Ricard Meyerhofer Parra

LP 2015-16 Q2

Professor: Albert Rubio

Índex

Introducció

Història del llenguatge

Vista la necessitat d'una alternativa a les pàgines webs estàtiques Javascript, va ser creat el maig de 1995 per Brendan Eich sota el nom de Mocha. El setembre d'aquell mateix any, va ser reanomenat a LiveScript i finalment, el desembre va adoptar el nom de JavaScript després de rebre llicència de Sun (el canvi de nom va ser una espècie moviment de marquèting doncs en aquell moment, Java era molt popular).

Als seus inicis JavaScript aparentment no requeria d'una gran habilitat per a ser usat, no tenia un IDE, ni un debugger multiplataforma, només podia ser provat via navegador i tot això, juntament amb els errors de seguretat que tenia coneguts i els llibres per a no-programadors que hi havia, va fer que molta gent titllés JavaScript com a un llenguatge simple (per a principiants) ignorant així el potencial d'aquest.

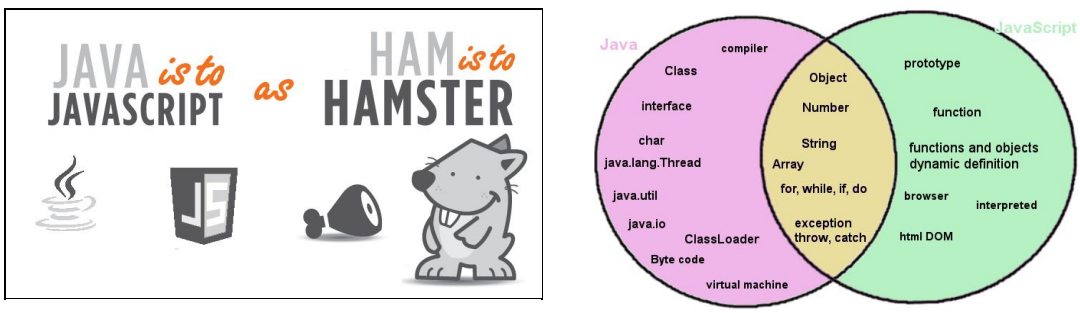
Una mica de cronologia del llenguatge:

- 1996 Microsoft va treure JScript com a resposta a JavaScript (per al seu ús a Internet Explorer).
- 1997 participa a ECMA on s'estandarditza sota el nom de ECMAScript.
- 1998 i 1999 surten ECMAScript2 i ECMAScript3, els quals són la base del JavaScript actual.
- A partir dels 2000 apareix una proliferació d'implementacions per la part del servidor Node.js essent un dels més destacables.
- Amb l'arribada d'Ajax proliferen frameworks i llibreries augmentant el seu ús fora de l'àmbit de web. 2009 CommonJS apareix per a especificar una llibreria d'ús comú principalment per a entorns fora de l'àmbit web.
- 2015 surt ECMAScript 6 on JavaScript incorpora mòduls d'organització del codi, vertaderes classes, iteradors, etc

Llenguatges similars

Encara que pugui semblar que pel nom Java i JavaScript siguin similars, no és així. Java i JavaScript tenen sintaxi i propòsits molt diferents. La sintaxi de JavaScript de fet deriva de C i a nivell de disseny està influenciat per Self i Scheme tot i que és cert que adopta noms i convencions de Java.

A part dels ja mencionats també s'inspira de Perl on incorpora les expressions regulars i Python d'on treu la forma d'iterar els objectes, les corutines i la generació de llistes i expressions per comprensió.



Imatge 0: Imatges que ilusten la diferència entre Java i JavaScript.

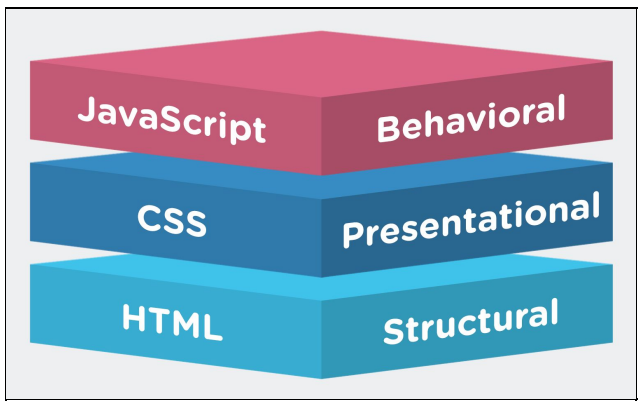
JavaScript per una part ha inspirat a molts llenguatges i per altra part es va inspirar en uns altres. Així doncs JavaScript per exemple ha donat lloc a llenguatges com:

- ObjectiveJ.
- JScript.
- JScript.NET.
- TIScript.
- CoffeeScript.
- IO.
- TypeScript.

