(19) OFICIUL DE STAT
PENTRU INVENŢII ŞI MĂRCI
Bucureşti

ROMÂNIA

(11) **RO 128167 A2**

(51) Int.Cl.
G06F 9/44 (2006.01),
G06F 15/00 (2006.01),
H04N 7/24 (2006.01)

(12) # CERERE DE BREVET DE INVENŢIE

(21) Nr. cerere: **a 2011 00323**

(22) Data de depozit: **07.04.2011**

(41) Data publicării cererii:
28.02.2013 BOPI nr. **2/2013**

(71) Solicitant:
• **CUCU ANDRIAN**, *STR. DUETULUI NR. 57-61, AP.6, SECTOR 1, BUCUREŞTI, B, RO;*
• **TĂNASE ADRIAN**, *STR. TRESTIANA NR. 7, BL. 10, SC. 1, ET. 8, AP. 35, SECTOR 4, BUCUREŞTI, B, RO*

(72) Inventatori:
• **CUCU ANDRIAN**, *STR. DUETULUI NR. 57-61, AP.6, SECTOR 1, BUCUREŞTI, B, RO;*
• **TĂNASE ADRIAN**, *STR. TRESTIANA NR. 7, BL. 10, SC. 1, ET. 8, AP. 35, SECTOR 4, BUCUREŞTI, B, RO*

(74) Mandatar:
CABINET ENPORA S.R.L.,
STR. GEORGE CĂLINESCU NR. 52A, AP. 1, SECTOR 1, BUCUREŞTI

(54) # METODĂ ŞI SISTEME PENTRU SUPORTAREA UNUI API DE RENDERING PRIN UTILIZAREA UNUI MEDIU RUNTIME

(57) **Rezumat:**

Invenţia se referă la o metodă şi la un sistem pentru suportarea unei interfeţe de programare a aplicaţiilor (API) de randare video, prin utilizarea unui mediu de execuţie. Sistemul conform invenţiei cuprinde un dispozitiv de calcul (**202**) ce are în componenţă o interfaţă I/E (**210**) şi un element de prelucrare conectat la interfaţa I/E (**210**), elementul de prelucrare implementând un mediu de execuţie (**128**) şi punând în aplicare o componentă de program care face ca elementul de prelucrare să expună o API de randare, care nu este suportată nativ de către mediul de execuţie (**128**), API de randare putând fi invocată printr-un cod cuprins într-un document de marcare, şi fiind expusă prin determinarea mediului de execuţie (**128**) să răspundă la şi să actualizeze un obiect proxy (**126**) ce reflectă proprietăţile, metodele şi comportamentul definite de API de randare. Metoda conform invenţiei constă din determinarea, prin intermediul unui dispozitiv de prelucrare, a unei stări a unui obiect proxy, păstrată într-un mediu citibil de către calculator, obiectul proxy reflectând atribute ale unei API de randare video, şi, ca răspuns la determinarea stării obiectului proxy şi pe baza stării obiectului proxy, invocarea de către dispozitivul de prelucrare a unei metode suportate de un mediu de execuţie.

Revendicări: 20
Figuri: 4

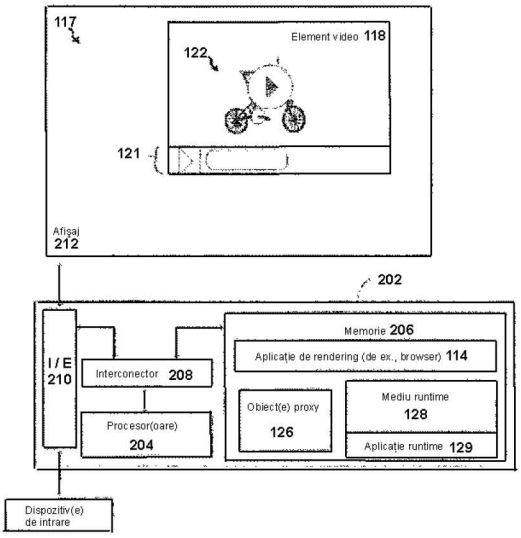

Fig. 2

RO 128167 A2

Stadiul tehnicii mondiale în domeniu

[0001] Utilizatorii de internet consumă cu mult entuziasm conținut video şi alte tipuri de continut multimedia. Un astfel de conținut este adesea distriubuit folosind coduri, cum ar fi codul HTML sau u alt cod de mark-up, configurat pentru a invoca posibilitățile unei aplicații de a recupera conținutul şi de a rendera conținutul în conformitate cu o interfață de programare de aplicație de rendering video (API) expusă de către aplicație. Ca un exemplu, un mediu runtime cum este Adobe® Flash® player sau mediul runtime AIR® (ambele disponibile de la Adobe Systems Incorporated of San Jose, California) poate să fie folosit pentru a rendera video. Cu toate acestea, unii utilizatori pot să nu aibă acces la un dispozitiv care suportă un astfel de mediu runtime.

[0002] Recent, standardul HTML a fost extins pentru a include API-urile de rendering video amintite prin includerea <video> şi al altor elemente în codul HTML prin care poate să fie utilizat un browser pentru a rendera video nativ. În ciuda recentelor expansiuni ale ARI-urilor pentru a adăuga browserelor suport video şi de altă natură, rămâne încă loc suficient pentru o îmbunătățire atât în ceea ce priveşte experiența utilizatorilor cât şi în ceea ce priveşte experiența dezvoltatorilor care lucrează pentru a furniza conținut utilizatorilor.

Rezumat

[0003] Un dispozitiv de calcul poate avea în componență o interfață de intrare-ieşire (I/E) şi un element de procesare conectat la interfața I/E şi care pune în aplicare un mediu runtime. Elementul de procesare poate pune în aplicare o componentă de program care face ca elementul de procesare să expună un API de rendering care nu este nativ suportat de către mediul runtime, API-ul de rendering invocabil prin cod conținut într-un document markup accesat de către procesor prin intermediul interfeței I/E. API-ul de rendering poate să fie expus prin a provoca răspunsul mediului de runtime la şi a actualiza un obiect proxy care vede în oglindă proprietățile, metodele şi comportamentele definite de către API-ul de rendering.

[0004] De exemplu, prin utilizarea obiectului proxy, mediul runtime poate fi făcut să răspundă la codul care invocă comenzile specificate în conformitate cu API <video> HTML. În felul acesta, dezvoltatorii pot să scrie codul care recurge la API <video> cu încrederea că acelaşi cod poate să fie utilizat cu sau fără mediul runtime. În plus, dezvoltatorii pot să scrie codul care încă foloseşte avantajele posibilităţilor mediului runtime care se extind dincolo de posibilităţile browserului sau altei aplicaţii de rendering.

[0005] Aceste modalităţi de realizare ilustrative sunt discutate nu pentru a limita prezentul subiect, ci numai pentru a asigura o scurtă introducere. Modalităţi de realizare suplimentare includ medii citibile pe computer pentru expunerea şi utilizarea funcţionalităţii API de rendering prin intermediul utilizării de obiecte proxy, împreună cu metode puse în aplicare pe computere care fac acelaşi lucru. Aceste modalităţi de realizare cât şi altele sunt descrise în cele ce urmează în cadrul Descrierii detaliate. Obiecte şi avantaje ale prezentului subiect pot să fie determinate după trecerea în revistă a specificaţiei şi/sau aplicaţiei practice a unei modalităţi de realizare configurată în conformitate cu unul sau mai multe aspecte dezvăluite aici.

Scurtă descriere a desenelor

[0006] O expunere detaliată completă şi accesibilă este accesibilă în partea care a rămas din această descriere a brevetului. Descrierea brevetului face referire la următoarele figuri care o însoţesc.

[0007] Figura 1 este o diagramă a fluxului de date care ilustrează despre cum unul sau mai multe obiecte proxy pot să fie folosite pentru ilustra funcţionalitatea API prin intermediul unui mediu runtime.

[0008] Figura 2 este o diagramă care arată un sistem ilustrativ configurat pentru a utiliza un obiect proxy pentru a ilustra funcţionalitatea API prin intermediul unui mediu runtime.

[0009] Figura 3 prezintă paşii unei scheme logice de program a unei metode ilustrative care ilustrează funcţionalitatea API prin intermediul unui obiect proxy şi utilizând o astfel de funcţionalitate.

