(19) Deutsches Patent- und Markenamt

(10) **DE 20 2013 012 499 U1** 2017.03.09

(12) # Gebrauchsmusterschrift

(21) Aktenzeichen: **20 2013 012 499.7**
(22) Anmeldetag: **25.06.2013**
(67) aus Patentanmeldung: **EP 13 73 7038.3**
(47) Eintragungstag: **27.01.2017**
(45) Bekanntmachungstag im Patentblatt: **09.03.2017**

(51) Int Cl.: ***G06F 9/44*** (2006.01)

(30) Unionspriorität:
13/570,962 **09.08.2012** **US**

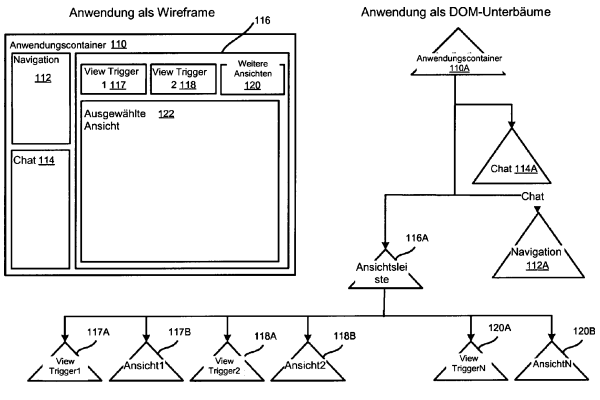
(73) Name und Wohnsitz des Inhabers:
GOOGLE INC., Mountain View, Calif., US

(74) Name und Wohnsitz des Vertreters:
Betten & Resch Patent- und Rechtsanwälte PartGmbB, 80333 München, DE

Die folgenden Angaben sind den vom Anmelder eingereichten Unterlagen entnommen

(54) Bezeichnung: **Hintergrundseite auf Browserebene zur Bereitstellung mehrfacher Ansichten**

(57) Hauptanspruch: System, das Folgendes umfasst:
einen Speicher; und
einen Prozessor, operativ verbunden mit dem Speicher, und so konfiguriert, dass er Code ausführt, um Folgendes zu erreichen:
das Bereitstellen eines HTML-Dokuments einschließlich einer Liste mit Modellen, worin jedes Modell in der Liste ein Tochterobjekt des Textkörpers des HTML-Dokuments ist;
das Erstellen eines Shadow DOM-Unterbaums im Textkörper des HTML-Dokuments, worin der Shadow DOM-Unterbaum eine oder mehrere Einfügemarken umfasst; und
das Festlegen, welches Modell aus der Liste mit Modellen in einer Ansicht einer Webseite angezeigt wird, mithilfe von Übereinstimmungskriterien einer oder mehrerer Einfügemarken.



Beschreibung

VERWEIS

[0001] Unter Schutz gestellt werden und Gegenstand des Gebrauchsmusters sind dabei, entsprechend den Vorschriften des Gebrauchsmustergesetzes, lediglich Vorrichtungen wie in den beigefügten Schutzansprüchen definiert, jedoch keine Verfahren. Soweit nachfolgend in der Beschreibung gegebenenfalls auf Verfahren Bezug genommen wird, dienen diese Bezugnahmen lediglich der beispielhaften Erläuterung der in den beigefügten Schutzansprüchen unter Schutz gestellten Vorrichtung oder Vorrichtungen.

TECHNISCHES GEBIET

[0002] Diese Beschreibung bezieht sich im Allgemeinen auf ein Model-View-Controller-Framework.

HINTERGRUND

[0003] Im Allgemeinen versuchen Webanwendungen heute, einem MVC-Designmuster (Model-View-Controller) für Benutzeroberflächen zu folgen. Das MVC-Designmuster unterteilt eine Anwendung in drei Verantwortungsbereiche: (a) das Modell (Model): die Domain-Objekte oder Datenstrukturen, die den Status der Anwendung repräsentieren; (b) die Ansicht (View), die den Status beobachtet und Ausgabedaten für die Nutzer generiert; und (c) die Steuerung (Controller), die die Eingabedaten der Nutzer in Vorgänge im Modell übersetzt. Ein Problem, auf das die Webanwendungen häufig stoßen, ist, dass ein DOM-Baum (Document Object Model), der die Webanwendung repräsentiert, einheitlich ist und keine Einkapselungsabstraktionen liefert. Dies führt zum Problemen bei der Aufrechterhaltung der logischen Trennung primitiver Datentypen (wie Model und View).

[0004] Ein Workaround für das MVC-Designmuster liefert alternative Betriebsmittel für den Benutzeroberflächenentwickler. Dieser Workaround resultiert jedoch in einer riesigen Menge von benutzeroberflächenspezifischen APIs. Deshalb besteht ein Bedarf an Systemen und Verfahren, um das Defizit der gegenwärtigen Technologie zu bewältigen und andere neue und innovative Eigenschaften zu bieten.

ZUSAMMENFASSUNG

[0005] Nach einem allgemeinen Aspekt kann ein Verfahren zur Implementierung von Webanwendungen das Projizieren eines Webseitenelements einer Webanwendung mithilfe eines Computergerät-Prozessors in die Ansicht der Webseite sein, wobei die Ansicht eine visuelle Ansicht des Webseitenmodells ist, und das Modell Anwendungsdaten und -regeln umfasst. Ein Controller kann Eingaben vermitteln und Eingaben in Befehle für die Ansicht oder das Modell umwandeln. Der Controller kann das eine, in die Ansicht der Webseite projizierte Element und ein anderes Element mithilfe einer Einfügemarke transponieren. Die Einfügemarke ist ein festgelegter Standort in einem Shadow-DOM-Unterbaum, die keinen Einfluss auf den DOM-Baum der Webseite nimmt.

[0006] Nach einem anderen allgemeinen Aspekt kann ein System einen Speicher und einen damit verknüpften Prozessor umfassen, welcher dazu konfiguriert wurde, Code auszuführen, um ein HTML-Dokument einschließlich einer Liste mit Modellen bereitzustellen, wobei jedes Modell in der Liste ein Tochterobjekt des Textkörpers des HTML-Dokuments ist, ein Shadow DOM-Unterbaum im Textkörper des HTML-Dokuments zu erstellen, wobei der Shadow DOM-Unterbaum eine oder mehrere Einfügemarken beinhaltet. In der Liste der Modelle wird festgelegt, welches Modell in der Ansicht einer Webseite angezeigt wird, und zwar in Übereinstimmung mit den Kriterien einer oder mehrerer Einfügemarken.

[0007] Nach einem weiteren allgemeinen Aspekt kann ein nicht-transitorisches, computerlesbares Medium ausführbaren Code enthalten, der bewirkt, dass ein Computergerät mithilfe eines Computergerätprozessors ein Webseitenelement in die Ansicht der Webseite projiziert, und dieses in die Ansicht der Webseite projizierte Element sowie ein anderes Element mithilfe einer Einfügemarke, die für einen bestimmten Standort in einem Shadow DOM-Unterbaum steht, transponiert, ohne die Hintergrundseite auf Browserebene darüber zu benachrichtigen. Die Hintergrundseite auf Browserebene kann eine Skriptdatei sein, die über keine eigene Benutzeroberfläche verfügt und als Container für alle Ansichten einer Webanwendung einschließlich der Ansicht der Webseite agiert. Jedes von der Webanwendung erstellte Fenster kann eine andere Ansicht der Hintergrundseite auf Browserebene darstellen.

[0008] Die Details einer oder mehrerer Implementierungen sind in den nachstehenden zugehörigen Zeichnungen und der Beschreibung dargelegt. Andere Features anhand der Beschreibung und Zeichnungen sowie anhand der Ansprüche ersichtlich werden.

KURZBESCHREIBUNG DER ZEICHNUNGEN

[0009] **Fig.** 1 ist ein exemplarisches Blockdiagramm eines Webanwendungs-Frameworks.

[0010] **Fig.** 2 ist ein exemplarisches Blockdiagramm eines Shadow DOM-Frameworks.

[0011] **Fig.** 3 ist ein exemplarisches Blockdiagramm eines anderen Shadow DOM-Frameworks.

[0012] **Fig.** 4A ist eine exemplarischer Benutzeroberfläche für eine Webanwendung, in Übereinstimmung mit den hier beschriebenen Technologien.

[0013] **Fig.** 4B ist eine andere exemplarische Benutzeroberfläche für eine Webanwendung, in Übereinstimmung mit den hier beschriebenen Technologien.

[0014] **Fig.** 5A, **Fig.** 5B und **Fig.** 5C beinhalten Beispiele für den Code der Webanwendung, die in **Fig.** 4A und **Fig.** 4B ausführlicher dargestellt sind.

[0015] **Fig.** 6 ist ein Flussdiagramm mit Darstellungen von exemplarischen Systemvorgängen, dargestellt in den **Fig.** 1–**Fig.** 5

[0016] **Fig.** 7 ist ein Blockdiagramm mit exemplarischen repräsentativen Computergeräte und zugehörigen Elementen, die zur Implementierung von Systemen und Verfahren verwendet werden können, wie in **Fig.** 1–**Fig.** 6

[0017] Ähnliche Referenzsymbole in den verschiedenen Zeichnungen verweisen auf ähnliche Elemente.

AUSFÜHRLICHE BESCHREIBUNG

[0018] Ein Document Object Model (DOM) ist eine plattformübergreifende und sprachenunabhängige Konvention für die Darstellung und Interaktion mit Objekten in HTML-, XHTML- und XML-Dokumenten. Wie hier verwendet, bezieht sich ein „Dokument" auf das zugrunde liegende Dokument für das DOM. Ein „Knoten" bezieht sich auf ein beliebiges DOM-Objekt in einem Baum. Ein DOM-„Baum" bezieht sich auf jeden Baum aus DOM-Objekten. Objekte in einem DOM-Baum können mithilfe von Objektverfahren adressiert und manipuliert werden. Eine „DOM-Struktur" bezieht sich auf einen DOM-Baum oder ein Fragment eines DOM-Baums.

[0019] Webanwendungen können einem MVC-Designmuster (Model-View-Controller) für Benutzeroberflächen folgen, das eine Anwendung in drei Verantwortungsbereiche unterteilt: (a) das Modell (Model): die Domain-Objekte oder Datenstrukturen, die den Status der Anwendung repräsentieren; (b) die Ansicht (View), die den Status beobachtet und Ausgabedaten für die Nutzer generiert; und (c) die Steuerung (Controller), die die Eingabedaten der Nutzer in Vorgänge im Modell übersetzt. Ein Problem, auf das die Webanwendungen häufig stoßen, ist, dass ein DOM-Baum (Document Object Model), der die Webanwendung repräsentiert, einheitlich ist und keine Einkapselungsabstraktionen liefert. Dies führt zum Problemen bei der Aufrechterhaltung der logischen Trennung primitiver Datentypen (wie Model und View).

[0020] Mithilfe der Implementierungen eines „Shadow DOM" (wie Einfügemarken und deren übereinstimmende Kriterien, ausführlich beschrieben auf FIG. (**Fig.** 1–**Fig.** 7) ist es möglich, eine bessere Aufteilung der Bedenken zwischen den primitiven Ansichts- und Modell-Datentypen zu gewährleisten. Mit einem Shadow DOM besteht ein HTML-Dokument aus einer Liste mit Modellen, wobei jedes Modell ein Tochterobjekt des Textkörperabschnitts des Dokuments ist. Ein Shadow DOM-Unterbaum wird im Textkörper des Dokuments erstellt. Dieser Unterbaum repräsentiert die Ansicht. In diesem Unterbaum werden eine oder mehrere Einfügemarken als Öffnungen der Ansicht verwendet, durch welche die Model-Bits durchscheinen. Die übereinstimmenden Einfügemarken werden vom Controller verwendet, um festzulegen, welches Modell in einer Ansicht gezeigt werden soll. Die Übereinstimmungskriterien können durch ein „select"-Attribut im (content) HTML-Element repräsentiert werden.

