UD01

Arquitectures i Eines de Programació

Índex

Introducció

Què és JavaScript?

JavaScript va ser introduït en 1995 com una manera d'agregar programes a pàgines web en el navegador Netscape Navigator. Des de llavors, el llenguatge ha estat adoptat per tots els altres navegadors web gràfics principals. Ha fet possibles aplicacions web modernes, és a dir, aplicacions amb les quals pots interactuar directament sense haver de recarregar la pàgina per a cada acció. JavaScript també s'utilitza en llocs web més tradicionals per a proporcionar diferents formes d'interactivitat i enginy.

És important tenir en compte que JavaScript gairebé no té res a veure amb el llenguatge de programació anomenat Java. El nom similar va ser inspirat per consideracions de màrqueting en lloc d'un bon judici. Quan s'estava introduint JavaScript, el llenguatge Java s'estava comercialitzant molt i guanyava popularitat. Algú va pensar que era una bona idea intentar aprofitar aquest èxit. Ara estem atrapats amb el nom.

Després de la seva adopció fora de Netscape, es va escriure un document estàndard per a descriure la forma en què hauria de funcionar el llenguatge JavaScript perquè les diverses peces de programari que afirmaven suportar JavaScript poguessin assegurar-se que realment proporcionaven el mateix llenguatge. Això es diu l'estàndard ECMAScript, segons l'organització Ecma International que va dur a terme l'estandardització. En la pràctica, els termes ECMAScript i JavaScript es poden usar indistintament, són dos noms per al mateix llenguatge.

Hi ha els qui diran coses terribles sobre JavaScript. Moltes d'aquestes coses són certes. Quan vaig començar a utilitzar-ho, ràpidament vaig arribar a detestar-ho. Acceptava gairebé qualsevol cosa que escrivia, però ho interpretava d'una manera completament diferent del que jo volia dir. Això tenia molt a veure amb el fet que no tenia ni idea del que estava fent, per descomptat, però hi ha un problema real aquí: JavaScript és ridículament liberal en el que permet. La idea darrere d'aquest disseny era que faria la programació en JavaScript més fàcil per a principiants. En realitat, això fa que trobar problemes en els teus programes sigui més difícil perquè el sistema no te'ls assenyalarà.

Aquesta flexibilitat també té els seus avantatges. Deixa espai per a tècniques impossibles en llenguatges més rígids i permet un estil de programació agradable i informal. Després d'aprendre el llenguatge adequadament i treballar amb ell durant un temps, ha arribat a realment agradar-me JavaScript.

Hi ha hagut diverses versions de JavaScript. La versió ECMAScript 3 va ser la versió àmpliament suportada durant l'ascens al domini de JavaScript, aproximadament entre 2000 i 2010. Durant aquest temps, s'estava treballant en una versió ambiciosa 4, la qual planejava una sèrie de millores i extensions radicals al llenguatge. Canviar un llenguatge viu i àmpliament utilitzat d'aquesta manera va resultar ser políticament difícil, i el treball en la versió 4 va ser abandonat en 2008. Una versió 5, molt menys ambiciosa, que només realitzava algunes millores no controvertides, va sortir en 2009. En 2015, va sortir la versió 6, una actualització important que incloïa algunes de les idees previstes per a la versió 4. Des de llavors, hem tingut noves actualitzacions petites cada any.

El fet que JavaScript estigui evolucionant significa que els navegadors han de mantenir-se constantment al dia. Si estàs usant un navegador més antic, és possible que no admeti totes les funcions. Els dissenyadors del llenguatge s'asseguren de no fer canvis que puguin trencar programes existents, per la qual cosa els nous navegadors encara poden executar programes antics. En aquest curs, utilitzarem la versió 2023 de JavaScript.

Els navegadors web no són les úniques plataformes en les quals s'utilitza JavaScript. Algunes bases de dades, com MongoDB i CouchDB, utilitzen JavaScript com el seu llenguatge de seqüències de comandos i consulta. Diverses plataformes per a programació d'escriptori i servidors, especialment el projecte Node.js, proporcionen un entorn per a programar en JavaScript fora del navegador.

