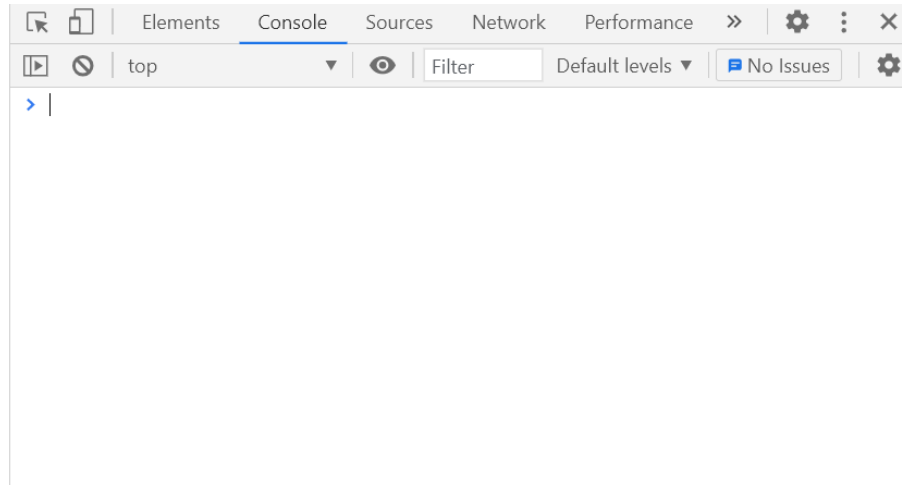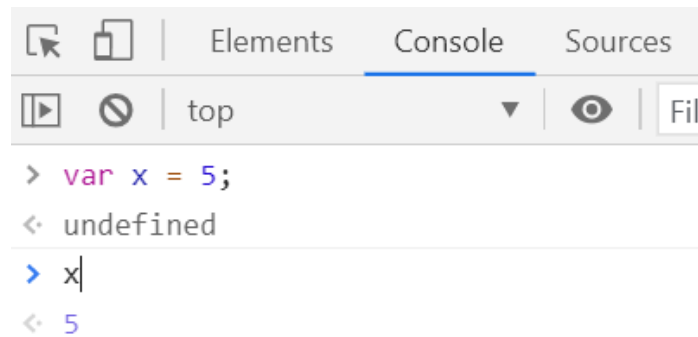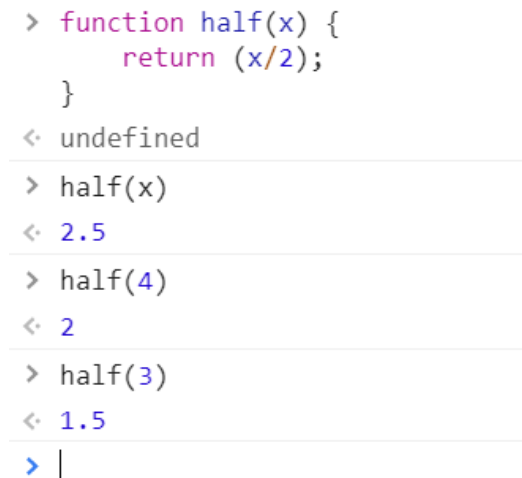# Exercises – Basic JS (2)

