NEW FEATURES OF ECMASCRIPT6

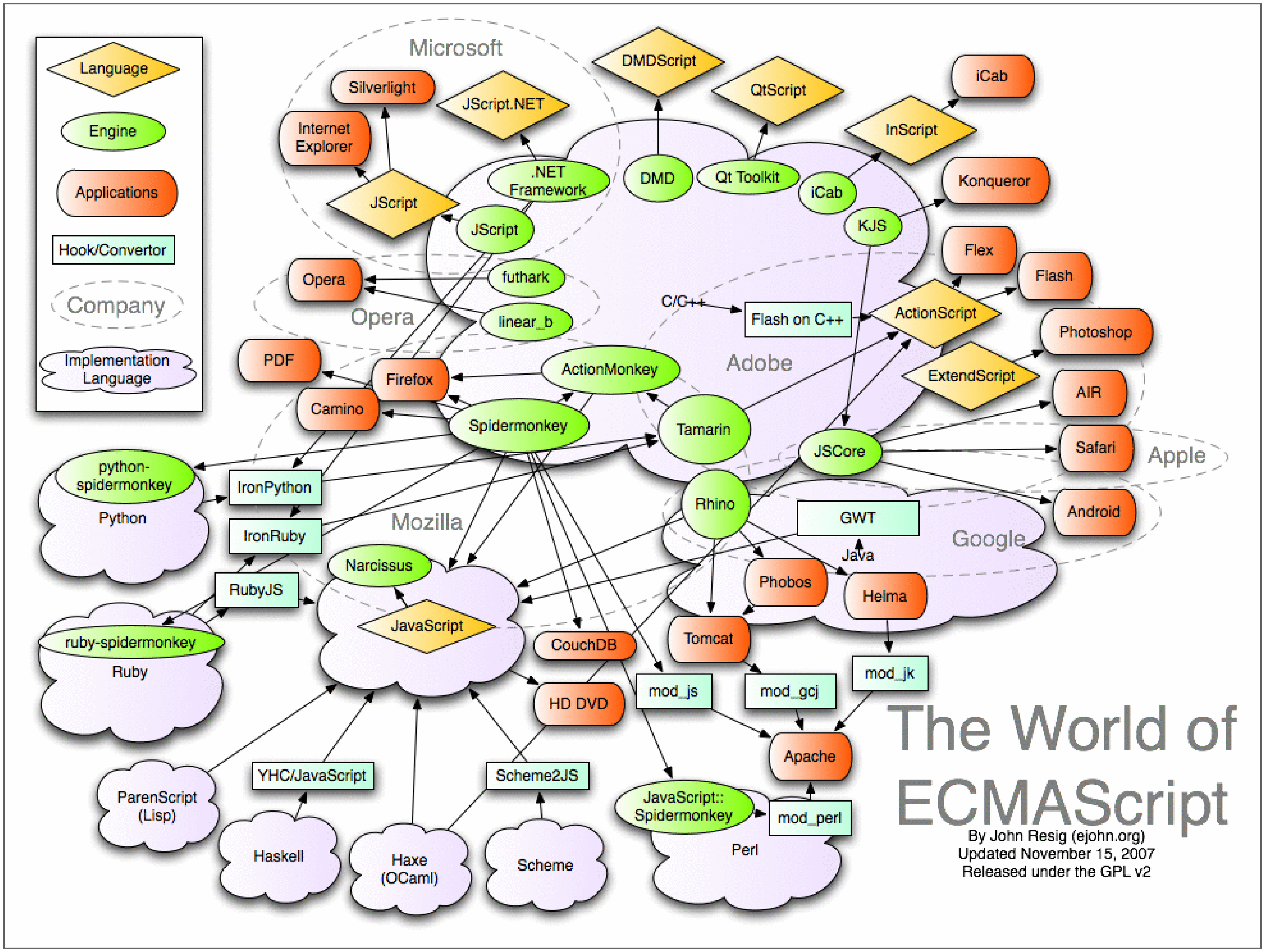
ECMASCRIPT VS JAVASCRIPT

ECMAScript dialects:

- JavaScript,
- JScript,
- ActionScript

JavaScript in browser host = ECMAScript implementation + DOM

ECMASCRIPT WORLD



LEARN FROM OTHER LANGUAGES

- Compile-to-JavaScript Languages
    - CoffeeScript
    - Dart
    - TypeScript
- CommonJS, Node.js
- Python, Scheme, Lua, Java…

ECMASCRIPT PROPOSALS

Currently its version is ECMAScript 5. ECMAScript.next (final name ECMAScript 6) will be finished by 2013.