Principals aplicacions

La principal aplicació de JavaScript és l'entorn web juntament amb HTML i CSS, és una de les tres principals tecnologies WWW (World Wide Web) amb les quals es genera contingut. Gairebé totes les webs l'incorporen i tots els navegadors hi tenen suport.
L'ús més comú de JavaScript el trobem per a programar funcionalitats a les pàgines en la part del client (tot i que actualment també hi ha JavaScript a la part de servidor). Aquests Scripts són afegits a les pàgines HTML per interactuar amb el DOM (Document Object Model) de la pàgina. Per exemple JavaScript s'empra per:

- Cercador (és amb diferència l'entorn més habitual on s'usa).
- Carregar nou contingut d'un servidor o enviar-ne.
- Validar els continguts d'un Webform tal que els paràmetres siguin correctes.



Imatge 1: Imatges representatives de l'entorn més emprat a la web i la seva funcionalitat.

Altres aplicacions

JavaScript a part de ser emprat per a Web, també s'utilitza per a:

- Desktop widgets.
- Desenvolupament de videojocs.
- Aplicacions mòbils i escriptori.
- Programació d'un servidor amb Node.js.

Compilat o interpretat?

JavaScript en els seus origen és interpretat doncs no requereix d'una compilació abans de ser executat i en el seu lloc un intèrpret llegeix el codi i l'interpreta i executa línia a línia. Tot i això actualment JavaScript pot ser compilat gràcies a la compilació just in time (JIT) el qual compila JavaScript a un bytecode executable just abans de ser executat.

Sistema de tipus de JavaScript

El tipat de JavaScript es tracta d'un tipat feble, dinàmic i de tipus segur.
Tipat feble doncs podem trobar els següents errors de tipus:

- Alliberament explícit de memòria (deallocate/delete) no és dóna doncs té un Garbage collector que ho gestiona.
- Operacions entre diferents tipus.
- No hi ha aritmètica de punters a JavaScript.

Dinàmic doncs es comprova el tipus d'una variable en temps d'execució i per tant no l'haurem d'especificar o no és necessari saber-lo en temps de compilació com en els tipats estàtics. Això ens porta un avantatge i un desavantatge:
Per una part, tenim que podem anar més ràpidament a picar el codi i que si tot el que fem és correcte ens sortirà el que volem. El problema és que no tindrem els errors de compilació d'un C++ o un Java per qüestions de tipus el qual ens pot fer perdre força temps i ser un mal de cap.

Tot i que no té molt de sentit parlar de si és un llenguatge de tipus segur pel fet de ser interpretat des de fa un temps, JavaScript incorpora compilació JIT (just in time) . Així doncs, JavaScript internament sempre farà les conversions necessàries per a què no hi hagi errors de tipus.
JavaScript disposa de la variable var la qual engloba qualsevol tipus de dades (com podríen ser strings, chars, ints etc)

Paradigmes de programació

Què és un paradigma?
Entenem com a paradigma a una orientació o filosofia de programació que s'utilitza al moment d'implementar un programa. Hi ha certs paradigmes que solen ser millors que d'altres per a resoldre certs problemes i viceversa.

Paradigmes de JavaScript
JavaScript actualment disposa dels següents paradigmes:

- Imperatiu.
- Procedimental.
- Estructurat.
- Orientat a objectes
- Prototípic

- Funcional

Imperatiu
El paradigma imperatiu on majoritàriament les sentències són assignacions, descriu com realitzar el càlcul en una sèrie de sentències, executades segons un control de flux explícit que modifiquen l'estat del programa el qual són cel·les de memòria que contenen dades/referències i representen l'estat del programa.

