EXERCISES

Ljupco Shemov

CODE ACADEMY codeacademy.mk

CONTENTS

INTRODUCTION

I'm a believer in learning by doing. Programming exercises are a useful tool to practice and improve your coding skills. After all, practice makes one, if not perfect, at least pretty darn good. One of the best ways to learn, practice, and enhance your skills with this code editor is through basic exercises and practice projects. These will allow you to understand the language better.

Learning to program means learning how to solve problems using code. Conceptually it is not very difficult to write a program that solves a problem that you can solve yourself. The skill you need to acquire is thinking very precisely about how you solve the problem and breaking it down into steps that are so simple that a computer can execute them. I encourage you to first solve a few instances of a problem by hand and think about what you did to find the solution. For example, if the task is sorting lists, sort some short lists yourself. A reasonable method would be to find the smallest element, write it down and cross it out of the original list and repeat this process until you have sorted the whole list. Then you have to teach the computer 1) how to find the smallest element, 2) how to write it down, 3) how to cross it out, and wrap this in a loop. Then continue this task breakdown process until you're confident you know how to write the necessary program.

To make good progress in your programming task, you need to test your work as early and as thoroughly as possible. Everybody makes mistakes while programming and finding mistakes in programs consumes a very large part of a programmer's workday. Finding a problem in a small and easy piece of code is much simpler than trying to spot it in a large program. This is why you should try to test each sub-task you identified during your task breakdown by itself. Only after you're confident that each part works as you expect you can attempt to plug them together. Make sure you test the complete program as well, errors can creep in the way the different parts interact. You should try to automate your tests. The easier it is to test your program, the freer you are in experimenting with changes.

Hope, these exercises help you to improve your coding skills.

VANILLA JAVASCRIPT STOPWATCH

Coding a JavaScript stopwatch is an easy little project you can build in one day even as a beginner. Your stopwatch needs three buttons for user interaction:

1. Start
2. Stop
3. Reset

Play around with some CSS to make it look pretty, and you're all done!

HTML

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Stopwatch | Exercise</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <div class="wrapper">
        <h1>Stopwatch</h1>
        <h2>Vanilla JavaScript Stopwatch</h2>
        <p><span id="seconds">00</span>:<span id="tens">00</span></p>
        <button id="button-start">Start</button>
        <button id="button-stop">Stop</button>
        <button id="button-reset">Reset</button>
    </div>

    <script src="main.js"></script>
</body>
</html>
```

CSS

```
body {
    background: #ffa600;
    font-family: "HelveticaNeue-Light", "Helvetica Neue Light", "Helvetica Neue",
Helvetica, Arial, "Lucida Grande", sans-serif;
    height: 100%;
  }

  .wrapper {
    width: 800px;
    margin: 30px auto;
    color: #fff;
    text-align: center;
  }

  h1, h2, h3 {
    font-family: "Roboto", sans-serif;
    font-weight: 100;
    font-size: 2.6em;
    text-transform: uppercase;
  }

  #seconds, tens {
    font-size: 2em;
  }

  button {
    -moz-border-radius: 5px;
    -webkit-border-radius: 5px;
    border-radius: 5px;
    -khtml-border-radius: 5px;
    background: #ffa600;
    color: #fff;
    border: solid 1px #fff;
    text-decoration: none;
    cursor: pointer;
    font-size: 1.2em;
    padding: 18px 10px;
    width: 180px;
    margin: 10px;
    outline: none;
  }
```

```
button:hover {
  -webkit-transition: all 0.5s ease-in-out;
  -moz-transition: all 0.5s ease-in-out;
  transition: all 0.5s ease-in-out;
  background: #fff;
  border: solid 1px #fff;
  color: #ffa600;
}
```

JAVASCRIPT

Starting JavaScript file.

```
window.onload = function () {
	//Logic here
}
```

HOW TO:

1. First, we need to declare two global variables assign double zero value to them:
    a. ***seconds* = 00**
    b. ***tens* = 00**

2. Second, we need to declare and assign global variables for the following HTML elements:
    a. ***tens (tensElement)***
    b. ***seconds (secondsElement)***
    c. ***button-start (buttonStart)***
    d. ***button-stop ()***
    e. ***button-reset ()***

3. We need to declare a global variable ***Interval***.

4. First for the logic we need to create a function called ***startTimer*** without parameters. Inside the block scope of the function the next steps are required:
    a. First, we need to increment the ***tens*** variable by one.
    b. Second, we need four if statements for the logic of changing the inner text of the ***tensElement*** and ***secondsElement***
        i. In the first if statement we need to check if ***tens*** variable is less or equal to 9. If it is we need to change the ***innerText*** of the ***tensElement*** to the value of ***"0" + tens.***
        ii. In the second if statement we need to check if ***tens*** variable is bigger then 9. If it is we need to change the ***innerText*** of the ***tensElement*** to the value of ***tens.***

iii. The third if statement should have to check if tens is bigger than 99. If it is we need to increment the ***seconds*** variable by one, then we need to change the ***innerText*** of the ***secondsElement*** to the value of ***"0" + seconds***, then set the ***tens*** value to ***0*** and in the end set the ***tensElement innerText*** value to ***"0" + 0***.

iv. And for the last if statement we need to check if the ***seconds*** variable is bigger then 9. If it is we need to set the ***secondsElement innerText*** value to ***seconds***

5. We need to create onclick function for the ***buttonStart*** element that does the following:
   a. First, we need to call the function ***clearInterval()*** (with the ***Interval*** variable as an argument)
   b. Second, we need to assign a new value to the ***Interval*** variable that is equal to the function ***setInterval()*** (with the ***startTimer*** function as the first argument and ***10*** as the second)

6. We need to create onclick function for the ***buttonStop*** element that does the following:
   a. We need to call the function ***clearInterval()*** (with the ***Interval*** variable as an argument)

7. And lastly we need to create onclick function for the ***buttonReset*** element that does the following:
   a. We need to call the function ***clearInterval()*** (with the ***Interval*** variable as an argument)
   b. Assign a new value to ***tens*** variable with the value of ***"00"***
   c. Assign a new value to ***seconds*** variable with the value of ***"00"***
   d. Change the ***innerText*** of the ***tensElement*** with the value of **tens**.
   e. Change the ***innerText*** of the ***secondsElement*** with the value of ***seconds***