Externe scripts

De scripts in een volwassen library zoals jQuery worden ook 'pre-made' scripts genoemd omdat de code al is geschreven. Het doel van een pre-made script is om het ontwikkelen in JavaScript gemakkelijker te maken, maar niet om de functionaliteit uit te breiden.

Een extern script is een script dat in een apart bestand wordt geplaatst en dat wordt uitgevoerd door het te koppelen aan de HTML-pagina (via het kenmerk src) in plaats van het in de HTML in te sluiten. Veel vooraf gemaakte scripts in een library zijn externe scripts, hoewel sommige in HTML kunnen worden ingesloten.

Het uitvoeren van externe scripts heeft verschillende voordelen:

- Het houdt de HTML-code schoon. De HTML is gemakkelijker te lezen en bij te werken als u niet door extra script-, opmaak- of programmeercode hoeft te waden.
- Het zorgt ervoor dat de scripts van een ontwikkelaar herbruikbaar worden en zelfs opnieuw verspreid als de ontwikkelaar ze nuttig vindt voor de ontwikkelingsgemeenschap. Alle JavaScript-ontwikkelaars kunnen hun scripts naar bibliotheken sturen om ze te delen.
- U kunt de code één keer maken en op veel pagina's gebruiken. Ontwikkelaars doen dit met CSS en hetzelfde kan met externe scripts. Deze aanpak is handig wanneer u veel webpagina's gebruikt die vergelijkbare functionaliteit nodig hebben, zoals invoervalidatie, menu's of andere items die van pagina tot pagina worden herhaald.

Library plug-ins

De meeste externe scripts die een code library vormen, worden plug-ins genoemd. Een plug-in is meestal een gespecialiseerd stuk JavaScript code dat is ontworpen om een enkele functie of een reeks vergelijkbare functies uit te voeren. Plug-ins worden ook vaak externe scripts, vooraf gemaakte scripts of add-on-scripts genoemd. Enkele veelgebruikte plug-ins zijn formuliervalidatie, menu-aanmaak, afbeeldingseffecten en nog veel meer.

Plug-ins versus vooraf gemaakte scripts

Elke plug-in in een library is een vooraf gemaakt script, maar niet elk vooraf gemaakt script is een plug-in. Plug-ins hebben alleen betrekking op een enkele library, terwijl sommige scripts in een library alleen kunnen worden gebruikt. Plug-ins bieden functies die specifieker zijn dan andere scripts, die doorgaans meer algemene basisfuncties bieden. Een plug-in kan niet worden gebruikt zonder de library waar deze vandaan komt, terwijl andere typen scripts dat doorgaans wel kunnen.

In wezen zijn library scripts voor algemeen gebruik die zijn ontworpen om het algehele gebruik van JavaScript te vergemakkelijken, terwijl plug-ins gespecialiseerde scripts zijn die zijn ontworpen om afzonderlijke functies of functiesets eenvoudiger te implementeren.

Externe scripts laden

U laadt externe scripts en bibliotheken in HTML-bestanden met behulp van de <script> -tag, meestal in het <head> </head> -gedeelte van het document. U kunt meer dan één extern script per HTML-pagina laden, maar er moet naar elk script worden verwezen met een afzonderlijk <script> -tagitem.

U kunt naar een extern JavaScript-bestand verwijzen met de volgende syntaxis, waarbij script.js een standaardbestandsnaam is voor het tekstbestand met uw JavaScript-code (of een bibliotheekplug-in) voor de functionaliteit die u aan de pagina wilt toevoegen:

<script src = "script.js" type = "text / javascript" charset = "utf-8"> </script>

Met deze tag instrueert de ontwikkelaar de HTML-interpreter om een extern script (src) te laden, met het type text/javascript, en om de UTF-8-tekenset te gebruiken.

De .js-bestanden hoeven niet in dezelfde map te staan als het HTML-bestand, maar als u ze niet samen opslaat, moeten ze worden benaderd ten opzichte van waar de scripts zich bevinden. U kunt zelfs .js-scripts van andere webdomeinen gebruiken als u dat wilt. Doe dit echter niet van onbekende bronnen, het script kan door kwaadwillenden gemanipuleerd worden.

Als u zelf geen JavaScript-bibliotheken zoals jQuery wilt hosten, kunt u deze vanaf een CDN in uw webtoepassing toevoegen. CDN's of inhoudsleveringsnetwerken worden gedefinieerd als een geografisch gedistribueerd netwerk van proxyservers die de service gelijkmatig globaal verspreiden.

Een extern script aanroepen met onload

Een andere manier wordt soms op internet gebruikt om een script aan te roepen vanuit een extern JavaScript-bestand. De eventhandler onload kan worden toegevoegd tussen de <script> </script> -tags in het HTML-bestand om een functie op te roepen die in het externe script is geplaatst. Dit zorgt ervoor dat de functie wordt aangeroepen wanneer de webpagina is geladen.