Procedimental
Indica un model d'organització de programes els quals s'organitzen com col·leccions de subrutines mitjançant invocacions entre elles. Es compleixen les propietats d'encapsulament, independència entre procediments…

Estructurat
És un subparadigma del paradigma imperatiu. El punt més notori d'aquest paradigma és que els codis guanyen una llegibilitat i facilitat de veure errors a reduir doncs no incorporen elements com els GOTO (els quals generen un model de codi espagueti).

Orientat a objectes
Les classes tenen objectes que contenen mètodes de classes i variables, instàncies d'objectes que contenen instàncies de mètodes i variables. L'herència i les interfícies proporcionen polimorfisme.

Prototípic
És un paradigma que proporciona molta interacció entre l'usuari i el desenvolupador. Ens permet fer representacions parcials ràpidament del que es vol fer. S'usa sobretot quan no els requeriments no estan del tot definits i hi ha molta interacció amb l'usuari, per a mostres al client d'una idea sobre el que es farà.. Els processos de verificació i validació tendeixen a ser mínims.

Funcional
El paradigma funcional està basat en el càlcul de lambda i es caracteritza pel fet que el valor generat d'una funció depèn només dels arguments de la funció. Això fa que sigui més previsible i per tant més fàcil d'entendre que el paradigma imperatiu la mateixa expressió pot tenir valors diferents en moments diferents del programa.
Dins del paradigma funcional ens trobem amb propietats les gairebé totes ja vam poder veure a haskell com:

- Pattern matching.
- Monads.
- Closures
- Funcions d'ordre superior.
- Funcions anònimes.

Altres característiques

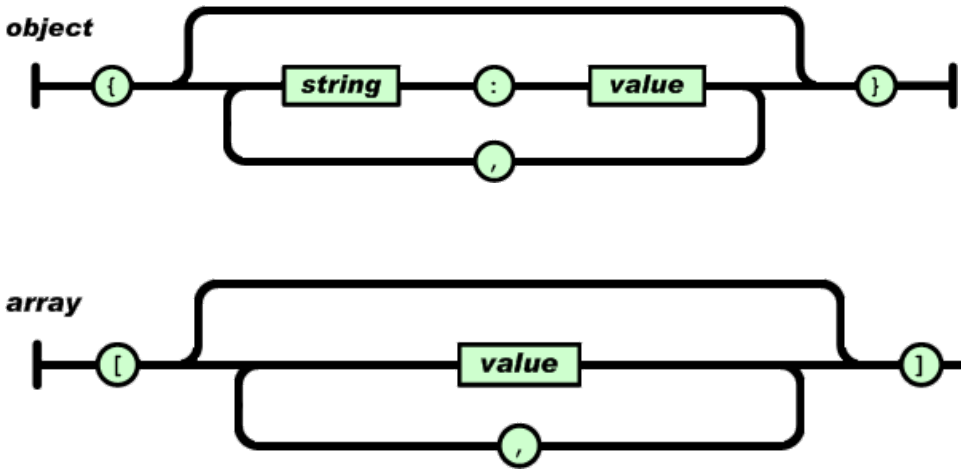
- Expressions regulars.
- JSON

JSON (JavaScript Object Notation) es tracta d'un format lleuger d'intercanvi de dades que actualment és molt popular i està començant a ser una competència directe del XML degut a que és molt llegible i fàcil de tractar.

JSON està constituït per a dues estructures:

- Objectes (diccionaris)
- Vectors

A continuació adjuntem unes imatges que expliquen què pot compendre cada tipus de dades:



object
{}
{ *members* }
members
pair
pair , *members*
pair
string : *value*
array
[]
[*elements*]
elements
value
value , *elements*
value
string
number
object
array
true
false
null

Un exemple de lo senzill que és JSON i per què està guanyant popularitat respecte XML seria el següent:

```
{"employees":[
    {"firstName":"John", "lastName":"Doe"},
    {"firstName":"Anna", "lastName":"Smith"},
    {"firstName":"Peter", "lastName":"Jones"}
]}
```

```
<employees>
    <employee>
        <firstName>John</firstName> <lastName>Doe</lastName>
    </employee>
    <employee>
        <firstName>Anna</firstName> <lastName>Smith</lastName>
    </employee>
    <employee>
        <firstName>Peter</firstName> <lastName>Jones</lastName>
    </employee>
</employees>
```

Exemples

Exemple 1: Funció anònima + clausura.

```
var displayClosure = function() {
    var count = 0;
    return function () {
        return ++count;
    };
}
var inc = displayClosure();
inc(); //retorn count+1
```

