EDA095
JavaScript

Per Andersson

Lund University
http://cs.lth.se/pierre_nugues/

Maj 4, 2017

Innehåll: *JavaScript*

JavaScript

JavaScript:

- syntax som Java
- interpreterat
- dynamiskt typat
- få varningar och run-time fel
- prototyp-funktions-objektorienterat
- enkeltrådat
- versioner: ECMAScript 1, . . . ECMAScript 6 (ECMAScript 2015), ECMAScript 7 (ECMAScript 2016)
- 'polyfill' för att gamla runtile-miljöer ska stödja ny funktionalitet

Typer

Typer:

- `boolean`
- `null`
- `number`
- `string`
- `object`
- `undefined`
- `function`

jämförelse

JavaScript gör typkonvertering vid jämförelse

- `3.14 == "3.14" // true`
- använd alltid `===` eller `!==`
- `typeof` är ofta del av jämförelse

false

Värden som evaluerar till *false*:

- `false`
- `null`
- `undefined`
- empty string ''
- number `0`
- number `NaN`

literal

literaler:

- `3.14`
- `'3.14'`
- `"3.14"`
- `{ }`
- `{text: "hej", done: false, "a-b": null }`
- `[1, false, null, [], { a: 1, b: 2}]`

variabler

- behöver inte deklareras
- dynamisk typning
- "call by reference", objekt kopieras aldrig
- ingen varning om namn deklareras flera gånger
- access till okända namn evaluerar till `undefined`
- function-scope

namnrymd

- functions-scope
- "best practice": deklarera alla variabler överst i funtioner
- minimera golbala namnrymden: ett globalt objekt med alla variabler som "property"
- `var MYAPP = {};`
  `MYAPP.myVar = "some value";`
- Block scope med let och const (ECMAScript 6)

namnrymd

```
let weapon = "sword";
if (true) {
  let offhand = "shield";
}
if(true){
  var head = "helmet";
}

console.log(weapon); // sword
console.log(offhand); // undefined
console.log(head); // helmet
```

namnrymd

```
(function(app) {
  app.DataService = DataService;

  function DataService() {
    this.myList = [...];
  }

  DataService.prototype.getTODOs = function() {
    return this.myList;
  };

  DataService.prototype.addTodo = function(todo) {
    this.myList.push({text: todo.text, done: todo.done});
  };

})(window.app = window.app || {});
```

objekt

- klasser finns inte
- mängd av name/värde-par
- dold länk till ett prototypobjekt, `prototype`
- tilldelning till egna namn/värden
- vid läsning följs prototyp-kedjan
- `prototype` är dynamisk, ändras super-objektet ändras barnen
- `Object.hasOwnProperty(string)` följer inte prototyp-kedjan
- `for (name in myObject)` följer prototypkedjan
- `delete` tar bort namn/värde från objekt, rör ej prototyp-kedjan

Förstärkta typer

förstärkta typer (augmented types)

- modifiera prototyperna:
  - `Object.prototype`
  - `Function.prototype`
  - `Number.prototype`
  - `Array.prototype`

array

- array är objekt (namn/värde)
- dynamisk typning
- `var a = ['zero', 'one';]`
- `var numbers_object = {'0': 'zero', '1': 'one'};`
- `a[1] === 'zero'; //true`
- `a.length === 3; //true`
- `b[10000] = true;`
- `b.length === 10000; //true`
- `numbers[numbers.length] = 'tre';`
  `numbers.push('tre');`
- `numbers.splice(2, 1);`
- `typeof numbers === 'object'//true`

access

```
var obj = { name: "Per", age: 42 }
```

- `var a = obj.name; //Per`
- `var b = obj["name"]; //Per`
- `var c = obj.pnr; //undefined`
- `var d = obj.parent || "unknown"; //unknown`

funktioner

funktioner

- är objekt
- dolda "properties": funktionens kontext och implementeringen
- `var add = function (a, b) { return a + b; };`
- default returvärde: `undefined`
- `this` beror på anropsmönstret
  - method: `obj.foo();`, `this` är objektet
  - funktion: `bar();` `this` är det globala objektet
  - konstruktor: `new MyObject();` `this` är ett nytt objekt
    OBS, default returvärde är `this`
  - apply: `add.apply(null, array);` `this` anges explicit

pil-fuktioner (ECMAScript 6)

```
var fruits = ['banana', 'apple', 'pear'];

fruits.forEach(function (f) { // det gamla sättet
  console.log(f); // banana, apple, pear
});

fruits.forEach(f => { // det nya sättet!
  console.log(f); // banana, apple, pear
});

var numbers = [24, 11, 7, 13, 8];

numbers.sort((a, b) => {
  return a - b;
});

console.log(numbers); // 7, 8, 11, 13, 24
```

closure

closure

- funktioner ser omgivningen
- omgivningen lever så länge funktionen lever

```
var myObject = function () {
  var value = 0;
  return {
    increment: function (inc) {
      value += typeof inc === 'number' ? inc : 1;
    },
    getValue: function () {
      return value;
    }
  };
}( );
```

oops

```
 // BAD EXAMPLE
var add_the_handlers = function (nodes) {
  var i;
  for (i = 0; i < nodes.length; i += 1) {
    nodes[i].onclick = function (e) {
      alert(i);
     };
  }
};
// END BAD EXAMPLE
```

works

```
var add_the_handlers = function (nodes) {
  var i;
  for (i = 0; i < nodes.length; i += 1) {
    nodes[i].onclick = function (i) {
      return function (e) {
        alert(e);
      };
    }(i);
  }
};
```

asynkrona händelser

```
request = prepare_the_request();
response = send_request_synchronously(request);
display(response);
```

```
request = prepare_the_request();
send_request_asynchronously(request, function (response) {
            display(response);
        });
```

promise

```
promise.then(function(result) {
  console.log(result); // "Stuff worked!"
}, function(err) {
  console.log(err); // Error: "It broke"
});
```

serierkoppling

cascade, låt metoder returnera this

```
getElement('myBoxDiv').
        move(350, 150).
        width(100).
        height(100).
        color('red').
        border('10px outset').
        padding('4px').
        appendText("Please stand by").
        on('mousedown', function (m) {
            this.startDrag(m, this.getNinth(m));
         }).
         on('mousemove', 'drag').
         on('mouseup', 'stopDrag'). later(2000, function () {
            this.
                color('yellow').
                setHTML("What hath God wraught?").
                slide(400, 40, 200, 200);
         }).
         tip('This box is resizeable');
```

namngivna argument

långa listor med argument är svåra att läsa...

```
var foo = function(a, b, c, d, e, f, g, h){
  ...
  alert( f );
}
var bar = function(args){
  ...
  alert( args.f );
}

foo(1, 1, 1, 1, 1, 1, 2, 1, 1);
bar({a: 1, b: 1, c: 1, d: 1, e: 1, f: 2, g: 1, h: 1});
```