[0010] Figura 4 prezintă paşii unei scheme logice de program a unei metode ilustrative a utilizării selective a mediului runtime pentru a ilustra un APi de rendering video care nu este suportat în mod nativ de către mediul runtime.

Descrierea detaliată

[0011] Modalităţile de realizare discutate în cele ce urmează includ sisteme de calcul, metode, medii citibile pe calculator care încorporează cod. De exemplu, un API de rendering precum API video „HTML5" poate să fie prevăzut la începutul unui runtime cum ar fi plug-in-ul de la Flash® player astfel încât dezvoltatorii de web pot să utilizeze un API standardizat, compatibil fără a fi nevoie de a negocia cu Flash® player sau cu un alt API specific runtime-ului. În mod suplimentar, pot să fie consolidate resurse (de exemplu, nu este nevoie să ai biblioteci separate atât pentru playerul video nativ HTML cât şi pentru playerul video Flash) atunci când o pagină de web are atât Flash player cât şi video native HTML pentru conţinut video pentru utilizarea de către dispozitive de client diferite. În unele dintre punerile în aplicare, gettere şi settere bazate pe script sunt utilizate pentru a mapa apelări de Apl de rendering pe apelări de runtime şi pentru a furniza răspunsuri de API de rendering care se bazează pe schimbări/evenimente de stare de runtime prin obţinerea şi setarea proprietăţilor văzute în oglindă ale API <video>.

[0012] Întorcându-ne la Figura 1, de exemplu, o diagramă a fluxului de date arată cum pot să fie folosite unul sau mai multe elemente proxy pentru a ilustra funcţionalitatea API care nu este suportată nativ de un mediu runtime.

[0013] În particular, codul 102 include unul sau mai multe elemente 104 care recurg la API de rendering împreună cu unul sau mai multe elemente 106 care apelează un API de scripting pentru a seta obiectele proxy. Codul 102 include de asemenea o referinţă 108 la o aplicaţie runtime 129. În cadrul acestui exemplu, după cum este arătat la 110, apelurile la API de rendering şi apelurile la API de scripting arătate la

112 sunt cel puțin procesate prin intermediul unei aplicații de rendering 114, care reprezintă un browser sau o altă aplicație care parsează codul 102. Aplicația de rendering 114 are de asemenea în componență funcționalitatea corespunzătoare pentru a recunoaşte apelurile API de scripting; de exemplu, aplicația de rendering 114 poate să includă un JavaScript™ sau un alt motor care suportă apelurile de scripting.

[0014] Aplicația de rendering 114 poate să includă funcționalitatea pentru a suporta nativ cel puțin o parte din apelurile API de rendering definite de elementele 104. În felul acesta, după cum este arătat la 116, aplicația de rendering furnizează cel puțin o anumită porțiune sau anumite porțiuni dintr-o interfață grafică de utilizator 117 care are în componență (în acest exemplu) un element video 118 şi butonul de control 120, care poate să fie folosit pentru a manevra evenimente de intrare care controlează redarea conținutului video 122 (de exemplu, pentru play/pauză ca răspuns la intrarea direcționată către butonul triunghiular „play" prezentat aici) şi asigură actualizările pe măsură ce conținutul 122 se schimbă. Ca un alt exemplu al unui element de control, un controlbar 121 cu un scrub bar poate să fie folosit în timpul redării, iar scrub bar-ul poate să fie deplasat de la stânga la dreapta pe măsură ce avansează video-ul.

[0015] În concordanță cu prezentul subiect, totuşi, aplicația de rendering 114 configurează unul sau mai multe obiecte proxy 126 astfel încât, după cum este prezentat la 124, unul sau mai multe apeluri către API de rendering pot să fie manevrate prin intermediul unui mediu runtime 128 care execută/interpretează aplicația runtime 129 astfel încât, în mod efectiv, mediul de runtime 128 expune API de rendering. De exemplu, obiectul proxy 126 poate să fie configurat să aibă aceleaşi nume de proprietate, nume de metode, şi comportament definite de către API de rendering, astfel încât obiectul proxy 126 distribuie aceleaşi evenimente pe baza schimbărilor de stare în video sau al altui conținut renderat după cum ar fi de aşteptat de la o punere în aplicare nativă.

[0016] Obiectul proxy 126 monitorizează evenimentele, cum ar fi intrarea de utilizator sau alte evenimente care recurg la API de rendering pentru a schimba o proprietate asociată cu APi de rendering. După cum este arătat în la 127, ca răspuns

la o schimbare de proprietate, obiectul proxy 126 recurge la funcționalitatea corespondentă a playerului runtime prin intermediul API-ului de apel extern asigurat de către runtime. După cum este arătat de asemenea la 127, obiectul proxy răspunde de asemenea la o schimbare în starea mediului runtime 128 prin propagarea schimbării de stare la una sau la mai multe dintre proprietățile sale şi prin expediarea unui eveniment corespunzător sau unor evenimente corespunzătoare după cum este definit în API de rendering.

[0017] De exemplu, după cum este prezentat la 130, mediul runtime 128 poate să fie utilizat pentru a furniza câteva dintre sau toate elementele interfeței grafice de utilizator 117. Ca un exemplu particular, codul 102 poate să specifice faptul că mediul runtime 128 trebuie să furnizeze elementul video 118 în timp ce posibilitățile API native ale browserului 114 sunt utilizate pentru a rendera butonul de control 120 şi un alt „chrome", ca şi prin utilizarea unui controlbar 121 definit utilizând codul HTML şi CSS. Aceasta poate da posibilitatea de a se utiliza aceleaşi controale pe platforme indiferent dacă acele paltforme suportă cu toatele mediul runtime 128 şi/sau pentru a da posibilitatea de a se utiliza posibilitățile mediului runtime 128 ca un suport pentru multicasting, pentru protecția conținutului (de exemplu, streamingul codat de conținut). În felul acesta, evenimentele direcționate către elementele UI ale controlbar-ului furnizat de către browser pot să fie propagate către obiectele proxy 126 care astfel recurg la funcționalitatea mediului runtime 128. De exemplu, video-ul poate să fie specificat ca un SWF sau un alt fişier care a fost redat de către mediul runtime în cazul în care a fost disponibil.

[0018] Ca răspuns la un eveniment de intrare direcționat către butonul de control 120, starea obiectului proxy 126 poate să fie actualizată. Ca răspuns la schimbarea în ceea ce priveşte starea lui, obiectul proxy 126 poate recurge la funcționalitatea mediului runtime 128, cum ar fi prin furnizarea unei comenzi de play() sau de pauză() expusă de către mediul runtime 128 ca parte a API-ului de control exterior al mediului runtime. În mod similar, pe măsură ce starea mediului runtime 128, de exemplu pe măsură ce indicele de timp al video-ului creşte, indicele de timp poate să fie propagat la obiectul proxy 126 care distribuie evenimentele corespunzătoare pentru a face butonul de control vizibil sau invizibil. În plus față de sau în locul stocării schimbărilor de stare de către obiectul proxy 126, una sau mai multe dintre

proprietăți pot să fie redate în oglindă de obiectul proxy 126 prin intermediul utilizării valorilor de proprietate pentru obiecte asociate cu API-ul de rendering pentru a declanşa comenzi ale mediului runtime 128 prin intermediul metodelor obişnuite de getter şi/sau prin intermediul actualizării valorilor de proprietăți pentru obiecte asociate cu API de rendering ca răspuns la activitatea mediului runtime 128 prin metode de setter obişnuite.

[0019] Ca un alt exemplu, codul 102 poate să recurgă la un API de rendering video pentru a furniza elementul video 118 şi butonul de control 120. Totuşi, pentru a folosi în mod avantajos capacitățile de raportare ale unei aplicații runtime (de exemplu, pentru a detecta dacă utilizatorii sar peste reclame) şi/sau capacitățile grafice avansate ale unui mediu runtime, codul 102 poate să fie configurat cu elementele portivite de scripting 106 astfel încât să fie utilizată o aplicație runtime corespunzătoare pentru a asigura controlbar-ul. De exemplu, codul 102 poate să includă un element care se referă la un SWF sau la un aslt fişier de rutină care furnizează controlbar-ul.