1. Mecanismes d'execució de codi en un navegador web

1.1. Introducció

És important entendre com funciona JavaScript per dins, saber què passa en els seus budells ens permet crear un codi optimitzat i ràpid d'execució. I no només això, sinó que també ens ajuda a entendre la naturalesa dels errors que veiem per consola.

JavaScript neix el 1995 creat per Brendan Eich, treballador en aquell temps de Netscape, com a projecte intern per dotar la web de dinamisme i velocitat.

Gràcies a l'èxit i àmplia acceptació d'aquest llenguatge de scripting del costat del client, Microsoft va llançar a Internet Explorer el seu propi llenguatge "com un fork" de JavaScript, batejat amb el nom de JScript, partint de JavaScript però amb algunes variacions i implementacions afegides com , per exemple, el maneig de dates per a l'"efecte 2000", entre altres.

Tenir diferents aproximacions per a un mateix fi es va convertir en un caos per a la indústria i, per tant, per als usuaris. Algunes pàgines web no funcionaven bé en tots els navegadors i per esmenar aquest comportament era necessari programar diversos codis específics amb l'impacte en costos extra que això comporta a l'equip de desenvolupament.

Per evitar una guerra de tecnologies finalment Netscape envià l'especificació de JavaScript a ECMA International per a la seva estandardització. La primera versió oficial seria ECMAScript 1 i es va publicar en 1997, amb diverses revisions posteriors fins ES6 o ES2015, moment en què l'estàndard es va actualitzant anualment.

1.2. El motor de JavaScript

Per interpretar el llenguatge els navegadors inclouen un motor de JavaScript, sent alguns dels més coneguts dels següents:

- SpiderMonkey (Firefox)
- V8 usat en navegadors basats en Chromium (Chrome, Microsoft Edge, Opera) i en NODE.JS i Deno.
- JavaScriptCore usat en navegadors basats en Webkit (Safari).
- Carakan (versions antigues d'Opera).
- Chakra intèrpret de JScript (Internet Explorer).

JavaScript és un llenguatge single-thread, és a dir, té només un fil d'execució, de manera que les tasques es van executant seqüencialment.

El llançament el 2008 del motor V8 com a projecte de codi obert, desenvolupat per Google, va suposar una gran millora en els motors de JS en crear una combinació d'intèrpret i compilador. Actualment, la resta de motors moderns utilitzen aquesta tècnica, JIT (Just In Time compiler).

1.3. Com funciona?

El compilador tradueix el codi font a bytecode de la manera més ràpida possible i, aquest bytecode és executat per l'intèrpret. El sistema també compta amb un «Profiler» que analitza les operacions i identifica el codi que es pot optimitzar per millorar el rendiment. Després es reemplacen les seccions de bytecode pel codi optimitzat. Això implica que la velocitat d'execució del codi JavaScript millora gradualment mentre s'executa.

A continuació, desglossarem alguns elements fonamentals que necessitem comprendre per poder entendre el funcionament intern de JavaScript.

- **Call stack i Memory Heap**: JavaScript maneja una sola pila de trucades (Call Stack) i un Memory Heap, que fa referència a la part de la memòria no estructurada on es guarden els objectes i funcions. Les funcions del Call Stack es processaran en l'ordre en què es diu, comunament conegut com a Last-In, First-Out (LIFO).
- **Event loop i Callback queue**: Per manejar tasques que requereixen un major temps de processament o triguen a retornar una resposta, com per exemple, una crida a servidor, es fa ús dels callbacks, funcions que es criden quan finalment es rep el resultat de l'operació.
  Els callbacks es van encolant en el que anomenem "Callback Queue" en espera a ser passats a la pila d'execució.
  L'Event Loop és un observador que s'encarrega d'escoltar si hi ha tasques pendents en el Callback Queue i les afegeix al Call Stack quan aquest es buida.
- **Web API**: Fan referència a les API integrades en el navegador, estan creades amb JavaScript i faciliten la implementació d'algunes funcionalitats. Algunes de les interfícies més conegudes són:
  - DOM (Document Object Model)
  - XMLHttpRequest
  - Geolocalització
  - Funcions de timer (setTimeout, setInterval)

- **Hoisting**: El terme Hoisting es refereix al comportament de JavaScript pel qual les declaracions de variables amb ***var*** i funcions són assignades en memòria durant la fase de compilació, això vol dir que es pot utilitzar una variable / funció abans que sigui declarada.
  Per contra, quan es declaren variables amb ***let*** i ***const*** apareix un error, ja que al contrari que ***var***, només estan disponibles el scope en el qual són declarades i el hoisting no realitza l'assignació de la variable a undefined.

