



Functions & Abstraction

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

© 2016, Lacharal Soudic



The Package

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

```
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



Writing Functions

Most programming is done by writing functions, so learning the form is key

```
function bmiE ( weightLBS, heightIn ) {  
    // Figure Body Mass Index in English units  
    var heightFt = heightIn / 12; // Change to feet  
    return 4.89 * weightLBS / (heightFt * heightFt);  
}
```



Declarations

A function is declared by writing down the "package" ... the function is used when it is called

Declaration

```
function BMI ( units, height, weight ) {  
    // Compute BMI in either metric or English  
    if (units == "English")  
        return bmiE(weight, height);  
    else  
        return bmiM(weight, height);  
}
```

Calls



Summarizing

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"

Worth = (Weight*12)*368.4

```
function      (      ) {
// Compute the dollar value
// of weight at $368.40/tz
}
```

Begin with the form ...



Gold Function

Suppose we compute "weight in Au"

Worth = (Weight*12)*368.4

```
function worthau (      ) {
// Compute the dollar value
// of weight at $368.40/tz
}
```

Pick a Name



Gold Function

Suppose we compute "weight in Au"

Worth = (Weight*12)*368.4

```
function worthau ( weight ) {
// Compute the dollar value
// of weight at $368.40/tz
}
```

Pick a Name

Pick the Parameter



Gold Function

Suppose we compute "weight in Au"

Worth = (Weight*12)*368.4

```
function worthau ( weight ) {
// Compute the dollar value
// of weight at $368.40/tz
return weight * 12 * 368.4;
}
```

Pick a Name

Pick the Parameter

Define the Computation



Testing Template

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>

Testing Template

```
<html><head><title>My Test Page</title></head>
<body>
<script language="JavaScript">
function worthau ( weight ) {
// Compute the dollar value
// of weight at $368.40/troy oz
return weight * 12 * 368.4;
}
alert(worthau(1/12));
</script>
</body>
</html>
```

Try The Function

Unquestionably, the best practice is to test everything

Function Features

Reviewing properties of functions

- Selecting names ... don't use alert()
- Parameter variables ... don't need to be declared, but work like local variables

```
function bmiE ( weightLBS, heightIn ) {
  // Figure BMI in English
  var heightFt = heightIn / 12; // Change to feet
  return 4.89 * weightLBS / (heightFt * heightFt);
}
function bmiE ( weightLBS, height ) {
  // Figure BMI in English (height in in)
  height = height / 12; // Change to feet
  return 4.89 * weightLBS / (height * height);
}
```

Function Features (cont.)

- Scope of Reference ... refers to where in the program a variable is "known," i.e. where its value can be referenced and/or assigned

```
function bmiE ( weight, heightIn ) {
  // Figure BMI in English
  var heightFt = heightIn / 12; // Change to feet
  return 4.89 * weight / (heightFt * heightFt);
}
function BMI (units, height, weight )
{
  // Compute BMI
  if (units == "English")
    return bmiE(weight, height);
  else
    return bmiM(weight, height);
}
```

Local to function

Function Features (cont.)

- Global ... declared outside of functions

```
<script LANGUAGE='JavaScript'>
var priceperoz = 368.40;
...
function worthau ( weight ) {
  // Compute the dollar value
  // of weight at priceperoz
  return weight * 12 * priceperoz;
}
...

```

Global to function

Function Features (cont.)

- Parameters vs Arguments ... parameters are the "formal" variables in the function declaration; arguments are the same thing in the call

```
function BMI (units, height, weight )
{
  // Compute BMI
  if (units == "English")
    return bmiE(weight, height);
  else
    return bmiM(weight, height);
}
...
index = BMI("English", 72, 200);
...

```

Parameters

Arguments

Summary

Functions are packages for algorithms

- They follow a series of rules, that quickly become intuitive
- 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