

## Mandatory exercise 2 (module 2 – client programming)

Hand in the exercise within the deadline in Canvas. The exercise must also be approved - and given a grade - by the teacher (Jostein) at the computer lab – or at jostein's office (A3 094). During the approval you must expect questions about the code. You should be able to explain the code in detail, e.g.:

- Why have you written these lines of code?
- What does these lines of code do in the program?

The approval can be done after the hand-in deadline in Canvas, but before the project period.

*Note 1:* Don't wait too long with the approval – it is easy to forget details in the code.

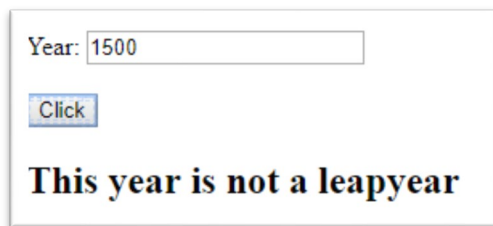
*Note 2:* You can look at examples and read documentation on Internet, but you should write all the code yourself. Then you will be much more able to understand and explain the code.

*Note 3:* The points values for each task indicates the workload and/or level of difficulty/complexity

---

### Task 1 (5p)

Create a program where the user can type in a year and click on a button. When the user clicks on the button, the program should display if the year is a leap year or not.

A screenshot of a web form. At the top, there is a text input field labeled "Year:" containing the value "1500". Below the input field is a button labeled "Click". Below the button, the text "This year is not a leapyear" is displayed in a large, bold, black font.

A leap year can be defined like this:

- If the year can't be evenly divided by 4 (tip: use modulus), it is not a leap year
- Else, if the year can be evenly divided by 100 and not by 400, it is not a leap year
- Else, it is a leap year.

Use CSS to make the user interface look better than the default HTML-look shown above.

### Task 2 (5p)

Explain the code snippets below. Write *short* answers.

a)

I. `let myPoint = {x: 23, y: 12, z: 10};`

II. `let myPoint = {  
    x: 23,  
    y: 12,  
    z: 10  
};`

III. `let myPoint = {};  
myPoint.x = 23;  
myPoint.y = 12;  
myPoint.z = 10;`

b)

```
let myPoint = {};  
myPoint.x = 23;  
myPoint.y = 12;  
myPoint.z = 10;  
  
myPoint.increaseHeight = function() {  
    this.z += 5;  
}  
  
//-----  
myPoint.increaseHeight();  
console.log(myPoint.z);
```

c)

```
function createPoint(x, y, z) {  
  
    let myPoint = {};  
  
    myPoint.x = x;  
    myPoint.y = y;  
    myPoint.z = z;  
  
    myPoint.increaseHeight = function() {  
        this.z += 5;  
    }  
  
    return myPoint;  
}  
  
// -----  
myPoint1 = createPoint(23, 12, 10);  
myPoint2 = createPoint(45, 17, 22);  
  
myPoint1.increaseHeight();  
  
console.log(myPoint1.z);  
console.log(myPoint2.z);
```

### Task 3 (10p)

- a) Change the code in Task 2c) so that *createPoint* becomes a constructor-function instead of a factory-function. Change the name of the function to *Point*.
- b) Can a constructor-function return objects of another type than “pure” JavaScript-objects? Could it for example return an HTML-element or a Date-object? What about a factory-function? Try/discuss...
- c) Change the code in Task 3a) so that we can use a function called *about* to get general info about Point-objects (the text: *Point handles 3D points*). Write the code so that we **don't not need to create a point-object** to use the *about*-function. I.e. the function can be used like this:

```
let info = Point.about();  
console.log(info); //Point handles 3D points
```

### Task 4 (5p)

Date objects in JavaScript deals with dates and time. You can find documentation on the Date-object at w3schools: [https://www.w3schools.com/jsref/jsref\\_obj\\_date.asp](https://www.w3schools.com/jsref/jsref_obj_date.asp)

- a) Is the function Date() a constructor-function or a factory-function?
- b) Do you need to know anything about the code “inside” date-objects in order to use them?
- c) Use a date object to find if 20<sup>th</sup> October 1631 was a Sunday. Be aware that numeric values for weekdays and months, start with 0. That means that January is the 0<sup>th</sup> month, February the 1<sup>th</sup>, ... and December the 11<sup>th</sup> month. Sunday is the 0<sup>th</sup> day, Monday is the 1<sup>th</sup>, and so on. Dates starts with 1.
- d) Are there any functions in Date that can be used without having to create a date-object?

### Task 5 (15p)

Create a program where you can type in a year and select a month. When you click on a button, all the days in the specified month should be displayed in a list where the Sundays are marked in red:

The screenshot shows a web form with the following elements:

- Year:
- Month:  (with a dropdown arrow)
- Show month:

Below the form, a list of days is displayed, each on a new line with a horizontal separator:

- 1  
Friday
- 2  
Saturday
- 3  
Sunday
- 4  
Monday

The number 3 is highlighted in red, indicating it is a Sunday.

Tip:

Alternative 1:

Step 1. Find out how many days there are in the selected month. You can for example create an array with the values: 31, (28 or 29), 31, 30, 31, 30... and retrieve the correct number of days from there. Correct for number of days in February by finding out if the year is a leap year.