[0021] Wenn der Controller (z. B. Webanwendung) diesen Attributwert modifiziert, ändert sich das Übereinstimmungskriterium, was wiederum eine Änderung des Elements, welches anstelle dieses <content>-HTML-Elements dargestellt wird, auslöst. Der Entwickler einer Webanwendung kann DOM-Elemente im Dokument als Ansichten zuweisen und die Übereinstimmungskriterien ändern, damit sie mit beliebigen dieser DOM-Elemente übereinstimmen, welche anstelle des <content>-HTML-Elements erscheinen. Dadurch wird eine klare Trennung zwischen dem Content der Ansicht und der Vorgehensweise bei der Kontrolle seiner Auswahl sichergestellt. Die Model-Bits werden niemals geändert oder vom Controller (oder der Ansicht) informiert, selbst wenn die Ansicht geändert wurde. Auf diese Weise können die Benutzeroberflächen der Webanwendungen und die Benutzeroberflächenkomponenten manipuliert, modifiziert und verbessert werden, ohne dass der Basiscode des Dokumenten-Markups des Webanwendungmodells geändert werden muss.

[0022] **Fig.** 1 ist ein exemplarisches Blockdiagramm eines Webanwendungs-Frameworks. In diesem Zusammenhang kann eine „Webanwendung" für eine einzige Aufgabe oder multiple Aufgaben für den Benutzer konfiguriert sein. In einer solchen Implementierung kann die Webanwendung so konfiguriert werden, dass sie durch den Webbrowser ausgeführt oder interpretiert werden kann. Das wird verglichen mit den nativen Anwendungen, die maschinenausführbaren Code beinhalten und konfiguriert sind, um direkt von einem Prozessor oder durch das Betriebssystem des Client-Gerätes ausgeführt zu werden, wohingegen eine Webanwendung ohne die Hilfe des Webbrowsers zur Ausführung oder Anzeige unfähig sein kann. Folglich können Webanwendungen innerhalb eines Browsers mit einem dedizierten User-Interface ausgeführt werden und eine Funktionalität und Nutzererfahrung verschaffen, die reichhaltiger und interaktiver als eine eigenständige Website ist und gleichzeitig weniger umständlich und monolithisch im Vergleich zu einer Desktopanwendung. Beispiele für Webanwendungen beinhalten Spiele, Foto-Editoren und Videoplayer, die innerhalb des Browsers ausgeführt werden.

[0023] Entwickler von Webanwendungen müssen oft eine Einkapselung der DOM-Struktur liefern. Obwohl sie Teil des Dokumenten-Baums sind, kann es viele funktionelle Fragmente des DOM (oder der DOM-Unterbäume) sowie Annahmen über diese unabhängig agierenden Fragmente geben. Dieser Einkapselungstyp wird als die „funktionelle Einkapselung" bezeichnet, im Gegensatz zu einer „Vertrauenseinkapselung", die sich auf die Begrenzung des Informationsflusses basierend auf Vertrauen und Gewährleistung der Sicherheit der Daten und des Status innerhalb einer Anwendung bezieht. Die funktionelle Einkapselung legt Funktionsgrenzen in einem Dokumentenbaum fest. Eine Funktionsgrenze (oder „Grenze") ist eine Beschreibung funktioneller Bedenken zwischen zwei lose verknüpften Funktionseinheiten.

[0024] Die Benutzeroberfläche einer Webanwendung kann aus mehreren Benutzeroberflächenelementen (oder „Widgets") 9 bestehen, wobei jede von ihnen einen DOM-Unterbaum repräsentiert. In den Fällen, wo ein Widget mit dem Hosten anderer Widgets beauftragt ist, muss das Widget verstehen, wo sein DOM-Unterbaum endet und der DOM-Unterbaum eines anderen Widgets beginnt.

[0025] Dieses Bedürfnis nach Verständnis der Funktionsgrenzen in einem Dokumentenbaum ist noch ausgeprägter, wenn ein Widget von einem externen Akteur, wie der Webanwendung, die diese Widgets konsumiert, verwaltet wird – hinzugefügt, bewegt oder aus der Dokumentenstruktur gelöscht. Wenn ein Widgetnutzer nicht genau versteht, wie die DOM-Struktur eines Widgets gestaltet ist, ist es für ihn nicht möglich, vernünftig mit dem Widget zu arbeiten.

[0026] Wie in der Implementierung auf der **Fig.** 1 gezeigt, kann ein Muster-Anwendungscontainer **110** ein Navigationswidget **112**, ein Chat-Widget **114**, ein „View Trigger 1"-Widget **116**, ein „View Trigger 2"-Widget **118**, ein „Mehr Ansichten"-Widget **120** und ein „Ausgewählte Ansicht"-Widget **122** umfassen.

[0027] Wenn die Webanwendung in Form von DOM-Unterbäumen gestaltet ist, wie auf **Fig.** 1 dargestellt, kann ein Anwendungscontainer-Knoten **110A** einen Knoten-Chat **114A** als Tochterobjekt enthalten, welches wiederum eine Ansichtsleiste **116A** und eine Navigationsleiste **112A** als untergeordnete Knoten enthält. Die Ansichtsleiste **116A** kann als ihre Tochterobjekte einen View Trigger **1117A**, seine Ansicht **117B**, den View Trigger **2118A**, seine Ansicht **118B** und so weiter enthalten, bis zu einer beliebigen Anzahl von View Trigger, dargestellt durch den View Trigger N **120A** und seine Ansicht N **120B**.

[0028] **Fig.** 2 ist ein exemplarisches Blockdiagramm eines Shadow DOM-Frameworks. Wie in **Fig.** 2, ein Dokumentenbaum **210** kann einen Shadow Host **212** und einen oder mehrere Tochterknoten **213**, **214**, **215**, **216**, **217** und **218** enthalten. Der Dokumentenbaum **210** kann eine beliebige Anzahl von Tochterknoten enthalten. Die Shadow DOM-Unterbäume **220** können beispielsweise von einem Entwickler der Webanwendung, der den Dokumentenbaum **210** entwickelt hat, erstellt werden.

[0029] Eine „Shadow DOM"-Struktur ermöglicht es, dass multiple Shadow DOM-Unterbäume **220** (zusätzlich zum Dokumentenbaum **210**) bei der Wiedergabe zu einem riesigen Baum zusammengefasst werden (beispielsweise in einem Tenderer-Browservorgang, wie nachfolgend ausführlich beschrieben). Die Existenz der multiplen Shadow DOM-Unterbäume **220** wird dadurch gewährleistet, dass jedes Element im Dokumentenbaum **210** einen oder mehrere zusätzliche DOM-Unterbäume hostet (wie Shadow DOM-Unterbäume **220**). Diese Shadow DOM-Unterbäume **220** können mithilfe eines Satzes von Regeln, die Einkapselungsgrenzen unter Beibehaltung der standardmäßigen DOM-Kompositionssemantik festlegen, gesteuert werden.

[0030] Die Einkapselungsgrenzen zwischen den Shadow DOM-Unterbäume **220** werden als Shadow Boundaries **230** bezeichnet. Die Elemente, die Shadow DOM-Unterbäume hosten, werden als Shadow Hosts **212** bezeichnet und die Root-Knoten der Shadow DOM-Unterbäume werden Shadow Roots **240** genannt. Die Shadow Roots **240** können einen oder mehrere Tochterknoten haben, wie Knoten **242**, **243**, **244**, **245**, **246** und **247**.

[0031] Ein Webbrowser, der die Webanwendungen, die in den **Fig.** 1 und **Fig.** 2 beschrieben sind, implementiert, kann in manchen Implementierungen in einer Multiprozessarchitektur arbeiten, sodass ein einziger Browserprozess alle Tabs, Fenster und den „Chrome" des Webbrowsers (z. B. die Benutzeroberfläche wie Suchleiste, Symbole, Bereich außerhalb einer Webseite des Webbrowsers) verwaltet. Der Browserprozess kann mehrere Tenderer-Prozesse (auch „Tenderer" genannt) erstellen, von denen jeder Prozess für die Wiedergabe von Webseiten zuständig ist. Die Tenderer-Prozesse können die gesamte komplexe Logik für die Arbeit mit HTML, JavaScript, CSS, Bildern etc. enthalten. Jeder Tenderer-Prozess wird in einer Sandbox-Umgebung ausgeführt, was bedeutet, dass er keinen direkten Zugriff auf die Festplatte, das Netzwerk oder die Anzeige eines Computergeräts hat. Alle Interaktionen mit Webanwendungen, einschließlich der Nutzereingaben und Screen Painting können über den Browserprozess ausgeführt werden. Dadurch kann der Browserprozess die Tenderer auf verdächtige Aktivitäten überwachen und sie bei Verdacht auf Exploit-Aktivitäten stilllegen. Der Browserprozess kann außerdem einen separaten Prozess für jede Art von verwendeten Plug-ins, wie FLASH, erstellen. Diese Prozesse können nur die Plug-ins selbst enthalten, zusammen mit einigem Code für die Interaktion mit dem Browser und den Wiedergabeprozessen. Sobald ein Webbrowser seine Browserprozesse erstellt hat, kann er in manchen Implementierungen auch einen Tenderer-Prozess für jede Instanz der Website, die ein Nutzer besucht, erstellen. Dieser Ansatz zielt darauf ab, Seiten von unterschiedlichen Websites von einander zu isolieren. Andere Browsertypen mit Tenderern können mithilfe einer einzigen Prozessarchitektur ausgeführt werden.

[0032] Bei der Wiedergabe beispielsweise durch einen Tenderer eines Webbrowsers kann der Shadow DOM-Unterbaum **220** den Contentplatz des Shadow Hosts **212** einnehmen. Wenn wiedergegeben beinhaltet der Content des Shadow Hosts **212** (repräsentiert durch den Knoten **212A**, gezeigt in der **Fig.** 2) die Tochterobjekte **242A**, **243A**, **244A**, **245A**, **246A** und **247A**, die den Inhalt des Shadow DOM-Unterbaums **220** repräsentieren. Wenn wiedergegeben kann der Dokumentenbaum **210** den Content eines oder mehrerer Shadow DOM-Unterbaums **220** enthalten, sodass eine Webanwendung Teile der Ansicht freilegen und zur Freilegung anderer Teile einer Webseite wechseln kann. Dadurch kann ein Entwickler der Webanwendung die Notwendigkeit für das Neucodieren einer ganzen Webanwendung oder Webseite sowie die Komplexität der Webseitenverwaltung mithilfe des MVC-Designmusters vermeiden.

[0033] **Fig.** 3 ist ein exemplarisches Blockdiagramm eines anderen Shadow DOM-Frameworks. Um die Komposition der Tochterobjekte des Shadow Hosts und des Shadow DOM-Unterbaums zu ermöglichen, kann ein Satz von Einfügemarken **326** und **329** verwendet werden. Eine Einfügemarke bezieht sich auf einen definierten Standort im Shadow DOM-Unterbaum **220**, wohin die Tochterobjekte des Shadow Hosts bei ihrer Wiedergabe transponiert werden.

