

IT 2320 – Interactive Internet Programming

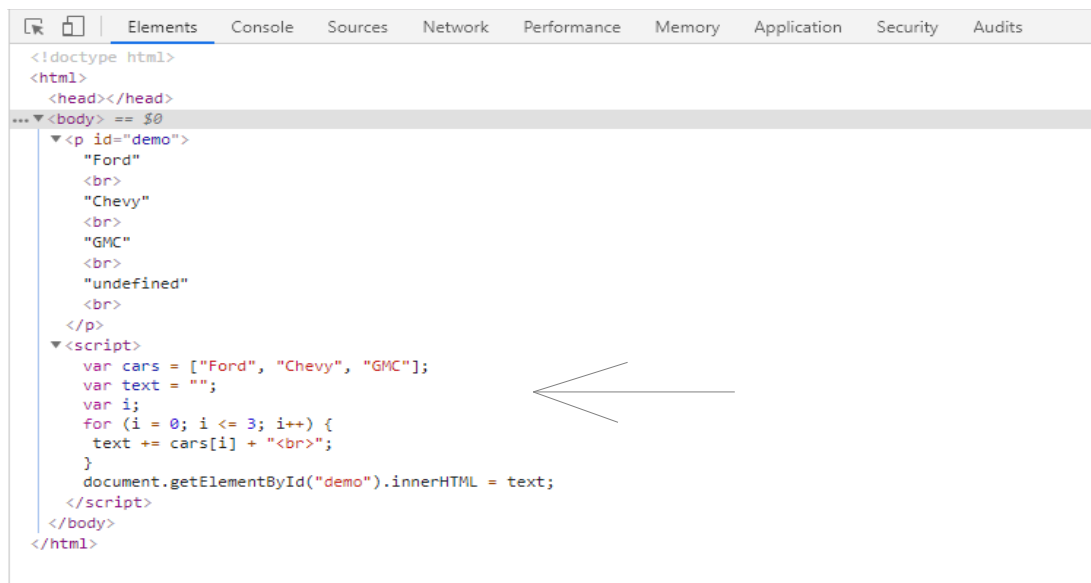
Lab 7

15 Points Total

1. In order to open developer tools in most browsers, we can use what function key? (1 point)

Answer: CTRL + SHIFT + I

2. We can view the values of the variables defined in our JavaScript code using Developer Tools in Chrome. Show an example (paste an image) (2 points).



3. In the following example, is arearect a local or global variable given that we have not declared strict mode? Explain two ways that we could fix this. Read the example carefully! (3 points)

```
var areaRectangle = function (width, height) {  
  
  var areaRect = width * height;  
  
  arearect = parseFloat(areaRect.toFixed(2));  
  
  return areaRect; }  

```

Strict mode is an optional mode for JavaScript that alters the semantic requirements for JavaScript code. Basically, it uses different semantics for the JavaScript code that may or may not be supported by browsers, and causes previously ignored errors to change to throw errors.

Var arearect should be a local variable to the function created above, but the value of arearect is stored in the global variable areaRectangle.

Personally, I would simplify the function by making arearect a global variable outside of the function and have it hold the value of width * height. Then, I would have areaRect hold the parsed value of arearect.

The problem with the code above is it is returning the value of areaRect, instead of the value of arearect! See below for my ideas.

```
var areaRectangle = function (width, height) {  
  
    var areaRect = width * height;  
  
    arearect = parseFloat(areaRect.toFixed(2));  
  
    return arearect; }
```

or

```
var arearect = width * height; //hold value in arearect, use as parameter for function  
  
var areaRectangle = function (arearect) {  
  
    var areaRect = parseFloat(areaRect.toFixed(2)); //hold parsed arearect value  
  
    return areaRect; } //return the parsed value
```

NOTE: I use two separate variables because they may hold two different types of variables (I.e int vs double or float).

