



JavaScript

Llenguatges de Programació

Albert López Alcácer



Origen del llenguatge

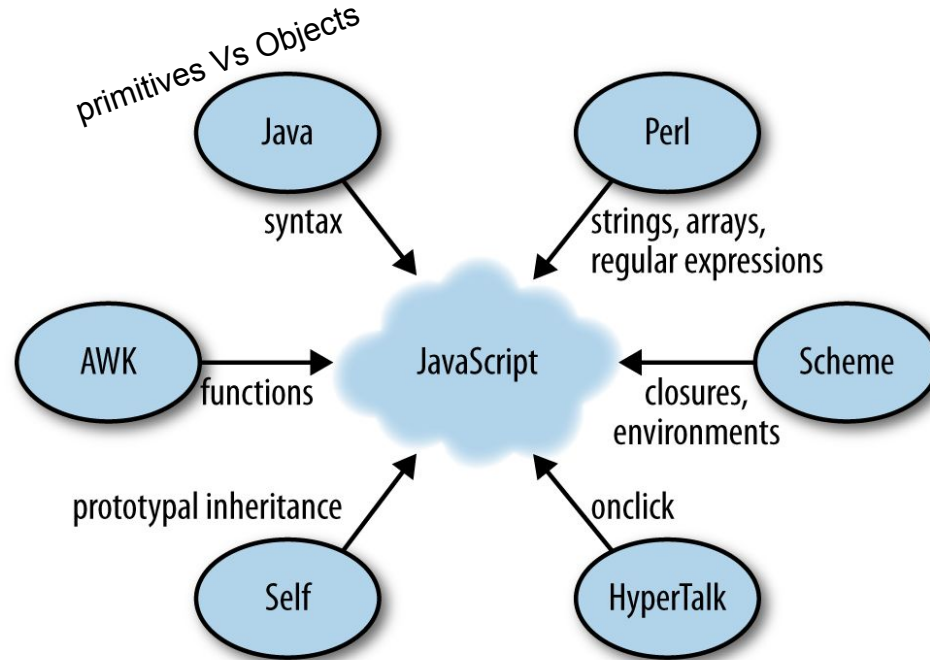
- Neix de la necessitat de fer les pàgines web interactives
- Creat el 1995 per Brendan Eich com a prototipus a incorporar al navegador Netscape Navigator
- De Mocha a LiveScript, i de LiveScript a JavaScript
- Altres implementacions: 1996, Microsoft amb el seu JScript.

Estandarització i versions

- 1997, Netscape treballa en la estandarització ECMAScript amb Ecma International.
- ECMAScript 1 (1997), ECMAScript 2 (1998), ECMAScript 3 (1999), ECMAScript 5 (2009)
- ECMAScript 5.1 (2011), ECMAScript 6(2015)
- A partir del 2000... Del costat del servidor amb Node.js o Common.js (2009)

Influències

```
function name(parameter-list)
{
    body-of-function
}
```



`<button onclick="myFunction()">Click me</button>`



Paradigmes de programació

Llenguatge multiparadigma:

- Funcional
- Basat en prototipus
- Orientat a objectes
- Imperatiu i procedural



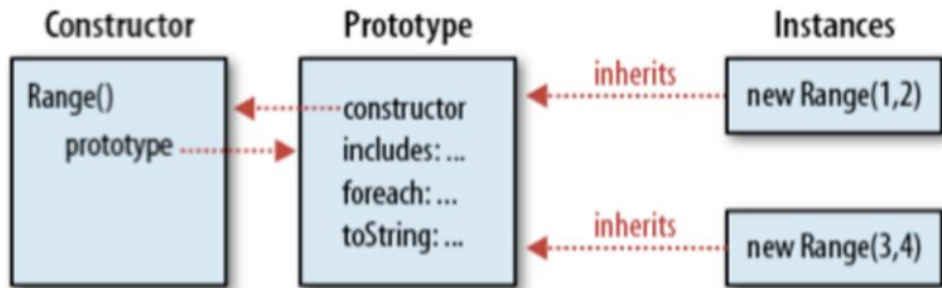
JavaScript funcional

Programes fent servir:

- Funcions pures
- Composició de funcions
- Currificació de funcions
- Implementacions de funcions d'ordre superior, Monades, Functors (ben coneguts en llenguatges com Haskell)
- Exemple pràctic: Rxjs for async. programming

Basat en prototipus

- Podem simular les classes amb els prototipus
- Si un objecte no entén una propietat, es busca a l'objecte que te com a prototipus.

