P.O. BOX 6802 DC ARNHEM

JavaScript Labs

Luminis Arnhem B.V.

Westervoortsedijk 73
Postbus 3147
6802 DC Arnhem
The Netherlands

+31 88 586 46 20
info@luminis.eu
www.luminis.eu

DATUM	PLAATS	ATTACHMENTS	PAGINA
9 oktober, 2015	Arnhem		1/5

My world is:

Exercises – JavaScript

Lab 1

Install Browser and your favorite IDE

Install the Google Chrome Browser (recommended) on your machine (if it is not already there) and make it your default browser.

Install your favorite IDE. Suggestions are:

- Web Storm
- Visual Studio Code

Get familiar with JavaScript

Create two files:

    `Lab 1.html`

And

    `Lab 1.js`

a. In `Hello World.js`, write a JavaScript function that prints “Hello World” in the console.
b. In `Hello World.html`, make a reference to `Hello World.js` and make sure the function is called.
c. Open `Hello World.html` in Chrome, open the console (F12) and observe the result.
d. Declare a variable: `person`
e. Assign an object literal to `person` that holds your name and age
f. Print it in the console
g. Create a function that behaves like a C# / Java Constructor. Call it Person, with a capital A. Instantiate a new Person with a name and an age. Print the newly created object to the console.
h. Extra: Give the Person a function that prints “Hello from <name>” and call it. It should behave exactly like a Java or C# method!

Lab 2 – Type system

Questions

a. Answer the following questions!
b. Use your browser console or a nodejs console to test your answers. There are no solutions provided for this lab. Test them yourself!

1. null === null
2. null == null
3. undefined == null
4. undefined === null
5. NaN == false
6. NaN == NaN
7. 4 == “4”
8. 4 + “4”
9. “3” * “3”
10. 3 == true
11. 3 === true
12. !false == 1

Another little puzzle

Look at the code below.
What would be printed to the console?

```
function puzzle () {
    var a = b = 5;
};

puzzle();
console.log(a);
console.log(b);
```

And another ☺

a. Find a value x for which:
   `+!x === 1`
b. Check your answer by writing a little JavaScript program.
c. Reason about for which different values this holds.
d. Try to see if you can find a way to determine of a variable holds an array!

Lab 3 – Control flow

Some simple excercises

a. Write a function named greaterNum that:
   - takes 2 arguments, both numbers.
   - returns whichever number is the greater (higher) number.

b. Call that function 2 times with different number pairs, and log the output to make sure it works (e.g. "The greater number of 5 and 10 is 10.").

c. Write a function named `helloWorld` that:
   - takes 1 argument, a language code (e.g. "es", "de", "en")
   - returns "Hello, World" for the given language, for atleast 3 languages. It should default to returning English.
   - Call that function for each of the supported languages and log the result to make sure it works.

d. Write a function named `assignGrade` that:
   - takes 1 argument, a number score.
   - returns a grade for the score, either "A", "B", "C", "D", or "F".
   - F is everything below 5. Greater than 6 is a D. Greater than 7 a C etc.

e. Call that function for a few different scores and log the result to make sure it works.

f. Test the functions with grades from 1 to 10 with a for loop

WordGuesser

You'll create a simple word guessing game where the user gets infinite tries to guess the word (like Hangman without the hangman, or like Wheel of Fortune without the wheel and fortune).

g. Create two global arrays: one to hold the letters of the word (e.g. 'F', 'O', 'X'), and one to hold the current guessed letters (e.g. it would start with '_', '_', '_' and end with 'F', 'O', 'X').

h. Write a function called `guessLetter` that will:
   - Take one argument, the guessed letter.
   - Iterate through the word letters and see if the guessed letter is in there.
   - If the guessed letter matches a word letter, changed the guessed letters array to reflect that.
   - When it's done iterating, it should log the current guessed letters ('F__')
   - and congratulate the user if they found a new letter.
   - It should also figure out if there are any more letters that need to be guessed,

- and if not, it should congratulate the user for winning the game.
- Try to solve also with a for in loop

i. Pretend you don't know the word, and call `guessLetter` multiple times with various letters to check that your program works.

j. **Bonus:** Make it more like Wheel of Fortune:
   - Start with a reward amount of $0
   - Every time a letter is guessed, generate a random amount and reward the user if they found a letter (multiplying the reward if multiple letters found), otherwise subtract from their reward.
   - When they guess the word, log their final reward amount.

k. **Bonus:** Make it like Hangman:
   - Keep track of all the guessed letters (right and wrong) and only let the user guess a letter once. If they guess a letter twice, do nothing.
   - Keep track of the state of the hangman as a number (starting at 0), and subtract or add to that number every time they make a wrong guess.
   - Once the number reaches 6 (a reasonable number of body parts for a hangman), inform the user that they lost and show a hangman on the log.