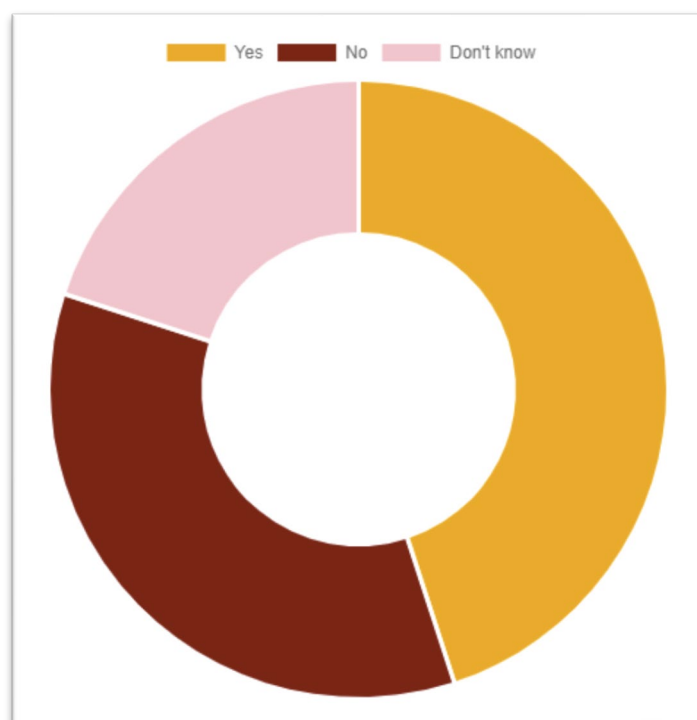
Step 2. Create a loop (from 1 to the number of days) to retrieve each day in the month from a Date-object.

Alternative 2:

Another, and maybe more elegant way is to use the date object itself to find the number of days in a month. You can for example create a loop, retrieve each day, and then end the loop (use *break*) if the date switches to next month.

### Task 6 (10p)

- Open the file *FancyChart.pdf*. This is the documentation of a JavaScript library called *FancyChart*. Is the function *FancyChart* a constructor-function or a factory-function?
- Assume you have downloaded *fancychart.js*. What code must you write to create a pie-chart with the values 23, 56, 130, 44, 30 and where the colors of the pie-slices are red, blue, green, pink and orange? (be aware that *fancychart.js* doesn't exist, so you are not able to test the code).
- Chart.js* is a popular JavaScript library for creating charts on a webpage. You will find some documentation here: <http://www.chartjs.org/docs/latest/> Is the function named *Chart* a constructor-function or a factory-function?
- Assume you would use *Chart.js* to create a doughnut-chart. What code would you write to make a chart with the values 45, 35 and 20 and the labels *Yes*, *No* and *Don't know*:



### Task 7 (10p)

- Open the file *SuperList.pdf* and read the documentation. Is *superlist* a constructor-function or a factory-function?
- Use *superlist* to create a list on your webpage. Test out the two color themes:



- Change *superlist.css* and add a new color theme named *candy*:



- Change *superlist.js* and add a function you can use to retrieve the number of elements in the list (the length of the list). Name the function *getLength*

### Task 8 (15p)

Create a file you call *fancychart.js* and create a constructor-function so that the first part of the code (not the event-handler) in the example in *FancyChart.pdf* will work (you don't need to write code for pie-charts). To help you out, the following code draws a bar-chart in a canvas-element. You are free to copy from this code:

```
var cnv = document.getElementById("myCanvas");
var ctx = cnv.getContext("2d");
ctx.lineWidth = 1;
ctx.clearRect(0, 0, cnv.width, cnv.height);
ctx.fillStyle = "#ffaa00";
ctx.strokeStyle = "#aa0000";

let values = [23, 50, 67, 10];

var numberOfBars = values.length;
var barDistance = 10;
var barWidth = (cnv.width / numberOfBars) - barDistance;

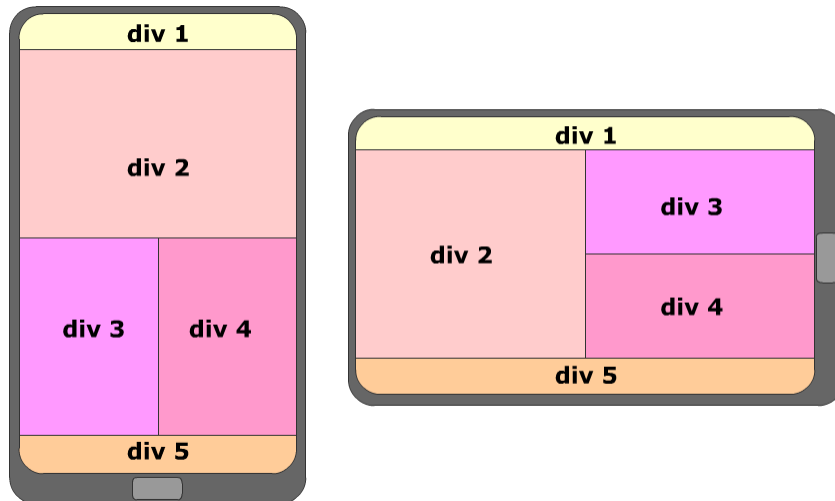
var x = 0;
for (var i = 0; i < numberOfBars; i++) {
    let barHeight = values[i];
    ctx.fillRect(x, 400-barHeight, barWidth, barHeight);
    x = x + (barWidth + barDistance);
}
```

### Task 9 (15p)

Change the constructor-function in Task 8 so that the event-handler in the example in *fancyChart.pdf* works. You don't have to implement events for pie-charts, only bar-charts.

### Task 10 (10p)

- a) Create the layout of divs shown below using a CSS grid. Div 1 and div 5 should have a height of 80px. Use media queries so that the layout changes with the orientation of the screen (portrait or landscape).



- b) Create the layout of divs shown below. Let *header* and *footer* have *fixed* or *sticky* position so that they stay still when the user scrolls the content.

