Instructions

Homework 4

This assignment deals primarily with functions, objects and DOM interaction in a browser.

You can use your preferred development environment and editor for writing the HTML file which will include all JavaScript code for this assignment. Make sure your editor supports UTF8 Unicode.

To view the HTML and run your code, just open the .html file in a browser of your choice (e.g. Firefox or Chrome).

1. Create a file hw4.html with the following contents:

```
<!DOCTYPE html>
<html>
<head>
  <title>HW4</title><meta charset="utf-8">
  <style>
  body,div { background-color: #eee; font-family: sans; padding: 1em; }
  </style>
</head>
<body>
  <form action="#">
    <input type="text" id="from" /> Celsius
    =
    <input type="text" id="to" /> Fahrenheit
  </form>
  <script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/2.1.3/jquery.min.js">
  </script>
  <script>
  $(function() {
    // FIXME
    $("#from").val("Hello");
    $("#to").val("World");
  });
  </script>
</body>
</html>
```

   Test this file by opening it in a web browser. You should see something like "Hello Celsius = World Fahrenheit".

   Make this temperature converter work by changing the JavaScript code in a way that any keyup event in the first textfield causes the second textfield to show the Celsius temperature from the first textfield converted to Fahrenheit.

   Example: After entering "100" into the first textfield, the second one should display "212".

   The precision for displaying the result is not important as long as the absolute part matches.

   Hint: The jQuery .val() function can both get and set values of form controls like text boxes. The jQuery .keyup() function can be used to attach a function as event handler for the "keyup" event.

(10 points)

26. Extend your temperature converter from question 1 by replacing the static "Fahrenheit" with a dropdown box:

```
<select id="toUnit">
  <option value="c">Celsius</option>
  <option value="f">Fahrenheit</option>
  <option value="k">Kelvin</option>
</select>
```

    You also need to change your code, so it does the right conversion based on the value of the first textfield and the selected unit in the dropdown box.

    Note: You also need to react to changes in the dropdown box.

    Example: After changing the target unit to Kelvin and entering "0" into the first textfield, the second textfield should display "273". Changing the target unit to Fahrenheit should immediately update the second textfield to "32".

    Hint: The jQuery .change() function can be used to attach a function as event handler for the "change" event.

    (5 points)

31. Extend your temperature converter from question 2 by replacing the static "Celsius" with another dropdown box. This means that both the source and the target unit of the conversion are selectable by the user. Despite having 9 different possible conversions in the user interface, you should *not* have 9 different cases in your code.

    Changing either of the units should result in updating the second text field.

    (5 points)

32. Extend your temperature converter from question 3 by making it work both ways, i.e. changing the left left value will update the right field and changing the right field updates the left value.

    Hint: You can either bind different callbacks for each of the two fields or you can access the source of an event in the event handler, e.g.

```
$("#from, #to").keyup(function() {
   if (this.id === "to") ...
});
```

    (10 points)

    This is an example screenshot of a complete solution of question 1 - 4:

33. In the JavaScript part of your code, add a function `randInt(min, max)` that returns and integer between `min` (included) and `max` (included).

```
function randInt(min, max) {
    ...
}
```

You can test your function by executing code directly in a JavaScript web console of your browser or by adding a some test button to your HTML.

Example:

```
> randInt(2,3)
2
> randInt(2,3)
3
```

Hint: Math.random() returns a floating point number between 0 (included) and 1 (excluded).Math.floor()` rounds down to the closest integer.

(5 points)

36. Write a function `createDice` that takes the number of faces and returns a of function for rolling a single die with.

Example:

```
var d6 = createDice(6);
d6();  // returns 1,2,3,4,5 or 6
d6();

var d20 = createDice(20);
d20(); // return 1, 2, ..., 19 or 20
```

Hint: In Haskell you could use currying, in JavaScript you can return a closure.

