A function is a package for an algorithm; once written, it can be use over and over.
Example Function

A function to compute a person’s weight in gold would be

function worthInAu (weight) {
    return weight*12*566.99;
}

This computation is what’s being packaged
Functions have a specific syntax

```
function <name> ( <parameter list> ) {
  <function definition>
}
```

- `<name>` names are identifiers; start w/letter
- `<parameter list>` is the input variables, a list separated by commas
- `<function definition>` is just the program to do the work

Brackets appear here by convention
A Sample Function

Compute the Body Mass Index when the inputs are in metric

```javascript
function <name> ( <parameter list> ) {
   <function definition>
}
```

function bmiM ( weightKg, heightM ) {
   // Figure Body Mass Index in metric units
   return weightKg / (heightM * heightM);
}

Identify the corresponding parts
Most programming is done by writing functions, so learning the form is key.

```javascript
function bmiE ( weightLBS, heightIn ) {
    // Figure Body Mass Index in English units
    var heightFt = heightIn / 12; // Change to feet
    return 4.89 * weightLBS / (heightFt * heightFt);
}
```
A function is declared by writing down the "package" ... the function is used when it is called

```javascript
function BMI (units, height, weight ) {
    // Compute BMI in either metric or English
    if (units == "English")
        return bmiE(weight, height);
    else
        return bmiM(weight, height);
}
```
Declaration: the function “package,” says what happens when the function runs

Call: the function use, asks for the computation to be run

- There is only one function declaration
- There can be many calls … functions are reusable
- In JS, functions tend to be grouped together but the calls go where they are needed
Gold Function

Suppose we compute “weight in Au”

\[
\text{worth in gold} = (\text{weight} \times 12) \times 566.99
\]

function (        ) { 

Begin with the form ...
Suppose we compute “weight in Au”

\[
\text{worthInAu} = (\text{weight} \times 12) \times 566.99
\]

```javascript
function worthInAu () {
    // Compute the dollar value
    // of weight at $566.99/tz
}
```
Gold Function

Suppose we compute “weight in Au”

\[ \text{worthInAu} = (\text{weight} \times 12) \times 566.99 \]

```javascript
function worthInAu ( weight ) {
    // Compute the dollar value
    // of weight at $566.99/tz
}
```

Pick a Name
Pick the Parameter
Gold Function

Suppose we compute “weight in Au”

\[ \text{worthInAu} = (\text{weight} \times 12) \times 566.99 \]

```javascript
function worthInAu ( weight ) {
  // Compute the dollar value
  // of weight at $566.99/tz
  return weight * 12 * 566.99;
}
```
No one writes perfect programs the first time … smart programmers check
To test, have a standard page handy

<html>
<head>
<title>My Test Page</title>
</head>
<body>
  <script language="JavaScript">
    Put your JavaScript code here
  </script>
</body>
</html>
Declare the Function

Put a function declaration in `<script>`

```html
<html><head><title>My Test Page</title></head>
<body>
<script language="JavaScript">
    function worthInAu ( weight ) {
        // Compute the dollar value
        // of weight at $566.99/troy oz
        return weight * 12 * 566.99;
    }
    alert(worthInAu(1/12));
</script>
</body>
</html>
```
Try The Function

Unquestionably, the best practice is to test everything.
Demonstration
Functions are packages for algorithms

- They follow a series of rules, that quickly become routine
- Functions have both a declaration and a call
- Functions have both parameters (in the declaration) and arguments (in the call)
- Scope refers to the region of a program where a variable is “known”

Functions are the secret to building complex systems