[0034] Wie in **Fig.** 3, ein Dokumentenbaum **302** kann einen Shadow Host **304** und einen oder mehrere Tochterknoten **306**, **308**, **310**, **312**, **314** und **316** enthalten. Das Shadow DOM **220** kann Shadow Roots **322**, ein Tochterobjekt **324** und eine Einfügemarke **236** enthalten. Das Tochterobjekt **324** kann außerdem einen Tochterknoten **328** und eine Einfügemarke 2 **329** enthalten. Wenn der Baum **340** wiedergegeben wird, kann der Shadow Host **304A** ein Tochterobjekt **306A** enthalten, welches wiederum die Tochterknoten **310A** und **312A** enthält. Der Knoten **312** kann das Tochterobjekt des Shadow Hosts anstelle der Einfügemarke 2 **329** sein. Der Shadow Host **304A** kann ein Shadow Host-Tochterobjekt **308A** statt der Einfügemarke **326** enthalten, welche die Tochterknoten **314A** und **316A** umfassen kann. Wie oben beschrieben beziehen sich die Einfügemarken **326** und **329** auf einen definierten Standort im Shadow DOM-Unterbaum **220**, wohin die Tochterobjekte des Shadow Hosts (wie **306**, **308**) bei ihrer Wiedergabe transponiert werden.

[0035] Um die Einkapselung mit niedrigen Boundaries aufrechtzuerhalten, beinhaltet die Verteilung der Tochterknoten des Shadow Hosts auf die Einfügemarken im damit zusammenhängenden Shadow DOM-Unterbaum mehrere Eigenschaften. Beispielsweise wird sich die Verteilung nicht auf den Status des Dokumentenbaums oder der Shadow DOM-Unterbäume auswirken. Jede Einfügemarke partizipiert an der Verteilung, indem sie Übereinstimmungskriterien für die Tochterknoten bereitstellt. Die Übereinstimmungskriterien legen fest, ob ein bestimmter Knoten an eine bestimmte Einfügemarke ausgeliefert werden kann. Die Verteilung ist ein Ergebnis der Ausführung eines stabilen Algorithmus. Die Verteilung selbst bewirkt keine Änderung der Variablen, die sich auf die Verteilung auswirken. Die Verteilung tritt immer dann auf, wenn eine Variable, die sich auf sie auswirkt, geändert wird

[0036] Der Verteilungsalgorithmus kann ein Ergebnis erzeugen, das dem Ergebnis bei der Verarbeitung der folgenden Schritte entspricht: Eingabe: BAUM, der ein Shadow DOM-Unterbaum ist; POOL, welcher eine Liste mit DOM-Knoten ist. Ausgabe: Die Knoten im POOL werden auf die Einfügemarken im BAUM verteilt. Wiederholen für jede aktive Einfügemarke im BAUM, in der Baumabfolge:
POINT ist die aktuelle Einfügemarke
Für jeden Knoten im POOL wiederholen:
NODE ist der aktuelle Knoten
Wenn NODE den Übereinstimmungskriterien des POINT entspricht:
NODE nach POINT verteilen
NODE aus dem POOL entfernen
Andernfalls weiter wiederholen
Weiter wiederholen

Übereinstimmende Einfügemarken

[0037] Die Übereinstimmungskriterien für eine Einfügemarke werden als ein Satz von Selektorfragmenten definiert. Jedes Selektorfragment ist ein Fragment im Selektor (Shadow Host) **>** (Fragment), wo (Shadow Host) ein Selektor ist, der den Shadow Host identifiziert, und das (Fragment) das Selektorfragment ist.

Übereinstimmender Host und Tochterobjekte, verteilt auf Einfügemarken

[0038] Zwei Arten von Selektoren, deklariert in den Übereinstimmungselementen der Shadow DOM-Unterbäume außerhalb des Baums, in dem sie deklariert sind: (1) A :host-Pseudoklasse, die mit dem Shadow Host eines Shadow DOM-Unterbaums übereinstimmt; und (2) ein Referenzauswahl-Combinator, der mit den Knoten, die aktuell einer Einfügemarke zugeteilt wurden, übereinstimmt. Die :host-Pseudoklasse repräsentiert den Shadow Host eines Shadow DOM-Unterbaums. Wenn der kontextbezogene Referenzelementsatz leer ist oder Elemente außerhalb eines Shadow DOM-Unterbaums enthält, übereinstimmt :host mit keinem Element. Die Referenz-Combinatoren übereinstimmen mit den Tochterobjekten eines Shadow Hosts, verteilt auf die Einfügemarken innerhalb eines Shadow DOM-Unterbaums. Es gelten folgende Bedingungen: Der Combinator-Wert ist „select"; der erste zusammengesetzte Selektor des Combinators übereinstimmt mit einer Einfügemarke und der zweite zusammengesetzte Selektor übereinstimmt mit einem Element, das an diese Einfügemarke verteilt wurde.

[0039] Beispielweise wird .some-insertion-point/select/div.special mit allen div-Elementen übereinstimmen, die an eine Einfügemarke mit einem auf some-insertion-point festgesetzten class-Attribut verteilt wurden.

[0040] Ein Shadow Host kann mehr als einen Shadow DOM-Unterbaum hosten. In solchen Fällen werden die Unterbäume in der Abfolge gestapelt, in der sie zum Host hinzugefügt wurden, beginnend mit dem zuletzt hinzugefügten Unterbaum. Dieser Unterbäume-Satz wird als Baumstapel bezeichnet. Der in letzter Zeit hinzugefügte Unterbaum wird als der jüngere Baum bezeichnet, und der zuerst hinzugefügte Unterbaum ergo der älteste Baum. Der zuletzt hinzugefügte Unterbaum wird als der jüngste Baum bezeichnet.

[0041] Um das Zusammenstellen multipler Shadow Unterbäumen desselben Hosts zu vereinfachen, wird ein spezieller Typ der Einfügemarke definiert. Die Shadow Einfügemarke weist einen Ort im Shadow DOM-Unterbaum zu, und zwar den Ort, an dem der ältere Baum eingefügt wird.

[0042] Vergleichsweise kann ein Shadow DOM-Unterbaum als eine Stelle zwischen einem DOM-Unterbaum in einem Dokument und einem Dokumentfragment gesehen werden. Da er wiedergegeben wird, zielt ein Shadow DOM-Unterbaum darauf ab, die Eigenschaften eines typischen DOM-Unterbaums in einem Dokument beizubehalten. Gleichzeitig handelt es sich hierbei um eine Einkapselungsabstraktion, weshalb es keine Aus-

wirkungen auf den DOM-Unterbaum des Dokuments geben darf. So verhalten sich die HTML-Elemente wie in den Shadow DOM-Unterbäumen festgelegt, mit einigen Ausnahmen.

Beispiel für ein Shadow DOM

[0043] Beispielsweise soll ein Entwickler eine einfache Liste mit Links in ein Nachrichtenwidget, das Links in zwei Kategorien, nämlich „Breaking News" und einfach News organisiert, umwandeln. Das aktuelle Dokumenten-Markup für die Nachrichten kann wie folgt aussehen:

```
<ul class-„stories“>
<li><a href=„//examplc.com/stories/1“>Eine Nachricht</a></li>
<li><a href=„//examplc.com/stories/2“>Andere Nachricht</a></li>
<li class=„breaking“><a href-„//example.com/stories/3“>Noch eine Nachricht</a></li>
<li><a href=„//examplc.com/stories/4“>Eine weitere Nachricht</a></li>
<li><a href=//example.com/stories/4„>Tolle Nachricht</a></li>
<li class=“breaking„><a href-“//example.com/stories/5„>Schreckliche Nachricht</a></li> </ul>
```

[0044] Um die Nachrichten zu organisieren, entscheidet sich der Entwickler für ein Shadow DOM. Auf diese Weise kann Bob das Dokument-Markup übersichtlich gestalten, dank der Nutzung von Einfügemarken wird das Sortieren von Nachrichten nach Klassennamen zu einer sehr einfachen Aufgabe.

[0045] Der Entwickler erstellt eine Simulation des folgenden Shadow DOM-Unterbaums, der von dem folgenden ul-Element gehostet wird:

```
<div class=“breaking„>
<ul>
<content select=“.breaking„></content> <!— Einfügemarke für Breaking News -->
</ul>
</div>
<div class=“other„>
<ul>
<content></content> <!— Einfügemarke für restliche Nachrichten —>
</ul>
</div>
```

[0046] Der Entwickler gestaltet das neue Widget anschließend gemäß den Vorgaben des Designers und fügt diese Simulation dem Shadow DOM-Unterbaum hinzu:

```
〈style scoped〉
div.brcaking {
color: Red;
font-size: 20px;
border: lpx dashed Purple;
}
div.other {
padding: 2px 0 0 0;
border: lpx solid Cyan;
}
</stylc>
```

[0047] Der Entwickler wandelt die Simulation zum Code um:

```
function crcateStoryGroup(className, contentSelector)

{
var group = documcnt.createElcmcnt('div');
group.className = className;
// Leere Zeichenfolge im Select-Attribut oder die Abwesenheit dessen führen zum gleichen
Ergebnis, weshalb keine speziellen Aktionen erforderlich sind.
group.innerHTML = '<ul><content select="' + contentSelector + n'></content></ul>'; return
group;
}
function createStyle()
{

var style = document.createElement('style');
style.scoped = true;
style.textContent = 'div.breaking { color: Red;font-size: 20px; border: lpx dashed Purple;} ' +
'div.other { padding: 2px 0 0 0; border: lpx solid Cyan;} ';
return style;
}
function makeShadowSubtree(storyList)
{
var root = new ShadowRoot(storyList);
root.appendChild(createStyle());
root.appendChild(createStoryGroup('breaking', '.breaking'));
root.appendChild(createStoryGroup('other',“));
}
document.addEventListener('DOMContentLoaded', function() {
[ ].forEach.call(document.querySelectorAll('ul.stories'), makeShadowSubtree);
});
```

[0048] **Fig.** 4A ist eine exemplarischer Benutzeroberfläche für eine Webanwendung, in Übereinstimmung mit den hier beschriebenen Technologien. Die Benutzeroberfläche **410** kann eine Benutzeroberfläche für einen Teil der Webanwendung (z. B. Webseite, die im Webbrowser angezeigt wird) sein, die eine Hintergrundseite auf Browserebene und eine Shadow DOM-Architektur verwendet, wie vorstehend beschrieben in Bezug auf **Fig.** 1–**Fig.** 3. Die Benutzeroberfläche **410** kann unterschiedliche Content-Elemente umfassen, wie Textelement **420**, welches Texte wie „Das ist der erste Bildschirm” und „Hallo und willkommen!” enthalten kann. Die Benutzeroberfläche **410** kann außerdem ein Link-Element **430** wie „Zum zweiten Bildschirm wechseln” enthalten. Wie vorstehend in Bezug auf **Fig.** 1–**Fig.** 3 beschrieben, wenn der Controller (z. B. Webanwendung) einen Attributwert für ein Content-Element modifiziert, ändern sich die Übereinstimmungskriterien, was wiederum eine Änderung des Elements, welches anstelle dieses content-Elements dargestellt wird (wie Elemente **420** oder **430**), auslöst. Der Entwickler einer Webanwendung kann danach die DOM-Elemente (wie Elemente **420** und **430**) im Dokument als Ansichten zuweisen und die Übereinstimmungskriterien so ändern, dass sie mit diesen DOM-Elementen übereinstimmen, damit diese statt der content-Elemente angezeigt werden, wie nachstehend beschrieben in Bezug auf **Fig.** 5.

