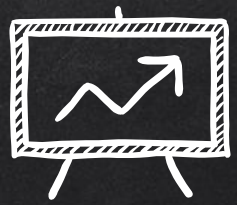


Functions and data structures

Some code - What does it do?

```
var firstName = "Henrik";
var lastName = "Thorn";
var age = 25;
var timesIveCelebrated25 = 7;
var gender = "male";
if(age == 25 && timesIveCelebrated25 == 7) {
        console.log("When did I become this old");
}else {
        console.log("You're still young");
}
if(gender != "male" || gender != "female"){
        console.log("When will this code execute?");
}
```

Some code - What does it do?

```
var persons = ["Maria", "Jennifer", "Peter", "Carl"];

persons.forEach(function(person){
        console.log(person);
});

var i = 0;

while(i < persons.length){
        console.log(persons[i]);
        i++;
}
```

Functions

Functions are the bread and butter of JavaScript programming. The concept of wrapping a piece of program in a value has many uses. It gives us a way to structure larger programs, to reduce repetition, to associate names with subprograms, and to isolate these subprograms from each other.

```
const makeNoise = function() {
  console.log("Pling!");
};

makeNoise();
```

Functions is a way of gathering functionality in order to reuse it multiple times.

```
const power = function(base, exponent) {
  let result = 1;
  for (let count = 0; count < exponent; count++) {
    result *= base;
  }
  return result;
};
```

Most often function will return a new value to us, using the return statement

Functions are being declared, which can be done in several ways.

```
const makeNoise = function() {
  console.log("Pling!");
};

function makeNoise() {
   console.log("Pling!");
}

makeNoise();
```

These two ways of creating a function does the exact same thing.

Scoping

Each binding has a *scope*, which is the part of the program in which the binding is visible.

JavaScript has two scopes – *global* and *local*.

Any variable declared outside of a function belongs to the global scope, and is therefore accessible from anywhere in your code.

Each function has its own scope, and any variable declared within that function is only accessible from that function and any nested functions.

Every country in our world has frontiers. Everything inside these frontiers belongs to the country's scope.

In every country there are many cities, and each one of them has its own city's scope.

The countries and cities are just like JavaScript functions – they have their local scopes.

The same is true for the continents. Although they are huge in size they also can be defined as locales.

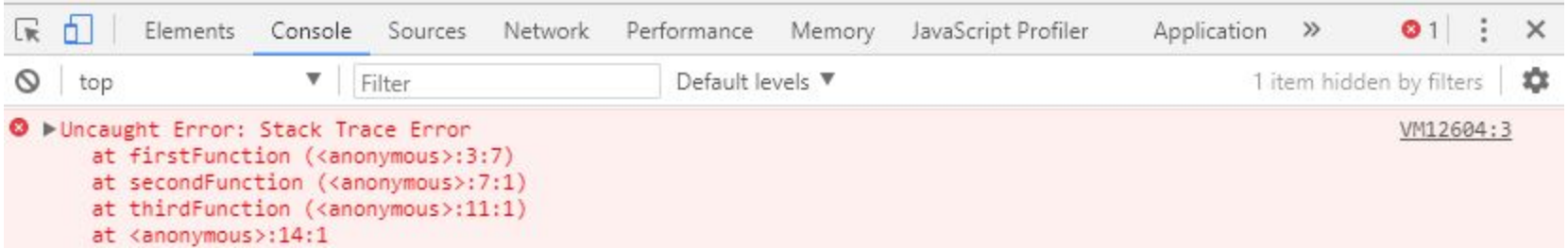
On the other hand, the world's oceans can't be defined as having local scope, because it actually wraps all local objects – continents, countries, and cities – and thus, its scope is defined as global.

```
let x = 10;
if (true) {
  let y = 20;
  var z = 30;
  console.log(x + y + z);
  // → 60
}
// y is not visible here
console.log(x + z);
// → 40
```

Here we see the real difference between let and var!

Call Stack

The call stack is primarily used for function invocation (call). Since the call stack is single, function(s) execution, is done, one at a time, from top to bottom. It means the call stack is synchronous.


