JavaScript

Ricard Meyerhofer

2015-2016 Q2

Albert Rubio

Introducció

- Creat el 1995 per Brendan Eich
- Molta gent pren com a llenguatge per a principiants.
- Llenguatge enfocat a la part del client web
- 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
- 2000 proliferació d'implementacions per la part del servidor

Introducció

- 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

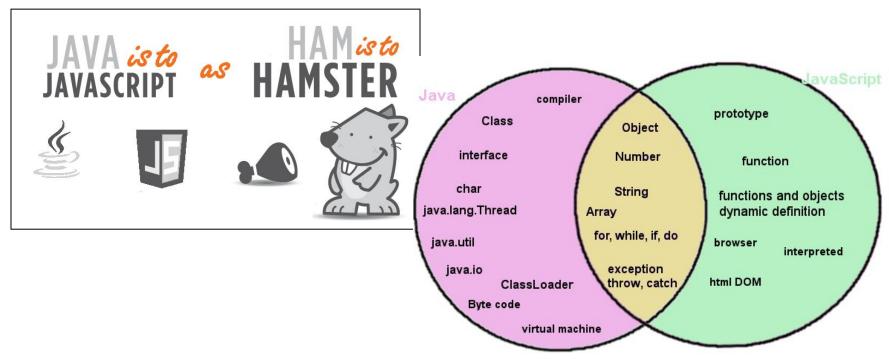
JavaScript s'ha inspirat de:

- C
- Scheme
- Self
- Perl
- Python

S'han inspirat de JavaScript:

- ObjectiveJ
- JScript
- JScript.NET
- TIScript.
- CoffeeScript
- 10
- TypeScript

Llenguatges Similars



Paradigmes

- Imperatiu
- Procedimental
- Estructurat
- Orientat a objectes
- Basat en prototipus
- Funcional

Spaghetti Sample Structured Sample 1.058 JMPL RBW SWITCHING 1.058 JMPL ResolutionBandwidth 1.059 LABEL FREQUENCY READOUT 1.059 LABEL ResolutionBandwidth done 1.060 CALL Sub Frequency Readout 1.061 CALL Sub Frequency Span 1.060 JMPL ResolutionSwitching 1.061 LABEL ResolutionSwitching done 1.062 CALL Sub Noise Sidebands 1.063 JMPL SYSTEM SIDEBANDS 1.062 JMPL DisplayedAverage 1.064 LABEL WIDE OFFSETS 1.063 LABEL DisplayedAverage done 1.065 CALL Sub Noise Wide Offsets 1.064 JMPL ResidualResponses 1.066 JMPL SPURIOUS 1.065 LABEL ResidualResponses_done 1.067 LABEL SYSTEM SIDEBANDS 1.066 JMPL AbsoluteAmplitude 1.068 CALL Sub System Sidebands 1.067 LABEL AbsoluteAmplitude done 1.069 CALL Sub Residual 1.068 JMPL FrequencyReadout 1.070 CALL Sub Display Switching 1.071 JMPL SCALE FIDELITY 1.069 LABEL FrequencyReadout done 1.072 LABEL SWEEP TIME 4 1.070 JMPL FrequencySpan 1.073 CALL Sub Sweep Time 1.071 LABEL FrequencySpan done 1.074 JMPL WIDE_OFFSETS 1.072 JMPL NoiseSidebands 1.075 LABEL SCALE FIDELITY 1.073 LABEL NoiseSidebands done 1.076 CALL Sub Scale Fidelity 1.074 JMPL SystemSidebands 1.077 CALL Sub Input Switching 1.078 CALL Sub Reference Level 1.075 LABEL SystemSidebands done 1.079 JMPL FREQUENCY_RESPONSE 1.076 JMPL Residual 1.080 LABEL RBW SWITCHING 1.077 LABEL Residual done 1.081 CALL Sub Resolution BW Switching 1.078 JMPL Display Switching 1.082 JMPL RESOLUTION BANDWIDTH 1.079 LABEL Display Switching done 1.083 LABEL ABSOLUTE AMPLITUDE -1.080 JMPL ScaleFidelity 1.084 CALL Sub Absolute Amplitude 1.085 JMPL FREQUENCY READOUT 1.081 LABEL ScaleFidelity_done 1.086 LABEL RESOLUTION BANDWIDTH 4 1.082 JMPL InputSwitching 1.087 CALL Sub Resolution Bandwidth 1.083 LABEL InputSwitching_done 1.088 JMPLAVERAGE NOISE 1.084 JMPL ReferenceLevel 1.089 LABEL FREQUENCY RESPONSE 1.085 LABEL ReferenceLevel_done 1.090 CALL Sub Freg Response 1.086 JMPL FregResponse 1.091 JMPL SWEEP_TIME 1.092 LABELAVERAGE NOISE 1.087 LABEL FreqResponse done 1.088 JMPL SweepTime 1.093 CALL Sub Displayed Average 1.094 JMPL RESIDUAL RESPONSES 1.089 LABEL SweepTime done 1.095 LABEL SPURIOUS -1.090 JMPL NoiseWideOffsets 1.096 CALL Sub TOL 1.091 LABEL NoiseWideOffsets done 1.097 JMPL GAIN COMPRESSION · 1.092 JMPL TOI 1.098 LABEL 2ND HARMONIC 1.093 LABEL TOI done 1.099 CALL Sub 2nd Harmonic 1.094 JMPL GC 1.100 CALL Sub Other Spurs 1.101 JMPL END 1.095 LABEL GC done 1.102 LABEL GAIN_COMPRESSION 1.096 JMPL 2ndHarmonic 1.103 CALL Sub GC 1.097 LABEL 2ndHarmonic 1.104 JMPL 2ND HARMONIC -1,098 JMPL OtherSpurs 1.105 LABEL RESIDUAL RESPONSES 1.099 LABEL OtherSpurs 1,106 CALL Sub Residual Responses 1.107 JMPLABSOLUTE AMPLITUDE 1.108 LABEL END -