[0049] **Fig.** 4B ist eine andere exemplarische Benutzeroberfläche für eine Webanwendung, in Übereinstimmung mit den hier beschriebenen Technologien. Die Benutzeroberfläche **410** kann unter Verwendung desselben DOM für die Webanwendung, die in Bezug auf **Fig.** 4A beschrieben ist, mit Änderungen der Ansicht der Benutzeroberfläche **410** programmiert werden. Solche Modifizierung der Ansicht kann ausgeführt werden, ohne dass die Hintergrundseite auf Browserebene, die mit der Webanwendung zusammenhängt, benachrichtigt oder modifiziert werden muss, wie vorstehend beschrieben in Bezug auf **Fig.** 1–**Fig.** 3.

[0050] Wie in **Fig.** 4B beschrieben, kann die Benutzeroberfläche **410** immer noch das Textelement **420** enthalten, welches nun eine andere Auswahl im Vergleich zu **Fig.** 4A aufweist, wie z. B. „Das ist der zweite Bildschirm” und „Auf Wiedersehen”. Die Benutzeroberfläche **510** kann auch ein Link-Element **430** enthalten, welches nun einen anderen auswählbaren Link darstellt, wie „Zum ersten Bildschirm zurückkehren”. Im Beispiel, dargestellt in **Fig.** 4B, ein Entwickler der Webanwendung mit der Benutzeroberfläche **410** hat die DOM-Elemente **420** und **430** im Dokument als Ansichten zugewiesen und die Übereinstimmungskriterien so geändert,

dass sie mit den DOM-Elementen übereinstimmen, damit der neue Content (z. B. Text und Links) statt des Content für dieselben Content-Elemente **420** und **430**, wie in **Fig.** 4A gezeigt wird.

[0051] **Fig.** 5A, **Fig.** 5B und **Fig.** 5C beinhalten Beispiele für den Code der Webanwendung, die in **Fig.** 4A und **Fig.** 4B ausführlicher dargestellt sind. Der Code für die Beispiel-Webanwendung, besprochen in Bezug auf **Fig.** 4A und **Fig.** 4B kann eine index.HTML-Datei, eine views.js JAVASCRIPT-Datei und eine views.css CSS-Datei enthalten. Die HTML-Datei kann einen Code wie in **Fig.** 5A gezeigt, enthalten. Die views.js-Datei kann einen Code wie in **Fig.** 5B gezeigt, enthalten. Die views.css-Datei kann einen Code wie in **Fig.** 5C gezeigt, enthalten.

[0052] **Fig.** 6 ist ein Flussdiagramm mit Darstellungen von exemplarischen Systemvorgängen, dargestellt in den **Fig.** 1–**Fig.** 5 Der Prozess in **Fig.** 6 kann zumindest von einer Webanwendung ausgeführt werden, die durch einen Prozessor eines Computergeräts ausgeführt wird. Beispiele für Prozessoren und Computergeräte, die für die Ausführung von Webanwendungen in Frage kommen, sind nachstehend detailliert in Bezug auf **Fig.** 7. Wie in **Fig.** 6 beschrieben, der Prozess **600** beinhaltet die Bereitstellung eines HTML-Dokuments einschließlich einer Liste mit Modellen (**610**). Jedes Modell in der Liste kann ein Tochterobjekt des Textkörpers eines HTML-Dokuments (**612**) sein. Der Prozess **600** beinhaltet die Erstellung eines Shadow DOM-Unterbaums im Textkörper des HTML-Dokuments, wobei der Shadow DOM-Unterbaum eine oder mehrere Einfügemarken (**620**) umfasst. Die eine oder mehrere Einfügemarken können jeweils einen definierten Standort im Shadow DOM-Unterbaum (**622**) haben. Beim Prozess **600** kann die Webanwendung festlegen, welches Modell aus der Liste mit Modellen in einer Webseiten-Ansicht angezeigt wird, mithilfe von Übereinstimmungskriterien einer oder mehrerer Einfügemarken (**630**). Der Prozess **600** kann die Wiedergabe in einem Webbrowser, auf einer Webseite einschließlich des Ersatzes eines Contents eines Shadow Hosts der Webseite mit einem Shadow DOM-Unterbaums (**640**) enthalten. In manchen Implementierungen kann die Wiedergabe die Transposition jeglicher Tochterobjekte eines Shadow Hosts an den definierten Standort bzw. Standorte (**642**) beinhalten.

[0053] **Fig.** 7 ist ein Blockdiagramm mit exemplarischen repräsentativen Computergeräte und zugehörigen Elementen, die zur Implementierung von Systemen und Verfahren verwendet werden können, wie in **Fig.** 1–**Fig.** 6 Das Computergerät **700** ist zur Darstellung verschiedener Formen von Digitalcomputern vorgesehen, wie Laptops, Desktops, Workstations, Personal Digital Assistants, Server, Blade-Server, Mainframes und andere geeignete Computer. Computergerät **750** soll verschiedene Formen mobiler Geräte, wie Personal Digital Assistants, Mobiltelefone, Smartphones und andere ähnliche Computergeräte, darstellen. Die hier gezeigten Komponenten, ihre Verbindungen und Beziehungen und ihre Funktionen sollen nur exemplarisch sein und sollen Implementierungen der in diesem Dokument beschriebenen und/oder beanspruchten Erfindungen nicht einschränken.

[0054] Das Computergerät **700** beinhaltet einen Prozessor **702**, einen Speicher **704**, ein Speichergerät **706**, eine Hochgeschwindigkeitsschnittstelle **708**, die mit dem Speicher **704** und Hochgeschwindigkeits-Erweiterungsanschlüssen **710** verbindet, und eine langsame Schnittstelle **712**, die mit einem langsamen Bus **714** und dem Speichergerät **706** verbindet. Alle der Komponenten **702**, **704**, **706**, **708**, **710** und **712** sind mithilfe verschiedener Busse miteinander verbunden und können an einer gemeinsamen Hauptplatine oder auf andere Weise, wie geeignet, angebracht sein. Der Prozessor **702** kann Anweisungen für die Ausführung im Computergerät **700** verarbeiten, zum Beispiel Anweisungen, die im Speicher **704** oder Speichergerät **706** gespeichert sind, um grafische Informationen für eine grafische Benutzeroberfläche auf einem externen Eingabe-/Ausgabegerät anzuzeigen, zum Beispiel Display **716**, das mit der High-Speed-Schnittstelle **708** gekoppelt ist. In anderen Implementierungen können mehrere Prozessoren und/oder mehrere Busse verwendet sein, wie angemessen, zusammen mit mehreren Speichern und Speichertypen. Außerdem können mehrere Computergeräte **700** verbunden sein, wobei jedes Gerät Teile der benötigten Operationen bereitstellt (z. B. als eine Serverbank, eine Gruppe von Blade-Servern oder ein Multi-Prozessor-System).

[0055] Der Speicher **704** zeichnet Informationen im Computergerät **700** auf. In einer Implementierung ist der Speicher **704** aus einer nichtflüchtigen Speichereinheit oder -einheiten ausgebildet. In einer anderen Implementierung ist der Speicher **704** eine nichtflüchtige Speichereinheit oder -einheiten. Der Speicher **704** kann auch eine andere Form von computerlesbarem Medium sein, zum Beispiel ein magnetischer oder optischer Datenträger.

[0056] Das Speichergerät **706** ist geeignet, Massenspeicherung für das Computergerät **700** bereitzustellen. In einer Implementierung kann das Speichergerät **706** ein computerlesbares Medium sein oder beinhalten, wie ein Floppy-Disk-Laufwerk, ein Festplattenlaufwerk, ein Optikplattenlaufwerk, ein Magnetbandlaufwerk, ein Flash-Speicher oder anderes ähnliches Solid-State-Speichergerät oder eine Reihe anderer Geräte, sein, ein-

schließlich Geräten in einem Speichernetzwerk oder anderer Konfigurationen. Ein Computerprogrammprodukt kann konkret in einem Informationsträger ausgeführt sein. Das Computerprogrammprodukt kann auch Anweisungen enthalten, die, wenn sie ausgeführt werden, ein oder mehrere Verfahren ausführen, wie die oben beschriebenen. Der Informationsträger ist ein computer- oder maschinenlesbares Medium, wie der Speicher **704**, das Speichergerät **706** oder der Prozessorspeicher **702**.

[0057] Der Hochgeschwindigkeits-Controller **708** verwaltet bandbreitenintensive Vorgänge für das Computergerät **700**, während der langsame Controller **712** Vorgänge mit niedrigerer Bandbreite verwaltet. Eine solche Zuordnung von Funktionen ist nur exemplarisch. In einer Implementierung ist der Hochgeschwindigkeits-Controller **708** an den Speicher **704**, die Anzeige **716** (z. B. durch einen Grafikprozessor oder -beschleuniger) und an die Hochgeschwindigkeits-Erweiterungsanschlüsse **710** gekoppelt, die verschiedene Erweiterungskarten aufnehmen können (nicht gezeigt). In der Implementierung ist der langsame Controller **712** an das Speichergerät **706** und an den langsamen Erweiterungsanschluss **714** gekoppelt. Der langsame Erweiterungsanschluss, der verschiedene Kommunikationsanschlüsse (z. B. USB) beinhalten kann, kann an ein oder mehrere Eingabe-/Ausgabe-Geräte, wie eine Tastatur, ein Zeigegerät, einen Scanner oder ein Netzwerkgerät, wie einen Switch oder Router, z. B. durch einen Netzwerkadapter gekoppelt sein.

[0058] Das Computergerät **700** kann in einer Reihe verschiedener Formen implementiert werden, wie in der Figur gezeigt. Es kann zum Beispiel als Standardserver **720** oder mehrmals in einer Gruppe solcher Server implementiert sein. Es kann auch als Teil eines Rackserversystems **724** implementiert sein. Außerdem kann es in einem Personal Computer, wie Laptop-Computer **722**, implementiert sein. Alternativ können Komponenten von Computergerät **700** mit anderen Komponenten in einem mobilen Gerät kombiniert sein (nicht dargestellt), z. B. Gerät **750**. Jedes solcher Geräte kann eines oder mehrere Computergeräte **700**, **750** enthalten, und ein gesamtes System kann aus mehreren Computergeräten **700**, **750** bestehen, die miteinander kommunizieren.

[0059] Das Computergerät **750** beinhaltet einen Prozessor **752**, Speicher **764**, ein Eingabe-/Ausgabegerät, wie eine Anzeige **754**, eine Kommunikationsschnittstelle **766** und einen Sende-Empfänger **768**, unter anderen Komponenten. Das Gerät **750** kann ebenfalls mit einer Speichervorrichtung z. B. einem Microdrive oder einem anderen Gerät ausgestattet werden, um zusätzlichen Speicher bereitzustellen. Alle der Komponenten **750**, **752**, **764**, **754**, **766** und **768** sind mithilfe verschiedener Busse miteinander verbunden und mehrere der Komponenten können an einer gemeinsamen Hauptplatine oder auf andere Weise, wie geeignet, angebracht sein.

[0060] Der Prozessor **752** kann Anweisungen im Computergerät **750** ausführen, zum Beispiel Anweisungen, die in Speicher **764** gespeichert sind. Der Prozessor kann als ein Chipsatz von Chips implementiert werden, die separate und mehrere analoge und digitale Prozessoren beinhalten. Der Prozessor kann zum Beispiel für die Koordination der anderen Komponenten des Geräts **750** sorgen, zum Beispiel die Kontrolle von Benutzeroberflächen, Anwendungen, die vom Gerät **750** ausgeführt werden, und die drahtlose Kommunikation durch Gerät **750**.

