Einführung in Web Technologien (WS 22/23)

Übungsblatt 03 – JavaScript

Aufgabe 1 – Plenum

Welche Ausgaben werden durch den jeweils gegebenen JavaScript-Code in der Konsole ausgegeben werden und welche Begründung gibt es hierfür?

https://codesandbox.io/s/webtech-ubung-3-plenum-1-bkt6nk

	Code	Ausgabe	Begründung
1)	var a = 5 + 3 + "4"; a;		
2)	4 < 5 < 6;		
3)	6 < 5 < 4;		
4)	6 == "6";		
5)	false == 0;		
6)	null == 0;		
7)	null < 1;		
8)	6 === "6";		
9)	undefined || "hallo";		
10)	"hallo1" && "hallo2";		
11)	("hallo" && "hallo") === ("hallo" || "hallo");		
12)	!("hallo1" && "hallo2");		
13)	!("4"+3==7);		
14)	!(undefined && undefined);		

15)	var a = (4 || 3 || 5);
	var b = (4 && 3 && 5);
	var c = (4 && (3 || 5));
	a+b+c;		

Aufgabe 2 – Plenum

a) Was passiert bei diesem Code?

```
alert("Alarm");
var alert = function (msg) {
        document.write(msg);
}
alert("Alarm");
```

b) Welche Ausgabe erstellt diese Funktion? Erläutern Sie was in x_var() geschieht. Welche Schwierigkeiten hat diese Definition?

```
function x_var() {
  var i = 5;
  for (var i = 0; i < 10; i++) {
    console.log(i);
  }
  console.log(i);
}

x_var();
```

c) Welche Ausgabe hat folgender Code? Was ist der Unterschied zwischen x_var() und x()? Welche Schwierigkeiten

```
function x() {
  i = 5;
  for (i = 0; i < 10; i++) {
    console.log(i);
  }
  console.log(i);
}

x();
```

d) Was geschieht, wenn nur „var i = 5;“ aus x_var() zu „let i = 5;“ geändert wird und warum?

e) Sei x_let() eine Funktion, mit dem Inhalt von x_var(), bei der statt „var“-Deklarationen stattdessen „let“ verwendet werden? Welche Ausgabe liefert x_let()? Begründen Sie ihre Antwort.

f) Const verbietet das Verändern eines Wertes durch den „=“ Operator. Warum ist dieser Codeschnipsel trotzdem gültig?

```
function x_const() {
  const i = 5;
  for (let i = 0; i < 10; i++) {
    console.log(i);
  }
  console.log(i);
}

x_const();
```

g) Welche Ausgabe liefert x_func()?

```
function x_func() {
  var i = 5;
  function y() {
    for (var i = 0; i < 10; i++) {
      console.log(i);
    }
  }
  y();

  console.log(i);
}

x_func();
```

Aufgabe 3 – Gruppe

Betrachten Sie die Datei stock.js. Diese enthält ein Modul Stock (Lager), geschrieben mit dem Revealing Module Pattern, das die Verwaltung des Lagerinhalts Ihrer Mensa übernimmt. Das Modul erlaubt das Einsehen des Lagerbestands und das Hinzufügen und Löschen von Lagerinhalten. Dazu wird der Lagerinhalt in einer privaten Variable _stock gespeichert und eine Funktion zum Zurückgeben dieser Variable bereitgestellt. Um den Lagerbestand zu verändern, gibt es eine Funktion zum Hinzufügen von Inhalten und eine zum Herausnehmen von Inhalten. Keine Funktion unterstützt die Verwendung von negativen Mengenangaben (es kann also nicht durch negatives Hinzufügen etwas entfernt werden).

Ihre Aufgabe ist es nun, eine Erweiterung für dieses Modul – ebenfalls mit dem Revealing Module Pattern – zu schreiben. Die Erweiterung soll ein Submodul Display hinzufügen, das eine Funktion **displayStockVolume** bereitstellt, welche die Inhalte Ihres Lagers als Tabelle formatiert und den HTML5-Code als String zurückgibt. Jede Tabellenzeile soll aus zwei Zellen bestehen. In der ersten Zelle soll die Menge und die Einheit stehen, wobei die Einheit Stück nicht ausgeschrieben werden soll. Die zweite Zelle soll den Namen des Lagerpostens enthalten. Unten sehen Sie, wie der generierte String formatiert aussehen kann. Nutzen Sie hierfür die Datei display.js.

In der Datei exampleExtension.js finden Sie eine beispielhafte Implementierung einer Erweiterung für das Stock-Modul.
Um Ihre Implementierung zu testen, nutzen Sie die Datei index.html, und rufen Sie die Funktion testDisplay über die Konsole auf.

```
<table>
    <tr>
        <td>Menge Einheit</td>
        <td>Zutat</td>
    </tr>
    ...
</table>
```