えて、一緒に更新されたCSSプロパティが同じサブスタイルの中にあるように、サブスタイルがグループ化されてよく、これは、更新が行われるときのサブスタイルのコピーを最小限にし得る。この構成により、ある位置/コンポーネントで行われる変更が他の位置/コンポーネントから見えない状態で、DOM、レイアウト、およびレンダリングコンポーネントが同じCSSスタイルを参照することが可能になる。同様のコピーオンライト手法が、HTMLキャンバス要素のために使用され得る。

【０２３１】

DOMツリーからのレイアウトツリーの分離は、レンダリングエンジンサブシステム560における、粒度の粗い並列性を可能にする。ウェブページが初めてユーザに対して表示される準備ができると、ブラウザシステム500は、レイアウトツリーを初期化してそれを処理のためにレンダリングエンジンサブシステム560に渡す、作業項目を作成することができる。レイアウトおよびレンダリング動作を異なるスレッドへと分離することで、ブラウザシステム500の残りが処理を進めることが可能になり、たとえばJavaScript(登録商標)を実行すること、ユーザインターフェース(UI)イベントを処理すること、CSSスタイリングを計算することなどができる。

【０２３２】

レンダリングエンジンサブシステム560がタスクを終え、スクリーンにページを表示すると、レンダリングエンジンサブシステム560は「LR作業項目」を提出してレイアウトツリーを更新し、処理を再び最初から始めることができる。「LR作業項目」のみがDOMへの独占的なアクセス権を必要とし、ツリーが更新されると、他の動作は並列にかつ/または非同期的に実行され得る。

【０２３３】

いくつかのJavaScript(登録商標) DOM API(たとえば、getComputedStyleおよびoffsetTop)は、レイアウトアルゴリズムが計算する結果についての情報を必要とし得る。レンダリングエンジンサブシステム560は、結果が利用可能になるまで休止することを求められ得る。レンダリングエンジンサブシステム560がメインスレッドでレイアウトを実行する場合、レンダリングエンジンサブシステム560は、LR作業項目(またはLRスレッド)で実行されている計算を重複して行うことがあり、これは時間とエネルギーを無駄にし得る。

【０２３４】

ある態様では、レンダリングエンジンサブシステム560は、レイアウトツリーが最新のレイアウト情報を有するかどうかを記憶するように構成され得る。最新のレイアウト情報を有する場合、同期的なレイアウト要求が直ちに返され得る。最新のレイアウト情報を有さない場合、レイアウト動作はいつもの通りLRスレッドで実行されてよく、レンダリングエンジンサブシステム560は、レイアウトプロセスが完了したときをメインスレッドに通知するように求められ得る。これにより、重複した作業を防ぎながら、可能な限り迅速に必要な結果が伝えられる。

【０２３５】

並列性に加えて、レイアウトツリーとDOMを分離することの別の利点には、レンダリングエンジンサブシステム560がウェブページ間で共有されるサービスとして扱われ得るということがある。レイアウトツリーはレイアウトツリーの構築の元となったDOMを参照し

ないので、同じレンダリングエンジンサブシステム560が、すべてのレイアウトツリーを、その源と無関係に管理することができる。これは、グラフィクスバッファのような、高価で有限のレンダリング関連のリソースが、ブラウザシステム500全体において1つのインスタンスしか必要ではないことを意味する。

【０２３６】

レイアウトツリーによってもたらされるさらに別の利点には、高速に変化しているページとユーザが対話するときにユーザの意図を判定する際の柔軟性の向上がある。たとえば、JavaScript(登録商標)によって画面中を動いているボタンをユーザがクリックする場合、DOMを変化させるJavaScript(登録商標)と、画面上に現れる結果との間には遅延があり、それは、レイアウトおよびレンダリング動作に時間がかかるからである。ユーザのクリックが登録されるときまでに、DOMは更新されている可能性があり、ブラウザから見たボックスの位置が変化している可能性がある。ユーザのマウスポインタがボックスのすぐ上にある場合であっても、クリックの試みが成功しないことがある。しかしながら、レイアウトツリーがDOMから分離されているので、ブラウザシステム500は、現在のワーキングツリーおよび画面に表示された最後のツリーへのアクセス権を有し得る。これにより、ブラウザシステム500は、DOMの現在の状態ではなく、ユーザがクリックしたときにユーザが見ていたものに基づいて、ユーザがクリックすることを意図していたオブジェクトを判定することが可能になり、知覚される応答性を向上させ、ユーザ体験をより良くする。

【０２３７】

様々な態様が、種々のモバイルコンピューティングデバイス上に実装されてよく、それらの一例が図12に示されている。具体的には、図12は、態様のいずれかとともに使用するのに適したスマートフォン/携帯電話1200の形式の、モバイル送受信機デバイスのシステムブロック図である。携帯電話1200は、内部メモリ1202と、ディスプレイ1203と、スピーカー1208とに結合された、プロセッサ1201を含み得る。加えて、携帯電話1200は、プロセッサ1201に結合されたワイヤレスデータリンクおよび/またはセルラー電話送受信機1205に接続され得る、電磁放射を送信および受信するためのアンテナ1204を含み得る。携帯電話1200は、通常、ユーザ入力を受け取るためのメニュー選択ボタンまたはロッカースイッチ1206も含む。