[0061] Prozessor **752** kann mit einem Benutzer über Steuerschnittstelle **758** und Displayschnittstelle **756** kommunizieren, die mit einem Display **754** gekoppelt ist. Die Anzeige **754** kann zum Beispiel eine TFT-LCD-(Thin-Film-Transistor Liquid Crystal Display) oder eine OLED-Anzeige (organische Leuchtdiode) oder eine andere angemessene Anzeigetechnologie sein. Die Anzeigeschnittstelle **756** kann angemessene Verschaltung zum Treiben der Anzeige **754** umfassen, um einem Benutzer grafische und andere Informationen zu präsentieren. Die Steuerschnittstelle **758** kann Befehle von einem Benutzer empfangen und sie für die Sendung an Prozessor **752** umwandeln. Außerdem kann eine externe Schnittstelle **762** in Kommunikation mit Prozessor **752** bereitgestellt werden, um die Nahbereichskommunikation von Gerät **750** mit anderen Geräten zu ermöglichen. Die externe Schnittstelle **762** kann beispielsweise in manchen Implementierungen eine drahtgestützte Verbindung oder in anderen Implementierungen eine drahtlose Verbindung aufbauen, und es können auch mehrere Schnittstellen zur Anwendung kommen.

[0062] Der Speicher **764** zeichnet Informationen im Computergerät **750** auf. Der Speicher **764** kann als ein computerlesbares Medium bzw. als eines von mehreren computerlesbaren Medien, als flüchtiger Speicher bzw. als flüchtige Speicher oder als ein nichtflüchtiger Speicher bzw. als nichtflüchtige Speicher implementiert werden. Erweiterungsspeicher **774** kann ebenfalls bereitgestellt und mit dem Gerät **750** über Erweiterungsschnittstelle **772** verbunden werden, die zum Beispiel eine SIMM(Single In Line Memory Module)-Kartenschnittstelle umfassen kann. Ein solcher Erweiterungsspeicher **774** kann zusätzlichen Speicherplatz für Gerät **750** bereitstellen oder er kann auch Anwendungen oder andere Informationen für Gerät **750** speichern. Insbesondere kann Erweiterungsspeicher **774** Anweisungen zum Ausführen oder Ergänzen der oben beschriebenen Prozesse enthalten und er kann außerdem sichere Informationen enthalten. Somit kann Erweiterungsspeicher

774 zum Beispiel als Sicherheitsmodul für Gerät **750** bereitgestellt werden und er kann mit Anweisungen programmiert sein, die die sichere Verwendung von Gerät **750** erlauben. Zusätzlich dazu können über die SIMM-Cards sichere Anwendungen bereitgestellt werden, zusammen mit zusätzlichen Informationen, wie dem Ablegen von Identifizierungsinformationen auf der SIMM-Card auf eine Weise, die nicht gehackt werden kann.

[0063] Der Speicher kann beispielsweise Flash Speicher und/oder NVRAM-Speicher beinhalten, wie nachstehend erörtert. In einer Implementierung, ist ein Computerprogrammprodukt konkret in einem Informationsträger ausgeführt. Das Computerprogrammprodukt enthält Anweisungen, die, wenn sie ausgeführt werden, ein oder mehrere Verfahren ausführen, wie die oben beschriebenen. Der Informationsträger ist ein computer- oder maschinenlesbares Medium, wie der Speicher **764**, die Speichererweiterung **774** oder der Prozessorspeicher **752**, das beispielsweise über den Transceiver **768** oder die externe Schnittstelle **762** empfangen werden kann.

[0064] Das Gerät **750** kann drahtlos über die Verbindungsschnittstelle **766** kommunizieren, die digitale Signalverarbeitungsschaltkreise beinhalten kann, falls erforderlich. Die Verbindungsschnittstelle **766** kann Verbindungen mit verschiedenen Kommunikationstypen oder -protokollen aufbauen, darunter GSM-Sprachanrufe, SMS, EMS, oder MMS-Messaging, CDMA, TDMA, PDC, WCDMA, CDMA2000 oder GPRS unter anderen. Eine solche Kommunikation kann zum Beispiel über Funkfrequenzempfänger **768** erfolgen. Zusätzlich kann eine Kurzstreckenkommunikation stattfinden, wie unter Verwendung eines Bluetooth-, WLAN- oder anderen solchen Sende-Empfängers (nicht gezeigt). Außerdem kann GPS(Global Positioning System)-Empfängermodul **770** zusätzliche mit der Navigation und dem Ort verbundene drahtlose Daten für Gerät **750** bereitstellen, die ggf. von Anwendungen verwendet werden können, die auf Gerät **750** ausgeführt werden.

[0065] Das Gerät **750** kann mithilfe des Audio-Codec **760** auch akustisch kommunizieren, das gesprochene Informationen von einem Benutzer empfangen und diese in nutzbare digitale Informationen konvertieren kann. Audio-Codec **760** kann ebenfalls akustische Töne für einen Benutzer erzeugen, zum Beispiel durch einen Lautsprecher zum Beispiel in einem Handgerät von Gerät **750**. Solche Töne können Töne von Sprachtelefonanrufen beinhalten, können aufgezeichnete Töne (z. B. Sprachnachrichten, Musikdateien usw.) beinhalten und können auch Töne, die von Anwendungen, welche auf Gerät **750** operieren, beinhalten.

[0066] Das Computergerät **750** kann in einer Reihe verschiedener Formen implementiert werden, wie in der Figur gezeigt. Zum Beispiel kann es als Mobiltelefon implementiert werden **780**. Es kann außerdem als Teil eines Smartphones **782**, Personal Digital Assistant oder eines anderen ähnlichen mobilen Geräts implementiert werden.

[0067] Verschiedene Implementierungen der hier beschriebenen Systeme und Techniken können in digitalen elektronischen Schaltkreisen, integrierten Schaltkreisen, speziell konzipierten ASICs (anwendungsorientierten integrierten Schaltkreisen), Computerhardware, Firmware, Software und/oder Kombinationen davon realisiert werden. Diese verschiedenen Implementierungen können eine Implementierung in einem oder mehreren Computerprogrammen beinhalten, die auf einem programmierbaren System ausführbar und/oder interpretierbar sind, das mindestens einen programmierbaren Prozessor beinhaltet, der ein spezieller Prozessor oder ein Prozessor für allgemeine Zwecke sein kann, und der zum Empfangen von Daten und Anweisungen von und zum Übertragen von Daten und Anweisungen an ein Speichersystem, mindestens eine Eingabevorrichtung und mindestens eine Ausgabevorrichtung gekoppelt ist.

[0068] Diese Computerprogramme (auch bekannt als Programme, Software, Softwareanwendungen oder Code) beinhalten Maschinenanweisungen für einen programmierbaren Prozessor und können in einer höheren prozeduralen und/oder objektorientierter Programmiersprache und/oder in Assembler-/Maschinensprache implementiert werden. Wie hier verwendet, bezeichnen die Begriffe „maschinenlesbares Medium", „computerlesbares Medium" ein beliebiges Computerprogrammprodukt, eine beliebige Vorrichtung und/oder ein beliebiges Gerät (z. B. Magnetplatten, optische Platten, Speicher, programmierbare Logikbausteine (PLDs)), die verwendet werden, um einem programmierbaren Prozessor Maschinenanweisungen und/oder Daten bereitzustellen, einschließlich eines maschinenlesbaren Mediums, das Maschinenanweisungen als ein maschinenlesbares Signal empfängt. Der Begriff „maschinenlesbares Signal" bezeichnet ein beliebiges Signal, das verwendet wird, um einem programmierbaren Prozessor Maschinenanweisungen und/oder Daten bereitzustellen.

[0069] Zur Interaktion mit einem Benutzer können die hier beschriebenen Systeme und Techniken auf einem Computer mit einem Anzeigegerät (z. B. ein CRT-[Kathodenstrahlröhre] oder ein LCD-[Flüssigkristallanzeige] Monitor) implementiert werden, um Informationen für den Benutzer anzuzeigen, und eine Tastatur und ein Pointergerät (z. B. Maus oder Trackball), mit denen der Benutzer Eingaben in den Computer vornehmen kann. Andere Arten von Geräten können auch verwendet werden, um eine Interaktion mit einem Benutzer bereitzu-

stellen; zum Beispiel kann eine dem Benutzer bereitgestellte Rückmeldung irgendeine Form von Sinnesrückmeldung sein (z. B. visuelle Rückmeldung, auditive Rückmeldung oder Tastrückmeldung); und eine Eingabe vom Benutzer kann in einer beliebigen Form empfangen werden, einschließlich akustischer, Sprach- oder Tasteingaben.

[0070] Die hierin beschriebenen Systeme und Techniken können in einem Computersystem implementiert werden, das eine Back-End-Komponente beinhaltet (z. B. als Datenserver), oder das eine Middleware-Komponente beinhaltet (z. B. einen Anwendungsserver), oder das eine Front-End-Komponente beinhaltet (z. B. ein Client-Computer mit einer grafischen Benutzeroberfläche oder einem Webbrowser, über welche ein Benutzer mit einer Implementierung der hier beschriebenen Systeme und Techniken interagieren kann) oder Kombination derartiger Back-End-, Middleware- und Front-End-Komponenten. Die Komponenten des Systems können durch eine beliebige Form oder ein beliebiges Medium von digitaler Datenkommunikation (z. B. ein Kommunikationsnetzwerk) miteinander verbunden sein. Beispiele von Kommunikationsnetzwerken beinhalten ein lokales Netzwerk („LAN"), ein Fernnetz („WAN"), und das Internet.

[0071] Das Computersystem kann Clients und Server beinhalten. Ein Client und Server befinden sich im Allgemeinen ortsfern voneinander und interagieren typischerweise über ein Kommunikationsnetz. Die Beziehung zwischen Client und Server entsteht aufgrund von Computerprogrammen, die auf den jeweiligen Computer laufen und die eine Client-Server-Beziehung zueinander haben.

[0072] Einige Implementierungen sind beschrieben worden. Dennoch ist selbstverständlich, dass verschiedene Modifizierungen vorgenommen werden können.

[0073] Außerdem erfordern die in den Figuren dargestellten logischen Abläufe nicht die bestimmte dargestellte Reihenfolge oder sequenzielle Reihenfolge, um wünschenswerte Ergebnisse zu erzielen. Darüber hinaus können andere Schritte vorgesehen oder Schritte aus den beschriebenen Abläufen eliminiert werden und andere Komponenten können zu den beschriebenen Systemen hinzugefügt werden oder von diesen entfernt werden. Dementsprechend liegen andere Implementierungen im Geltungsbereich der folgenden Ansprüche.

Schutzansprüche

1. System, das Folgendes umfasst:
einen Speicher; und
einen Prozessor, operativ verbunden mit dem Speicher, und so konfiguriert, dass er Code ausführt, um Folgendes zu erreichen:
das Bereitstellen eines HTML-Dokuments einschließlich einer Liste mit Modellen, worin jedes Modell in der Liste ein Tochterobjekt des Textkörpers des HTML-Dokuments ist;
das Erstellen eines Shadow DOM-Unterbaums im Textkörper des HTML-Dokuments, worin der Shadow DOM-Unterbaum eine oder mehrere Einfügemarken umfasst; und
das Festlegen, welches Modell aus der Liste mit Modellen in einer Ansicht einer Webseite angezeigt wird, mithilfe von Übereinstimmungskriterien einer oder mehrerer Einfügemarken.

2. System nach Anspruch 1, worin der Prozessor folgendermaßen konfiguriert ist: Wiedergabe in einem Webbrowser, auf einer Webseite durch das Ersetzen des Contents eines Shadow Hosts der Webseite mit einem Shadow DOM-Unterbaum.

3. System nach Anspruch 2, worin der Content ein HTML-Content-Element ist.

4. System nach Anspruch 1, worin eine oder mehrere Einfügemarken jeweils einen definierten Standort im Shadow DOM-Unterbaum enthalten.