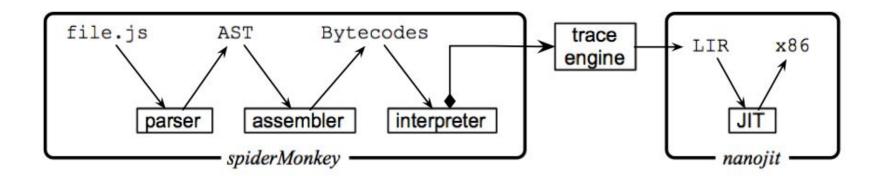
Paradigma funcional

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

```
var $ = fun.parameter,
   fact = fun(
       [0, function () { return 1; }],
       [$, function (n) { return n * fact(n - 1); }]
   );
 var displayClosure = function() {
     var count = 0;
      return function () {
          return ++count;
     };
 var inc = displayClosure();
 inc(); // devuelve 1
 inc(); // devuelve 2
 inc(); // devuelve 3
```

Compilat/Interpretat?

- Interpretat
- Compilació JIT (Just In Time)



Sistema de Tipus

- Feble
 - Operacions entre diversos tipus
- Dinàmic
- Llenguatge de tipus segur
- JavaScript disposa de la variable var que s'usa per a tot

Aplicacions

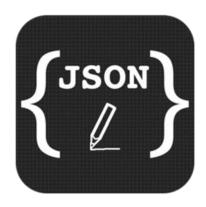
- Entorn web
 - Cercador
 - Enviar informació
 - Validació de continguts
- Programació de servidors amb Node.js
- Desenvolupament de videjocs
- Aplicacions



Altres Característiques

- Expressions regulars
- JavaScript Object Notation
 - Format lleuger d'intercanvi de dades
 - Constituit per dues estructures:
 - Parells nom/valor (diccionari)
 - Llista ordenada de valors (vectors)
 - o Actualment molt popular per la seva facilitat de lectura

i tractament, competència directe del XML



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

false null

Altres Característiques

```
{"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>
```

Exemple

```
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()}
```