U kunt bijvoorbeeld de functie myNewName () aanroepen vanuit een extern JavaScript-bestand met de naam external.js wanneer de HTML-pagina is geladen met behulp van de volgende instructie:

```
<script type="text/javascript" src="external.js" onload="myNewName()">
```

Server-side JavaScript

Door oplossingen als Node.js wordt er steeds meer JavaScript op de server gebruikt dan voorheen, ten koste van talen als PHP en ASP. Het heeft veel voordelen:

1. Node.js biedt een gemakkelijke schaalbaarheid.
   Een van de belangrijkste voordelen van Node.js is dat ontwikkelaars het gemakkelijk vinden om de applicaties zowel in horizontale als in verticale richting te schalen. Door de toevoeging van extra nodes aan het bestaande systeem kunnen de applicaties horizontaal geschaald worden.
2. Gemakkelijk te leren.
   Omdat JavaScript een van de meest populaire programmeertalen is, hebben de meeste frontend ontwikkelaars er goed begrip van. Het wordt veel gemakkelijker voor hen om de Node.js aan de backend te gebruiken. Het is gemakkelijker om Node.js te leren en kost minder tijd om ermee te werken.

3. Node.js wordt gebruikt als een enkele programmeertaal.
   Node.js biedt de ontwikkelaars de luxe om de server-side applicaties in JavaScript te schrijven. Hierdoor kunnen de Node.js-ontwikkelaars zowel de frontend als de backend webapplicatie in JavaScript schrijven met behulp van een runtime-omgeving.
   En ze hoeven geen andere programmeertaal aan de serverzijde te gebruiken. Het maakt ook de implementatie van de webapplicaties eenvoudiger omdat bijna alle webbrowsers JavaScript ondersteunen.
4. Het voordeel van Fullstack JS.
   Node.js wordt beschouwd als een full-stack JavaScript voor zowel de client als de server-side applicaties. Het voordeel is daarom dat u geen aparte ontwikkelaars hoeft in te huren voor zowel de backend als de frontend ontwikkeling. Het bespaart zowel uw kostbare geld als tijd.
5. Bekend om zijn hoge prestaties.
   Node.js interpreteert de JavaScript-code via de V8 JavaScript-engine van Google. Deze engine past de JavaScript-code rechtstreeks in de machinecode in. Dit maakt het eenvoudiger en sneller om de code effectief te implementeren.
6. De ondersteuning van een grote en actieve gemeenschap.
   Node.js is gezegend met een grote en actieve gemeenschap van ontwikkelaars die continu blijven bijdragen aan de verdere ontwikkeling en verbetering.
7. Het voordeel van caching.
   De open-source runtime-omgeving van de Node.js biedt ook de mogelijkheid om afzonderlijke modules in de cache op te slaan. Wanneer er een verzoek voor de eerste module is, wordt deze in het toepassingsgeheugen opgeslagen. De ontwikkelaars hoeven de codes niet opnieuw uit te voeren, omdat caching ervoor zorgt dat applicaties de webpagina's sneller kunnen laden en sneller reageren op de gebruiker.
8. Biedt de vrijheid om apps te ontwikkelen.
   Een ander voordeel dat Node.js de ontwikkelaars biedt, is de vrijheid om de apps en software te ontwikkelen.
9. Ondersteuning krijgen voor veelgebruikte tools.
   Met Node.js kunnen de ontwikkelaars uitgebreide ondersteuning krijgen voor de verschillende veelgebruikte tools. Laten we een voorbeeld nemen. Stel dat u de broncode van de Node.js-toepassing wilt testen; u kunt dit doen door de Jasmine en andere dergelijke unit-testing tools te gebruiken.
   Evenzo, als u de projectafhankelijkheden wilt identificeren en installeren, kunt u gebruikmaken van npm, een krachtige pakketbeheerder. U kunt grunt gebruiken voor het uitvoeren van taken van het project.
10. Behandelt de verzoeken tegelijkertijd.
    Omdat de Node.js de mogelijkheid biedt om I/O-systemen niet te blokkeren, helpt het relatief om meerdere verzoeken tegelijkertijd te verwerken. Het systeem kan de gelijktijdige verwerking van verzoeken efficiënter afhandelen dan andere, waaronder Ruby of Python. De binnenkomende verzoeken worden op een rij gezet en worden snel en systematisch uitgevoerd.
    Node.js is zeer uitbreidbaar.
    De Node.js staat bekend als zeer uitbreidbaar, wat betekent dat u Node.js kunt aanpassen en verder uitbreiden volgens hun vereisten. U kunt ook gebruikmaken van JSON om ruimte te bieden voor de uitwisseling van gegevens tussen de webserver en de client. Het wordt ook mogelijk gemaakt met ingebouwde API's voor het ontwikkelen van HTTP-, TCP- en DNS-servers enz.

Nadelen

Er kleven ook nadelen aan het werken met Server Side JavaScript:

- Bestaande backend ontwikkelaars die gewend zijn PHP en ASP te schrijven zullen moeten leren werken met JavaScript.
- Het asynchrone gedrag van JavaScript-applicaties zorgt soms voor verrassingen en is soms lastig te doorgronden.

JavaScript heeft geen specifieke functies en routines om bijvoorbeeld met een database te communiceren terwijl andere server-sided talen als PHP en ASP dat wel hebben.