5. System nach Anspruch 4, worin der Prozessor für die Wiedergabe einer Webseite durch die Transposition von Tochterobjekten eines Shadow Hosts an den festgelegten Standort konfiguriert ist.

6. System nach Anspruch 1, worin der Prozessor für die Wiedergabe einer Webseite durch die Komposition multipler DOM-Unterbäumen in einen riesigen Baum.

7. System nach Anspruch 1, worin die Übereinstimmungskriterien ein Satz von Selektorfragmenten sind.

8. Nicht flüchtiges, computerlesbares Medium mit ausführbarem Code, der bewirkt, dass ein Computergerät Folgendes ausführt:
das Projizieren eines Webseitenelements in die Ansicht der Webseite mithilfe eines Prozessors des Computergeräts, und
das Transponieren des in die Ansicht der Webseite projizierten Elements und eines anderen Elements mithilfe einer Einfügemarke, die ein festgelegter Standort in einem Shadow-DOM-Unterbaum ist, durch den Prozessor und ohne die Hintergrundseite auf Browserebene darüber zu benachrichtigen,
worin die Hintergrundseite auf Browserebene eine Skriptdatei sein kann, die über keine eigene Benutzeroberfläche verfügt, und als ein Container für alle Ansichten einer Webanwendung einschließlich der Ansicht der Webseite agiert, und
worin jedes von der Webanwendung erstellte Fenster eine andere Ansicht der Hintergrundseite auf Browserebene darstellt.

9. Nicht flüchtiges, computerlesbares Medium nach Anspruch 8, worin die Hintergrundseite auf Browserebene von keiner aktuell angezeigten Ansicht beeinflusst wird.

10. Nicht flüchtiges, computerlesbares Medium nach Anspruch 8, worin die Hintergrundseite ein HTML-Dokument ist.

11. Nicht flüchtiges, computerlesbares Medium nach Anspruch 8, worin ein Element und das andere Element HTML-Content-Elemente sind.

12. Nicht flüchtiges, computerlesbares Medium nach Anspruch 8, worin die Übereinstimmungskriterien von der Einfügemarke für ihre Tochterknoten bereitgestellt werden, um festzulegen, ob der bestimmte Knoten für eine bestimmte Einfügemarke ausgeliefert werden könnte.

13. Nicht flüchtiges, computerlesbares Medium nach Anspruch 8, worin der Shadow DOM-Unterbaum von einem Element im DOM-Baum der Webseite gehostet wird.

Es folgen 10 Seiten Zeichnungen

Anhängende Zeichnungen

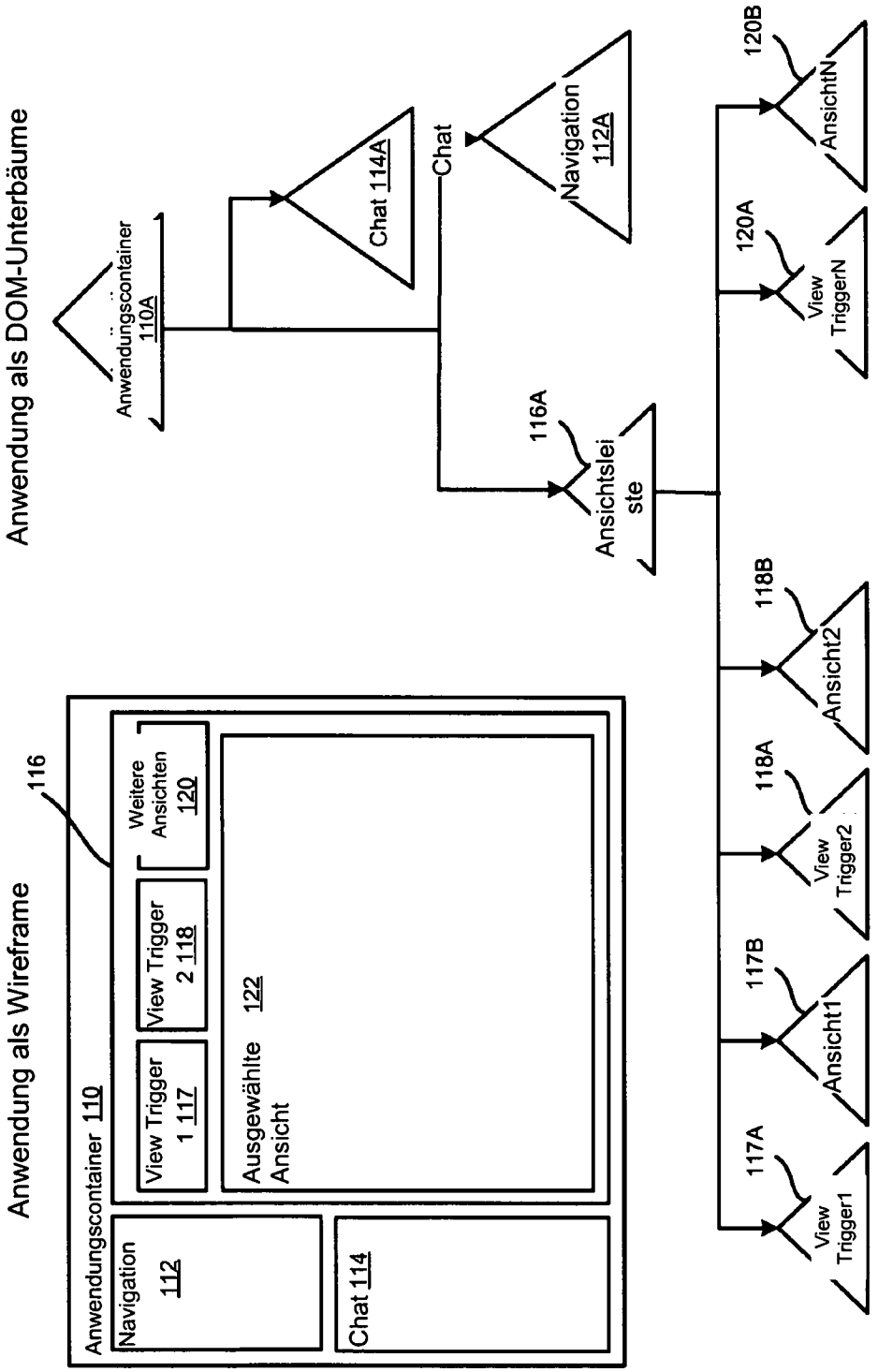


FIG. 1

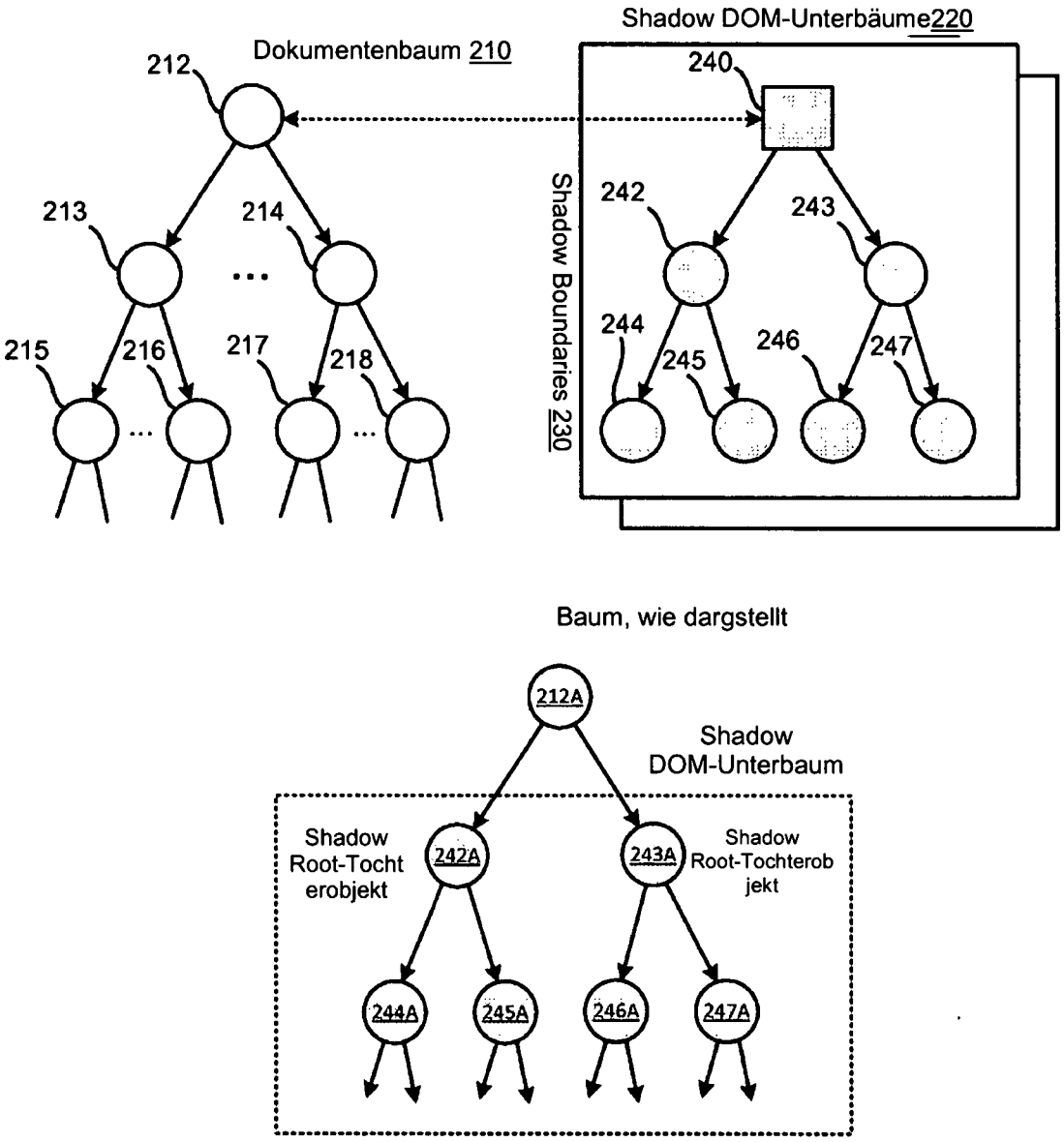

Shadow DOM-Unterbäume220
Dokumentenbaum 210
212
240
213
214
242
243
Shadow Boundaries 230
215
216
217
218
244
245
246
247
Baum, wie dargestellt
212A
Shadow
DOM-Unterbaum
Shadow
Root-Tocht
erobjekt
242A
243A
Shadow
Root-Tochterob
jekt
244A
245A
246A
247A

