JS: Good Parts

/

4 good parts?

-

1) first class functions

2) loose typing

3) dynamic objects

4) expressive object literal notation (JSON)

/

What is probably JavaScript’s worse feature

-

The global object

/

Why avoid /\* \*/ style commets on blocks of code?

-

Because those patterns can be found in regular expressions in the code.

/

What is JavaScripts single number type?

-

64 bit floating point

/

Which statements can have an optional label to interact with the break statement?

-

switch, while, for, do

/

Why should variables be declared at the top of a function?

-

Because blocks do not create new scopes. Scope depends upon functions.

/

What are the 6 falsy values?

-

1) false

2) null

3) undefined

4) Empty string ‘’

5) 0

6) NaN

/

What statement is used to iterate over the keys of an object?

Is order guaranteed?

What property should you use to make sure the property is not part of the prototype chain?

-

for in

no

object.hasOwnProperty(variable)

/

What 6 values are produce by typeof?

-

1) ‘number’

2) ‘string’

3) ‘boolean’

4) ‘undefined’

5) ‘function’

6) ‘object’

/

What are two unexpected results of typeof?

-

Both array and null produce ‘object’

/

What to watch out for with the + symbol?

-

It is overloaded for addition and concatenation, so make sure both operands are of the correct type.

/

What is a convenient notation for specifying new objects?

What is interesting about the names and values?

-

Object literals

Name must be strings fixed at compile time (no variables), while values are any expression excluding undefined

/

What do you call a function?

Is the name optional?

-

A function literal.

Yes, it can be used to call itself recursively.

/

What are the five simple types?

What are all other values?

-

numbers, booleans, strings, null, undefined

ojbects

/

How may object expressions be retrieved?

-

string expression enclosed by []

or a valid literal by . notation

/

How do you set defaults?

-

with ||

>> var x = foo.bar || ‘bar’

/

How are type errors guarded against (when try to access property of undefined)?

-

with &&

>> var x = foo && foo.bar

(only last one returns a value if both true, otherwise returns false)

/

How are objects passed?

-

By reference

/

What is delegation?

-

How javascript first checks the object for a property, all the way up the prototype chain until Object, then returns undefined if not found.

/

What does it mean that the prototype relationship is a dynamic relationship?

-

That if you add a property to a prototype, all objects derived from that prototype will be able to see the property, even if they are already instantiated.

/

What are two ways of dealing with unwanted properties in the prototype chain?

-

Use typeof to reject functions

Use hasOwnProperty to make sure isn’t part of the prototype

/

How do you minimize the use of global variables?

-

Have a single global variable for you app.

var MYAPP = {}

/

What is the craft of programming?

-

Factoring a set of requirements into a set of functions and data structures.

/

Are functions objects?

What are they linked to?

What are two other hidden properties?

-

Yes

Function.prototype then Object.prototype

The function’s context (closure) and the code that implements it

/

What is a closure?

-

When an inner function has access to the outer functions variables (the outer context). So even if the outer function has returned (popped off the stack), it’s variables (on the heap) are still available to the inner function as long as it is alive with references to it.

/

What two additional parameters does a function receive?

-

1) this – determined by the invocation patterns

2) arguments – a list of all parameters, so variable size parameter lists can be used.

/

What are the four invocation patterns?

1) Method – where a function stored as a property can be invoked with the . operator and this refers to the object

myObject.increment()

2) Function – where this is bound to the global object.

helper()

3) Constructor – where if a function is invoked with the new prefix, this is bound to the new object.

4) Apply – because javasript is function & object-oriented, functions can have methods. fn.apply() allows us to pass this and a list of arguments.

var Quo = new Quo()

/

Why is the function invocation pattern a mistake?

-

Because inner helper functions don’t have access to an outer object. A work around is to set a variable var that = this so that it is available to the inner function.

ex.

myOjbect.double = function () {

var that = this;

var helper =function() {

that.value = 2 \* that.value;

}

}

/

Why is the constructor pattern the worst of both worlds?

-

It is a weak attempt to make JavaScript (a prototypical inheritance language) appear more like a class based (classical inheritance) language. By convention constructor functions are capitalized, but if the new operator is forgotten bad things happen (a global this?).

/

Do functions always return a value?

-

Yes, if none specified then undefined.

/

Can a try statement have multiple catch blocks?

-

No, so to handle different exceptions you must inspect the name.

