Sheet 2

1- How many alert dialogs will the following JS generate, and what will be displayed in each of them?

```
1- var x = "10";
2- function f(){
3-    var x = "4";
4-       alert(this.x); print x=10
5-       function g(){alert(x);} print x=4
6-    g();
7- }
8- f();
```

2- Below you will see some snippets of HTML from a Web page. Fill in the body of the JS function *changeColor* so that the color of the text changes in response to the selection made in the menu

```
1- <style type="text/css">
2- .a {color:red;}
3- .b {color:green;}
4- .c {color:blue;}
5- </style>
6- ...
7- <div   id="colorText">Select   below   to
   change the color of this text</div>
8- <select
   onchange="changeColor(this.value)">
9- <option value="a">Red</option>
10-   <option value="b">Green</option>
11-   <option value="c">Blue</option>
12-   </select>
13-   <script type="text/javascript">
14-   //<![CDATA[
15-   function changeColor(value) {
        // Answer:
documnent.get elementbyid("colorText").classname=value
16-   }
17-   //]]>
18-   </script>
```

if i need to change color to red from the first time i click it :
<option value=" "> select your color </option>

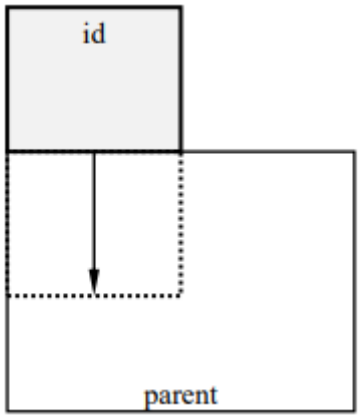
3- Repeat question (2) using JQuery.

Enjoy it ☺

4- Write a Javascript function `peekDown` that will animate the appearance of an HTML element within its parent. The function will be invoked as follows:

```
peekDown(id, duration)
```

`Id` is the HTML identifier of an element, and duration indicates how long the animation should take, in milliseconds. `PeekDown` should initially position the given element just above the upper left corner of its parent as shown in the figure below; the element will not be visible because it is entirely outside the bounds of its parent. Then `peekDown` should gradually slide the element down; as it does this, the bottom part of the element will become visible underneath the top edge of the parent. Eventually, duration *ms* later, `peekDown` should stop, leaving the entire element just visible at the top of its parent (dotted line in the figure). The result is an animation where the element "peeks" in from the top of its parent.



Additional notes and requirements:

- In order to get full credit, `PeekDown` must not use any global variables (except for the `PeekDown` function itself) and it must be able to support multiple simultaneous transitions.
- You can assume that the element and its parent have been created already and configured so that the child element is clipped by the boundaries of its parent as described above (CSS "overflow: hidden;"); all you need to do in your code is to move the child to create the animation effect.

- You can assume that neither the element nor its parent uses padding, spacing, or a border.
- You may find the following Javascript functions useful:
    - `setTimeout(funcOrCode, ms);`
    - id = setInterval(funcOrCode, ms);
    - cancelInterval(id);

      setTimeout returns immediately but arranges for funcOrCode to be executed (exactly once) ms milliseconds in the future; funcOrCode can be either a function to invoke or a string to eval. setInterval is similar to setTimeout except that funcOrCode is executed repeatedly every ms milliseconds; setInterval returns a token that can be passed to cancelInterval to stop the repeating execution.