Lab 4 Arrays

Arrays

Write the following functions on arrays:

```
function contains(array, item)
        // returns true if the array contains the item
        // returns false otherwise
function add(array, item)
        // adds the item to the array, if it is not yet included
        // does nothing, otherwise
function remove(array, item)
        // removes the item from the array, if it is included
        // does nothing, otherwise
function sum(array)
        // returns the sum of all elements
```

Rewrite the word guesser from the previous exercise to never use a for loop. Use array functions instead. There are multiple answers possible!

Lab 5 Objects

The Reading List

Keep track of which books you read and which books you want to read!

a. Open the starter folder of lab 5. There is a html page and javascript file. You will be adding logic to this javascript file.
b. Implement the add method. Store the books in an array, where each object describes a book and has properties for the title (a string), author (a string), a release date.
c. Iterate through the array of books. For each book, log the book title and book author like so: "The Hobbit by J.R.R. Tolkien". Print the info to the console
d. It's going to become a bit more difficult. Save the list of books to localstorage.
e. Implement the load function. It should load the books from storage. Make sure release dates are actual dates and it should populate the books are with books sorted on release date.
    - Hints
        - The localstorage api can be found here http://www.w3schools.com/html/html5_webstorage.asp
        - It is okay to populate the global scope for now

- Use a revive function when deserializing
- When sorting arrays use can use a compare function to compare two objects! Look it up ☺.

Lab 6 – functions

Functions

Create a function that concatenates an unlimited amount of words. It should accept any number of strings. Program it defensively, check for types and null or undefined values.

Practice with namespaces

Make the above function part of a namespace. Something like <yourcompany>.<department>

Use a closure to create an object with private data and getters

Use the closure pattern to create a function which creates objects that have private data and public get methods.

You can call this function createPerson, with parameters for firstname, lastname and age. On the returned object there should be methods for getting the name, getting the age and getting the full name. Make sure only the methods are on the returned object! Not the private data.

Call your method twice with different data. Save the results in a variable called person and a variable called person2. Print the results to the console.

If you understand what happens here, you have ventured into the first basics of object oriented JavaScript.

Lab 7 – Object Oriented Javascript

a. Create every class in a separate file and include them in a html page in the right order!
b. Create a BankAccount class. It has an account number and a balance. Create getters and setters for these members.
c. A BankAccount must always be created with a identification number.
d. Create public methods to withdraw and deposit. Make sure you do the proper validations and throw an error if the balance drop below zero.
e. Create a Bank class which holds all the BankAccounts. Give it a method to store new bank accounts. It should accept the initial balance for an account.
f. Create a method on Bank to find an Account given it's number
g. Create a method on Bank to TransferMoney from one account to another.
h. Create a method on Bank to remove an account, given it's number.
i. Create a SavingsAccount class. The SavingsAccount class should override the withdraw method. The balance on a SavingsAccount may drop below zero. Add an extra Boolean to CreateAccount to signal a SavingsAccount.
j. Create some code to test if everything works! You can test everything in the console. You can also some UI to add accounts or place some buttons with window.prompt() behind them. Ask your teacher to explain about some of this if you like a bit of UI! You can also take a look the localstorage solution. There is a bit of UI involved there.
k. Bonus: Add them all to your namespance!

Lab 8 – Typescript

This lab is the same as above but now we use Typescript.:

a. Create the BankAccounts classes in a separate module
b. Create the Bank class in a separate module
c. Create every class in it's own file
d. Type everything, the return values, the parameters etc.
e. Use an interface called IBankAccount that defines the Withdraw and Deposit methods.
f. Define a function interface that describes an AccountFinder function. This AccountFinder accepts an account and returns a Boolean.
g. The FindAccount method should not accept a number, but should accept a function that with the signature of the AccountFinder interface. You should use this function to find the account.
h. All the other functions that use the find method should also accept the AccountFinder.
i. Make proper use of inheritance.
j. Create a program.js file that references the Bank and the Account files. Use this file to test everything.
k. Instruct your compiler to concat all the output to one javascript file.
l. Instruct your compiler to use sourcemaps
m. Add a script tag to your html page that references just that one file.
n. Use the chrome dev tools to observe that one big javascript file get's downloaded. But you can also debug the separate typescript files.

Bonus:

After you are done. Convert it to external modules using requirejs! Verify in your browser everything get's downloaded separately again.