【０２３８】

一般的な携帯電話1200はまた、マイクロフォンから受け取られた音をワイヤレス送信に適したデータパケットへとデジタル化し、受け取られた音のデータパケットを復号し、スピーカー1208に供給され音を発生させるアナログ信号を生成する、音声符号化/復号(CODEC)回路1213を含む。また、プロセッサ1201、ワイヤレス送受信機1205およびCODEC 1213のうちの1つまたは複数は、デジタルシグナルプロセッサ(DSP)回路(個別に示されず)を含み得る。携帯電話1200はさらに、ワイヤレスデバイス間の低電力短距離通信のためのZigBee(登録商標)送受信機(すなわち、IEEE 802.15.4送受信機)1213、または他の同様の通信回路(たとえば、Bluetooth(登録商標)またはWiFiプロトコルなどを実装する回路)を含み得る。

【０２３９】

様々な態様は、図13に示されるサーバ1300のような、種々の市販のサーバデバイスのいずれにおいても実装され得る。そのようなサーバ1300は通常、揮発性メモリ1302と、ディスクドライブ1303のような大容量の不揮発性メモリとに結合された、プロセッサ1301を含む。サーバ1300はまた、プロセッサ1301に結合されたフロッピー(登録商標)ディスクドライブ、コンパクトディスク(CD)またはDVDディスクドライブ1311を含み得る。サーバ1300はまた、他の通信システムコンピュータおよびサーバに結合されたローカルエリアネットワークなどのネットワーク1305とデータ接続を確立するための、プロセッサ1301に結合されたネットワークアクセスポート1306を含み得る。

【０２４０】

他の形式のコンピューティングデバイスも、様々な態様の恩恵を受け得る。そのようなコンピューティングデバイスは、通常、例示的なパーソナルラップトップコンピュータ14

00を示す図14に示されたコンポーネントを含む。そのようなパーソナルコンピュータ1400は、一般に、揮発性メモリ1402、およびディスクドライブ1403などの大容量不揮発性メモリに結合されたプロセッサ1401を含む。コンピュータ1400はまた、プロセッサ1401に結合されたコンパクトディスク(CD)および/またはDVDドライブ1404を含み得る。コンピュータデバイス1400はまた、プロセッサ1401をネットワークに結合するためのネットワーク接続回路1405などの、データ接続を確立し外部メモリデバイスを受け入れるための、プロセッサ1401に結合されたいくつかのコネクタポートを含み得る。コンピュータ業界ではよく知られているように、コンピュータ1400はさらに、キーボード1408、マウスなどのポインティングデバイス1410、およびディスプレイ1409と結合され得る。

【０２４１】

プロセッサ1201、1301、1401は、以下で説明される様々な態様の機能を含む、様々な機能を実施するためのソフトウェア命令(アプリケーション)によって構成され得る、任意のプログラマブルマイクロプロセッサ、マイクロコンピュータ、または1つもしくは複数の多重プロセッサチップであり得る。いくつかのモバイルデバイスでは、1つのプロセッサをワイヤレス通信機能専用にし、1つのプロセッサを他のアプリケーションの実行専用にするなどして、複数のプロセッサ1301が設けられ得る。通常、ソフトウェアアプリケーションは、アクセスされてプロセッサ1201、1301、1401にロードされる前に、内部メモリ1202、1302、1303、1402に記憶され得る。プロセッサ1201、1301、1401は、アプリケーションソフトウェア命令を記憶するのに十分な内部メモリを含み得る。

【０２４２】

様々な態様は、任意の数のシングルプロセッサシステムまたはマルチプロセッサシステムで実装され得る。一般に、複数のプロセスが単一のプロセッサで同時に実行されているように見えるように、プロセスがプロセッサ上で短いタイムスライスの間に実行される。プロセスがタイムスライスの終わりにおいてプロセッサから除去されると、プロセスの現在の動作状態に関する情報がメモリに記憶されるので、プロセスは、プロセッサ上での実行に戻るときに、シームレスに動作を再開することができる。この動作状態データは、プロセスのアドレス空間、スタック空間、仮想アドレス空間、レジスタセットイメージ(たとえば、プログラムカウンタ、スタックポインタ、命令レジスタ、プログラムステータス語など)、アカウンティング情報、パーミッション、アクセス制限、および状態情報を含み得る。

【０２４３】

プロセスは他のプロセスをスポーンすることがあり、スポーンされたプロセス(すなわち、子プロセス)は、スポーンしたプロセス(すなわち、親プロセス)のパーミッションおよびアクセス制限(すなわち、コンテキスト)の一部を受け継ぐことがある。プロセスは、複数の軽量のプロセスまたはスレッドを含む重量のプロセスであってよく、軽量のプロセスは、そのコンテキスト(たとえば、アドレス空間、スタック、パーミッション、および/またはアクセス制限など)のすべてまたは部分を他のプロセス/スレッドと共有するプロセスである。したがって、単一のプロセスは、単一のコンテキスト(すなわち、プロセッサのコンテキスト)を共有する、それへのアクセス権を有する、かつ/またはその中で動作する、複数の軽量のプロセスまたはスレッドを含み得る。

【０２４４】

上記の方法の説明およびプロセスフロー図は、単に説明のための例として提供され、様々な態様のブロックが提示された順序で実施されなければならないことを要求または暗示するものではない。当業者が諒解するように、上記の態様におけるブロックの順序は、どのような順序で実行されてもよい。「その後」、「次いで」、「次に」などの単語は、ブロックの順序を限定するものではなく、これらの単語は、方法の説明を通して読者を案内するために使用されるにすぎない。さらに、たとえば、冠詞「a」、「an」または「the」の使用による単数形での請求要素へのいかなる言及も、その要素を単数に限定するものとして解釈されるべきではない。