One of the reasons that the progress of ECMAScript specs is slow, is they care very much about the tons of existing code out there, which mustn't break.

ECMASCRIPT HARMONY

ECMAScript Harmony is a superset of ECMAScript.next.

Contains features not ready or high-priority enough for ECMAScript.next

STRAW-MAN PROPOSALS

This ‘strawman’ namespace is intended to contain proposals for the “ES-Harmony” language that are not yet approved harmony proposals
-- stawman wiki

REST PARAMETER

using 'arguments' object:

```
function format(pattern /*, rest */) {
  var rest = Array.prototype.slice.call(arguments, 1);
  var items = rest.filter(function (x) { return x > 1});
  return pattern.replace(“%v”, items);
}
format(“scores: %v”, 1, 5, 3); // scores: 5, 3
```

now...

```
function format(pattern, …rest) { // real array
  var items = rest.filter(function (x) { return x > 1});
  return pattern.replace(“%v”, items);
}
```

DESTRUCTURING ASSIGNMENT

for arrays:

```
let [x, y] = [10, 20, 30]; //non-strict matching
console.log(x, y); //10, 20
```

for objects:

```
let user = { name: 'Ann', location: {x:10, y:20} };
let {name:n, location:{x:x, y:y}} = user;
console.log(n, x, y); //'Ann', 10, 20
```

SETS: CONTAINER

A set is in an ordered list of values that cannot contain duplicates.

```
var items = new Set();
items.add(5);
items.add("5");

console.log(items.has(5));    // true
console.log(items.has(6));    // false
```

MAPS/WEAKMAPS

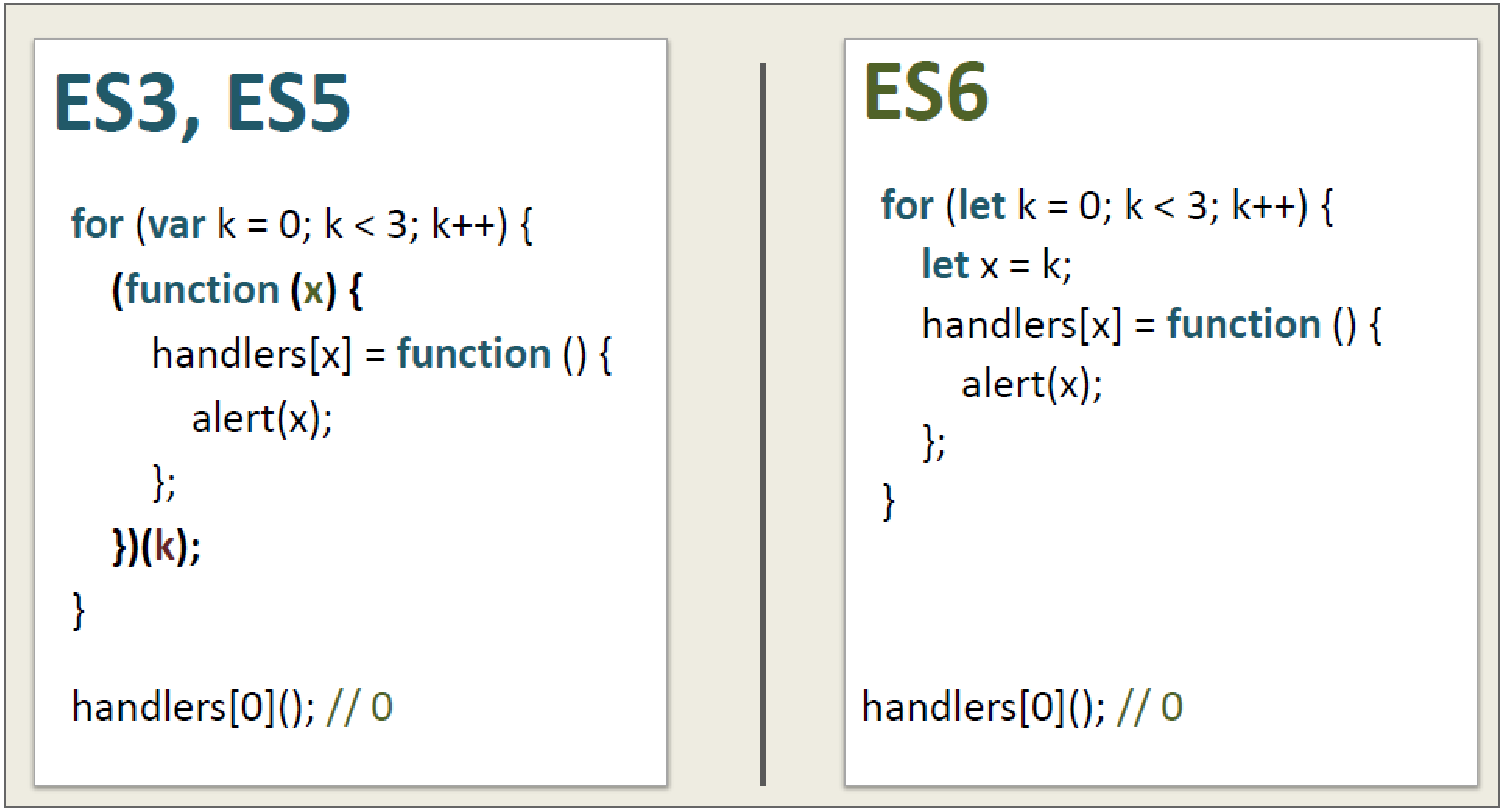
We can't get unique string keys from objects.

```
var map = new Map();
  map.set("name", "Nicholas");
  map.set(document.getElementById("my-div"), { flagged: false });

var name = map.get("name"),
  meta = map.get(document.getElementById("my-div"));
```