4. Use the console to log what will be output to the command prompt along with the index that being used to access each element. Should how you corrected the problem by outputting only elements that have been assigned values (4 points).

```
<!DOCTYPE html>
```

```
<html>
```

```
  <body>
```

```
    <p id="demo"></p>
```

```
    <script>
```

```
      var cars = ["Ford", "Chevy", "GMC"];
```

```
      var text = "";
```

```
      var i;
```

```
      for (i = 0; i <= 3; i++) {
```

```
        text += cars[i] + "<br>";
```

```
      }
```

```
      document.getElementById("demo").innerHTML = text;
```

```
    </script>
```

```
  </body>
```

```
</html>
```

ORIGINAL OUTPUT:

Ford
Chevy
GMC
undefined

Change the code to be the following and you will be able to remove the undefined shown above.

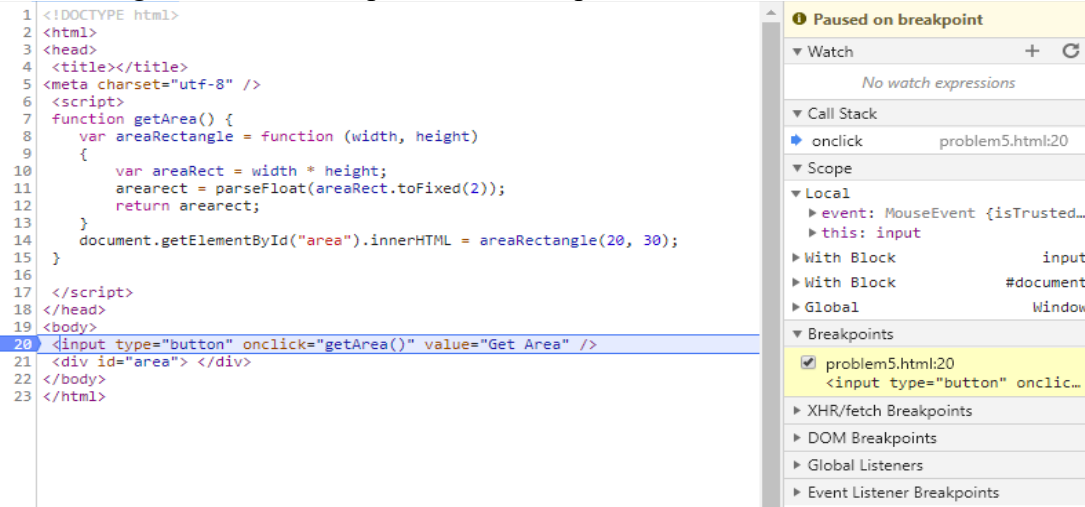
```
var i;  
  
for (i = 0; i < 3; i++) { //removed the = in i <= 3  
  
text += cars[i] + "<br>";
```

5. In the following code, the function assigned to areaRectangle is defined within a function. Move the function to the appropriate place. Show in the Call Stack how the method is now available. Use a BreakPoint to demonstrate the method running correctly. For credit, paste a screenshot showing the method running with a Break Point...the method name should appear in the Call Stack (5 points).

Fixed version of code. Placed the function where it belongs.

```
1 <!DOCTYPE html>  
2 <html>  
3 <head>  
4 <title></title>  
5 <meta charset="utf-8" />  
6 <script>  
7   function getArea() {  
8     var areaRectangle = function (width, height)  
9     {  
10       var areaRect = width * height;  
11       arearect = parseFloat(areaRect.toFixed(2));  
12       return arearect;  
13     }  
14     document.getElementById("area").innerHTML = areaRectangle(20, 30);  
15   }  
16 </script>  
17 </head>  
18 <body>  
19 <input type="button" onclick="getArea()" value="Get Area" />  
20 <div id="area"> </div>  
21 </body>  
22 </html>
```

Code running in web browser, paused at break point before function is executed.



```
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <title></title>
5 <meta charset="utf-8" />
6 <script>
7   function getArea() {
8     var areaRectangle = function (width, height)
9     {
10       var areaRect = width * height;
11       arearect = parseFloat(areaRect.toFixed(2));
12       return arearect;
13     }
14     document.getElementById("area").innerHTML = areaRectangle(20, 30);
15   }
16 </script>
17 </head>
18 <body>
19 <input type="button" onclick="getArea()" value="Get Area" />
20 <div id="area"> </div>
21 </body>
22 </html>
```

Paused on breakpoint

Watch

No watch expressions

Call Stack

onclick problem5.html:20

Scope

Local

event: MouseEvent {isTrusted...}

this: input

With Block input

With Block #document

Global Window

Breakpoints

problem5.html:20
<input type="button" onclick=...