FIG. 2

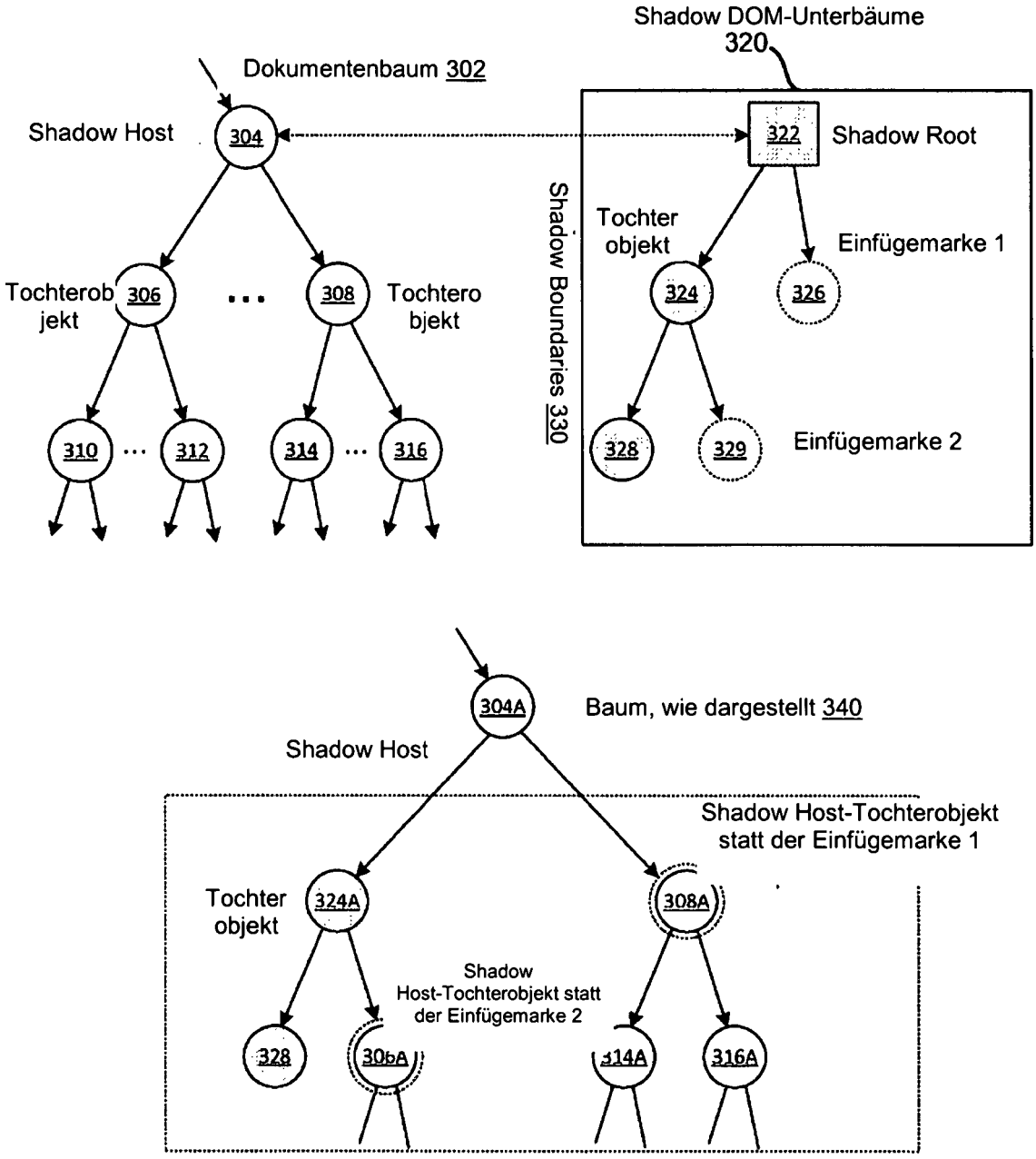

Shadow DOM-Unterbäume
320
Dokumentenbaum 302
Shadow Host
304
322
Shadow Root
Shadow Boundaries 330
Tochter objekt
Einfügemarke 1
Tochterob jekt
306
308
Tochtero bjekt
324
326
310
312
314
316
328
329
Einfügemarke 2
304A
Baum, wie dargestellt 340
Shadow Host
Shadow Host-Tochterobjekt statt der Einfügemarke 1
Tochter objekt
324A
308A
Shadow Host-Tochterobjekt statt der Einfügemarke 2
328
30bA
314A
316A

FIG. 3

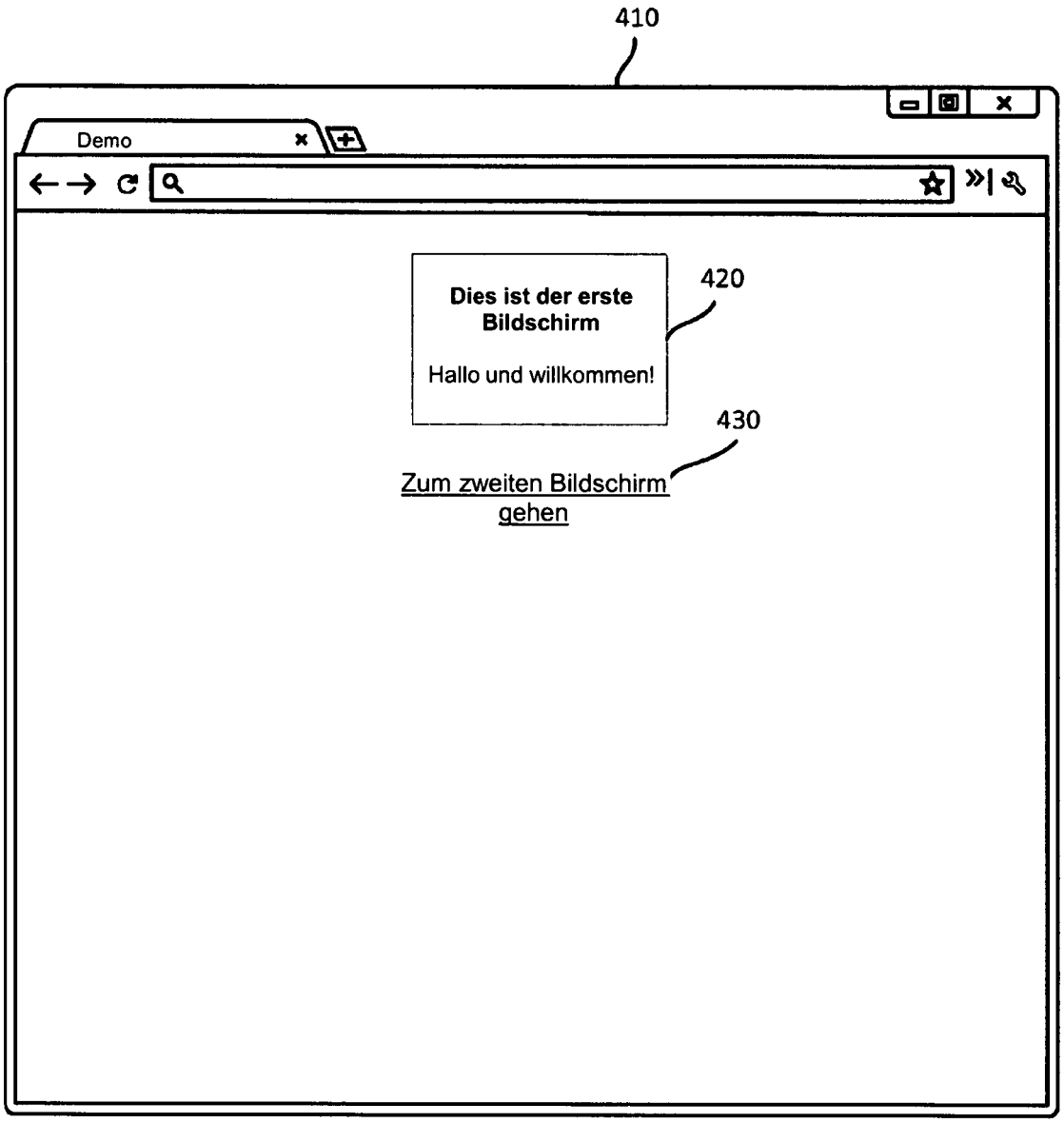

410
Demo
Dies ist der erste Bildschirm
Hallo und willkommen!
420
430
Zum zweiten Bildschirm gehen

FIG. 4A

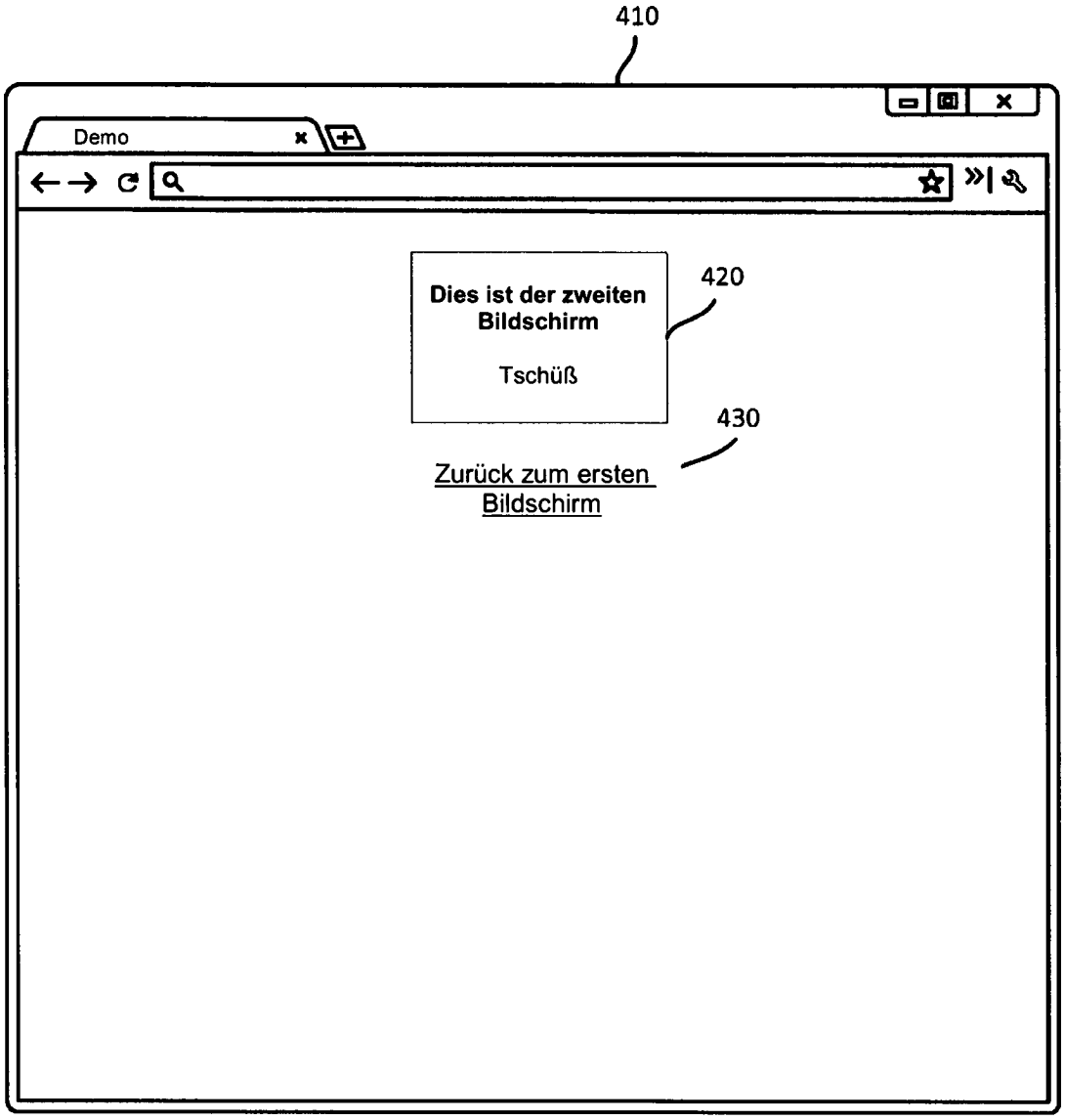

410
Demo
Dies ist der zweiten Bildschirm
Tschüß
420
430
Zurück zum ersten Bildschirm

FIG. 4B

```
<!DOCTYPE html>
<html>
<head>
  <title> Views Demo</title>
  <link rel="stylesheet" type="text/css" href="views.css">
  <script src="views.js"></script>
</head>
<body>
<h1> Views Demo</h1>
<!-- The "window" attribute picks the window (dialog, main window, etc?) in
which to show the view -->
<view id="main-view" window="main">
  <!-- scenes can be nested inside the view ... -->
  <scene id="first-screen">
    <h1>This is first screen</h1>
    <p>Hello and welcome!</p>
    <a href="#second-screen">Go to second screen</a>
  </scene>
</view>
<scene id="second-screen" view="main-view">
  <!-- or they can just reference a view -->
  <h1>This is second screen</h1>
  <p>Goodbye</p>
  <a href="#first-screen">Go back to first screen</a>
</scene>
</body>
</html>
```