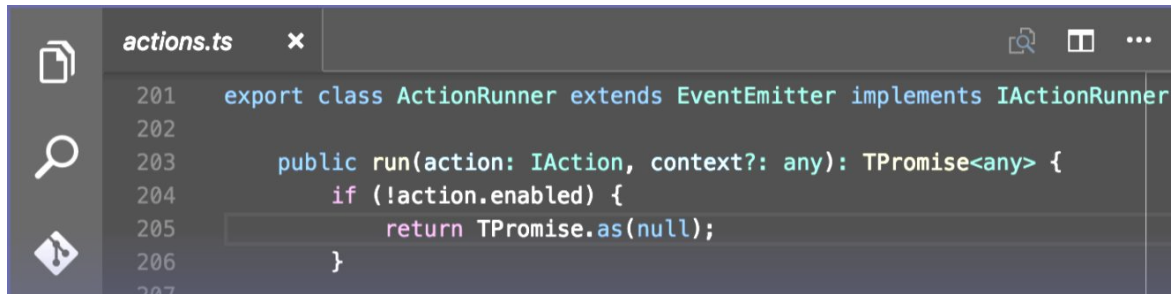


```
DynamicRange.prototype = inherit(Range.prototype)
```



Orientat a objectes

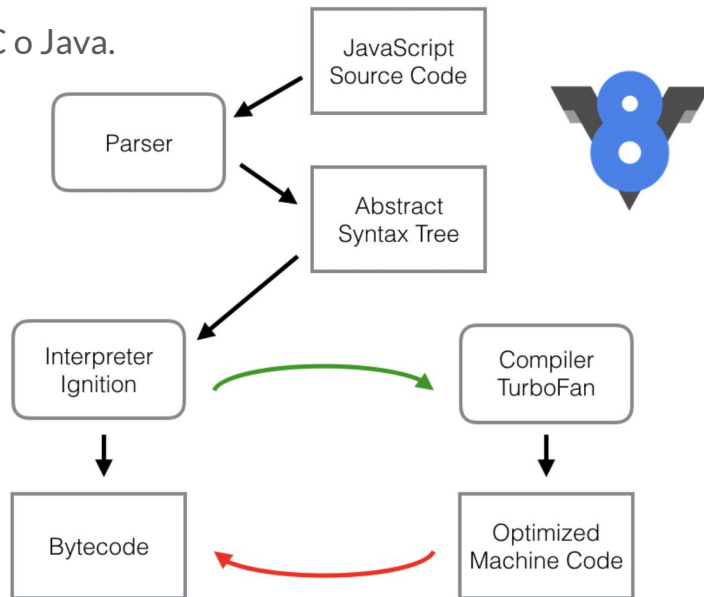
- Objectes amb estat emmagatzemat, que anem modificant instrucció a instrucció.
- Molts “frameworks” fan servir aquest paradigma
 - Angular2 + TypeScript



```
actions.ts x
201 export class ActionRunner extends EventEmitter implements IActionRunner
202
203     public run(action: IAction, context?: any): TPromise<any> {
204         if (!action.enabled) {
205             return TPromise.as(null);
206         }
207     }
```

Compilat o interpretat?

- Es JIT (“Just in Time”) compilat, i no pas AOT (“Ahead of Time”) com C o Java.
- Els programes es distribueixen en el seu codi font.
- L’engine realitza les tasques pròpies d’un compilador.
- En concret: V8 de Google.