[0020] În măsura în care evenimentele de intrare sunt direcționate către elementele puse la dispoziție de către aplicația de rendering 114, obiectul(ele) proxy 126 poate(pot) să fie actualizat şi utilizat pentru a expedia evenimentele corespunzătoare în conformitate cu API-urile de rendering video. De exemplu, o intrare „play" sau „pauză" către aplicația runtime poate să provoace actualizări corespunzătoare stării obiectului proxy 126, care expediază o comandă potrivită către componentele de rendering ale browserului video pentru a începe redarea sau respectiv pentru a-l trece în pauză.

[0021] Alte evenimente pot să fie utilizate pentru a recurge la API de rendering video. De exemplu, o pagină poate să includă logicul care indică faptul că un video trebuie să fie pus în pauză în timp ce este expediat un anunț, cum ar fi prin expedierea unei comenzi de pauză() atunci când video ajunge la un anumit moment. În cazul în care, de exemplu, mediul runtime este utilizat pentru redarea video-ului, atunci comanda de pauză() trimisă de către logicul paginii poate să fie mapată pe o comandă de pauză corespondentă pentru runtime.

[0022] Exemplele de mai sus sunt prezentate numai cu scop ilustrativ. În cadrul descrierii detaliate care urmează, sunt expuse numeroase detalii specifice pentru a se asigura în felul acesta o amănunțită cunoaştere a subiectului. Cu toate acestea, trebuie să fie înțeles de către specialiştii din domeniu că subiectul poate să fie pus în parctică fără aceste detalii specifice. În alte împrerjurări, metodele, aparaturile sau sistemele care ar fi cunoscute de către un specialist din domeniu nu au fost descrise în detaliu astfel încât să nu eclipseze subiectul.

[0023] Figura 2 este o diagramă care prezintă un dispozitiv iliustrativ de calcul 202 care este folosit pentru a asigura conținut video 122 după cum este arătat în cadrul Figurii 1 prin intermediul utilizării aplicației de rendering 114, obiectului proxy 126 şi a mediului runtime 128. La dispozitivul de calcul 202 se poate face referire în mod alternativ ca la un sistem pentru prelucrarea datelor, dispozitiv computerizat, sau simplu un „calculator". Dispozitivul de calcul 202 reprezintă un desktop, laptop, tablet sau oricare alt sistem de calcul. Alte exemple de sisteme de calcul 202 includ, dar nu sunt limitate la acestea, dispozitivele mobile (PDA-uri, smartphone-uri, media playeruri, sisteme de joc etc.), dispozitive pentru jocuri, televizoare şi set-top boxe pentru televizoare, precum şi sisteme încorporate (de exemplu, în vehicole, aparate electrocasnice, sau alte dispozitive).

[0024] În general vorbind, dispozitivul de calcul 202 reprezintă unul sau mai multe elemente de hardware pentru prelucrarea de date care implementează aplicația de rendering 114 şi mediul runtime 128. Aplicația de rendering 114 face ca dispozitivul de calcul 202 să parseze codul 102 din Figura 1 şi să stabilească obiectele proxy 126 şi, în situația în care mediul runtime 128 nu este utilizat, să răspundă la apelurile API de rendering specificate în codul 102. Desigur, codul 102 poate să fie prevăzut ca unul sau mai multe fişiere. În cadrul acestui exemplu, API de rendering este un API de rendering video, dar este de la sine înțeles că principiile care sunt discutate aici pot să fie aplicate altor API de rendering pentru care ar fi de dorit utilizarea unui mediu runtime.

[0025] Obiectul(ele) proxy 126 poate să fie implementat într-un mediu citibil pe computer ne-tranzitoriu într-un mod care asigură actualizarea atributelor obiectului(elor) proxy 126 şi permite ca metode definite prin obiectul(ele) proxy 126

să fie puse în aplicare prin intermediul unui hardware de prelucrare potrivit. De exemplu, aplicația de rendering 114 poate să includă un Javascript™ sau un alt motor de scripting a cărui funcționalitate este utilizată pentru a furniza obiectul(ele) proxy 126 pe baza apelurilor de API de scripting formulate în codul 102.

[0026] Mediul runtime 128 pune la dispoziție un mediu de execuție pentru una sau pentru mai multe aplicații de runtime 129 la care se face referință prin codul 102. De exemplu, aplicația de runtime 129 poate să aibă în componență un cod de byte tip cross-platform care asigură mediul runtime 128. În unele puneri în aplicare, mediul runtime 128 are în componență un mediu cum ar fi Adobe® Flash® or AIR®, amintite mai sus, Microsoft® Silverlight® (disponibil de la Microsoft Corporation din Redmond Washington), o Java® Virtual Machine (disponibilă de la Oracle Corporation din Redwood Shores, California), sau un alt runtime. Aplicația de runtime 129 poate să fie descărcată de la o conexiune de rețea sau poate să fie accesată din memorie ca răspuns la referința 108 în codul 102, sau poate fi accesată chiar direct cu codul 102 atunci când codul 102 este accesat inițial. Mediul de runtime 128 poate să execute în cadrul unui sistem de operare care suportă de asemenea aplicația de rendering 114. Totuşi, mediul runtime 128 poate ca el însuşi să aibă în componență sistemul de operare în unele dintre implementări. Atunci când este implementat ca un software, mediul runtime 128 poate fi el însuşi descărcat în cazul în care nu este deja rezident pe dispozitivul 202, şi mediul runtime 128 şi aplicația 129 pot să fie distribuite chiar ca o unitate integrată, atunci când se doreşte.

[0027] Aplicația de rendering 114 şi/sau mediul runtime 128 pot să fie puse în aplicare în hardware-ul accesibil prin intermediul sau ca o parte a elementului pentru prelucrarea datelor (de exemplu, ca un circuit integrat specific pentru aplicație, (ASIC), ca un dispozitiv logic programabil (de exemplu, PLA-uri, FPGA-uri etc.)). Ca un alt exemplu, aplicația de rendering 114 şi/sau mediul runtime 128 pot să fie puse în aplicare utilizând software sau firmware care configurează funcționarea unui procesor sau a unor procesoare.

[0028] În cadrul exemplului prezentat în Figura 2, dispozitivul de calcul 202 reprezintă un element de hardware pentru prelucrarea datelor care are în componență unul sau mai multe procesoare 204 şi un mediu citibil pe calculator

(memoria 206) interconectate prin intermediul unui interconector 208, reprezentând magistralele interne, conexiunile şi altele asemănătoare. Interconectorul 208 se conectează de asemenea la componentele I/E 210, cum sunt magistrala serială universală (USB), VGA, HDMI, serial şi alte conexiuni I/E către alt hardware al sistemului de calcul. Hardware-ul include de asemenea unul sau mai multe afişaje 212. Este de la sine înţeles că dispozitivul de calcul 202 poate să includă şi alte componente, cum sunt dispozitivele de stocare, dispozitivele de comunicaţie (de exemplu, Ethernet, componente radio) şi alte componente de I/E cum sunt difuzoarele, un microfon şi altele asemănătoare.

[0029] În general vorbind, interconectorul 208 este utilizat pentru a primi codul 102 şi conţinutul care trebuie să fie renderat, pentru a stoca codul 102 şi datele de conţinut după cum este cerut, cum ar fi într-un hard drive local, o memorie cache, stocare de reţea etc. şi să retransmită conţinutul renderat către interfaţa I/E 210 pentru afişare pe dispozitivul de afişare 122. Intrarea este asigurată porin intermediul dispozitivelor de intrare cum sunt mouse-ul, tastatura, interfaţa de touch screen etc.

[0030] Mediul citibil pe calculator 206 poate să aibă în componenţă memorie RAM, ROM sau o altă memorie şi în cadrul acestui exemplu încorporează codul aplicaţiei de rendering 114, obiectul proxy 126, mediul runtime 128, şi aplicaţia de runtime 129. Este de la sine înţeles că codul 102 poate să fie de asemenea stocat în cadrul unui mediu citibil pe calculator 206 sau în cadrul unui alt mediu.

