functions

What is a function?

A function is a repeatable process or procedure.

```
var arr = [5,4,3,2,1];
var poppedVal = arr.pop();
console.log(arr);
console.log(poppedVal);
```

the pop function takes no inputs, and it returns a value which is the last item in the array that has been removed from the array

Declaring Functions

```
function anyNameYouWantForTheFunction() {
  // As many lines of code as you want
}
```

This type of function syntax consists of four parts:

- The function keyword,
- The name of the function (in this case, *anyNameYouWantForTheFunction*),
- Any parameters for the function (parameters will go inside of the parentheses after the function name),
- The function body (the code for the function, which lives inside of the curly braces).

Calling Functions

Function definition

```
function firstFunction(){
  console.log("I just wrote my first function!");
}
```

Function Invocation

```
firstFunction();
```

Returning Values from Functions

Returning a string

Ignoring the returned value

```
// this is called the function definition -
// we are ONLY defining the function here

function firstFunction(){
  return "I just wrote my first function!";
}

// to call or invoke the function

firstFunction(); // now we don't see undefined anymore!
```

Capturing the return value and logging it

```
var returnValue = firstFunction();
console.log(returnValue);
```

Conditional Logic in Functions

Do we need the second else condition?

```
function isOverPointFive() {
  if (Math.random() > .5) {
    return true;
  } else {
    return false;
  }
}
```

Eliminate Unnecessary Else

```
function isOverPointFive() {
  if(Math.random() > .5){
    return true;
  }
  return false;
}
```

Simpler and clearer - eliminate second else condition

Ternary operator

The conditional (ternary) operator is the only JavaScript operator that takes three operands. This operator is frequently used as a shortcut for the if statement.

$$\boxed{\textit{condition ? expr1 : expr2}}$$

condition: an expression that evaluates to true or false

expr1, expr2: Expressions with values of any type

Ternary Operator

```
function isOverPointFive() {
  return Math.random() > .5 ? true : false;
}
```

```
function isOverPointFive() {
  if (Math.random() > .5) {
    return true;
  } else {
    return false;
  }
}
```

```
function isOverPointFive() {
  if(Math.random() > .5){
    return true;
  }
  return false;
}
```

```
function isOverPointFive() {
  return Math.random() > .5 ? true : false;
}
```

Simplifying Further

```
function isOverPointFive() {
  return Math.random() > .5 ? true : false;
}
```

Meth.random() > .5 is a boolean expression
So we can just return this:

```
function isOverPointFive(){
  return Math.random() > .5;
}
```

Function Scope Rules

- All variables that are defined outside of functions (and inside of functions without the var keyword) are declared in the global scope
- All variables defined inside of functions can only be accessed by those functions (and any inner functions).

```
var globalVariable = "I live in the global scope";

function makeNewScope(){
  var functionScopeVariable = "I live in the scope of the makeNewScope function";
}

globalVariable; // "I live in the global scope";
makeNewScope(); // maybe this will define the functionScopeVariable...

functionScopeVariable;
// This gives us an error! To be specific, a ReferenceError
//  because the functionScopeVariable is not defined.
```

What happens when we remove the var keyword?

```
// Since thse variable declaration is in the global scope, it will
// be a globalVariable with or without the var keyword.  It is a best
// practice to always use the var keyword though.
globalVariable = "I live in the global scope";

function makeNewScope(){
  // You do not want to do this in practice.  You should
  // always defined your variables with the var keyword.
  functionScopeVariable = "What happens now?";
}

globalVariable; // "I live in the global scope"
makeNewScope(); // now this will define the functionScopeVariable!

// The value of the variable will be "What happens now?"
functionScopeVariable;
```

var keyword

If we omit the var keyword inside of a function, we actually declare that variable in the global scope.

This is not best practice - as it makes the code very difficult to read

To enhance readability further - always use either **const** or **let**