XHR/fetch Breakpoints

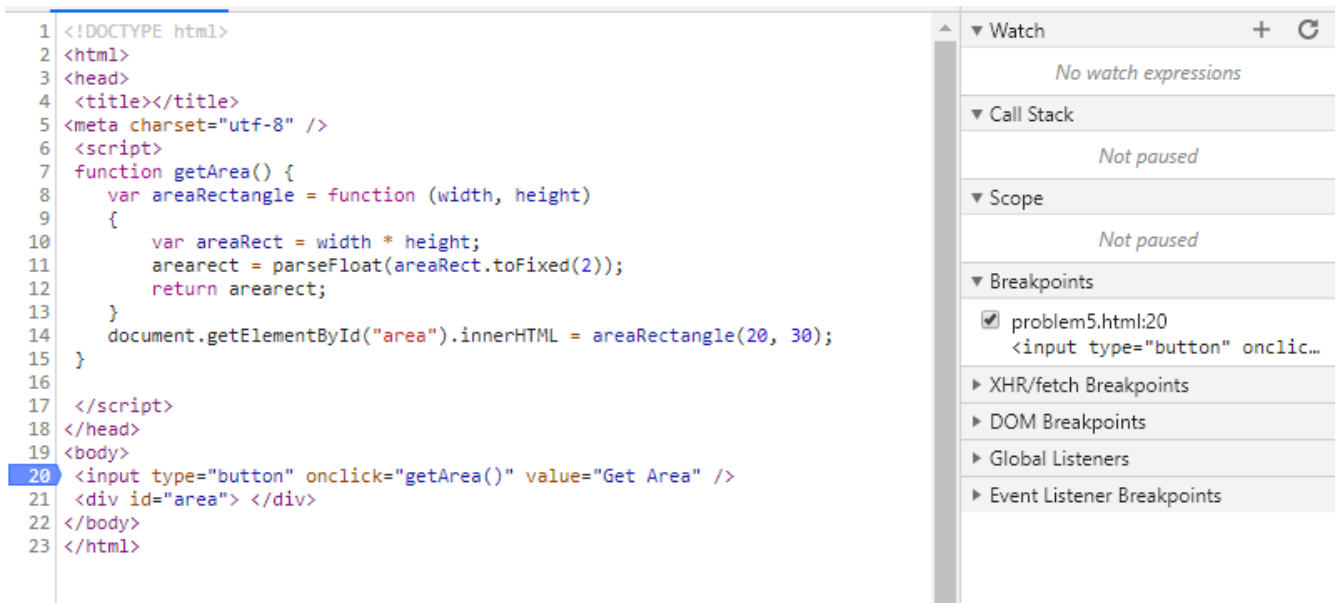
DOM Breakpoints

Global Listeners

Event Listener Breakpoints

Final result output with source code

Get Area
600



```
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <title></title>
5 <meta charset="utf-8" />
6 <script>
7   function getArea() {
8     var areaRectangle = function (width, height)
9     {
10       var areaRect = width * height;
11       arearect = parseFloat(areaRect.toFixed(2));
12       return arearect;
13     }
14     document.getElementById("area").innerHTML = areaRectangle(20, 30);
15   }
16 </script>
17 </head>
18 <body>
19 <input type="button" onclick="getArea()" value="Get Area" />
20 <div id="area"> </div>
21 </body>
22 </html>
```

Watch

No watch expressions

Call Stack

Not paused

Scope

Not paused

Breakpoints

problem5.html:20
<input type="button" onclick=...

XHR/fetch Breakpoints

DOM Breakpoints

Global Listeners

Event Listener Breakpoints