[0031] Figura 3 este o schemă logică de program care ne prezintă paşii unei metode ilustrative 300 pentru evidenţierea funcţionalităţii API prin intermediul unui obiect proxy şi utilizând o astfel de funcţionalitate. Blocul 302 reprezintă codul de parsare al unui markup sau alt document care recurge la un API de rendering. De exemplu, blocul 302 poate să fie executat de către un browser sau o altă aplicaţie de rendering 114 din Figurile 1 – 2.

[0032] Blocul 304 reprezintă instalarea unuia sau a mai multor obiecte proxy 126. De exemplu, documentul de markup poate să fie configurat cu cod care recurge la un API de scripting suportat de către aplicaţia de rendering 114 astfel încât obiectele proxy pot să fie instanţiate astfel încât să redea în oglindă proprietăţile, metodele şi

comportamentele definite de către API de rendering. În cadrul unor puneri în aplicare, obiectul(ele) proxy 126 furnizează fiecare şi oricare proprietate, metodă şi comportare a API-ului de rendering (sau API-urilor) pentru a fi expuse pentru a asigura că capacitățile aşteptate ale API-ului de rendering sunt disponibile.

[0033] În plus, documentul markup poate să definească metode de getter şi de setter care sunt suportate de către API de scripting în scopul mapării evenimentelor şi stărilor de runtime către API de rendering şi al recurgerii la funcționalitatea runtime/stabilirii stărilor de runtime ca răspuns la apelurile pentru API de rendering. Un getter este o metodă care obține valoarea unei proprietăți specifice. Un setter este o metodă care stabileşte valoarea unei proprietăți specifice. Astfel, la blocul 304 atunci când este instalat obiectul proxy, pot să fie inițializate gettere şi settere corespunzătoare pentru a face face legătura între API de rendering şi apelurile de interfață externă ale mediului runtime.

[0034] În cadrul unor puneri în aplicare poate să fie prevăzut un obiect proxy JavaScript care redă în oglindă elementul HTMLVideoElement, HTMLAudioElement, şi API-urile HTMLMediaElement după cum sunt definite în standardul HTML (vezi, de exemplu, http://www.whatwg.org/specs/webapps/current-work/multipage/video.html#video, care a fost încorporat aici, prin referire, în întregul său). Obiectul poate să aibă exact aceleaşi nume de proprietate, exact aceleaşi nume de metodă şi exact acelaşi comportament cu acela descris în specificații (de exemplu, pentru API-urile de mai sus, obiectul proxi expediază exact aceleaşi evenimente VideoElement/Media Element specificate în specificație care reflectă schimbările în starea punerii în aplicare de bază). Punerile în aplicare care oglindesc API la nivelul proprietății pot să permită în mod avantajos codului scris să recurgă la API de rendering pentru a fi reutilizat cu modificări minime sau chiar fără modificări pentru acomodarea utilizării proxy/runtime din moment ce specificația API este respectată în întregime. Desigur, şi alte API-uri decât cele de mai sus pot să fie oglindite. Exemplele suplimentare includ, fără a se limita la ele, HTMLFormElement, HTMLCanvasElement, HTMLTableElement, etc.

[0035] Blocul 306 reprezintă instalarea UI şi a altor elemente (de exemplu, elementul video 118) care trebuie să fie furnizate de către aplicația de rendering, în

timp ce blocul 308 reprezintă recurgerea la aplicația de runtime. Blocul 306 poate să includă stabilirea elementelor UI suplimentare de către o altă aplicație specifică. Operațiile de stabilire pot să includă instanțierea obiectelor şi a altor componente de program pentru a rendera elementele UI şi pentru a executa o altă funcționalitate. Echilibrul particular între ce funcții sunt folosite de către aplicația de rendering şi aplicația de runtime poate varia în concordanță cu punerile în aplicare particulare. Aplicația de runtime poate să fie descărcată sau accesată în alt mod de către mediul runtime pentru execuție după cum este nevoie ca răspuns la comnezile sau evenimentele de intrare.

[0036] De exemplu, documentul de markup va recurge în mod obişnuit la funcționalitate pentru a fi folosită nativ de către aplicația de rendering, cum ar fi diverse elemente de pagină web cum ar fi text şi imagini. Funcție de o anumită punere în aplicare, aplicația de rendering poate să fie utilizată pentru a rendera conținut video, controale video şi altele asemănătoare, cu aplicația de runtime jucând un rol de „background" pentru a monitoriza şi a raporta redarea. Ca un alt exemplu, aplicația de runtime poate să fie utilizată pentru a acționa redarea video în timp ce capacitățile native ale aplicației de rendering sunt utilizatre pentru a asigura controale şi „chrome" pentru un aspect consistent. Ca un exemplu suplimentar, aplicația de runtime poate să fie utilizată pentru „chrome" şi pentru controale în timp ce aplicația de rendering se ocupă de renderingul video din prezent.

[0037] În orice caz, pentru acele invocări ale API de rendering care să fie folosit de către aplicația de runtime, UI (sau alte elemente) sunt stabilite astfel încât utilizatorul sau alte evenimente care recurg la API de rendering provoacă o actualizare a stării obiectului(elor) proxy şi/sau sunt legate de metode de getter şi setter astfel încât schimbările stărilor şi evenimentelor sunt trecute direct către runtime fără a stoca starea din obiectul proxy.

[0038] În cadrul acestui exemplu, după cum este arătat la blocul 310, ca răspuns la un apel de API de rendering, starea obiectului proxy poate să fie actualizată. Apoi, la blocul 312 se recurge la o comandă de runtime pe baza stării obiectului proxy. Totuşi, în cazul în care metodele getter/setter sunt suportate, atunci un apel de API

de rendering poate să fie mapat direct pe un apel către una sau mai multe metode ale runtime pentru a face ca runtime să răspundă.

[0039] De exemplu, în unele implementări, apelurile de metodă JavaScript sunt proxiate (printr-un obiect proxy) la metode expuse de către mediul runtime. De exemplu, mediul runtime Flash® furnizează apeluri InterfațăExternă pe care obiectul proxy le poate invoca ca răspuns la UI sau la alte evenimente care invocă API de rendering. În felul acesta, schimbarea de stare a obiectului proxy poate fi pur şi simplu recepția apelului API, care apoi conduce direct la un apel corespondent către runtime.

[0040] În mod suplimentar sau alternativ, apelul API de rendering poate fi de un tip care schimbă un atribut definit de către API-ul de rendering, acel atribut fiind menținut ca un atribut care poate să fie scris al obiectului proxy. Schimbările făcute la atributele care pot să fie scrise ale obiectului proxy pot să fie folosite în diverse feluri funcție de capacitățile aplicației de rendering 114. De exemplu, motoarele de scripting ale celor mai recente versiuni de browsere permit ca getter-urile şi setter-urile să fie definite prin intermediul apelurilor către API de scripting după cum s-a remarcat mai sus. Pentru acele browsere, pot să fie definite gettere şi settere astfel încât schimbarea atributului care poate să fie scris face ca obiectul proxy să apeleze o metodă expusă de către mediul runtime. De exemplu, dacă o intrare sau un alt veniment apelează „current Time = 30" în JavaScript, obiectul proxy poate să facă un apel „setCurrent Time(30)" către runtime-ul Flash® (prin intermediul InterfețeiExterne)

[0041] Nu toate browserele suportă setterele şi getterele obişnuite. Pentru a rezolva astfel de situații, codul de scripting poate să includă apeluri de API de scripting pentru a păstra o primă copie a obiectului proxy şi o a doua copie a obiectului proxy, prima şi a doua copie fiind identice atunci când sunt inițializatre pentru prima dată. De exemplu, obiectul proxy poate să fie inițializat împreună cu un obiect care funcționează ca o copie la indigo a stării runtime-ului. Obiectul proxy poate să sondeze în mod regulat copia la indigo şi, atunci când copia la indigo şi obiectul proxy nu se mai potrivesc, obiectul proxy poate să determine dacă API de rendering

a fost invocat şi să facă un apel corespunzător la runtime (de exemplu, pentru runtime-ul Flash®, utilizând InterfațaExternă).

