JavaScript Language

In the program

1. **introduction**
2. **Types**
3. **functions**
4. **structures**
5. **conversions**
6. **objects**

In the program

1. **introduction**
2. Types
3. **functions**
4. **structures**
5. **conversions**
6. **objects**

JavaScript

- a scripting language, interpreted
  - scripts can be placed in HTML documents
  - the browser has a javascript interpreter
- the javascript code allows
  - act on the properties of the elements of an HTML document
  - manipulate the DOM tree
  - dynamicity of the displayed document

Integrating JavaScript into a web page

we use the element script, in the body or header of the document. 2 possible scenarios:

1. javascript code directly placed in the body of the html file:

```
<script type="text/javascript">
. . .  javascript code
       here </ script>
```

2. javascript code in a separate file specified by the attributesrcof the elementscript:

```
<script src="aFile.js“ type="text/javascript">
</script>
```

examples: ***code in html***–***code in file***

An imperative style of programming

- variables, data types,
- control structures: sequences, conditionals and iteratives modularization: functions
- Objects
- a new syntax
- some differences in the operating rules of new primitive functions to learn

In the program

1. introduction
2. **types**
3. functions
4. structures
5. conversions
6. objects

Primitives types

boolean

2 constants **true**,**false**
operators: negation **!**, and logical **&&**, or logic **|**

number

no clear separation between integers and floats
operators:**+**,**-**,*****,**/**(floating division),**%**(rest of division)
- Infinity,**Infinity**

string

no type ***character*** separated from string, we must consider strings of length 1
the strings are noted between " or ':*"example"*,*'another'*
concatenation operator:**+**
+ object **String** ⇒ many methods

Variables

statement

Variables must be declared using the keyword**var**. A variable must be declared before it can be used.

assignment

The assignment operator is noted**=**.
An uninitialized variable has the value**null**Or**undefined**

variable-types-and-predicates.js

In the program

1. introduction
2. types
3. **functions**
4. structures
5. conversions
6. objects

Functions

function-like value

- the keyword **function** is used to define a function type data,
- we specify in parentheses the **formal parameter**, separated by commas,
- the body of the function is denoted between braces,
- the return value of a function is specified by **return**,
- **return** it's not mandatory (⇒ return value =**undefined**)

```
function(param1,param2,...){
    ...function body return
    expression; }
```

Functions

Calling and naming

a function is called by specifying the **effective parameters** in parentheses

we can name a function by defining a variable whose value is of type function

Example

Functions

```
Function example (x, y) {// eqv to:
 var value = x+y; // var example = function(x,y) {
Return 2* value + 1;
}

example (2,3);   // equals 11
```

Functions

```
Function example (x, y) {// eqv to:
 var value = x+y; // var example = function(x,y) {
Return 2* value + 1;
}

example (2,3);   // equals 11
```

In the program

1. introduction
2. types
3. **functions**
4. structures
5. conversions
6. objects

In the program

1. introduction
2. types
3. functions
4. **structures**
5. conversions
6. objects

Sequences and blocks

Sequence

A sequence of instructions is constructed by separating them with a “**;**”.

instruction block

A block of instructions in sequence is noted between braces.

A block of instructions is an instruction.

Conditional structure

```
if (condition){
    instruction sequence iftrue
}
else{
    instruction sequence iffalse
}
```

```
var collatz = function(i){
if (i %2 == 0){
return i/2;
    }
    else{
        return 3*i+1;
    }
}
```

the part **else** it's not mandatory

false, *0*, *""*, **Nope**, **null**, **undefined** are worth **false**, everything else is worth**true**

Iterative structure

```
for (var i=lower;i<max;i=i+1){
    instruction sequence
}
```

//*i=i+1* is also written *i++*

```
var sumIntegers = function(maxbound) {
    var sum = 0;
    for(var i = 0; i < limitMax; i=i+1) { sum = sum + i;
    }
    return sum;
}
```

```
sumIntegers (100);                    // sum is 4950
```