【０２４５】

本明細書で開示された態様に関して説明された、様々な例示的な論理ブロック、モジュール、回路、およびアルゴリズムブロックは、電子ハードウェア、コンピュータソフトウェア、または両方の組合せとして実装され得る。ハードウェアとソフトウェアのこの互換性を明確に示すために、様々な例示的なコンポーネント、ブロック、モジュール、回路、およびブロックが、上記では概してそれらの機能に関して説明された。そのような機能がハードウェアとして実装されるか、またはソフトウェアとして実装されるかは、具体的な適用例および全体的なシステムに課される設計制約に依存する。当業者は、説明された機能を具体的な適用例ごとに様々な方法で実装し得るが、そのような実装の決定は、本発明の範囲からの逸脱を生じるものと解釈すべきではない。

【０２４６】

本明細書で開示された態様に関して説明された様々な例示的な論理、論理ブロック、モジュール、および回路を実装するために使用されるハードウェアは、汎用プロセッサ、デジタルシグナルプロセッサ(DSP)、特定用途向け集積回路(ASIC)、フィールドプログラマブルゲートアレイ(FPGA)もしくは他のプログラマブル論理デバイス、個別ゲートもしくはトランジスタ論理、個別のハードウェアコンポーネント、または、本明細書で説明された機能を実行するように設計されたそれらの任意の組合せで、実装または実行され得る。汎用プロセッサはマイクロプロセッサであり得るが、代替として、そのプロセッサは任意の従来のプロセッサ、コントローラ、マイクロコントローラ、または状態機械であり得る。プロセッサは、コンピューティングデバイスの組合せ、たとえば、DSPとマイクロプロセッサとの組合せ、複数のマイクロプロセッサ、DSPコアと連携する1つもしくは複数のマイクロプロセッサ、または任意の他のそのような構成としても実装され得る。代替として、いくつかのブロックまたは方法は、所与の機能に固有の回路によって実行され得る。

【０２４７】

1つまたは複数の例示的な態様では、説明された機能は、ハードウェア、ソフトウェア、ファームウェア、またはそれらの任意の組合せで実装され得る。機能は、ソフトウェアで実装される場合、1つまたは複数の命令またはコードとして、非一時的コンピュータ可読媒体または非一時的プロセッサ可読媒体に記憶され得る。本明細書で開示された方法またはアルゴリズムのステップは、非一時的コンピュータ可読記憶媒体またはプロセッサ可読記憶媒体上に存在し得るプロセッサ実行可能ソフトウェアモジュールで具現化され得る。非一時的コンピュータ可読記憶媒体またはプロセッサ可読記憶媒体は、コンピュータまたはプロセッサによってアクセスされ得る任意の記憶媒体であってよい。限定ではなく例として、そのような非一時的コンピュータ可読媒体またはプロセッサ可読媒体は、RAM、ROM、EEPROM、フラッシュメモリ、CD-ROMもしくは他の光ディスク記憶装置、磁気ディスク記憶装置もしくは他の磁気記憶デバイス、または、命令もしくはデータ構造の形式で所望のプログラムコードを記憶するために使用され得るとともに、コンピュータによってアクセスされ得る任意の他の媒体を含み得る。本明細書で使用されるディスク(disk)およびディスク(disc)は、コンパクトディスク(disc)(CD)、レーザディスク(disc)、光ディスク(disc)、デジタル多用途ディスク(disc)(DVD)、フロッピー(登録商標)ディスク(disk)、およびブルーレイディスク(disc)を含み、ディスク(disk)は、通常、磁気的にデータを再生し、ディスク(disc)は、レーザで光学的にデータを再生する。上記の組合せも、非一時的コンピュータ可読媒体およびプロセッサ可読媒体の範囲内に含まれる。加えて、方法またはアルゴリズムの動作は、コンピュータプログラム製品に組み込まれ得る、非一時的プロセッサ可読媒体および/またはコンピュータ可読媒体上のコードおよび/または命令の、1つまたは任意の組合せ、またはそのセットとして存在し得る。

【０２４８】

開示された態様の上記の説明は、任意の当業者が本発明を製作または使用できるように提供されたものである。これらの態様への様々な修正が当業者には容易に明らかであり、本明細書で定義された一般的な原理は、本発明の趣旨または範囲を逸脱することなく、他の態様に適用され得る。したがって、本発明は、本明細書で示された態様に限定されるものではなく、以下の特許請求の範囲ならびに本明細書で開示された原理および新規の特徴

