Stefano Schmidt
Launch School
Introduction to Programming With JavaScript
The Basics – Exercises

1: In this exercise we are asked to show how we can concatenate two or more strings to create our full name as value. One way of doing this is the following:

```
let name = "Stefano" + " " + "Schmidt";
// assigns value 'Stefano Schmidt' to variable name
```

2: This exercise asks us to use arithmetic to determine the individual digits of a 4-digit number like 4936. One way of doing this is the following:

```
let number = 4936;
let ones = number%10;
let tens = ((number - ones)/10)%10;
let hundreds = ((number - (tens*10 + ones))/100)%10;
let thousands = ((number - (hundreds*100 + tens*10 + ones))/1000)%10;

console.log('thousands place is ' + thousands);
// logs 'thousands place is 4'
console.log('hundreds place is ' + hundreds);
// logs 'hundreds place is 9'
console.log('tens place is ' + tens);
// logs 'tens place is 3'
console.log('ones place is ' + ones);
// logs 'ones place is 6'
```

3: This exercise asks us to identify the data type of the following values:

'true': the data type is String.
false: the data type is Boolean.
1.5: the data type is Number.
2: the data type is Number.
undefined: the data type is undefined.
{ foo: 'bar' }: the data type is Object.

4: This exercise asks us to identify which lines from the code snippet below are expressions:

```
var foo;
foo = 5;
foo;
```

The first line is a statement since we can't use the return value as part of an expression:

```
> "this string is " + (var foo)
"this string is " + (var foo)
                     ^^^

Uncaught SyntaxError: Unexpected token 'var'
```

The second line an expression since we can use it as part of another expression:

```
> "this string is " + (foo = 5)
'this string is 5'
```

The third line is an expression since we can use it as part of another expression:

```
> 2 + foo
7
```

5: This exercise asks us to explain why the following code logs '510' instead of 15:

```
console.log('5' + 10);
```

This happens because the + operator coerces the value 10 to the value '10' since the other operand is a string, and then performs a string concatenation.

6: This exercise asks us to refactor the code from the previous exercise so that it logs 15 instead. One way of doing so is the following:

```
console.log(Number.parseInt('5') + 10);
```

7: This exercise asks us to use template literal syntax along with the expression (Number('5') + 10) to log "The value of 5 + 10 is 15." to the console. We can do this as follows:

```
console.log(`The value of 5 + 10 is ${Number('5') + 10}.`);
```

8: This exercise asks us "Will an error occur if you try to access an array element with an index that is greater than or equal to the length of the array?":

Trying to access an array element with index greater than or equal to the length of the array will not cause an error to occur: what will happen is that the value undefined will be returned.

9: This exercise asks us to create an array named names that contains a list of pet names. We can do this as follows:

```
let names = ["asta", "butterscotch", "pudding", "neptune", "darwin"];
```

10: This exercise asks us to create an object named pets that contains a list of pet names and the type of animal. We can do this as follows:

```
let pets = {
  asta: 'dog',
  butterscotch: 'cat',
  pudding: 'cat',
  neptune: 'fish',
  darwin: 'lizard'
};
```

11: This exercise asks us what the expression below evaluates to:

```
'foo' === 'Foo'
```

This expression evaluates to false because the strings ‘foo’ and ‘Foo’ are not the same value: capitalization matters.

12: This exercise asks us what the expression below evaluates to:

```
parseInt('3.1415')
```

This expression evaluates to the value 3.

13: This exercise asks us what the expression below evaluates to:

```
'12' < '9'
```

This expression evaluates to true. This is because JavaScript orders strings using a dictionary ordering.