/

How are types augmented?

-

By adding methods to Object.prototype. This also works for funtions, arrays, strings, numbers, regular expressions, or Booleans.

/

Does JavaScript provide tail recursion?

-

No

/

What does var myObject = (function() {…})() mean?

-

That the function is being invoked and its results returned to myObject.

/

What is important to remember about inner functions (closures)?

-

They have access to the outer functions actual variables, not copies made at time of exectution.

/

Where should you avoid creating functions?

-

Inside loops.

/

What is a module?

What can we use to construct modules?

-

An interface that hides its state and implementation (private and public variables and methods).

Functions and closures.

/

What are cascades?

-

Having a function return this so that functions can be chained.

getElement().move().width().height()

/

What is currying?

-

When a function is combined with an argument to create a new function.

/

What is memorization?

-

Keeping the results of computations, i.e. in Fibonacci

Use objects and arrays.

/

What is the problem if you forget new in the constructor pattern?

-

This will be bound to the global object.

/

What is the main form of code reuse in classical languages?

What constraints is JavaScript free from?

-

The main construct is classical inheritance.

JS is free from complex class hierarchies that most classical languages have do to the static types system.

JS has more and better methods for dealing with code reuse then class inheritance.

/

How does the prototypal pattern avoid the need to break an application into a set of nested abstract classes?

-

We can just create a base object with an object literal, and then add methods to objects as we need them.

/

What is differential inheritance?

-

The process of customizing a new object and differentiating it from the object upon which it is based.

/

What are the four steps to create a functional constructor with private scope?

-

1) Create an object (literal, new, Object.create(), or function that returns an object)

2) Define private variables and methods as vars.

3) Add methods to that (a new object)

4) Return that

/

Template of functional constructor (no new)

-

var constructor = function(spec, my) {

var that, … // other private instance variables

my = my || {};

// add shared variables and methods to my

// spec is the constructor variables

my.member = value;

// create that

that = {}

// create functions and add to that as the interface

var method = function() {…}

that.method = method;

// return that, the public interface

// where methods have access to my, spec, and that

return that

/

Does JavaScript have real arrays?

-

No, it contains an array like object, that is significantly slower than a real array (where linear memory cells are accessed by offsets).

/

Do array elements need to be the same type?

How do you determine an arrays size?

-

No

array.length

/

What are two ways to add an element to the end of an array?

-

array[array.length] = variable

array.push(variable)

/

-

What is the issue with using delete to remove an element from an array?

What is an alternate method?

What is the issue of using the alternate method?

-

delete array[2] will leave a hole

array.splice(2, 1) is an alternate method that closes the hole.

All elements after the second will be removed and reinstated with a new key, so may go slow for large arrays.

/

How should you iterate over arrays?

Why?

-

for (var i=0; i < array.length; i++) and not for in

This guarantees order and avoids dredging up unwanted properties like length

/

What are two ways to detect arrays?

-

var is\_array = function(value) {

return value && typeof value === ‘object’ && value.constructor === Array;

}

(the above method fails to detect arrays from different windows or frames)

var is\_array = function(value) {

return Object.prototype.toString.apply(value) === ‘[object Array]’

}

/

What is the fastest way to concatenate a bunch of strings.

-

Put them in an array and join.

/

Does JavaScript sort number arrays correctly?

-

No, it sorts lexicographically; you must specify the sort operator.

/

What is the difference between slice and splice?

-

Slice makes a shallow copy of a portion of an array.

Splice removes elements from an array.

/

What is a use of apply?

-

Chaining constructors.

/

What are the three ways to declare a global variable?

-

1) declare outside a function: var x = value

2) declare without a var: x = value

3) attach to global object (window in browser): window.x = value

/

What are 4 of JavaScripts awful parts?

-

1) global variables

2) scope (blocks without block scope, only function scope)

3) semicolon insertion

4) typeof not distinguishing between object, null and array

/

Why should we use === instead of ==?

-

=== checks for object type identity and equality.

== checks for equality and performs auto casting, which is poorly done in javascript.

/

What are 4 other features to avoid or watch out for?

-

1) with

2) eval

3) switch fall through

4) blockless statements, such as with if

/

What is function hoisting?

-

All functions are pulled to the top of the scope in which they are defined. So functions can be used prior to being declared.

/

What are the 6 kind of values in a JSON object?

-

1) object

2) array

3) string

4) number

5) boolean

6) null