に合致する、最も広い範囲を与えられるべきである。

【符号の説明】

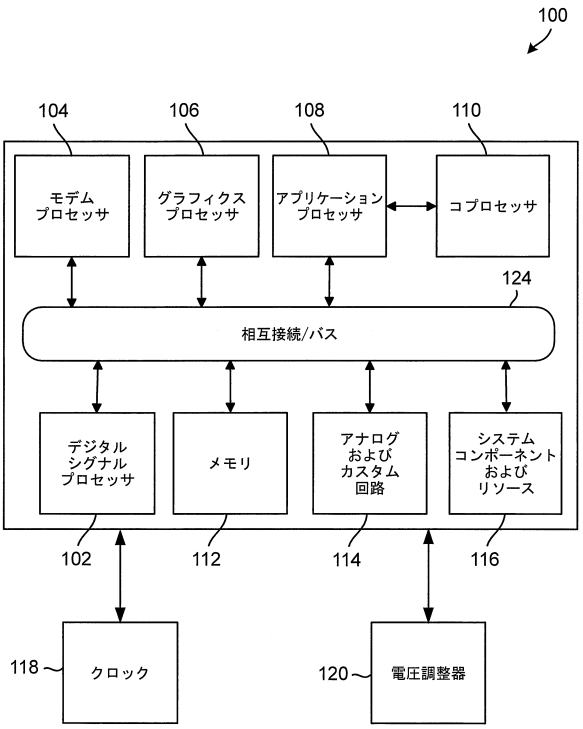
【０２４９】

100 システムオンチップ
102 デジタルシグナルプロセッサ
104 モデムプロセッサ
106 グラフィクスプロセッサ
108 アプリケーションプロセッサ
110 コプロセッサ
112 メモリ
114 アナログおよびカスタム回路
116 システムコンポーネントおよびリソース
118 クロック
120 電圧調整器
124 相互接続/バス
202 マルチコアプロセッサ
204 コア0
206 コア1
208 処理ユニット
210 処理ユニット
212 L1キャッシュ
214 L1キャッシュ
216 L2キャッシュ
218 バス/相互接続インターフェース
220 メインメモリ
222 入力/出力モジュール
224 外部メモリ/ハードディスク
226 L2キャッシュ
230 コア2
232 コア3
234 処理ユニット
236 処理ユニット
238 L1キャッシュ
240 L1キャッシュ
242 L2キャッシュ
350 ブラウザシステム
352 ウェブ
356 プログラミング命令
360 HTMLコード
362 スクリプトコンポーネント
366 ドキュメントオブジェクトモデル
370 修正されたDOMツリー
380 外部モジュール
500 ブラウザシステム
502 フェッチマネージャ
504 DOMディスパッチャ
506 HTMLパーサ
508 HTMLプレスキャナ
510 画像デコーダ
512 CSSエンジン

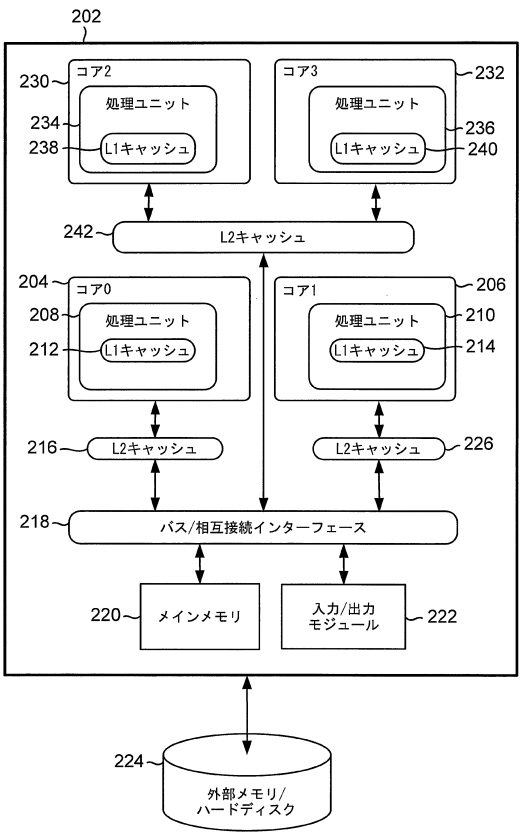
514 JavaScript(登録商標)エンジン
516 レイアウトおよびレンダリングエンジン
518 ユーザインターフェース
520 CSSリソースプリフェッチャ
522 CSSパーサ
524 DOMスタイラ
526 ライトコンパイラ
528 フルコンパイラ
530 サンドボックス化されたJavaScript(登録商標)エンジン
552 ユーザインターフェース
554 リソースマネージャ
556 ページ毎DOMエンジン
558 ページ毎JavaScript(登録商標)エンジン
560 レンダリングエンジン
562 プリフェッチ動作
563 HTMLプレスキャン動作
564 画像復号動作
566 CSSスキャン/プリフェッチ動作
567 JavaScript(登録商標)スキャン/プリフェッチ動作
568 HTML解析動作
570 CSS解析動作
571 JavaScript(登録商標)解析
572 タイマー動作
574 スタイリング動作
576 イベント動作
578 実行動作
580 コンパイル動作
582 レイアウト動作
584 レンダリング動作
902 CSSリソースプリフェッチ動作
904 CSS解析動作
906 DOMスタイリング動作
1200 スマートフォン/携帯電話
1201 プロセッサ
1202 内部メモリ
1203 ディスプレイ
1204 アンテナ
1205 ワイヤレス送受信機
1206 ロッカースイッチ
1208 スピーカー
1213 CODEC
1300 サーバ
1301 プロセッサ
1302 揮発性メモリ
1303 ディスクドライブ
1305 ネットワーク
1306 ネットワークアクセスポート
1400 パーソナルラップトップコンピュータ
1401 プロセッサ
1402 揮発性メモリ

1403　ディスクドライブ
1404　CDドライブ/DVDドライブ
1405　ネットワーク接続回路
1408　キーボード
1409　ディスプレイ
1410　ポインティングデバイス