FIG. 5A

```
(function() {
  window.View = function() {
    var root = new WebKitShadowRoot(this);
    this._content = root.appendChild(document.createElement('content'));
    this._content.select = 'scene:first-of-type';  }
  window.View.prototype = Object.create(WebKitShadowRoot.prototype, {
    setScene: {  value: function(scene)
      { if (scene.id) this._content.select = 'scene#' + scene.id;  } }  });
  function asArray(value) {return [].slice.call(value);}
  function morph(element, func) {
    element.__proto__ = func.prototype;
    func.call(element);
    return element; }
  window.Scene = function() {
    var view = document.getElementById(this.getAttribute('view'));
    if (!view) {
      if (this.parentElement.tagName == 'VIEW')
        view = this.parentElement;
    } else {        view.appendChild(this);      }
    this.view = view;  }
  window.Scene.prototype = Object.create(HTMLUnknownElement.prototype);
  function Controller() {
    document.addEventListener('DOMContentLoaded', this._initialize.bind(this));
    document.addEventListener('click', this._onClick.bind(this));  }
  Controller.prototype = { _initialize: function()
    {Array(document.querySelectorAll('view')).forEach(function(view) {
        morph(view, window.View); });
  asArray(document.querySelectorAll('scene')).forEach(function(scene) {morph(scene, window.Scene); });      },
    _onClick: function(evt)
    {if (evt.target.tagName != 'A')
        return;
      evt.preventDefault();
      this.route(evt.target.getAttribute('href'));      },
    route: function(url)
    {if (url.indexOf('#') != 0)
        return;
      var scene = document.getElementById(url.substring(1));
      if (!scene)
        return;
      var view = scene.view;
      if (!view)
        return;
      view.setScene(scene);      }  }
  window.controller = new Controller();})();
```

FIG. 5B

```
body {
    position:
absolute;
    width: 0px;
    height: 100%;
    overflow:
hidden;
    margin: 0;
    left: -1000px;
}

view {
    position:
absolute;
    top: 0;
    left: 1000px;
    width: 800px;
    height: 600px;
}
```

FIG. 5C

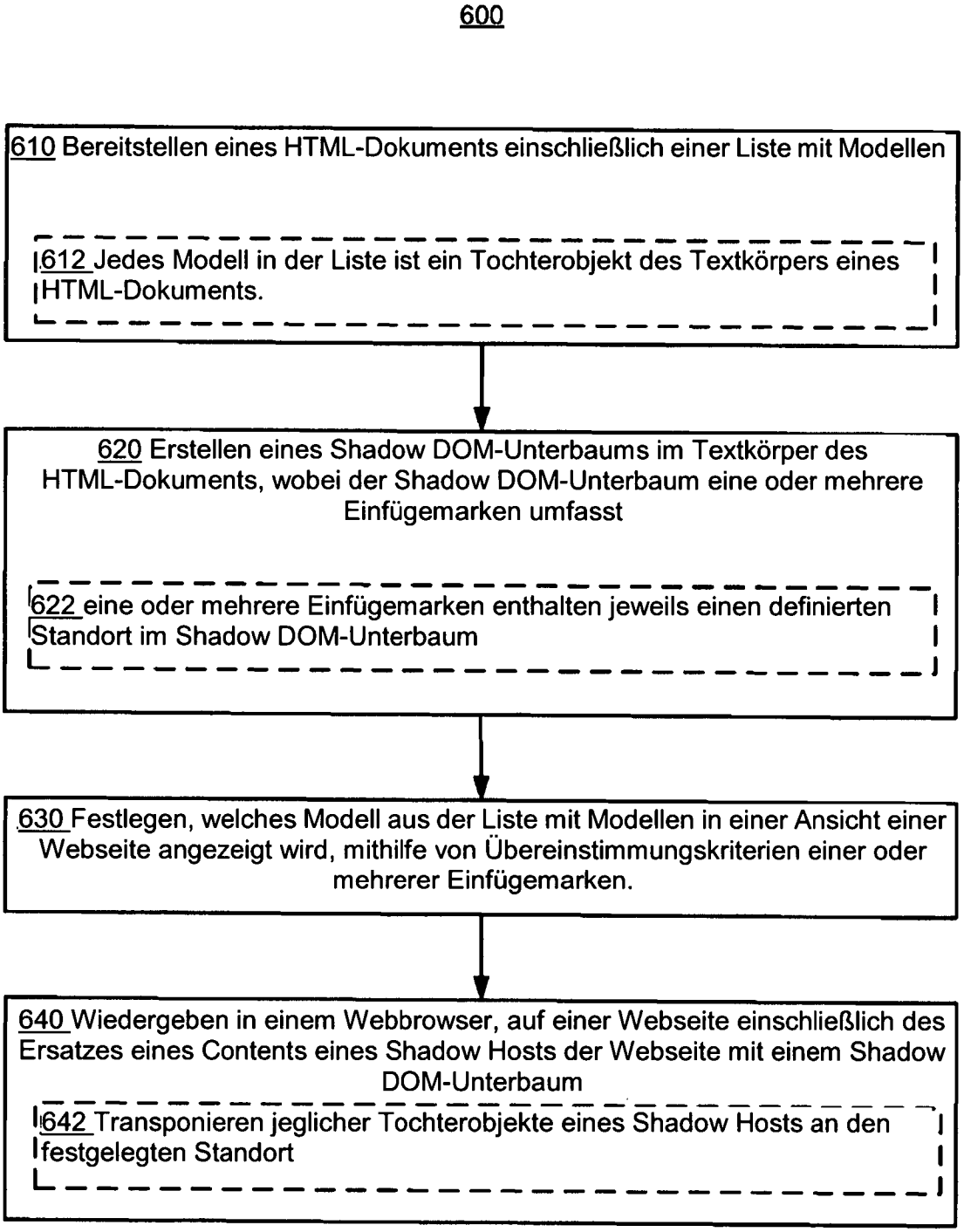

600
610 Bereitstellen eines HTML-Dokuments einschließlich einer Liste mit Modellen
612 Jedes Modell in der Liste ist ein Tochterobjekt des Textkörpers eines HTML-Dokuments.
620 Erstellen eines Shadow DOM-Unterbaums im Textkörper des HTML-Dokuments, wobei der Shadow DOM-Unterbaum eine oder mehrere Einfügemarken umfasst
622 eine oder mehrere Einfügemarken enthalten jeweils einen definierten Standort im Shadow DOM-Unterbaum
630 Festlegen, welches Modell aus der Liste mit Modellen in einer Ansicht einer Webseite angezeigt wird, mithilfe von Übereinstimmungskriterien einer oder mehrerer Einfügemarken.
640 Wiedergeben in einem Webbrowser, auf einer Webseite einschließlich des Ersatzes eines Contents eines Shadow Hosts der Webseite mit einem Shadow DOM-Unterbaum
642 Transponieren jeglicher Tochterobjekte eines Shadow Hosts an den festgelegten Standort

FIG. 6

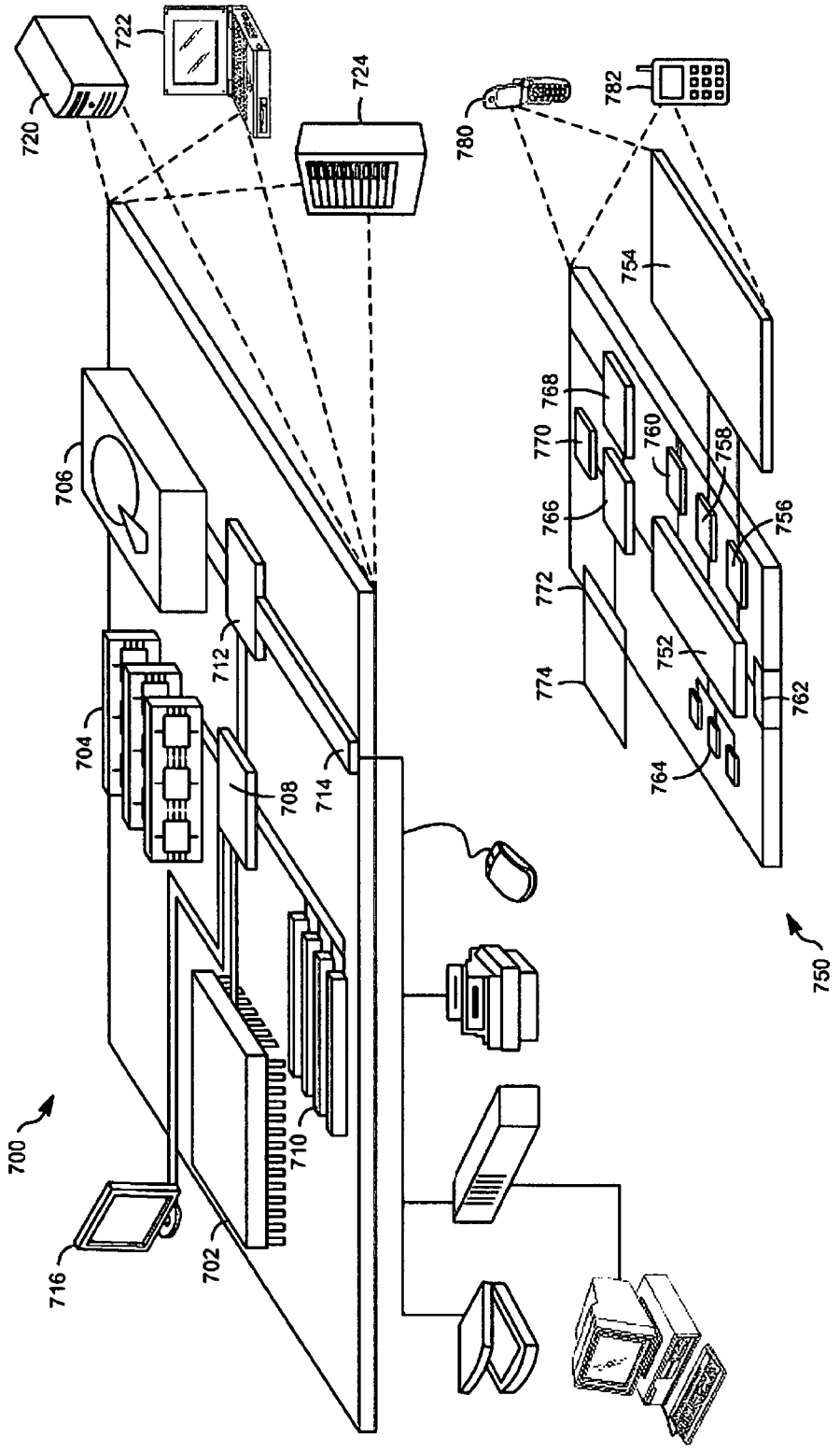

720
722
724
780
782
754
768
770
760
758
766
756
772
752
774
762
764
750
706
712
704
708
714
700
710
716
702

FIG. 7