2. Capacitats i limitacions d'execució

2.1. Compatibilitats

JavaScript és interpretat pel client. Actualment, existeixen múltiples clients o navegadors que suporten JavaScript, incloent-hi Firefox, Google Chrome, Safari, Opera, Internet Explorer, etc. Per tant, quan escrivim un script a la nostra pàgina web, hem d'estar segurs que serà interpretat per diferents navegadors i que aporti la mateixa funcionalitat i característiques en cada un d'ells. Aquesta és una altra de les diferències amb els scripts de servidor en què nosaltres disposarem del control total sobre la seva interpretació.

Cada tipus de navegador dona suport a diferents característiques de JavaScript i més també afegeixen els seus propis errors o fallades. Alguns d'aquests errors són específics de la plataforma sobre la qual s'executa aquest navegador, mentre que altres són específics del mateix navegador en si.

A vegades les incompatibilitats entre navegadors en interpretar el codi de JavaScript no són causades pel mateix codi en si, sinó que el seu origen prové del codi font HTML. Per tant, és molt important que el teu codi HTML seguiu les especificacions de l'estàndard W3C i per a això disposes d'eines com el validador HTML W3C
.

2.2. Limitacions

- No tots els navegadors suporten llenguatges de script (especialment JavaScript) al costat del client.
- Alguns dispositius mòbils tampoc podran executar JavaScript.
- Fins i tot les implementacions més importants de JavaScript en els diferents navegadors no són totalment compatibles entre elles: per exemple diferents incompatibilitats entre Firefox i Internet Explorer.
- L'execució de codi JavaScript al client podria ser desactivada per l'usuari de forma manual, de manera que no podrem tenir una confiança cega en què es vagi a executar sempre el codi de JavaScript.
- Alguns navegadors de veu, no interpreten el codi de JavaScript.

3. Llenguatges de programació en entorn client

Quan parlem de llenguatges de programació en clients web, podem distingir dues variants:

- Llenguatges que ens permeten donar **format i estil** a una pàgina web (HTML, CSS, etc.).
- Llenguatges que ens permet aportar **dinamisme** a pàgines web (llenguatges de scripting).

En aquest mòdul ens centrarem principalment en aquests últims, els llenguatges de scripting, i en particular en el llenguatge JavaScript que serà el llenguatge que utilitzarem al llarg de tot aquest mòdul formatiu.

Llenguatges de programació web client - servidor	
Cliente	**Servidor**
HTML	CGI
JavaScript	PERL
Applets de Java	ASP
VB Scritpt	PHP
Flash	JSP
Css	

3.1. Característiques

JavaScript està orientat a donar solucions a:

- Aconseguir que la nostra pàgina web respongui o reaccioni directament a la interacció de l'usuari amb elements de formulari i enllaços hipertext.
- La distribució de petits grups de dades i proporcionar una interfície amigable per a aquestes dades.
- Controlar múltiples finestres o marcs de navegació, plug-ins, o applets Java basats en les eleccions que ha fet l'usuari en el document HTML.
- Preprocessar dades en el client abans d'enviar-los a servidor.
- Modificar estils i contingut en els navegadors de forma dinàmica i instantàniament, en resposta a interaccions de l'usuari.

- Sol·licitar fitxers de servidor, i enviar peticions de lectura i escriptura als llenguatges de servidor.