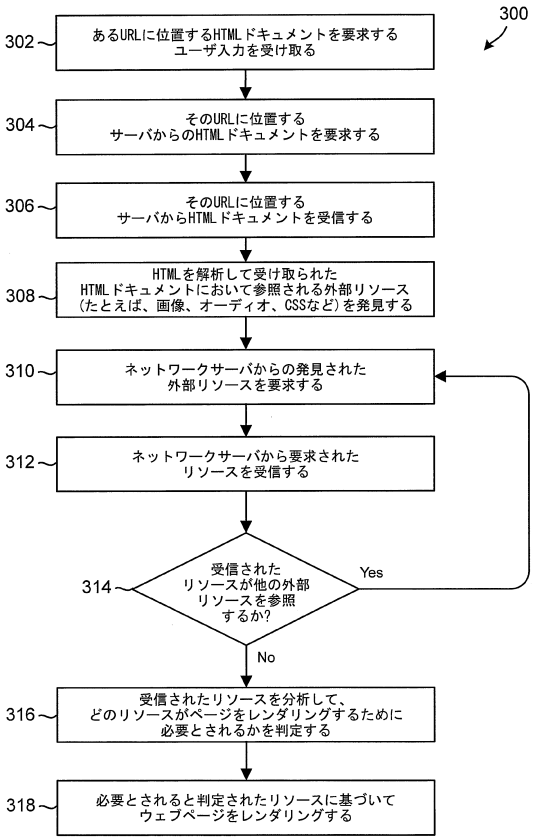
【図１】



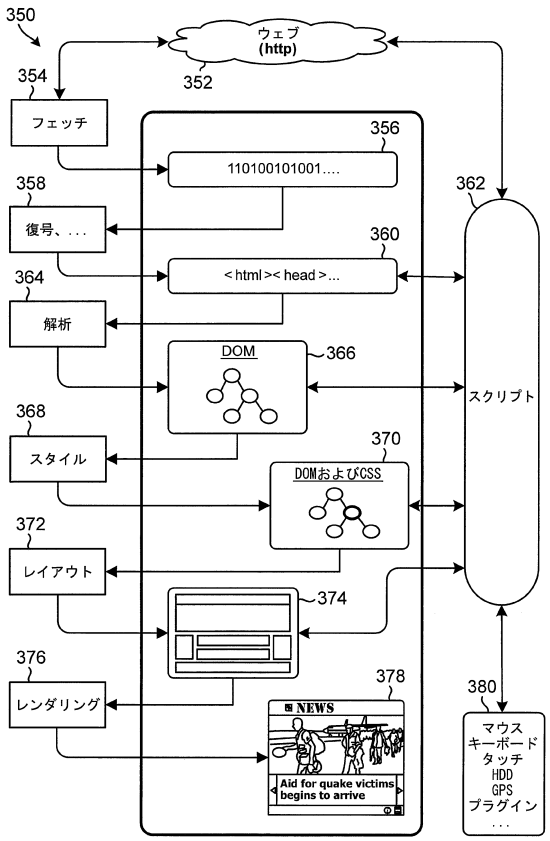
【図２】



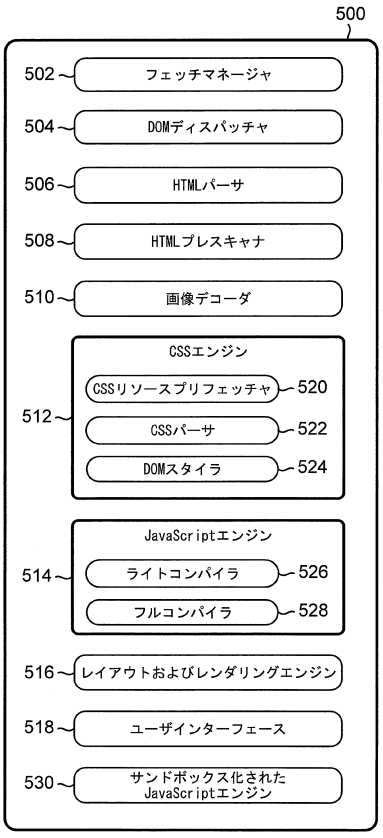
【図３Ａ】


300
302
あるURLに位置するHTMLドキュメントを要求する
ユーザ入力を受け取る
304
そのURLに位置する
サーバからのHTMLドキュメントを要求する
306
そのURLに位置する
サーバからHTMLドキュメントを受信する
308
HTMLを解析して受け取られた
HTMLドキュメントにおいて参照される外部リソース
(たとえば、画像、オーディオ、CSSなど)を発見する
310
ネットワークサーバからの発見された
外部リソースを要求する
312
ネットワークサーバから要求された
リソースを受信する
314
受信された
リソースが他の外部
リソースを参照
するか?
Yes
No
316
受信されたリソースを分析して、
どのリソースがページをレンダリングするために
必要とされるかを判定する
318
必要とされると判定されたリソースに基づいて
ウェブページをレンダリングする