A weakmap holds only a weak reference to a key(object only), which means doesn’t prevent garbage collection of that object.

```
var map = new WeakMap(),
  element = document.querySelector(".element");
map.set(element, "Original");
var value = map.get(element);
console.log(value);             // "Original"

// later still - remove reference
element.parentNode.removeChild(element);
element = null;
value = map.get(element);
console.log(value);             // undefined
```

LET: BLOCK SCOPE



ES3, ES5

```
for (var k = 0; k < 3; k++) {
    (function (x) {
        handlers[x] = function () {
            alert(x);
        };
    })(k);
}

handlers[0](); // 0
```

ES6

```
for (let k = 0; k < 3; k++) {
    let x = k;
    handlers[x] = function () {
        alert(x);
    };
}

handlers[0](); // 0
```

CLASS

class is syntactic sugar for a constructor – a function, to be invoked via new.

```
class Person {
  constructor(name) {
  this.name = name;
  }
  describe() {
    return "Person called "+this.name;
  }
}
// Subtype
class Employee extends Person {
  constructor(name, title) {
    super.constructor(name);
    this.title = title;
  }
  describe() {
    return super.describe() + " (" + this.title + ")";
  }
}
```

MODULES

```
module DBLayer {
  export function query(s) { ... }
  export function connection(...args) { ... }
}
import DBLayer.*; // import all
import DBLayer.{query, connection: attachTo}; // import only needed exports
```

GENERATOR

```
function* fibonacci() {
  let [prev, curr] = [0, 1];
  for (;;) {
    [prev, curr] = [curr, prev + curr];
    yield curr;
  }
}

let seq = fibonacci();
print(seq.next()); // 1
print(seq.next()); // 2
```

more examples…

ITERATOR

```
var s = Set([1, 3, 4, 2, 3, 2, 17, 17, 1, 17]);
for (var v of s) {
    alert(v);
}
```

The for-of loop syntax supports custom iteration behavior. When for (V of OBJ) STMT executes, first OBJ is evaluated, then an implicit call to OBJ.iterator() occurs. This method is expected to return an iterator.

```
  Array.prototype.iterator = function* iterator() {
    for (var i = 0; i < (this.length >>> 0); i++)
      yield this[i];
  };
```

FAT ARROWS

```
var fat3 = (a, b) => a + b;
['caption', 'select', 'cite', 'article'].map(word => word.toUpperCase());

//fat arrow function mixin is lexically bound (to global object in this case)
var withCircleUtilsFat = () => {
  this.area = function() {return this.radius * this.radius * Math.PI};
  this.diameter = function() {return this.radius + this.radius};
}
```

- no constructors
- no arguments
- no names
- "this" is lexically bounded

TAIL CALL

```
function factorial(remaining, accumulator){
  if (remaining === 0) {
    return accumulator;
  }
  return factorial(remaining - 1, remaining * accumulator); // <--- tail call
}
```

If remaining == 50000, "RangeError" for ES5. ES6 will has proper tail calls.

