JavaScript Timer Exercises

1. The main difference between these two statements lies in how the function booyah is passed to setTimeout.

- **‘setTimeout(booyah, 2000);’**

  In this line, booyah is passed as a reference to the function, which means that the function will be executed after **2000 milliseconds**.

- **‘setTimeout(booyah(), 2000);’**

  In this case, booyah() is invoked immediately, and its return value (if any) is passed to setTimeout as a callback. If booyah doesn't return a function, this will not work as intended and will not delay the execution of booyah.

2.

- alert(x) will output **6**.
  And
- alert(y(2,3)) will output **6**. (because y(2,3) is effectively the same as myfunc(2, 3), since y has been assigned the myfunc function).

3.

```
// booyah1 is a simple function that alerts "BOOYAH!"
function booyah1() {
  alert('BOOYAH!');
}
setTimeout(booyah1, 2000); // This will alert "BOOYAH!" after 2 seconds.


// booyah2 is a function that RETURNS another function that alerts "BOOYAH!"
function booyah2() {
  return function() {
    alert('BOOYAH!');
  };
}
setTimeout(booyah2(), 2000); // This will also alert "BOOYAH!" after 2 seconds.
```

4. Unobtrusive JavaScript is a way of separating webpages into different parts:
    - Html
    - CSS
    - JavaScript

   The practical application of unobtrusive JavaScript is:

- Improved maintainability: With separation of concerns, code is easier to manage, update, and debug.
- Accessibility: Ensures the website is accessible to as many users as possible, including those who have JavaScript disabled or use assistive technologies.
- Performance: Reduces the amount of code delivered to the client, which can improve page load times.
- Progressive Enhancement: Start with a baseline of essential features, then add more functionality for browsers that can handle them.