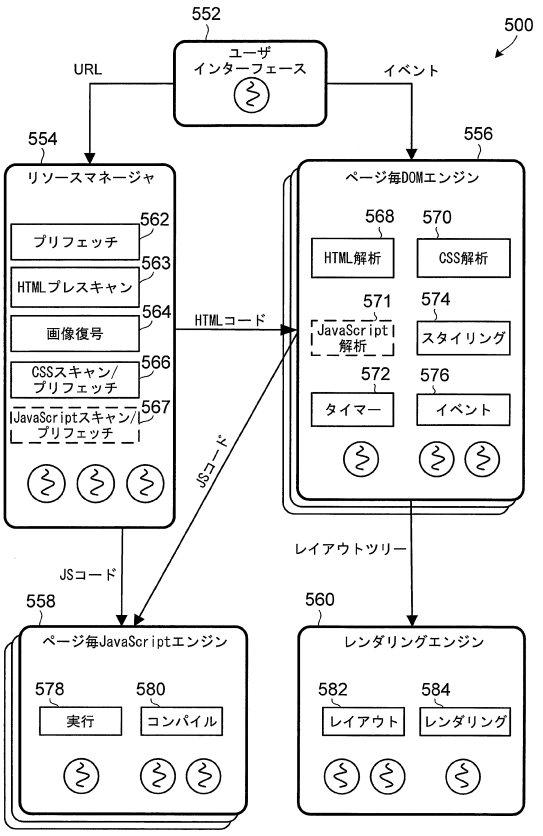
【図３Ｂ】


350
ウェブ
(http)
352
354
フェッチ
356
110100101001....
358
復号、...
360
<html><head>...
362
364
解析
DOM
366
スクリプト
368
スタイル
370
DOMおよびCSS
372
レイアウト
374
376
レンダリング
378
NEWS
Aid for quake victims begins to arrive
380
マウス
キーボード
タッチ
HDD
GPS
プラグイン
...

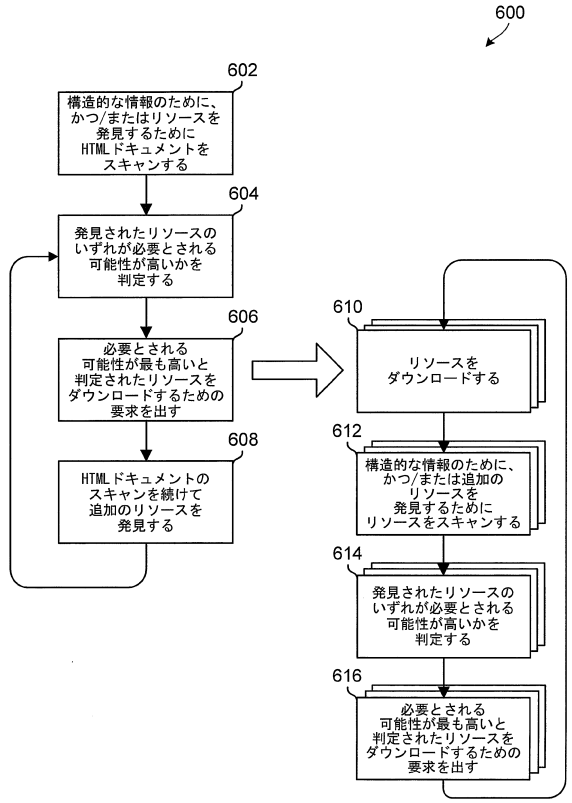
【図４】


500
502
フェッチマネージャ
504
DOMディスパッチャ
506
HTMLパーサ
508
HTMLプレスキャナ
510
画像デコーダ
512
CSSエンジン
CSSリソースプリフェッチャ
520
CSSパーサ
522
DOMスタイラ
524
514
JavaScriptエンジン
ライトコンパイラ
526
フルコンパイラ
528
516
レイアウトおよびレンダリングエンジン
518
ユーザインターフェース
530
サンドボックス化された
JavaScriptエンジン

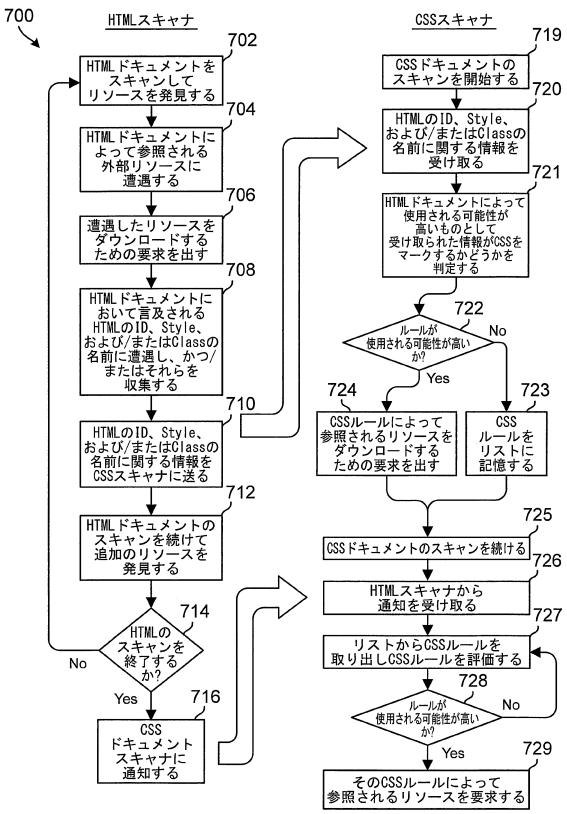
【図５】


500
552
ユーザ
インターフェース
URL
イベント
554
リソースマネージャ
556
ページ毎DOMエンジン
562
プリフェッチ
563
HTMLプレスキャン
564
画像復号
566
CSSスキャン/
プリフェッチ
567
JavaScriptスキャン/
プリフェッチ
HTMLコード
568
HTML解析
570
CSS解析
571
JavaScript
解析
574
スタイリング
572
タイマー
576
イベント
JSコード
JSコード
レイアウトツリー
558
ページ毎JavaScriptエンジン
560
レンダリングエンジン
578
実行
580
コンパイル
582
レイアウト
584
レンダリング