[0042] Blocul 314 reprezintă propagarea unei schimbări de stare dela un mediu runtime la un obiect proxy, în timp ce blocul 316 reprezintă actualizarea elementelor UI sau o altă acțiune care este luată în aplicația de rendering care se bazează pe schimbarea de stare propagată. Aceste blocuri sunt prezentate după blocurile 310 – 312, deşi este posibil ca ele dsă se găsească înaintea blocurilor 310 – 312 sau chiar în paralele cu acestea. În general vorbind, mediul runtime poate să facă un apel la API-ul de scripting pentru a actualiza starea obiectului proxy şi/sau pentru a invoca o metodă definită de către obiectul proxy, obiectul proxy fiind expediat după aceea ca potrivit către API-ul de rendering.

[0043] De exemplu, în cazul în care este folosit runtime-ul Flash®, InterfațaExternă poate să facă un apel către JavaScript pentru a actualiza una sau mai multe dintre proprietățile obiectului proxy al lui JavaScript. În unele dintre punerile în aplicare, actualizarea are loc fără o prelucrare ulterioară în afara situației în care există deja o schimbare a atributului care poate să fie scris, caz în care conflictul este rezolvat în favoarea schimbării existente a obiectului proxy. Ca răspuns la schimbare, este trimis un eveniment JavaScript care se potriveşte cu evenimentul(ele) specificat în API-ul de rendering (de exemplu, evenimente după cum sunt ele definite în API <video> HTML).

[0044] După cum s-a remarcat mai sus, nu toate browserele pot să suporte gettere şi settere suficiente pentru a identifica când au fost făcute schimbări la obiectul proxy. În felul acesta , abordarea cu copia la idigo poate de asemenea să fie utilizată pentru a se determina când a propagat un runtime o schimbare de stare.

[0045] După cum este prezentat la blocul 318, metoda poate să se bucleze prin blocurile 310 – 316 până când rendering-ul este complet. De exemplu, rendering-ul poate să fie complet atunci când conținutul video sau un alt conținut este terminat sau dacă un utilizator iese înainte de vreme. Desigur, pot să fie definite şi alte condiții de ieşire.

[0046] Figura 4 este o schemă logică de program care prezintă paşii unei metode ilustrative 400 pentru utilizarea selectivă a mediului runtime în scopul expunerii unui API de rendering video care nu este suportat în mod nativ de către mediul runtime. De exemplu, codul 102 din Figura 1 poate să facă dispozitivele de calcul (cum este dispozitivul de calcul 202) să efectueze fluxul prezentat la 400.

[0047] Blocul 402 reprezintă primirea şi parsarea, de către un browser, a unui fişier HTML cu un element <video> şi elemente de scripting configurate după cum a fost discutat în cele de mai sus. De exemplu, fişierul HTML poate recurge cel puțin la o parte din funcționalitatea API de rendering <video> HTML. Blocul 404 reprezintă determinarea dacă dispozitivul de calcul suportă un mediu runtime necesar pentru a utiliza o aplicație de runtime la care face referire fişierul HTML. De exemplu, codulJavaScript sau un alt cod poate fi folosit pentru a identifica un tip de dispozitiv şi/sau dacă este instalat runtime-ul Flash® sau un alt mediu runtime.

[0048] În cazul în care mediul runtime nu este disponibil, fluxul continuă la blocul 406, care reprezintă API de rendering video ca fiind suportat nativ de către browser. Totuşi, dacă mediul runtime este găsit la blocul 404, atunci fluxul continuă la blocul 408. Blocul 408 reprezintă stabilirea unui sau mai multor obiecte proxy cu settere şi gettere, sau un obiect proxy şi o copie la indigo, în timp ce blocul 410 reprezintă utilizarea obiectului(elor) proxy pentru a propaga comenzile la mediul runtime şi pentru a transmite mai departe schimbările de stare a runtime-ului la elementele UI prevăzute de către browser sau la alte elemente. De exemplu, aceşti paşi pot să fie îndepliniți în conformitate cu cele discutate mai sus la Figura 3.

[0049] Dezvoltatorii şi distribuitorii de conținut, prin configurarea markup sau a altui cod 102 pentru a face un dispozitiv de calcul să funcționeze după cum a fost discutat aici, pot să fie scutiți de problema de a pregăti versiuni multiple ale codului pentru diferite platforme. În loc de aceasta, dezvoltatorul/distribuitorul de conținut poate genera codul în conformitate cu un API de rendering suportat în mod obişnuit (de exemplu, API <video> HTML) dar fără a pierde avantajele folosirii unui mediu runtime (de exemplu, runtime-ul Flash® sau runtime-ul AIR®) pentru acele dispozitive care suportă mediul runtime.

[0050] De exemplu, după cum s-a observat mai sus, mediul runtime poate să fie utilizat pentru a asigura capacități, cum ar fi multicasting, distribuția de conținut protejat, depistarea utilizării şi altele asemănătoare care nu sunt suportate de către API de rendering. În felul acesta, acelaşi fişier HTML sau un alt fişier poate să fie distribuit, de exemplu, către set-top boxele pentru televiziune care suportă runtime, dispozitive mobile care suportă runtime, dar şi către dispozitive mobile, tablete etc. care s-ar putea să nu suporte runtime.

[0051] Operațiile de parsare şi de expunere API discutate mai sus pot să fie îndeplinite la un dispozitiv de client care afişează de asemenea conținutul renderat. Totuşi, în cadrul unor implementări alternative, operațiile de parsare şi de expunere API pot să fie desfăşurate la un server care apoi trimite mai departe datele care reprezintă conținutul ce trebuie să fie renderat la alte dispozitive decât cel care renderează în prezent conținutul.

[0052] Exemplul din Figura 4 şi alte exemple care au fost discutate mai sus un browser şi API de rendering<video> HTML. Este de la sine înțeles faptul că alte aplicații pot să suporte API de rendering şi pot să fie utilizate şi alte API-uri de rendering. În mod suplimentar, aplicația de rendering a fost descrisă ca aplicația pentru expunerea API-ului de scripting. Modalități de realizare suplimentare pot să utilizeze o aplicație separată sau un proces separat pentru a expune API de scripting, cu comunicare inter-proces între acea aplicație separată şi aplicația de rendering utilizată ca necesară pentru a coordona intrarea către şi ieşirea de la elementele prevăzute cu aplicație de rendering. Încă în mod suplimentar, mai multe exemple de mai sus au făcut referire la codul de markup, dar poate fi folosit şi alt cod (de exemplu, codul binar) care invocă API-urile de rendering şi de scripting şi invocă mediul de runtime în locul codului de markup (de exemplu, pentru codul 102).

[0053] Uneltele de dezvoltare pot să utilizeze principiile amintite mai sus pentru a da dezvoltatorilor posibilitatea de a implemeta codul care recurge la un API de rendering şi care de asemenea utilizează un mediu runtime atunci când este disponibil. De exemplu, o aplicație de dezvoltare a web-ului poate să includă o interfață de utilizator, un manager de cod şi alte componente care sunt folosite pentru a defini HTML-ul sau alt cod de pagină de web. În cadrul unei implementări,

aplicația pentru dezvoltarea de web poate să fie configurată pentru a identifica tag-uri <video>, de exemplu sau altă sintaxă care invocă un API de rendering. Pagina poate să fie actualizată pentru a utiliza un mediu runtime (de exemplu, prin referirea la un Flash® sau la alt plugin) şi pentru a include codul JavaScript pentru a mapa apelurile API către playerul Flash®player, păstrându-se în acelaşi timp apelurile API de rendering originale care nu utilizează Flash® (sau alt mediu runtime). În unele implementări, tagul <video> este înlocuit cu un pentru a evita controversele care apar pe unele dispozitive în cazul în care tagul<video> însuşi este înlocuit în întregime cu redarea pe bază de runtime.

Considerații generale General Considerations

[0054] Unele porțiuni ale descrierii detaliate au fost prezentate în termeni de reprezentare simbolică sau algoritmică a operațiunilor pe biți de date sau semnale digitale stocate în interiorul memoriei unui sistem de calcul, cum este o memorie de calculator. Aceste descrieri sau reprezentări algoritmice sunt exemple de tehnică folosită de către specialiştii din domeniul prelucrării datelor pentru a transmite substanța muncii lor altor specialişti din domeniu. Un algoritm este considerat aici şi în general a fi o secvență de operațiuni auto-suficientă sau procesare similară cu aceasta care conduce la un rezultat dorit. În acest context, operațiunile şi procesarea implică manipularea fizică şi mărimi fizice.

