JavaScript

#### Staement

Statement are syntax constructs and command that perform action. Statement can be separated with a semicolon.

#### Semicolon

A semicolon may be omitted in most case when a line break exist.

#### Variables

A variable is a “named storage” for data. To create a variable in Javascript, use “let, const, var”

* The “const” declare a constant (unchaning) variable.
* The “var” delaraction is similar to “let”. But they has litte different, “var” has no block scope, with “var” are either fn-scoped and global-scoped. “let” can’t delare the same variable twice in the same scope. If use “var”, can redelare a variable any number of times. “var” can be delared below their use.

Some good-to-follow rules are:

* Use human-readable.
* Stay away from short names like a,b,c.
* Make name maximally description and concise.
* Agree on terns within your team and in your mind.

#### IIFE(immediatelyl-invoked function expression)

IIFE is created and immediately called.

Construction: (function(){  
 //your code  
})();

Ex:   
var a = 10;  
(function () {  
 console.log(a); //undefined  
 a = 5;  
  console.log(window.a); //10  
 var a = 20;  
 console.log(a); //20  
})();

#### Data Types

Primitive  
string, number( -(253 -1) => (253 -1), Infinity, NaN), BigInt( …n), boolean, null, undefined, symbol.

Non-primitive  
object.

* typeof nulll; // “object”
* typeof undefined; // ”undefined”

#### Interaction

alert(title);  
prompt(title,[default]);  
confirm(title);

#### Type Conversions

We have : a ={a:12, b:13}

**Boolean**

* Falsy(false, “”, 0, null, undefined, NaN)
* Truthly( other value)

**Number**

* Number(“123z”); //NaN
* Number(a); //NaN
* Number(undefined) //NaN
* Number(null) // 0
* (“2” – “3”) // -1
* (“2” \* “3”) // 6
* (“2” / “3”) // 0.66666
* +(“2”) // 2, -(“2”) // -2
* +(“”) // 0
* +(“a”) //NaN
* (“3" -2) // 1

**String**

* (String(a)) //[object Object]
* (“2” + “3”) //”23”
* ( 3 + “”) // “3”
* (3 + “2”) // “32”

#### Comparisions

When comparing values of different types, Javascript converts the values to numbers.  
Special case :

* NaN === | == | >= | <= | > | < anything // false
* null == 0 // false
* null === undefined //false ( object != undefined )
* null == undefined //true
* null >= | <= 0 //true

#### Nullish coalescing operator “??”

Precedence : ?? (low precedence) > (? and =)

* Use sequence of “??”, “||” to select the first value from a list that isn’t null/undefined
* “??” returns the first defined value. It forbidden to use it with “||” or “??” without explicit parentheses.
* “||” returns the first truthly value. It doesn’t distinguish between false, 0, an empty string “” and null/undefined. They are all the same – falsy values. If any of these is the first, then we’ll get the second as the result.

Ex:

* ( 0 || 100 ) // 100 ❌
* ( 0 ?? 100 ) // 0 ✅

#### Functions

* A function in Javascript is a value and a function is object.
* A function with a empty return or without it returns undefined

function nothing() {  
return  
}  
nothing() // undefined

* Function Declaration

function name(parameter1, parameter2, … parameterN) {  
 //your code  
}

* Function Expressisons

const/let name = function(paramater1, paramater2, … parameterN) {  
 //your code  
}

* Arrow functions

Const/let name => (paramater1, paramater2, … parameterN){  
 // your code  
}

#### setTimeout and setInterval

**setTimeout**

*let timeId = setTimeout(func|code, [delay], [arg1], [arg2], …)*

* Canceling with *clearTimeout(timeId)*

**setInterval**

*let timeId = setInterval(func|code, [delay], [arg1], [arg2], …)*

* Canceling with clearInterval(timeId)

**Nested setTimeout and Nested setInterval**

Nested setTimeout allows to set the delay between the executions more precisely than setInterval.

Ex:  
let i =1;  
setInterval(function() {  
 func(i++);  
},100);

Ex:  
let i = 1;  
setTimeout(function run() {  
 func(i++);  
 setTimeout(run, 100);  
}, 100);

* The real delay between func calls for setInterval is less than in the code!
* The nested setTimeout guarantees the fixed delay (here 100ms)