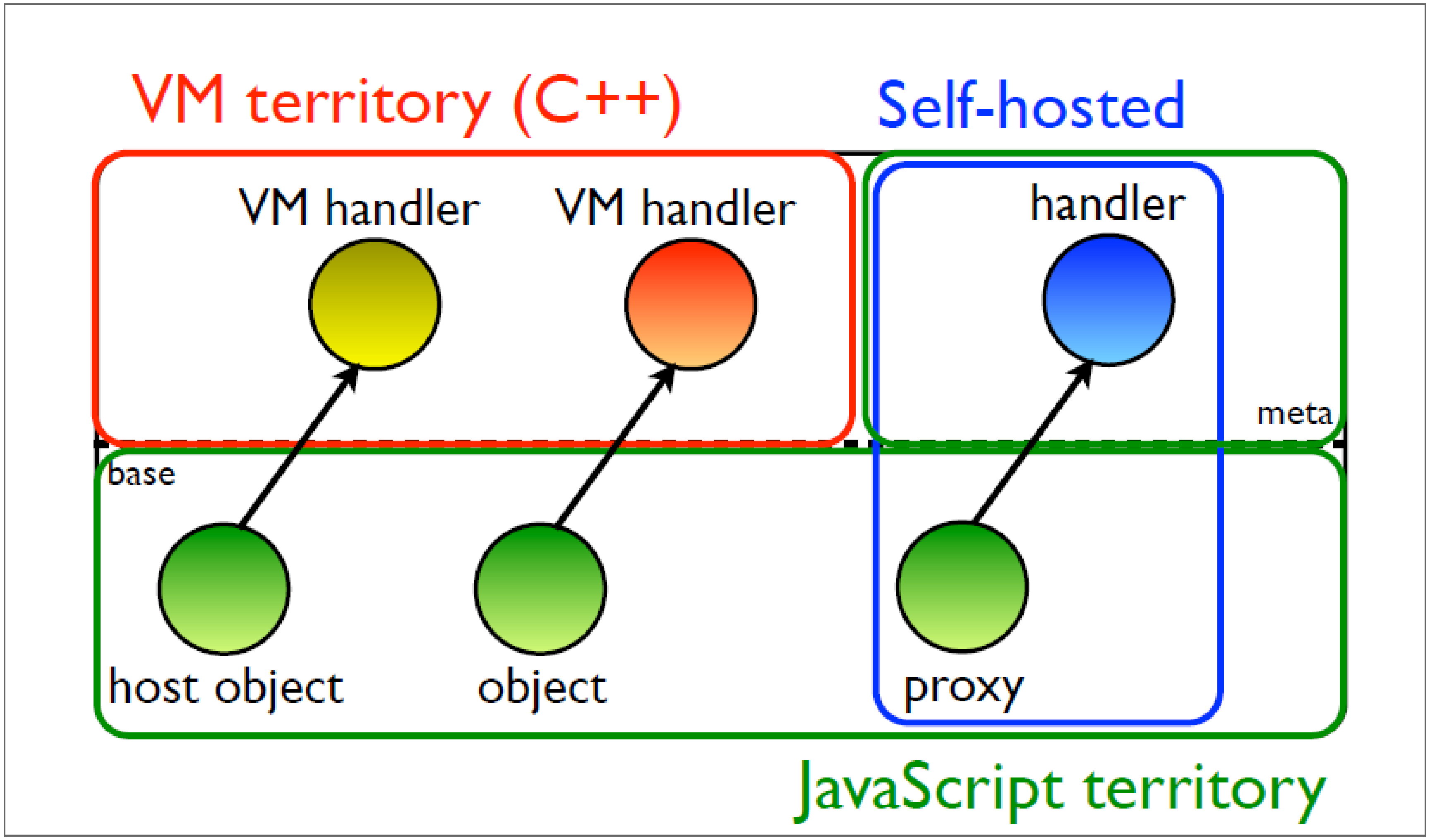
PROXIES(DIRECT PROXIES)

```
var proxy = Proxy(target, handler);

get:                      function(target,name,receiver) -> any
set:                      function(target,name,val,receiver) -> boolean
enumerate:                function(target) -> iterator
keys:                     function(target) -> [string]
apply:                    function(target,thisArg,args) -> any
construct:                function(target,args) -> any
......
```

meta domain... diagram

AOP, remote service proxy ...

PROXY CONT.


VM territory (C++)
Self-hosted
VM handler
VM handler
handler
meta
base
host object
object
proxy
JavaScript territory

THE END