Exemple 2: Funció com a paràmetre (Callback)

```
function loadStuff(callback) {
    // Go off and make an XHR or a web worker or somehow generate some data
    var data = ...;
    callback(data);
}
loadStuff(function(data){
    alert('Now we have the data');
});
```

Exemple 3: Expressions regulars

```
var input = "A string with 3 numbers in it... 42 and 88.";
var number = /\b(\d+)\b/g;
var match;
while (match = number.exec(input))
  console.log("Found", match[1], "at", match.index);
// → Found 3 at 14
//   Found 42 at 33
//   Found 88 at 40
```

Exemple 4: Nombres de Hamming.

```
function hamming() {
  var queues = {2: [], 3: [], 5: []};
  var base;
  var next_ham = 1;
  while (true) {
    yield next_ham;

    for (base in queues) {queues[base].push(next_ham * base)}

    next_ham = [ queue[0] for each (queue in queues) ].reduce(function(min, val) {
      return Math.min(min,val)
    });

    for (base in queues) {if (queues[base][0] == next_ham) queues[base].shift()}
  }
}

var ham = hamming();
var first20=[], i=1;

for (; i <= 20; i++)
  first20.push(ham.next());
print(first20.join(', '));
print('...');
for (; i <= 1690; i++)
  ham.next();
print(i + " => " + ham.next());
```

Bibliografia

Tota la informació que apareix sempre ha estat contrastada amb diverses fonts les quals, no apareixen totes doncs han servit per a validar que el contingut fos correcte.
En termes generals per a fer-me una mica d'idea del llenguatge sobre si és compilat/interpretat, en quins llenguatges s'inspira o es deixa d'inspirar, la seva història i fins i tot algún exemple, m'he basat en wikipedia doncs amb poc espai, dóna informació sobre gairebé tot el que es demanava. Però això no significa que hagi agafat literalment tot el que surt a la wikipedia.

Per a l'història del llenguatge per exemple he usat les tres fonts que es veuen avall on entre les tres i wikipedia, he intentat recopilar i quadrar tot el que ha passat amb JavaScript.

Sobre el tema de compilat/interpretat vaig mirar en diversos llocs (doncs sempre hi ha qui té una idea diferent de si és compilat/interpretat), sobretot pel fet de tenir JIT. I un cop vaig trobar l'enllaç que he adjuntat, contrastant informació de diverses fonts em va quedar clar.

Pel sistema de tipus, el que vaig fer basicament va ser buscar varies webs amb exemples i vaig provar si el que deien era cert o no.

Pels exemples algún de wikipedia em va semblar suficient, d'altres vaig mirar a rosettacode.

Finalment els llibres els he usat com a font de contrast tot i que no han estat la meva font més emprada i només m'hi he recolzat en algun moment. Però realment JavaScript té molta informació que es pot aconseguir sense accedir a llibres físicament (els dos llibres es poden trobar a la biblioteca BRGF.

Enllaços web:

Informació en general

- https://en.wikipedia.org/wiki/Self_%28programming_language%29
- http://web.stanford.edu/class/cs98si/slides/overview.html

- **Compilat/Interpretat**
  - http://techiejs.com/Blog/Post/Yes-Modern-JavaScript-is-Compiled
- **Tipus**
  - http://www.etnassoft.com/2011/01/27/tipado-blando-en-javascript/
- **Història**
  - http://www.crockford.com/javascript/javascript.html
  - http://archive.oreilly.com/pub/a/javascript/2001/04/06/js_history.html
  - http://www.techotopia.com/index.php/The_History_of_JavaScript
- **Altres característiques**
  - http://www.json.org

- **Exemples**
    - **https://es.wikipedia.org/wiki/JavaScript**
    - **https://rosettacode.org/wiki/Hamming_numbers**
- **Imatges:**
    - **http://qlikshare.com/wp-content/uploads/2016/03/jsvsjava.jpg**
    - **http://j2s.sourceforge.net/j2s-user-guide/dist/j2s-user-guide-htmls/chap-java-to-js-translation.html**
    - **http://blog.teamtreehouse.com/wp-content/uploads/2014/11/progressive-enhancement.png**
    - **http://www.json.org**

Llibres:

- Learning JavaScript / Shelley Powers
- JavaScript fácil / Arnaldo Pérez Castaño