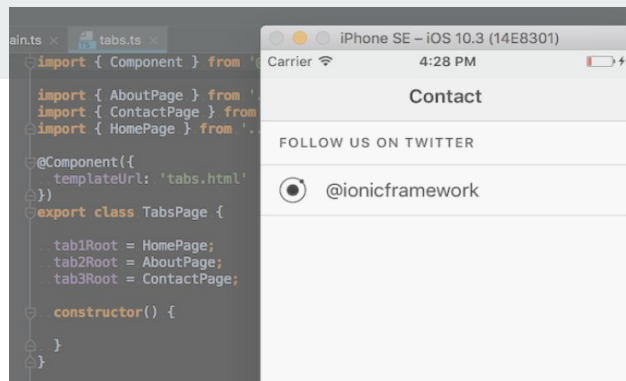




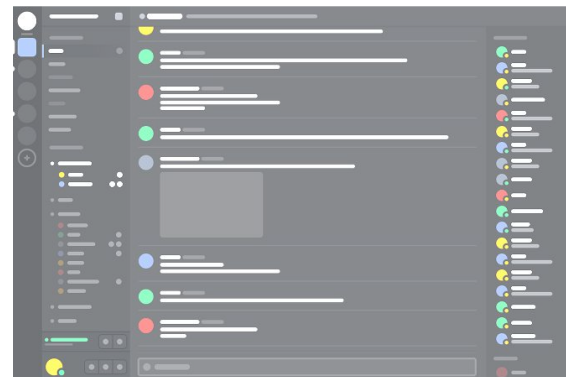
El sistema de tipus

- Tipatge dinàmic
 - No donem informació prèvia al compilador del tipus de les variables.
 - Variables contenen tipus primitius o objectes (col·leccions dinàmiques de parells clau, valor)
 - La paraula clau “var” (i “let”)
 - Positiu (en part) per a desenvolupadors, negatiu per als compiladors

Principals aplicacions



- Del costat del navegador o Frontend (han anat apareixent frameworks per a fer SPA “Single page Applications”)
- Al servidor (com Node.js i bases de dades NoSQL funcionant amb JSON)
- Aplicacions mòbil multiplataforma (com Ionic)
- Aplicacions d’escriptori multiplataforma (com Electron)
- O fins i tot sistemes operatius! (Chrome OS)





Altres característiques

- JSON

```
{  
  "first": "Jane",  
  "last": "Porter",  
  "married": true,  
  "born": 1890,  
  "friends": [ "Tarzan", "Cheeta" ]  
}
```

- Scope, hoisting i closures



Exemples de codi

- Pila fent servir closures

```
function StackCr () {  
  var index = 0;  
  var arr = [];  
  return {  
    push: function(val) {  
      arr[index] = val;  
      index++;  
    },  
    pop: function() {  
      /* arr[-1] = num afegiria '-1' com a  
      propietat de l'objecte arr i les funcions de  
      array (con arr.length no els tindria en compte*/  
      if (index > 0)  
        index--;  
    },  
    top: function() {  
      return arr[index - 1];  
    },  
    size: function() {  
      return index;  
    },  
    empty: function() {  
      return index == 0;  
    }  
  }  
}  
var pila = StackCr();
```



Exemples de codi

- Fibonnaci fent servir propietats com a cache

```
function factorial(n) {  
    if ( !(n in factorial) )  
        factorial[n] = n * factorial(n-1);  
    return factorial[n];  
}  
factorial[1] = 1;
```

```
/*De manera que aprofitem que les funcions com a  
tal son objectes, i per tant poden tenir  
propietats, per a anar guardant els resultats de  
les crides */
```



Exemples de codi

- Callbacks

```
var display = function() {...}  
pictureService.getPicture(id)  
    .subscribe(  
        (picture) => {  
            console.log('Foto rebuda correctament')  
            display(picture)  
        },  
        (error) => console.log("S'ha produït un error: " + error)  
    )
```



Gràcies per l'atenció