(5 points)

37. Create a prototype for six-sided dice objects. Each Dice object stores its current value (the current number shown on the top) and the corresponding HTML representation. Rolling a Dice updates both the value and the representation.

```
function Dice() {
    // creates a new instance
    // should assign the "this.el" property to be a jQuery-wrapped
    // <span> HTML element with the CSS class "d6" using one of the
    // icons as inner text.
    this.icons = "□□□□□□";
    this.el = ...
    ...
}

Dice.prototype.render = function() {
    // updates the HTML represenation by changing the icon in the
    // jQuery-wrapped element in the "this.el" property
```

```
    ...
}

Dice.prototype.roll = function() {
    // rolls this die and updates the represenation
    ...
}
```

Additionally, add a CSS definition for the CSS class "d6" to the style tag in the HTML. It should increase the font-size and possibly add other styles as well.

Example: Typing the following commands into the web console, should first show a dice in the browser window and then change it:

```
> var d = new Dice();
> $("body").append(d.el);
(shows a dice in the body)
> d.roll()
(rolls it and updates the representation)
```

(15 points)

57. Change your implementation of Dice in question 7, so that clicking on the HTML element of the dice will roll it.

    (5 points)

58. Create a new prototype for an UnfairDice that uses inheritance to behave just like a normal Dice object from question 8 but has a 50% chance of rolling '6' but only a 10% chance for each of the other sides.

    Example:

    ```
    > var d = new UnfairDice();
    > $("body").append(d.el);
    (shows a dice in the body)
    > d.roll()
    (rolls it and updates the representation - probably showing 6)
    ```

    (5 points)

59. Implement a minimal "Yatzy" game.

    You can start by adding the following HTML to your file, after the temperature converter:

    ```
    <div>
      <button id="restart">Restart</button>
      <p id="score">Score: _ (_ dice with _ dots each)</p>
      <p id="rolls">_ rolls left</p>
      <div id="dice"></div>
    </div>
    ```

The score paragraph will show the current score. In this minimal Yatzy game, the score is determined by only counting the dice that have a common number of dots and only those number appears the most often, e.g. the score for the five dice [1,1,1,4,4] would be 3 as 3 dice all show the number 1. If multiple numbers appear the same number of times, the higher one has precedence, e.g. the five dice [2,4,4,6,6] would have a score of 12 because of 2 dice with 6 dots each. The order does not matter.

In addition to the score paragraph, the game only allows a limited number of re-rolls on click as shown in the rolls element. You have to create a new kind of dice that decrements this number and prevents a roll if it already reached 0. You can use your Dice prototype from question 8 but you should use inheritance to create a YatzyDice without modifying the original Dice.

Opening the page or clicking the restart button will create five of these dice objects, add their elements to the dice element and reset the number of available re-rolls to 5.

In contrast to the regular Yahtzee game, this game only allows re-rolling of individual dice. Every time a dice is rolled, the total number of available re-rolls is decreased and the score for all five current dice gets calculated (ignoring the previous score).

(25 points)

This is an example screenshot of a complete solution of question 5 - 10:
Your solution might have a different style and layout but should include the same elements.

60. In the JavaScript part of your HTML, add a function memoize that takes a one argument function f(x) and returns a function that produces the exact same results as f but stores the return values, so it does not compute the same argument more than once.

    Example:

    ```
    > function square(x) { alert("computed sq " + x); return x * x; }
    > msq = memoize(square)
    > msq(2) + msq(2) + msq(3)
    computed sq 2
    computed sq 3
    17
    ```

    Hint: wrap the original function in a closure.

    (10 points)

61. **(Extra credit)**

    Make your solution for question 11 work for simple recursive functions (named functions that call themselves with that same name).

    Example:

    ```
    > i = 0
    ```

```
> var fib = function f(n) { i++; return n < 2 ? 1 : f(n-1) + f(n-2); }
> memoize(fib)(4)
5
> i
5
> fib(30)
1346269     (after waiting a second)
> memoize(fib)(30)
1346269     (immediately)
```

Hints:

Calling `toString` on a function returns its source code.

`eval` takes a string of JavaScript and evaluates it.

(10 points)