JavaScript

JavaScript adesea abreviat JS, este un limbaj de programare care reprezintă una dintre tehnologiile de bază ale World Wide Web, alături de HTML și CSS. Peste 97% dintre site-uri web folosesc Javascript pe partea de client pentru comportamentul paginii web, încorporând adesea biblioteci terțe. Toate **browser**-ele web majore au un motor JavaScript dedicat pentru a executa codul pe dispozitivul utilizatorului.

JavaScript este un limbaj de nivel înalt, adesea **just-in-time compiled**, cu mai multe paradigme. Este conformă cu specificația ECMAScript și are tastare dinamică, orientare pe obiect bazată pe prototip și funcții de primă clasă. Acceptă stiluri de programare bazate pe evenimente, funcționale și imperative. Are interfețe de programare a aplicațiilor (API) pentru lucrul cu text, date, expresii regulate, structuri de date standard și **Document Object Model** (DOM).

Standardul ECMAScript nu include nicio intrare/ieșire (**I/O**), cum ar fi facilități de rețea, stocare sau grafică. În practică, **browser**-ul web sau alt sistem de rulare furnizează API-uri JavaScript pentru **I/O**. Motoarele JavaScript au fost utilizate inițial numai în **browser**-ele web, dar acum sunt componente de bază ale unor servere și ale unei varietăți de aplicații. Cel mai popular sistem de rulare pentru această utilizare este Node.js.

Deși Java și Javascript sunt similare ca nume, sintaxă și bibliotecile standard respective, cele două limbi sunt distincte și diferă foarte mult în design.

Utilizarea **site-ului** pe partea client

JavaScript este limbajul de scriptare dominant pe partea client al **Web-ului**, cu 97% dintre **site-uri** **web** care îl folosesc în acest scop. Scripturile sunt încorporate sau incluse din documentele **HTML** și interacționează cu **DOM**. Toate **browser-ele web** majore au un motor JavaScript încorporat care execută codul pe dispozitivul utilizatorului.

Exemple de comportament scriptat:

* Încărcarea unui nou conținut de pagină **web** fără a reîncărca pagina, prin **Ajax** sau un **WebSocket**. De exemplu, utilizatorii rețelelor sociale pot trimite și primi mesaje fără a părăsi pagina curentă.
* Animații ale paginilor **web**, cum ar fi **fading objects in and out**, redimensionarea și mutarea acestora.
* Jucând jocuri de **browser**.
* Controlul redării conținutului media în flux.
* Generarea de reclame **pop-up**.
* Validarea valorilor de intrare ale unui formular **web** înainte ca datele să fie trimise către un server **web**.
* Înregistrarea datelor despre comportamentul utilizatorului, apoi trimiterea acestora către un server. Proprietarul **site-ului web** poate folosi aceste date pentru analiză, urmărire a reclamelor și personalizare.
* Redirecționarea unui utilizator către o altă pagină.

Biblioteci și **framework-uri**:

Peste 80% dintre **site-uri web** utilizează o bibliotecă JavaScript terță parte sau un cadru **web** pentru scripturile lor la nivel client.

**jQuery** este de departe cea mai populară bibliotecă, folosită de peste 75% dintre **site-uri web**. **Facebook** a creat biblioteca **React** pentru **site-ul** său **web** și ulterior a lansat-o **open source**; alte **site-uri**, inclusiv **Twitter**, îl folosesc acum. De asemenea, **framework-ul** **Angular** creat de **Google** pentru **site-urile** sale **web**, inclusiv **YouTube** și **Gmail**, este acum un proiect **open source** folosit de alții.

În schimb, termenul **„Vanilla JS”** a fost inventat pentru **site-urile web** care nu folosesc biblioteci sau **framework-uri**, ci se bazează în totalitate pe funcționalitatea standard JavaScript.

React (JavaScript library)

React (cunoscut și ca React.js sau ReactJS) este o bibliotecă JavaScript gratuită și open-source pentru construirea de interfețe cu utilizatorul bazate pe componente UI. Este întreținut de Meta (fostul Facebook) și de o comunitate de dezvoltatori individuali și companii. React poate fi folosit ca bază în dezvoltarea de aplicații cu o singură pagină sau mobile. Cu toate acestea, React se preocupă doar de gestionarea stării și de redarea acelei stări către DOM, astfel încât crearea aplicațiilor React necesită de obicei utilizarea de biblioteci suplimentare pentru routing, precum și anumite funcționalități pe partea de client.