[0055] În mod obişnuit, deşi nu este necesar, astfel de mărimi pot lua forma unor semnale electrice sau magnetice care au posibilitatea de a fi stocate, transferate, combinate, comparate sau în alt fel folosite. S-a dovedit convenabil uneori, în principal din motive de utilizare comună, să se facă referire la astfel de semnale ca la biți, date, valori, elemente, simboluri, litere, termeni, numere, numerale sau altele asemănătoare. Trebuie totuşi să se înțeleagă că toate acestea şi termenii similari trebuie să fie asociați cu mărimi fizice corespunzătoare şi sunt pur şi simplu etichete convenabile.

[0056] În afară de cazul în care este afirmat altceva, după cum reiese din discuția anterioară, se consideră că în întreaga această specificație discuțiile care utilizează termeni precum „prelucrare", „calculare", „care calculează", „care determină" sau

altele asemenea se referă la acțiuni sau la procese ale unei platforme de calcul, cum ar fi unul sau mai multe calculatoare şi/sau un dispozitiv sau dispozitive de calcul electronice similare, care folosesc sau transformă date reprezentate ca mărimi fizice electronice sau magnetice din memorii, registrii sau alte dispozitive de stocare a informației, dispozitive de transmisie, sau dispozitive de afişare ale platformei de calcul.

[0057] Deşi mai multe exemple au prezentat dispozitive mobile, diversele sisteme care au fost discutate aici nu sunt limitate la nicio arhitectură sau configurație particulară de hardware. Un dispozitiv de calcul poate să includă orice aranjament corespunzător de componente care furnizează un rezultat condiționat de una sau de mai multe intrări. Dispozitive de calcul potrivite includ sisteme multifuncționale de calculator bazate pe microprocesoar care accesează software-ul stocat, care programează sau configurează sistemul de calcul dintr-un aparat de calcul de destinație generală într-un aparat de calcul specializat prin punerea în aplicare a uneia sau mai multor modalități de realizare a prezentului subiect. Orice limbaj corespunzător de programare, de scripting sau un alt tip de limbaj sau combinație de limbaje poate să fie folosit pentru a pune în aplicare ştiința conținută aici în software-ul care urmează să fie folosit pentru programarea sau configurarea unui dispozitiv de calcul.

[0058] Un dispozitiv de calcul poate să acceseze unul sau mai multe medii ne-tranzitorii citibile pe calculator care încorporează instrucțiuni citibile pe calculator care, atunci când sunt executate de către cel puțin un calculator, fac ca acel cel puțin un calculator să implementeze una sau mai multe din modalitățile de realizare ale prezentului subiect. Atunci când este utilizat software-ul, software-ul poate să aibă în includă una sau mai multe componente, procese şi/sau aplicații. În plus sau alternativ la software, dispozitivul(ele) de calcul poate să aibă în componență circuite care fac dispozitivul(ele) funcțional pentru a implementa una sau mai multe metode ale prezentului subiect.

[0059] Exemplele de dispozitive de calcul includ, fără a fi limitate la acestea, servere, computere personale, dispozitive mobile (de exemplu, tablete, smartphone-uri, asistente digitale personale (PDA-uri) etc.), televizoare, set-top boxe pentru

televizoare, playere portabile pentru muzică şi dispzitive electronice pentru amatori cum ar fi aparatele de fotografiat, aparatele de filmat şi dispozitivele mobile. Dispozitivele de calculat pot să fie integrate în alte dispozitive, de exemplu electrocasnice „inteligente", automobile, chioşcuri şi altele asemenea.

[0060 Modalităţi de realizare a metodelor descrise aici pot să fie realizate în operaţiile dispozitivelor de calcul. Ordinea blocurilor prezentate în exemplele de mai sus poate să fie diferită – de exemplu, blocurile pot să fie re-ordonate, combinate şi/sau descompuse în sub-blocuri.
Anumite blocuri sau procese se pot desfăşura în paralel.

[0061] Oricare mediu sau medii citibile pe calculator ne-tranzitorii corepunzătoare pot să fie utilizate pentru a implementa sau aplica în practică subiectul descris aici, inclusiv, dar fără a se limita la, dischete, drive-uri, medii de stocare pe bază magnetică, medii de stocare optice (de exemplu, CD-ROM-uri, DVD-ROM-uri şi variante ale acestora), dispozitive de memorie flash, RAM, ROM şi altele.

[0062] Utilizarea aici a termenilor de „adaptat la" sau „configurat pentru" este înţeleasă ca limbaj deschis şi incluziv care nu exclude dispozitivele adaptate pentru sau configurate pentru a executa sarcini şi paşi suplimentari. În plus, utilizarea lui „pe baza" este înţeleasă ca fiind deschisă şi incluzivă, prin aceea că un proces, o etapă, un calcul sau o altă acţiune care "se bazează pe" una sau mai multe din condiţiile sau valorile relatate poate, în practică, să fie bazată pe condiţii sau valori adiţionale celor care au fost enunţate. Titlurile, listele şi numerotarea incluse aici sunt numai pentru uşurarea explicaţiei şi nu sunt intenţionate ca limitative.

[0063] Chiar dacă prezentul subiect a fost descris în detaliu cu referire la anumite modalităţi de realizare a acestuia, este de la sine înţeles pentru specialiştii din domeniu, după parcurgerea şi înţelegerea a celor anterioare că pot cu uşurinţă să aducă modificări la, variaţii la şi echivalente la aceste modalităţi de realizare. În consecinţă, este de la sine înţeles că prezenta descriere a fost prezentată numai pentru scopuri de exemplificare şi nu de limitare, şi că nu împiedică includerea unor astfel de modificări, variaţii şi/sau suplimentări la prezentul subiect după cum este evident pentru specialiştii din domeniu.

Revendicări

1. Computer care are în componență:
o interfață I/E; şi
un element de prelucrare conectat la interfața I/E şi care pune în aplicare un mediu runtime, elementul de prelucrare punând de asemenea în aplicare o aplicație de rendering care face ca elementul de procesare să pună la dispoziție un obiect proxy utilizat pentru a expune un API de rendering care nu este suportat în mod nativ de către mediul runtime, API-ul de rendering fiind invocabil prin codul cuprins într-un document markup accesat de către procesor prin intermediul interfeței I/E, API de rendering fiind expus prin aceea că face ca mediul de runtime să răspundă la proprietățile, metodele şi comportamentele definite de către API de rendering şi oglindite de către obiectul proxy.

2. Calculator în conformitate cu revendicarea 1, în care API-ul de rendering are în componență un API de rendering video.

3. Calculator în conformitate cu revendicarea 1,
în care documentul markup invocă funcționalitatea unui API de scripting care face ca elementul de procesare să mențină obiectul proxy într-un mediu de stocare ne-tranzitoriu.

4. Calculator în conformitate cu revendicarea 3, în care documentul de markup invocă o metodă getter a API-ului de scripting care face ca procesorul să obțină o valoare a unei proprietăți redate în oglindă şi, pe baza valorii proprietății oglindite, invocă o comandă a mediul de runtime, şi
în care documentul de markup invocă o metodă de setter a API-ului de scripting care face ca procesorul să stabilească o valoare a unei proprietăți oglindite de către obiectul proxy ca răspuns la un eveniment sau la o schimbare de stare a runtime-lui.

5. Calculator în conformitate cu revendicarea 3, în care mediul runtime face ca elementul de procesare să propage schimbările de stare de la mediul de runtime către obiectul proxy prin invocarea API-ului de scripting pentru a schimba o stare a obiectului proxy.

6. Calculator în conformitate cu revendicarea 3, în care obiectul proxy face ca procesorul să trimită un eveniment de scripting care se potriveşte cu un format de eveniment a unui API de rendering.

7. Calculator în conformitate cu revendicarea 1, în care documentul de markup invocă API de scripting pentru a face ca procesorul să păstreze o primă copie a obiectului proxy şi o a doua copie a obiectului poroxy, prima copie şi a doua copie fiind identice atunci când sunt iniţializate pentru prima dată, şi în care mediul runtime este configurat, după ce prima copie şi a doua copie sunt iniţializate prima dată, pentru a determina dacă APi de rendering a fost invocat prin determinarea dacă una dintre prima copie şi a doua copie ale obiectului proxy au suferit modificări.