1. Try JavaScript interactively.

2. Define a variable x and give it a value of 5. Evaluate x and verify it shows the value.

3. Enter the function into …… and see if you were right.

4. Try to predict what ...... Try it and see.
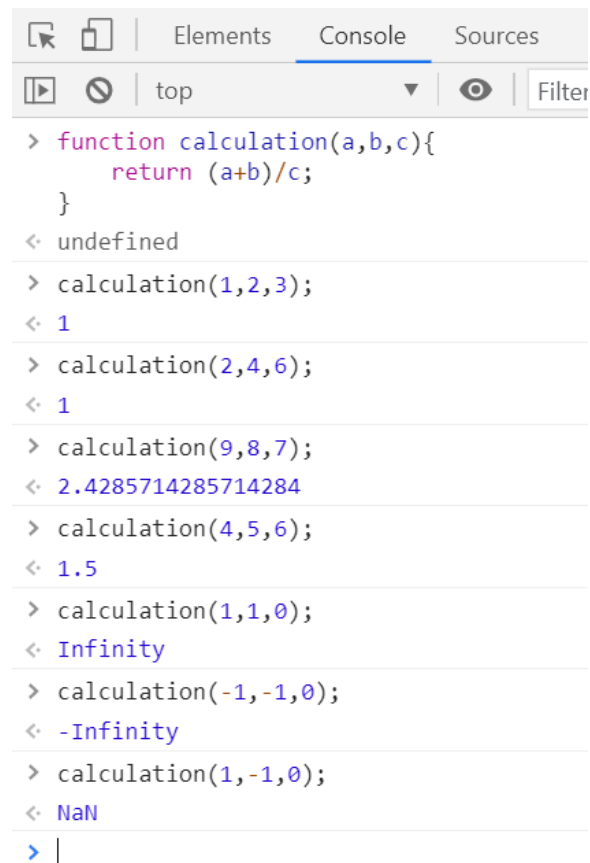
```
> x
< 5
> |
```

5. Enter this function into ...... you explain the surprising result?

```
> function seven(){
      x = 7;
      return (x);
  }
< undefined
> seven()
< 7
> x
< 7
> |
```

Since, variable **x** has been a variable with a **GLOBAL SCOPE**, it could be directly accessed with the **seven()** function and be manipulated from there to give the above output **x = 7**.
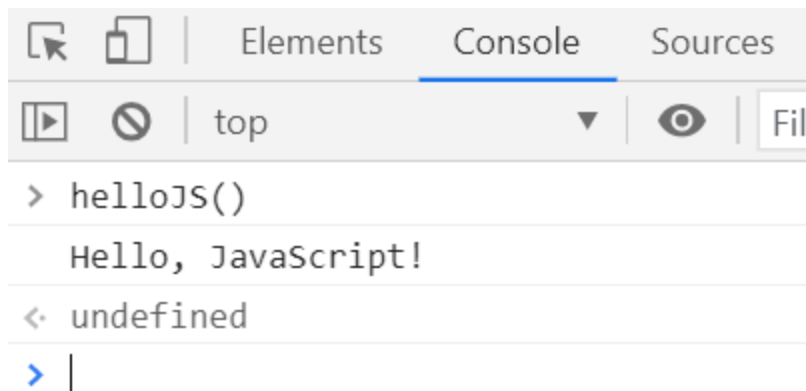
6. Make a function called calculation …… get what you expected.

```
          Elements    Console    Sources
    top                        ▼  ⦿  Filter
> function calculation(a,b,c){
      return (a+b)/c;
  }
← undefined
> calculation(1,2,3);
← 1
> calculation(2,4,6);
← 1
> calculation(9,8,7);
← 2.4285714285714284
> calculation(4,5,6);
← 1.5
> calculation(1,1,0);
← Infinity
> calculation(-1,-1,0);
← -Infinity
> calculation(1,-1,0);
← NaN
> |
```

7. Try the "more powerful practice" approach …… and call the new function.

```
//7. Try the "more powerful practice" approach …... and call the new function.
function helloJS() {
    console.log("Hello, JavaScript!");
}
```

```
          Elements    Console    Sources
    top                        ▼  ⦿  Fil
> helloJS()
  Hello, JavaScript!
← undefined
> |
```

```
//7. Try the "more powerful practice" approach …... and call the new function.
function helloJS() {
    console.log("Hello, JavaScript!");
}

function sumOfNums(numOne, numTwo) {
    return numOne + numTwo;
}
```

```
      Elements   Console   Sources

    top                    ▼  👁  Fil

> sumOfNums(23, 87)
← 110
> |
```
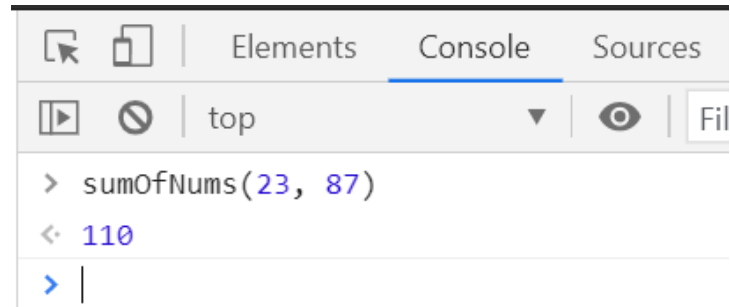
8. Write a function called isEven that …… so you cannot use either one.

```
//8. Write a function called isEven that …... so you cannot use either one.
function isEven(num) {
    if (num) {
        if (num % 2 === 0) {
            return true;
        } else {
            return false;
        }
    } else {
        return "Argument Not Found!";
    }
}
```

```
> isEven()
← "Argument Not Found!"

> isEven(2)
← true

> isEven(3)
← false

> |
```