Caracteristici notabile

Declarativ:

React aderă la paradigma de programare declarativă. Dezvoltatorii proiectează vizualizări pentru fiecare stare a unei aplicații, iar React actualizează și redă componente atunci când datele se modifică.

Componente:

Codul React este format din entități numite componente. Componentele pot fi redate la un anumit element din DOM folosind biblioteca React DOM. Când redați o componentă, se pot trece valori cunoscute sub numele de „recuzită”:

ReactDOM.render(<Greeter greeting="Hello World!" />, document.getElementById('myReactApp'));

Cele două moduri principale de declarare a componentelor în React sunt prin componentele de funcție și componentele bazate pe clasă.

Componentele funcționale:

Componentele funcției sunt declarate cu o funcție care returnează apoi niște JSX.

**const** Greeter = (props) => <div>Hello, {props.name}!</div>;

Componente bazate pe clasă:

Componentele bazate pe clasă sunt declarate folosind clase ES6.

**class** ParentComponent **extends** React.Component {

state = { color: 'green' };

render() {

**return** (

<ChildComponent color={**this**.state.color} />

);

}

}

DOM virtual

O altă caracteristică notabilă este utilizarea unui model de obiect de document virtual sau DOM virtual. React creează un cache pentru structura de date în memorie, calculează diferențele rezultate și apoi actualizează eficient DOM afișat în browser. Acest proces se numește reconciliere. Acest lucru permite programatorului să scrie cod ca și cum întreaga pagină este redată la fiecare modificare, în timp ce bibliotecile React redau doar subcomponentele care se schimbă efectiv. Această randare selectivă oferă o creștere majoră a performanței. Economisește efortul de a recalcula stilul, aspectul paginii și randarea pentru întreaga pagină.

Metode ale ciclului de viață

Metodele ciclului de viață folosesc o formă de agățare care permite executarea codului la punctele stabilite pe durata de viață a unei componente.

* shouldComponentUpdate permite dezvoltatorului să prevină redarea inutilă a unei componente, returnând false dacă nu este necesară o randare.
* componentDidMount este apelat odată ce componenta s-a „montat” (componenta a fost creată în interfața cu utilizatorul, adesea prin asocierea cu un nod DOM). Acesta este folosit în mod obișnuit pentru a declanșa încărcarea datelor de la o sursă la distanță printr-un API.
* componentWillUnmount este apelat imediat înainte ca componenta să fie dărâmată sau „demontată”. Acesta este folosit în mod obișnuit pentru a șterge dependențele care necesită resurse de componentă care nu vor fi eliminate pur și simplu odată cu demontarea componentei (de exemplu, eliminarea oricăror instanțe setInterval()care sunt legate de componentă sau un „eventListener” setat pe „ document" din cauza prezenței componentei)
* render este cea mai importantă metodă a ciclului de viață și singura necesară în orice componentă. Este de obicei apelat de fiecare dată când starea componentei este actualizată, ceea ce ar trebui să se reflecte în interfața cu utilizatorul.

Jsx

JSX, sau JavaScript XML, este o extensie a sintaxei limbajului JavaScript. Similar ca aspect cu HTML, JSX oferă o modalitate de a structura randarea componentelor folosind o sintaxă familiară multor dezvoltatori. Componentele React sunt scrise de obicei folosind JSX, deși nu trebuie să fie (componentele pot fi scrise și în JavaScript pur). JSX este similar cu o altă sintaxă de extensie creată de Facebook pentru PHP numită XHP.

Un exemplu de cod JSX:

**class** **App** **extends** React.Component {

render() {

**return** (

<div>

<p>Header</p>

<p>Content</p>

<p>Footer</p>

</div>

);

}

}

Arhitectură dincolo de HTML

Arhitectura de bază a React se aplică dincolo de redarea HTML în browser. De exemplu, Facebook are diagrame dinamice care se redau la etichete <canvas>, iar Netflix și PayPal folosesc încărcarea universală pentru a reda HTML identic atât pe server, cât și pe client.