8. Computer în conformitate cu revendicarea 1, în care obiectul proxy face ca elementul de procesare să răspundă la intrare prin schimbarea unui atribut care poate să fie scris definit de către API de rendering prin schimbarea unui atribut al obiectului proxy.

9. Calculator în conformitate cu revendicarea 1, în care elementul de procesare are în componenţă un procesor interfaţat la o memorie, mediul runtime fiind pus în aplicare prin executarea instrucţiunilor de program care sunt stocate în memorie.

10. Calculator în conformitate cu revendicarea 1, în care documentul de markup are în componenţă un document HTML, funcţionalitatea de scripting are în componenţă JavaScript, iar funcţionalitatea de scripting este invocată în conformitate cu un fişier JavaScript la care se face referire în cadrul documentului HTML.

11. Calculator în conformitate cu revendicarea 2, în care documentul de markup se referă la video care trebuie să fie redat utilizând mediul runtime cu cel puţin un element de control renderat de către un browser şi configurat pentru a invoca API de rendering video.

12. Produs program de calculator care are în componenţă codul încorporat într-un mediu citibil pe calculator ne-tranzitoriu, codul având în componenţă:

codul pentru invocarea funcționalității unei aplicații de runtime; şi
codul pentru invocarea unui API de scripting pentru a menține un obiect proxy, atributele care se oglindesc ale obiectului proxy definite prin intermediul unui API de rendering video şi configurate pentru a invoca o metodă care este suportată de către aplicația de runtime ca răspuns la primirea unei comenzi pentru a invoca un apel suportat de către API de rendering video.

13. Produs program de calculator cu revendicarea 12, în care codul pentru invocarea API-ului de scripting are în componență funcționalitatea de scripting care invocă codul de markup al unei aplicații de browser.

14. Produs program de calculator în conformitate cu revendicarea 13, în care obiectul proxy oglindeşte numele de proprietate, numele de metodă şi comportamente definite prin intermediul API-urilor HTMLVideoElement, HTMLAudioElement şi API-uri HTMLMediaElement.

15. Produs program de calculator în conformitate cu revendicarea 12,
în care codul pentru invocarea funcționalității aplicației de runtime are în componență un prim element de markup care face referire la o aplicație de runtime care să fie executată în mediul runtime pentru a reda un video.

16. Produs program de calculator în conformitate cu revendicarea 15, având în plus în componență:
codul pentru generarea unui element de interfață, codul pentru generarea elementului de interfață având în componență un al doilea element de markup care defineşte cum trebuie renderat elementul de interfață; şi
codul pentru furnizarea comenzii pentru a invoca un apel suportat de către API de rendering video ca răspuns la un eveniment de intrare generat de către elementul de interfață.

17. Produs program de calculator în conformitate cu revendicarea 16, în care apelul suportat de către API de rendering video are în componență o comandă de control de redare iar metoda invocată de către obiectul proxy are în componență o metodă corespondentă care controlează redarea video prin intermediul mediului runtime.

18. O metodă pusă în aplicare pe pe calculator, care are în componență:
determinarea, prin intermediul unui dispozitiv de procesare, a unei stări a obiectului proxy păstrată într-un mediu citibil pe computer, obiectul proxy oglindind atribute ale API de rendering video suportat de către o aplicație de browser;
ca răspuns la determinarea stării obiectului proxy şi pe baza stării obiectului proxy, invocarea, de către dispozitivul de procesare, a unei metode suportate de un mediu runtime.

19. Metodă în conformitate cu revendicarea 18, având în plus în componență:
ca răspuns la determinarea stării obiectului proxy şi pe baza stării obiectului proxy, expedierea unui eveniment definit de către API de rendering video.

20. Metodă în conformitate cu revendicarea 19, având în plus în componență:
actualizarea unui atribut al obiectului proxy ca răspuns la o schimbare de stare a mediului runtime, înainte de a determina starea obiectului proxy,
în care evenimentul care este expediat ulterior este definit de către obiectul proxy pentru a corespunde cu atributul în forma actualizată.

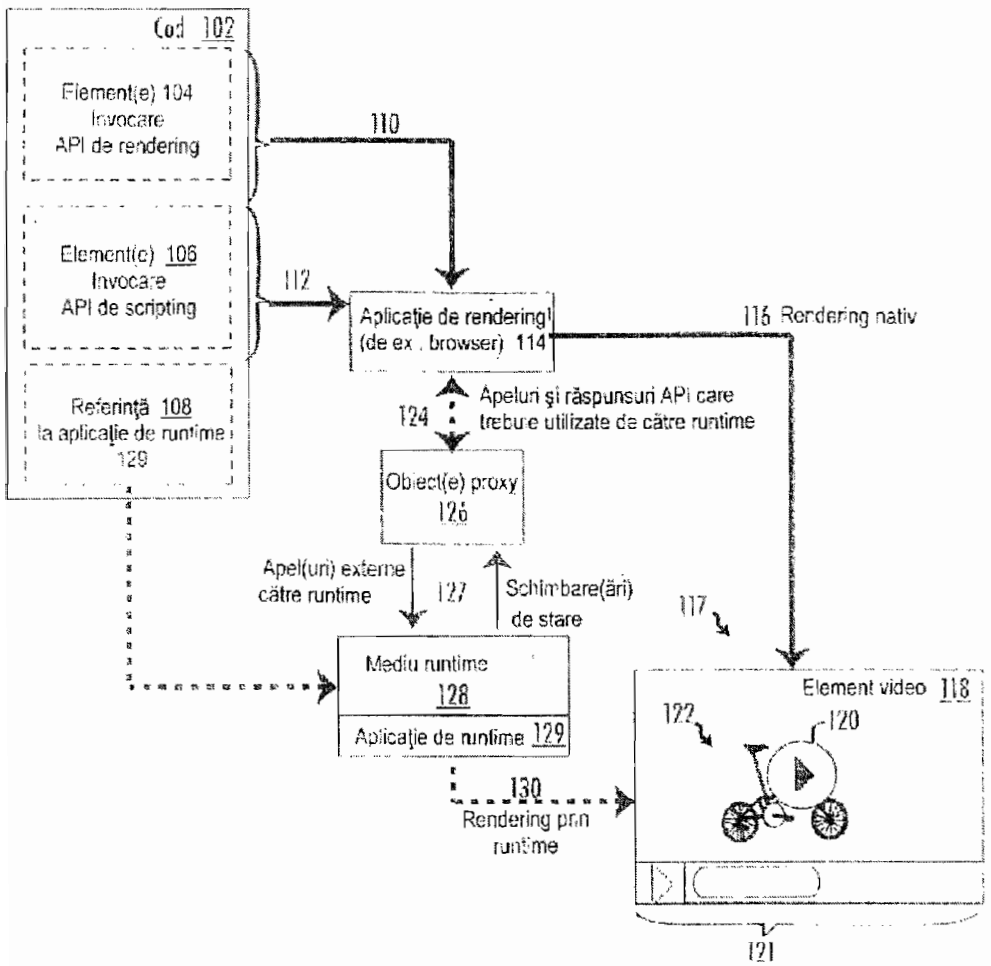
Cod 102
Element(e) 104
Invocare
API de rendering
Element(e) 106
Invocare
API de scripting
Referinţă 108
la aplicaţie de runtime
129
110
112
Aplicaţie de rendering
(de ex . browser) 114
116 Rendering nativ
124
Apeluri şi răspunsuri API care
trebuie utilizate de către runtime
Obiect(e) proxy
126
Apel(uri) externe
către runtime
127
Schimbare(ări)
de stare
117
Mediu runtime
128
Aplicaţie de runtime 129
Element video 118
122
120
130
Rendering prin
runtime
121

