Porque usar javascript para desenvolver solução de economia de energia em dispositivos moveis

Baltazar Tavares Vanderlei

Instituto de Computação - IC/UFAL

4 de abril de 2013

Sumário

Sumário

- Linguagem moderna
- O mais portavel das alternativas
- Suporta Annotations
- Serialização
- Reflexão
- Sobrescrever funções

Sumário

Na verdade... Não tem

Mas existe como contornar

- Fazer na mão(a seguir...)
- Declarar em um espaço de nomes conhecido
- Object.defineProperty(obj, propname, desc)

```
function Human(){
        /* @@-FirstName-
         * myType:TEXT,
         * myDefaultValue:John
         * @@ */
        this.FirstName;
        /* @@-LastName-
         * myType:TEXT,
         * myDefaultValue:Doe
         * @@ */
        this.LastName;

        this.toString = function(){
                return this.FirstName+" "+this.LastName;
        }
}
```

```
var h = new Human();

console.log(Annotations(Human));
h.FirstName = Annotations(Human)['FirstName']['myDefault
h.LastName = Annotations(Human).LastName.myDefaultValue;
console.log(h.toString()); /*display John Doe*/
```

Sumário

- Disponivel com JSON
- Se não disponovel, pode usar implementação propria
  - http://www.sitepoint.com/javascript-json-serialization/
  - https://github.com/douglascrockford/JSON-js

```
var obj1 = {
b1: true,
s1: "text string",
n1: 12345,
n2: null,
n3: undefined,
a1: [ 1,1,2,3,5,8, [13, 21, 34] ],
o1: {
        a: [3, 2, 1],
        b: {
                c: 42,
                d: [ 3.14, 1.618 ]
        }
}
};
```

```
// JSON stringify
var json = JSON.stringify(obj1);

// JSON parse
var obj2 = JSON.parse(json);
```

Sumário

- Object.getOwnPropertyNames(obj)
- Object.getOwnPropertyDescriptor(obj, "prop")

```
> var obj = { prop: "abc", method: function(x,y,z) {} };
> Object.getOwnPropertyNames(obj)
[ 'prop', 'method' ]
```

```
> Object.getOwnPropertyDescriptor(obj, "prop")
{ value: 'abc'
, writable: true
, enumerable: true
, configurable: true
}
```

```
> Object.getOwnPropertyDescriptor(obj, "method")
{ value: [Function]
, writable: true
, enumerable: true
, configurable: true
}
```

Extraind argumento de funcões

```
function argumentNames(fun) {
        var names = fun.toString().match(/^[\s\(]*functi
                .replace(/\/\/.*?[\r\n]|\/\*(?:.|[\r\n])
                .replace(/\s+/g, '').split(',');
        return names.length == 1 && !names[0] ? [] : nam
}
function foo(bar, baz) {}
argumentNames(foo)
```

Objeto proxy

```
var obj = {
        __noSuchMethod__: function(name, args) {
                print("Method " + name);
        }
};

obj.foo();
```

Getters e Setters

```
var obj = {
get foo() {
        return "getter";
},
set foo(value) {
        print("setter: "+value);
}
};
```

```
> obj.foo
getter
> obj.foo = "bla";
setter: bla
bla
> Object.getOwnPropertyDescriptor(obj, "foo")
{ get: [Function: foo]
, set: [Function: foo]
, enumerable: true
, configurable: true
}
```