React hooks

Hook-urile sunt funcții care permit dezvoltatorilor să se „prindă” în funcțiile React și ale ciclului de viață din componentele funcției. Hook-urile nu funcționează în cadrul claselor - vă permit să utilizați React fără clase.

React oferă câteva hook-uri încorporate, cum ar fi useState, useContext, useReducer , useMemo și useEffect. useState, useReducer și useEffect, care sunt cele mai utilizate, sunt pentru controlul stării și, respectiv, a efectelor secundare.

Regulile hook-urilor

Există reguli pentru hook-uri care descriu modelul de cod caracteristic pe care se bazează acestea. Este modalitatea modernă de a gestiona starea cu React.

1. Hook-urile ar trebui să fie apelate numai la nivelul superior (nu în interiorul buclelor sau declarațiilor if).
2. Hook-urile ar trebui să fie apelate numai din componentele funcției React, nu din funcții normale sau componente ale clasei.

Deși aceste reguli nu pot fi aplicate în timpul execuției, instrumentele de analiză a codului, cum ar fi linters, pot fi configurate pentru a detecta multe greșeli în timpul dezvoltării. Regulile se aplică atât utilizării cârligelor, cât și implementării de hook-uri personalizate, care pot face call catre ale hook-uri.

React Native

**React Native** este un **open-source UI software framework** creat de **Meta Platforms, Inc**. Este folosit pentru a dezvolta aplicații pentru **Android, Android TV, iOS, macOS, tvOS, Web, Windows** și **UWP**, permițând dezvoltatorilor să folosească **framework-ul** **React** împreună cu capabilitățile platformei native. De asemenea, este folosit pentru a dezvolta aplicații de realitate virtuală la **Oculus**.

Istorie

În 2012, Mark Zuckerberg a comentat: „Cea mai mare greșeală pe care am făcut-o ca companie a fost să pariezi prea mult pe **HTML**, spre deosebire de nativ”. Utilizarea **HTML5** pentru versiunea mobilă a **Facebook** a dus la o aplicație instabilă care a preluat datele lent. El a promis că **Facebook** va oferi în curând o experiență mobilă mai bună.

În cadrul **Facebook**, Jordan Walke a găsit o modalitate de a genera elemente de **UI** pentru **iOS** dintr-un fir **JavaScript** de fundal, care a devenit baza pentru cadrul **web React**. Ei au decis să organizeze un **Hackathon** intern pentru a perfecționa acest prototip pentru a putea construi aplicații native cu această tehnologie.

În 2015, după luni de dezvoltare, **Facebook** a lansat prima versiune pentru **React JavaScript Configuration**. În timpul unei discuții tehnice, Christopher Chedeau a explicat că **Facebook** folosea deja **React Native** în producție pentru aplicația de grup și pentru aplicația **Ads Manager**.

Implementarea

Principiile de lucru ale **React Native** sunt practic identice cu **React**, cu excepția faptului că **React Native** nu manipulează **DOM-ul** prin **Virtual DOM**. Se rulează într-un proces de fundal (care interpretează **JavaScript-ul** scris de dezvoltatori) direct pe dispozitivul final și comunică cu platforma nativă prin date serializate printr-o punte asincronă și **batched**.

Componentele **React** înglobează codul nativ existent și interacționează cu **API-urile** native prin paradigma declarativă a interfeței de utilizare **React** și **JavaScript**.

În timp ce stilul **React Native** are o sintaxă similară cu **CSS**, nu folosește **HTML** sau **CSS**. În schimb, mesajele din firul **JavaScript** sunt folosite pentru a manipula vizualizările native. **React Native** le permite dezvoltatorilor să scrie cod nativ în limbi precum **Java** sau **Kotlin** pentru **Android**, **Objective-C** sau **Swift** pentru **iOS** și **C++/WinRT** sau **C#** pentru **Windows 10**, ceea ce îl face și mai flexibil.

**Microsoft** creează și menține **React Native** pentru **Windows** și **React Native** pentru **macOS**.