Fig. 1

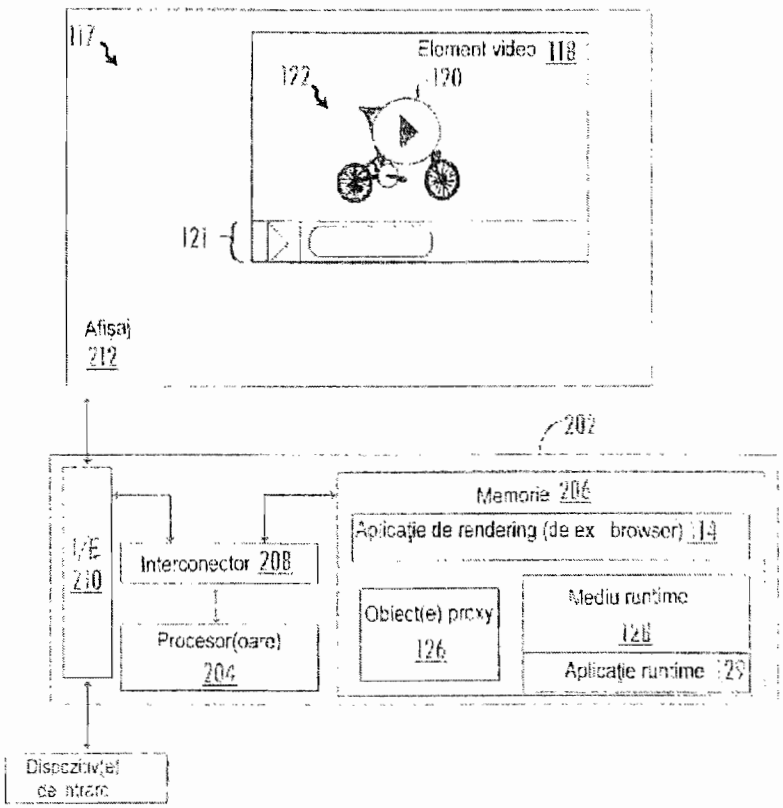
117
Element video 118
122
120
121
Afişaj
212
202
Memorie 206
Aplicaţie de rendering (de ex. browser) 114
I/E
210
Interconector 208
Obiect(e) proxy
126
Mediu runtime
128
Aplicaţie runtime 129
Procesor(oare)
204
Dispozitiv(e)

Fig. 2

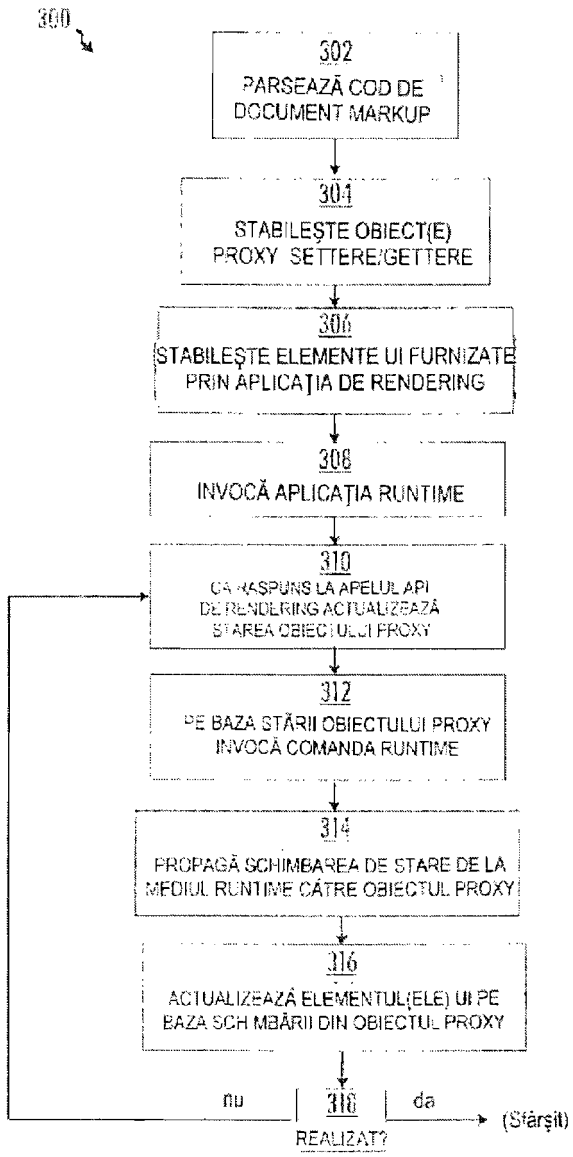
300
302
PARSEAZĂ COD DE DOCUMENT MARKUP
304
STABILEŞTE OBIECT(E) PROXY SETTERE/GETTERE
306
STABILEŞTE ELEMENTE UI FURNIZATE PRIN APLICAŢIA DE RENDERING
308
INVOCĂ APLICAŢIA RUNTIME
310
CA RĂSPUNS LA APELUL API DE RENDERING ACTUALIZEAZĂ STAREA OBIECTULUI PROXY
312
PE BAZA STĂRII OBIECTULUI PROXY INVOCĂ COMANDA RUNTIME
314
PROPAGĂ SCHIMBAREA DE STARE DE LA MEDIUL RUNTIME CĂTRE OBIECTUL PROXY
316
ACTUALIZEAZĂ ELEMENTUL(ELE) UI PE BAZA SCHIMBĂRII DIN OBIECTUL PROXY
nu
318
da
REALIZAT?
(Sfârşit)

Fig. 3

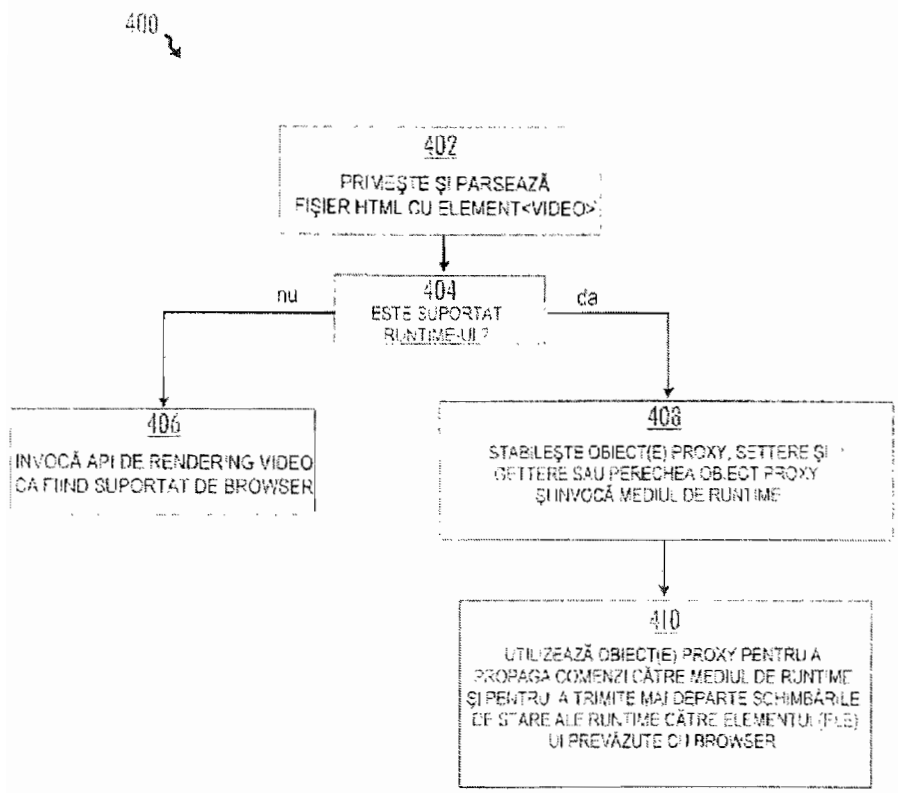
400
402
PRIMEŞTE ŞI PARSEAZĂ FIŞIER HTML CU ELEMENT<VIDEO>
404
ESTE SUPORTAT RUNTIME-UL ?
nu
da
406
INVOCĂ API DE RENDERING VIDEO CA FIIND SUPORTAT DE BROWSER
408
STABILEŞTE OBIECT(E) PROXY, SETTERE ŞI GETTERE SAU PERECHEA OBJECT PROXY ŞI INVOCĂ MEDIUL DE RUNTIME
410
UTILIZEAZĂ OBIECT(E) PROXY PENTRU A PROPAGA COMENZI CĂTRE MEDIUL DE RUNTIME ŞI PENTRU A TRIMITE MAI DEPARTE SCHIMBĂRILE DE STARE ALE RUNTIME CĂTRE ELEMENTUL (FILE) UI PREVĂZUTE DE BROWSER

Fig. 4