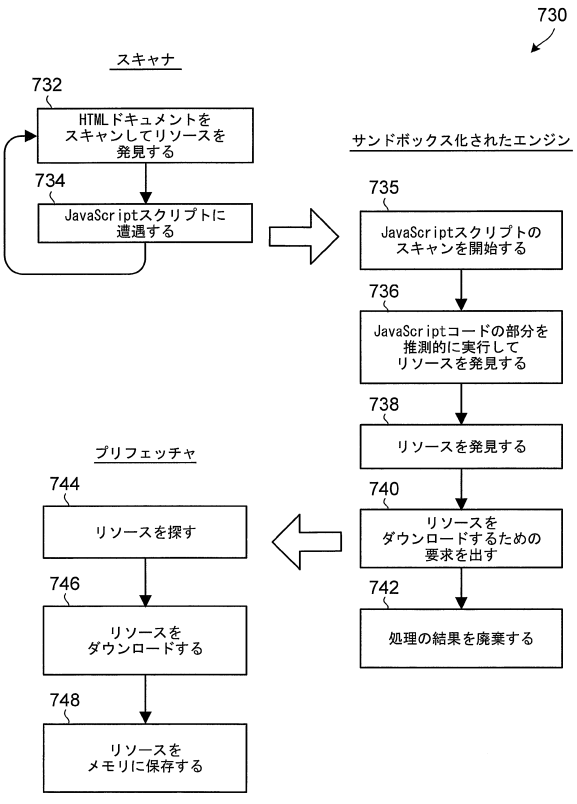
【図６】



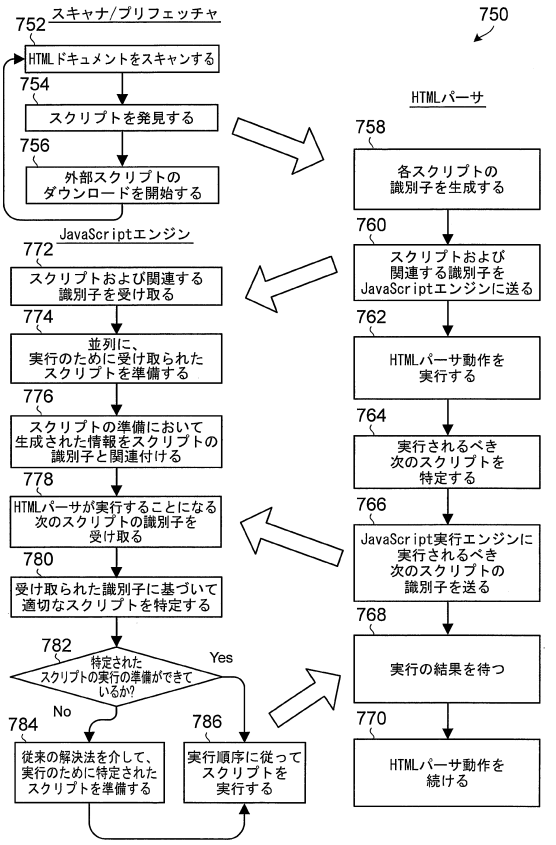
【図７Ａ】



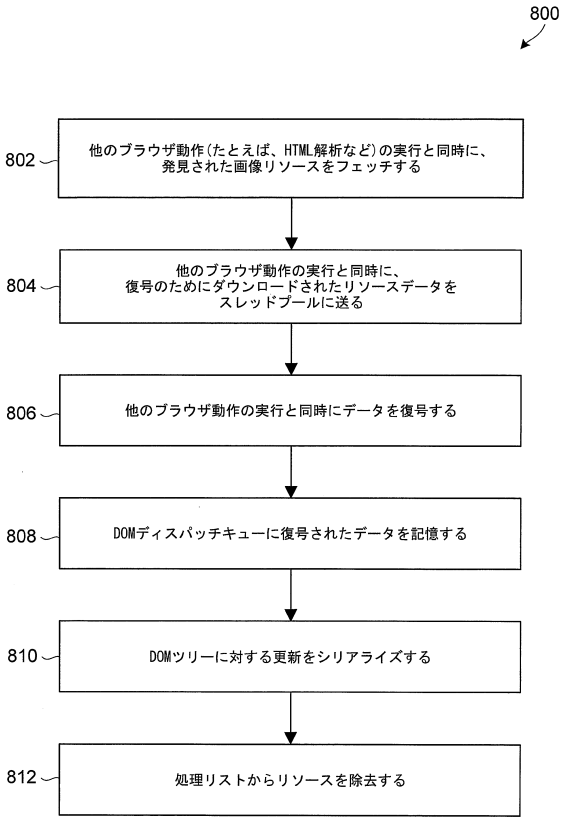
【図７Ｂ】



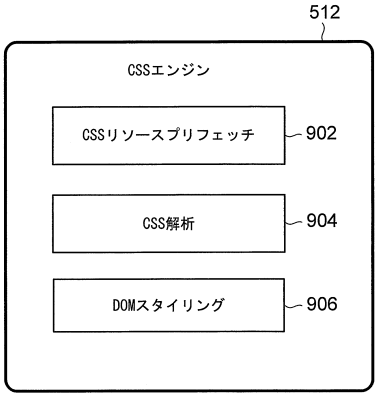
【図７Ｃ】



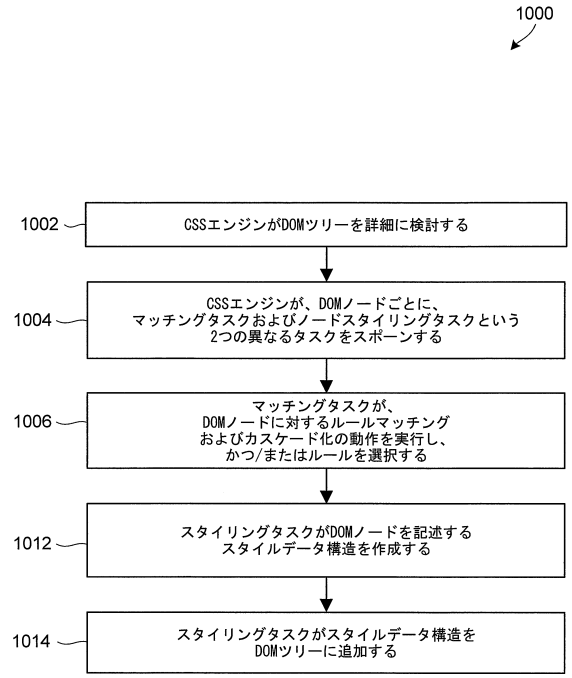
【図８】



【図９】



【図１０】



【図１１Ａ】

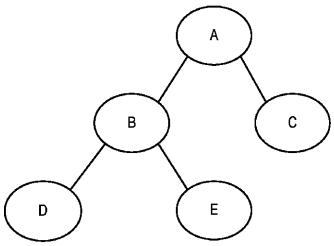
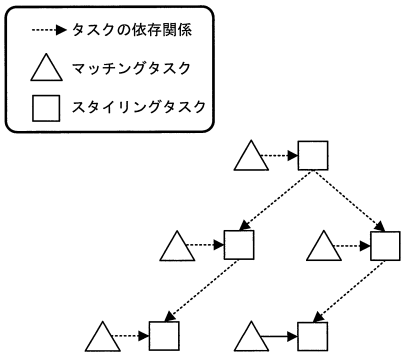


FIG. 11A

【図１１Ｂ】



【図１２】

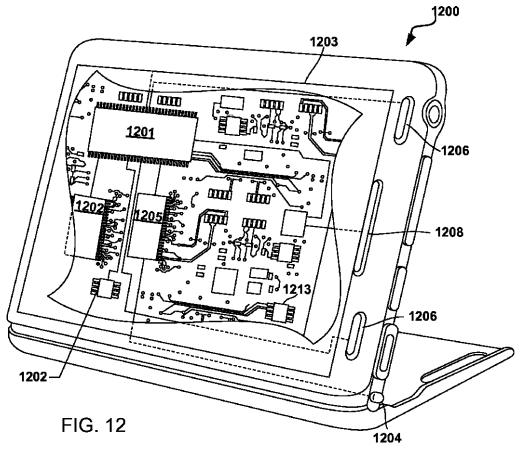


FIG. 12

【図１３】

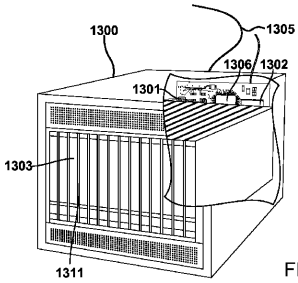