3.2. Compatibilitats

A vegades les incompatibilitats entre navegadors en interpretar el codi de JavaScript no són causades pel mateix codi en si, sinó que el seu origen prové del codi font HTML. Per tant, és molt important que el teu codi HTML seguiu les especificacions de l'estàndard W3C i per a això disposes d'eines com el validador HTML W3C:

The W3C Markup Validation Service

3.3. Seguretat

JavaScript no podrà realitzar cap de les següents tasques:

- Modificar o accedir a les preferències del navegador del client, les característiques d'aparença de la finestra principal de navegació, les capacitats d'impressió, o als botons d'accions del navegador.
- Llançar l'execució d'una aplicació a l'ordinador de client.
- Llegir o escriure fitxers o directoris a l'ordinador del client (amb l'excepció de les cookies).
- Escriure directament fitxers al servidor.
- Capturar les dades procedents d'una transmissió en streaming d'un servidor, per a la seva retransmissió.
- Enviar e-mails a nosaltres mateixos de forma invisible sobre els visitants a la nostra pàgina web (Tot i que sí que podria enviar dades a una aplicació en el costat del servidor capaç d'enviar correus).
- Interactuar directament amb els llenguatges de servidor.
- Les pàgines web emmagatzemades en diferents dominis no poden ser accessibles per JavaScript.
- JavaScript és incapaç de protegir l'origen de les imatges de la nostra pàgina.
- Implementar multiprocessament o multitasca.
- Un altre tipus de vulnerabilitats que podem trobar estan relacionades amb el XSS. Aquest tipus de vulnerabilitat viola la política de "mateix origen" i passa quan un atacant és capaç d'injectar codi maliciós a la pàgina web presentada a la seva víctima. Aquest codi maliciós pot provenir de la base de dades a la qual està accedint aquesta víctima. Generalment, aquest

tipus d'errors es deuen a errors d'implementació dels programadors de navegadors web.

4. Tecnologies i llenguatges associats

La idea és decantar-se per editors que posseeixin característiques que facilitin la programació web, com per exemple:

- **Sintaxi amb codificació de colors**, que ressalti automàticament en diferent color o família tipogràfica als elements del llenguatge com ara objectes, comentaris, funcions, variables, etc.
- **Verificació de sintaxi**, que marqui els errors en la sintaxi del codi que s'està escrivint.
- Que **generi automàticament** parts del codi com ara blocs, estructures, etc.
- Que disposi d'**utilitats addicionals** tals com a client FTP, servidor web, etc.

Un altre dels components obligatori per aprendre JavaScript és el **navegador web**. No cal connectar-se a Internet per comprovar els scripts realitzats amb JavaScript.

5. Integració del codi amb les etiquetes HTML

Els navegadors web et permeten diverses opcions d'inserció de codi de JavaScript:

- Podrem inserir codi usant les etiquetes **<script> </ script>** i emprant l'atribut **type** indicarem quin tipus de llenguatge de *script* estem utilitzant.
  Per exemple:

```
<script type="text/javascript">
  // El código de JavaScript vendrá aquí.
</scritp>
```

```
<!DOCTYPE html>
<html lang="es-es">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width,  itial-scale=1">
<script type="text/javascript">JAVASCRIPT SUELE COLOCARSE AQUÍ</script>
<title>Proves</title>
</head>
<body>
```

<script type="text/javascript">JAVASCRIPT TAMBIÉN PUEDE IR AQUÍ</script>
</body>
</html>

- Una altra forma d'integrar el codi de JavaScript és incrustar un **fitxer extern** que contingui el codi de JavaScript. Aquesta seria la forma més recomanable, ja que així s'aconsegueix una separació entre el codi i l'estructura de la pàgina web i com a avantatges addicionals podràs compartir el codi entre diferents pàgines, centralitzar el codi per a la depuració d'errors, tindràs més claredat en els teus desenvolupaments, més modularitat, seguretat del codi i aconseguiràs que les pàgines carreguin més ràpidament. La rapidesa de càrrega de les pàgines s'aconsegueix en tenir el codi de JavaScript en un fitxer independent, ja que si més d'una pàgina ha d'accedir a aquest fitxer ho agafarà automàticament de la memòria cau del navegador amb el qual s'accelerarà la càrrega de la pàgina.
  Per a això haurem d'afegir a l'etiqueta script l'atribut **src**, amb el nom del fitxer que conté el codi de JavaScript. Generalment, els fitxers que contenen text de JavaScript tindran l'extensió **.js**.
  Per exemple:
  <script type="text/javascritp" src="codigo.js"></script>
  El situarem just després de tancar el body.