General information

- The course has four compulsory laboratory exercises.
- You are to work in groups of two people. Sign up for the labs at `http://sam.cs.lth.se/Labs`
- The labs are mostly homework. Before each lab session, you must have done all the assignments in the lab, written and tested the programs, and so on. Contact a teacher if you have problems solving the assignments.
- Smaller problems with the assignments, e.g., details that do not function correctly, can be solved with the help of the lab assistant during the lab session.
- Extra labs are organized only for students who cannot attend a lab because of illness. Notify Per Andersson (`Per.Andersson@cs.lth.se`) if you fall ill, *before* the lab.

Laboratory Exercise 1

The first lab is about the JavaScript language, *objectives*:

1. Understanding how prototype based object orientation works.
2. Develop data structures and functions to be used in later labs.
3. Get some experience using functional style of programming.
4. Get familiar with Node.js.

Background

Later in the course you will develop a web application for ordering in a salad bar, similar to *Grönt o' Gott* at the LTH campus. The customer composes their own salads from a selection of ingredients. Each salad is composed of one foundation, one or more proteins, a selection of extras, and one dressing. For example, a Caesar salad is composed of: chicken breast, bacon, croutons, cherry tomato, lettuce, parmesan cheese, and Caesar dressing.

In addition to handling salad composition, the application should also provide additional information about the salad, for example the price and if it contains ingredients that could cause an allergic reaction.
All ingredients will be imported from a CommonJS module named `inventory.js`. It looks like this:

```
exports.inventory = {
  Sallad: {price: 10, foundation: true, vegan: true},
  'Norsk fjordlax': {price: 30, protein: true},
  Krutonger: {price: 5, extra: true},
  Caesardressing: {price: 5, dressing: true},
  /* more ingredients */
};
```

The properties `foundation`, `protein`, `extra`, and `dressing` indicate which part of the salad the ingredient is to be used for.

Node.js

In this lab you will use Node.js as execution environment. The tool is installed on the linux computers at LTH, but you need to run `initcs` to add it to the path. You can also install it on your own computer, see `https://nodejs.org/`. You start Node.js from the terminal with the command `node`. If you do not provide any arguments, you will start the REPL (Read-Eval-Print-Loop). Write `.exit` to quit the REPL, see `https://nodejs.org/api/repl.html`. This is great for testing stuff, but it is a good idea to save the code for the labs in a file. To execute the JavaScript code in a file, you simply give the file name as argument to `node`:

```
node lab1.js
```

Node.js does not support ECMAScript modules, so we will use CommonJS modules instead. Try the following code (you need to download ./ingredients.js from the course website or github first):

```
const imported = require("./inventory.js");
console.log(imported.inventory);
```

Have you forgotten about the terminal? Check out the introduction from LTH `http://www.ddg.lth.se/perf/unix/unix-x.pdf`.

IDE

Do you want to use an IDE when writing code? I recommend Visual Studio Code, see `https://code.visualstudio.com`. It has great support for JavaScript and TypeScript. We will use TypeScript later in the course which Eclipse has poor support for. TypeScript is JavaScript extended with static typing.

Assignments

1. Study the relevant material for lecture 1-2, see `http://cs.lth.se/edaf90/reding-instructions/lecture-1-2/`.

2. If you are using the linux system at LTH, remember to run `initcs` to add `node` to the path.

3. To get started you can either set the project up yourself or use the starter code from GitHub.

   #### Set up the project yourself

   Create a directory and add a new file named `lab1.js` containing the following code:

   ```
   'use strict';
   const imported = require("./inventory.js");
   console.log(imported.inventory['Sallad']);
   ```

   Download `inventory.js` to the same directory and run your `lab1.js` program:

   ```
   curl -o inventory.js http://fileadmin.cs.lth.se/cs/Education/EDAF90/labs/lab1/inventory.js
   node lab1.js
   ```

   #### Or use the starter code from GitHub

   Clone the GitHub repository, navigate to the lab1 folder, inspect the code and then run it:

```
git clone https://github.com/lunduniversity/webprog
cd webprog/labs/lab1
cat lab1.js
node lab1.js
```

4. In the `inventory.js` file you can find all options for composing a salad. Its structure is good for looking up properties of the ingredients, i.e. `imported.inventory['Krutonger']`. However this structure is not ideal for presenting the options for the customers. Your first assignment is to address this. Print the choices for composing a salad. Use separate lines for foundations, proteins, extras, and dressings. The output should look like this:

   ```
   Foundations: Sallad,Pasta,Sallad + Pasta,Sallad + Matvete,...
   Extras: Avocado,Fetaost,Gurka,Jalapeno,Krutonger,Lime,Majs,Oliver,...
   ```

   hint check out `Object.keys()`, see `https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Object/keys`. How does it compare to using a `for...in` loop instead?
   Note, only the properties that holds true are declared in the objects. Reading an object property that is not declared will evaluate to `undefined`, which is falsy, so there is no need to do explicitly store negative facts, for example {`lactose: false`}.

5. We need a representation for a salad. Create a JavaScript constructor/class named `Salad` for that. You need four properties: `foundation`, `proteins`, `extras`, and `dressing`. Create a constructor and functions to add and remove selections. You may use the ECMAScript 2015 `class` syntax, or the backwards compatible constructor function for this and the remaining assignments.

6. Create an object for a caesar salad:

   ```
   let myCaesarSalad = new Salad();
   /* code to add the ingredients */
   ```

7. Next task is to add additional functionality to the `Salad` class/prototype. Add a function, `price()` to calculate the price. The price is simply the sum of the prices of all ingredients. The computation should be done using functional style, i.e. no loops (`for`/`while`). *hint*: check out `reduce()` and the other functions in `Array.prototype`, see `https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array`. Test your price function with the caesar salad created above, i.e. run `myCaesarSalad.price()`.

8. The restaurant wants to introduce a discount salad. The price is reduced by making the foundation 30% larger, and all other parts 50% smaller. The principle for calculating the price is still the same, but you need to adjust the numbers from the ingredient object to compensate for the changed size (extras are 50% cheaper). Create a `ExtraGreenSalad` constructor/class for this. Most functionality can be inherited from `Salad` You should only change the `price()` function.

9. `let mySalad = ExtraGreenSalad()` creates an extra green salad object. Describe the prototype chain involving these objects. The explanation should include both the `mySalad` and `ExtraGreenSalad` objects.

10. The `ExtraGreenSalad` was not a success. Instead the customers asked for salads with less

lettuce and larger portions of the extras. Create a new class, `GourmetSalad`, for this. In a `GourmetSalad` the customer can specify the size of each ingredient when adding it to the salad. With a size of 1.0 for each ingredients, you get a normal salad. Using size of 1.3 for the foundation and extras with size 0.5 give you an extra green salad. Note, the ingredient objects imported are immutable. *Hint*: if you need to copy an object and add extra properties, use the spread operator in combination with object literals.

Extra assignments, if you have time left.

1. Create an object to manage an order, example of functions needed: create an empty shopping basket, add and remove a salad, calculate the total price for all salads in the shopping basket.

Editor: Per Andersson

Contributors in alphabetical order:
Oscar Ammkjaer
Per Andersson

Home: `https://cs.lth.se/edaf90`

Repo: `https://github.com/lunduniversity/webprog`

This compendium is on-going work.
Contributions are welcome!
Contact: `per.andersson@cs.lth.se`