FIG. 13

【図１４】

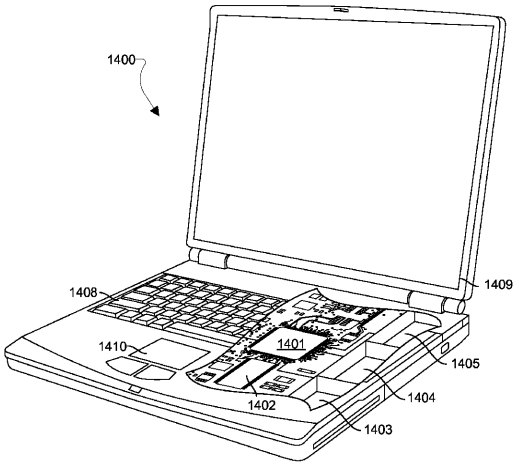


FIG. 14

フロントページの続き

(31)優先権主張番号　13/722,066
(32)優先日　平成24年12月20日(2012.12.20)
(33)優先権主張国　米国(US)

早期審査対象出願

(72)発明者　モハメド・エイチ・レシャディ
アメリカ合衆国・カリフォルニア・９２１２１・サン・ディエゴ・モアハウス・ドライヴ・５７７５
(72)発明者　ゲオルゲ・シー・カスカヴァル
アメリカ合衆国・カリフォルニア・９２１２１・サン・ディエゴ・モアハウス・ドライヴ・５７７５

審査官　木村　雅也

(56)参考文献　米国特許出願公開第２０１１／１８５２７１（ＵＳ，Ａ１）
米国特許出願公開第２０１１／１７３５９７（ＵＳ，Ａ１）
米国特許出願公開第２０１２／１１０４３３（ＵＳ，Ａ１）
米国特許出願公開第２０１１／０８２９８４（ＵＳ，Ａ１）

(58)調査した分野(Int.Cl.，ＤＢ名)
Ｇ０６Ｆ　１３／００