<script>
var guessInput = document.getElementById("guess");
var guess = guessInput.value;
var answer = null;

var answers = ["red",
    "green",
    "blue");

var index = Math.floor(Math.random() * answers.length);

if (guess == answers[index]) {
    answer = "You're right! I was thinking of " + answers[index];
}
else {
    answer = "Sorry, I was thinking of " + answers[index];
}

alert(answer);
</script>
var guessInput = document.getElementById("guess");
var guess = guessInput.value;

var answer = checkGuess(guess);
alert(answer);

<script>
var guessInput = document.getElementById("guess");
var guess = guessInput.value;
var answer = null;

var answers = ["red", "green", "blue"];

var index = Math.floor(Math.random() * answers.length);

if (guess === answers[index]) {
  answer = "You're right! I was thinking of " + answers[index];
} else {
  answer = "Sorry, I was thinking of " + answers[index];
}
alert(answer);
</script>
function checkGuess(guess) {
    var answers = ["red",
                   "green",
                   "blue"];

    var index = Math.floor(Math.random() * answers.length);

    if (guess == answers[index]) {
        answer = "You're right! I was thinking of " + answers[index];
    } else {
        answer = "Sorry, I was thinking of " + answers[index];
    }

    return answer;
}
function bark(dogName, dogWeight) {
    if (dogWeight <= 10) {
        return dogName + " says Yip";
    } else {
        return dogName + " says Woof";
    }
}

The Bark fxn:

→ what does it do?

→ what happens when it is invoked?
When we call `bark`, the arguments are assigned to the parameter names in the `bark` function. Anytime the parameters appear in the function, the values we passed are used.

The code in the body does all the work.

Here we are passing two arguments, the name & the weight.

The function name ↓

```
bark("Fido", 50);
```
Note: Functions are not required to return a value.
But in this case, the `bark` fn does return a value.

```javascript
function bark(dogName, dogWeight) {
    if (dogWeight <= 10) {
        return dogName + " says Yip";
    } else {
        return dogName + " says Woof";
    }
}
```

Call the fn:

```javascript
var sound = bark("Fido", 50);
alert(sound);
```

The string is returned & assigned to `sound`:

```
Fido says Woof
```
What are the advantages of using functions?

* Powerful
  - Write it once, use it many times 😊

* Flexible
  - Because they are parameterized:
    - Each time you use it, you can pass it different arguments → the results are specific to your problem
Returning a value

→ many functions return values
→ functions do not have to.

Ex. a function can update the DOM and then be done w/o returning a value (those funcs do not have a return stmt.)
Q: What is the difference between a parameter & an argument?

* You define a function with parameters.
You call a function with arguments.
Defining three parameters

```javascript
function cook(degrees, mode, duration) {
    // your code here
}
```

These are ARGUMENTS:

```javascript
cook(425.0, "bake", 45);
cook(350.0, "broil", 10);
```
Always start with the keyword "function".

Give your function a name.

```
function addScore (level, score) {
  var bonus = level * score * .1;
  return score + bonus;
}
```

Even if your function has no parameters, you still need an opening and closing set of parentheses.

The body is between the curly braces.

The return statement includes a statement which is returned as a result of calling the function.

A function can include a statement with the return keyword, but it does not need to.

Here is the closing brace!
Q: Why don’t the parameter names have var in front of them? A parameter is a new variable right?

A: Effectively yes. The function does all the work of instantiating the variable for you, so you don’t need to supply the var keyword in front of your parameter names.
Q: What are the rules for function names?

A: The rules for naming a function are the same as the rules for naming a variable.
Q: I'm passing a variable to my function—if I change the value of the corresponding parameter in my function does it also change my original variable?

A: No. When you pass a primitive value it is copied into the parameter. We call this "passing by value." So if you change the value of the parameter in your function body it has no affect on our original argument's value. The exception to this is passing an array or object.

Q: So how can I change values in a function?

A: You can only change the values of global variables (those defined outside of functions), or variables you've explicitly defined in your function. We're going to talk about that in a little more detail shortly.
Q: What does a function return if it doesn’t have a return statement?

A: A function without a return statement returns undefined.
function dogsAge(age) {
    return age * 7;
}
var myDogsAge = dogsAge(4);

function rectangleArea(width, height) {
    var area = width * height;
    return area;
}
var rectArea = rectangleArea(3, 4);

function addUp(numArray) {
    var total = 0;
    for (var i = 0; i < numArray.length; i++) {
        total += numArray[i];
    }
    return total;
}
var theTotal = addUp([1, 5, 3, 9]);
function getAvatar(points) {
    var avatar;
    if (points < 100) {
        avatar = "Mouse";
    } else if (points > 100 && points < 1000) {
        avatar = "Cat";
    } else {
        avatar = "Ape";
    }
    return avatar;
}
var myAvatar = getAvatar(335);
var avatar;
var levelThreshold = 1000;

function getScore(points) {
    var score;
    for (var i = 0; i < levelThreshold; i++) {
        // code here
    }
    return score;
}

If a variable is declared outside a function, it’s GLOBAL. If it’s declared inside a function, it’s LOCAL.
These four variables are globally scoped. That means they are defined and visible in all the code below.

Note: if you link to additional scripts in your page, they will see these global variables too!

Here we can use only the global variables, we have no access to any variables inside the functions because they are not visible in the global scope.

```javascript
var avatar = "generic";
var skill = 1.0;
var pointsPerLevel = 1000;
var userPoints = 2008;

function getAvatar(points) {
    var level = points / pointsPerLevel;
    if (level == 0) {
        return "Teddy bear";
    } else if (level == 1) {
        return "Cat";
    } else if (level >= 2) {
        return "Gorilla";
    }
}

function updatePoints(bonus, newPoints) {
    for (var i = 0; i < bonus; i++) {
        newPoints += skill * bonus;
    }
    return newPoints + userPoints;
}

userPoints = updatePoints(2, 100);
avatar = getAvatar(2112);
```

The level variable here is local and is visible only to the code within the getAvatar function.

That means only this function can access the level variable.

Don’t forget the points parameter, which also has local scope in the getAvatar function.

Note: getAvatar makes use of the pointsPerLevel global variable too.

In updatePoints there is a local variable i. i is visible to all the code in updatePoints.

bonus and newPoints are also local to updatePoints, while userPoints is global.
What happens if I name a local variable the same thing as an existing global variable?

- Global
  - var beanCounter = 10;
  - function getNumberOfItems (ordertype) {
    - var beanCounter = 0;
    - if (ordertype == "order") {
      // do some stuff with beans
    }
    - return beanCounter;
  }

- Local
  - var beanCounter = 0;
  - function getNumberOfItems (ordertype) {
    - if (ordertype == "order") {
      // do some stuff with beans
    }
    - return beanCounter;
  }
<!doctype html>
<html lang="en">
<head>
<title>Functions as Values - WOW!!</title>
<meta charset="utf-8">
</head>
<body>
<script>
function addOne(num) {
    return num + 1;
}
var plusOne = addOne;
var result = plusOne(1);
alert(result);
</script>
</body>
</html>
Creating a function without using a name!

function (num) {
  return num + 1;
}

not so useful.

var f = function (num) {
  return num + 1;
}

this is really useful 😊

use the variable to call the function 😊

var result = f(1);
alert(result);

what is the result? 😊
what do you think is going on with this code?

```
var element = document.getElementById("button");
element.onclick = function () {
    alert("clicked");
}
```

just think about it.