Elements
Console
Sources
Network
Performance
Memory
JavaScript Profiler
Application
1
top
Filter
Default levels
1 item hidden by filters
Uncaught Error: Stack Trace Error
VM12604:3
at firstFunction (<anonymous>:3:7)
at secondFunction (<anonymous>:7:1)
at thirdFunction (<anonymous>:11:1)
at <anonymous>:14:1

We use the call stack for debugging, when something does not go according to what we've designed.

The callstack gives us an idea about where the error has occurred and what has been happening up to that error.

Recursion

Stated more concisely, a recursive definition is defined in terms of itself. Recursion is a computer programming technique involving the use of a procedure, subroutine, function, or algorithm that calls itself in a step having a termination condition so that successive repetitions are processed up to the critical step where the condition is met at which time the rest of each repetition is processed from the last one called to the first.

recursion

Alle Billeder Videoer Maps Bøger Mere Indstillinger Værktøjer

Ca. 5.730.000 resultater (0,43 sekunder)

Mente du: ***recursion***

Recursion - Wikipedia

https://en.wikipedia.org/wiki/Recursion ▾ Oversæt denne side

Recursion occurs when a thing is defined in terms of itself or of its type. **Recursion** is used in a variety of disciplines ranging from linguistics to logic. The most ...

Recursion (computer science) · Recursion · Recursion (disambiguation)

The main point of it is that it is defined in terms of itself:

"Recursion: ... for more information, see Recursion."


EXAMPLE

Arrays

Fortunately, JavaScript provides a data type specifically for storing sequences of values. It is called an *array* and is written as a list of values between square brackets, separated by commas.

```
let listOfNumbers = [2, 3, 5, 7, 11];
console.log(listOfNumbers[2]);
// → 5
console.log(listOfNumbers[0]);
// → 2
console.log(listOfNumbers[2 - 1]);
// → 3
```

We use indexes to keep track of the values inside out array!

Array-variables contain some methods that we're able to use in order to control our array.

Two examples being: push() and shift()

In generally a lot of the variables you'll use will have some generic methods you're able to use when you program.

They can all be found in the official documentation.

developer.mozilla.org/bm/docs/Web/JavaScript

The go-to place when you're in doubt

OR

You could just google your problem ;)

Objects

Values of the type object are arbitrary collections of properties.

```
let person = {
  firstName: "Henrik",
  lastname: "Thorn",
  age: 25,
  ssn: "011085-1234",
  birthMark: true
}
```

The difficult thing about objects is to analyse their content!


TIME
FOR A
BREAK

Let's play a bit

Form a group of four people within the room

Which variables, functions, objects and arrays do we need to:

- ✘ Allow a user to log in to a system
- ✘ Purchase a pair of shoes from a webshop
- ✘ Buy a ticket to a Justin Bieber concert online

Pick one of the scenarios and analyse how you would structure the program.

In 15 minutes we're switching roles and you'll present to me.

Initial Idea

You'll have to hand in a list of requirements and a description of your idea to a system that you wish to build.

The description should in text clearly state what you wish to build and the requirements should be done in use cases. Furthermore, you should analyze which program structures you would need in order to build the system.

Deadline: 08/03

Defining your project

From your initial project, you're going to select parts of the system that you're actually going to build. You choose these based on what you wish to build and what is required as part of the exam description.

You're going to hand in a short motivation for the various parts you've selected, which will be further analyzed in the last project.

UML Diagrams

You're going to analyze and create use case diagrams for the parts of your system, which your going to build.

You have to hand in a series of UML Use Case Diagrams showing what the system will be able to do.

Furthermore, you have to hand in a UML Class Diagram that shows us how you're going to implement the system in JavaScript.

Online Quiz

A simple online quiz with questions regarding the syllabus and what we've discussed in class.

You have 60 minutes to complete the quiz, which can be completed from home (or any place you wish).

Mandatory Assignments

You're allowed to hand in the same assignment within you group, but each student has to hand in!

Thanks!

I wish you a nice weekend

If you have any questions

You can find me at @ht.digi at Slack