Iterative structure

```
while (condition){
    instruction sequence
}
varsumNumbers =function(not) {
   varresult = 0;
   while(n > 0) {
      result = result + (n % 10); n =  Math.floor(n/10);
   }
   return    result;
}
sumDigits (12345);                    //   worth   15


do{
    instruction sequence while
}   (condition)
```

In the program

1. introduction
2. types
3. functions
4. **structures**
5. conversions
6. objects

In the program

1. introduction
2. types
3. functions
4. structures
5. **conversions**
6. objects

Type conversion

- javascript is (very) flexible on the notion of typing.
- javascrit "automatically" applies certain type conversions on values when the context requires it:
  - to type **boolean**
  - to the type **string** to type **number**

- affects the notion of equality

Example

Conversion to Boolean and string

```
varBooleanValue =function(val) {
    if(val) {                          // in this context expected boolean value
        return"'"+val+"' is converted to true";                    // expected string
    }
    else{
        return"'"+val+"' is converted to false";
    }
}

BooleanValue("abcd");    // -> 'abcd' converts to true
  BooleanValue("");          // -> '' converts to  false


varx;
BooleanValue(x);                          // -> 'undefined' is converted to false
 x=0;
BooleanValue(x);                          // -> '0' is converted to false
x=1;
BooleanValue(x);                          // -> '1' is converted to true
```

Convert to numbers

- a string whose characters represent a number is converted to that number
- NB: in an expression with the operator + the conversion to string wins

- **NaN**: ***Not a Number***
- conversion value for any expression that cannot be converted to a number
- can test itself with function **isNaN**.

```
"12.5"*3; //   -> 37.5
"99"-5; //   -> 94
"99"+5  // -> "995"
"two"*3; //   -> NaN
isNaN("two"*3); //   -> true
```

ParseInt and ParseFloat

- convert a string to a number (integer or float)
- only the first number in the string is returned, other characters (including matching numbers) are ignored
- if the first character cannot be converted to a number, the result will be **NaN**
- leading spaces are ignored

```
parseFloat("1.24");                    //    1.24
parseInt("42");                        //    42
parseInt("42 is the answer");          //    42

parseInt("   42istheanswer");          //    42
parseInt("42 is the answer");          //    42
parseInt("42 43 44");                  //    42
parseInt("answer = 42");               //    NaN
```

Strange equalities

Because of the conversion, in some cases values of different types may be considered equal.

```
1  ==  "1"          //   ->  true
10  !=   "10"       //   ->  false
1  ==  "A"          //   ->  false
0  ==  false        //   ->  true
"0"==false          // -> true /!\ while "0" converts to true
```

The operator **===** tests both type and value (negation **!==**).

```
1  ===  "1"          //   ->  false
0  ===  false        //   ->  false
10  ===  9+1;        //   ->  true

1!=="1";            // -> true
```

In the program

1. introduction
2. types
3. functions
4. structures
5. conversions
6. objects

In the program

1. introduction
2. types
3. functions
4. structures
5. conversions
6. **objects**

objects

- objects have methods (= functions) a
- method is invoked on an object
- we use the "***dot notation***»

example: with the object**String**

```
vars=newString("timoleon"); varsub =          // creation of a String object // sub is
s.substring(2,6); s.charAt(4);                "mole"
                                              //   worth   "l"
s.length;                                     //   worth   8

                        // convert string values to String object
"abracadabra".charAt(2); "abracadabra"        // worth "r"
.substring(4,8);                              //   worth   "cada"
```

Windows and documents

For a document loaded in a browser, 2 object variables are defined by default:

- **Windows** represents the browser window in which the document is loaded.

  The object **windows** is thc basc objcct,

  an object **windows** per tab

- **Document** represents the DOM document loaded in the window.

first interactions with html document

waiting for better...

window.alertdisplays an information popup

window.prompt

displays a dialog box with a text input box results in the text entered

attention:the result is of type**string**, plan conversions with **parseInt**Or**parseFloat**if necessary.

document.write And**document.writeln**

write text to html stream

the written text is interpreted by the browser

attention:clears the contents of the document if the stream needs to be reopened

* **do not use**to modify a document, only during its creation!*(actually don't use it at all...)*

Example 1 ***Example2***