Functions & Abstraction

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

A Sample Function

Compute the Body Mass Index when the inputs are in metric

```javascript
function bmiM (weightKg, heightM) {

// Figure Body Mass Index in metric units
return weightKg / (heightM * heightM);
}
```

Writing Functions

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);
}
```

Declarations

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

```javascript
function BMI (units, weight, height) {

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

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
Suppose we compute “weight in Au”

\[ \text{Worth} = (\text{Weight} \times 12) \times 368.4 \]

Begin with the form ...

```
function worthAu() {
  // 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>`
Unquestionably, the best practice is to test everything.

Reviewing properties of functions

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

```javascript
function bmi(units, height, weight) {
  var heightFt = height / 12;  // Change to feet
  return 4.89 * weight / (heightFt * heightFt);
}
function bmiE(weightLBS, heightIn) {
  var heightFt = height / 12;  // Change to feet
  return 4.89 * weightLBS / (heightFt * heightFt);
}
function wortha(weight) {
  // Compute the dollar value
  // of weight at priceperoz
  return weight * 12 * priceperoz;
}
```

• Global: declared outside of functions

Scope of Reference...

- Refers to where in the program a variable is "known," i.e. where its value can be referenced and/or assigned
- Can be local to a function or global to the whole program

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

• Parameters vs Arguments

- Parameters are the "formal" variables in the function declaration; arguments are the same thing in the call

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

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"