Javascript Basics II

Justice Reskill

Conditionals

Conditionals are conditions that are required for an action to happen. For example

- If the time is after 5:00 pm turn on the street lights
- If the weather is currently raining, bring an umbrella
- If the current date is between memorial day and labor, and the temperature is above 80 degrees - open the pools

Conditionals As Variables

In coding, we using conditionals as variables.

- X > Y (when x is greater than y)
- cat.length <= dog.length (when the length of cat is less than or equal to dog)

Comparison Operators

To compare values in Javascript, we use the same comparison operators learned in algebra.

==	Equals, test if two values are equal to each other
!=	Not equals, test if two values are not equal to each other
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to
===	A special comparison operator in programing, that check if the values are equal AND they are the same data type

Defining An If Statement

If statements compare two more conditionals. If statement use the brackets ‘{’ and ‘}’ define the body.

If the conditional inside an if statement evaluates to true, the code inside the if statement is executed.

Reserved word

```
if (conditional) {
    //body
}
```

Where the code goes if conditionals a true

Example if statements

If the conditional of an if statement evaluates to true, then the code inside the body is executed. Otherwise it is skipped.

```
if (3>2) {
        console.log(“3 is greater than 2 is true”);
}

if (4==3) {
     console.log(“4 is not equal to 3, its fales”)
}
```

Variables and if statements

If statements can use variables as conditionals.

```
var x = 3;

var y = 5;

if (x>y) {
	console.log(“x is greater than y”);
}
```

Default else statement

If the if statement evaluates to false, we can define an else statement that will run as a default.

```
var x = 3;

var y = 5;

if (x>y) {
        console.log(“x is greater than y”);
} else {
        console.log(“Running default statement”);
}
```

Chaining If Statements

Multiple if statements can be evaluated before running the else to test multiple scenarios. If one if statement fails, it well test the others in the chain. If one evaluates to true, the remaining if statements will not be evaluated.

Which one will be executed here?

```
var x = 3;

var y = 5;

if (x>y) {
        console.log(“x is greater than y”);
} else if (x<y) {
        console.log(‘y is greater than x“);
} else if (x==y) {
        console.log(‘y is equal to x“);
} else {
        console.log(“Running default statement”);
}
```

Multiple Comparison Operators

Multiple comparisons can be made in a single conditional statement.

&&	AND - Requires that both conditionals are true
||	OR - requires that one of the conditionals are true

AND Conditionals

The && operator will test if both conditionals are true. In our example, both x>y AND string1 != string2 must evaluate to true.

```
var x = 3;

var y = 5;

var string1 = ‘Cat’;

var string2 = ‘Dog’;

if (x>y && string1 != string2) {
        console.log(“x is greater than y AND cat does not equal dog”);
}
```

OR Conditionals

The || operator will test if either one of the conditionals are true. In our example, either x>y OR string1 != string2 must evaluate to true.

```
var x = 3;

var y = 5;

var string1 = ‘Cat’;

var string2 = ‘Dog’;

if (x>y || string1 != string2) {
        console.log(“x is greater than y AND cat
does not equal dog”);
}
```

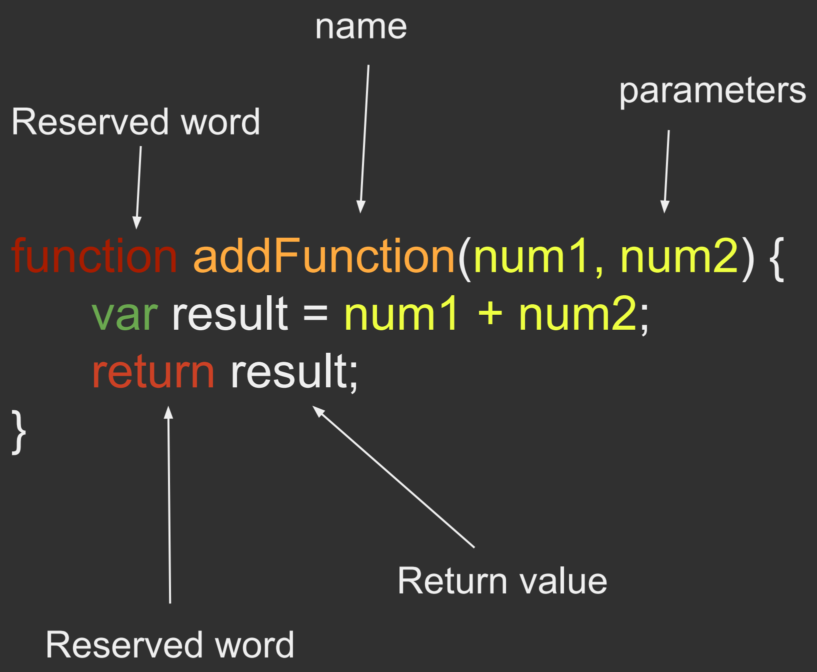
What Are Functions

Functions can be thought of as re-usable segments of code. You write the code once, and then you can call the code multiple times.

Defining a Function

Functions have several parts in its definition.

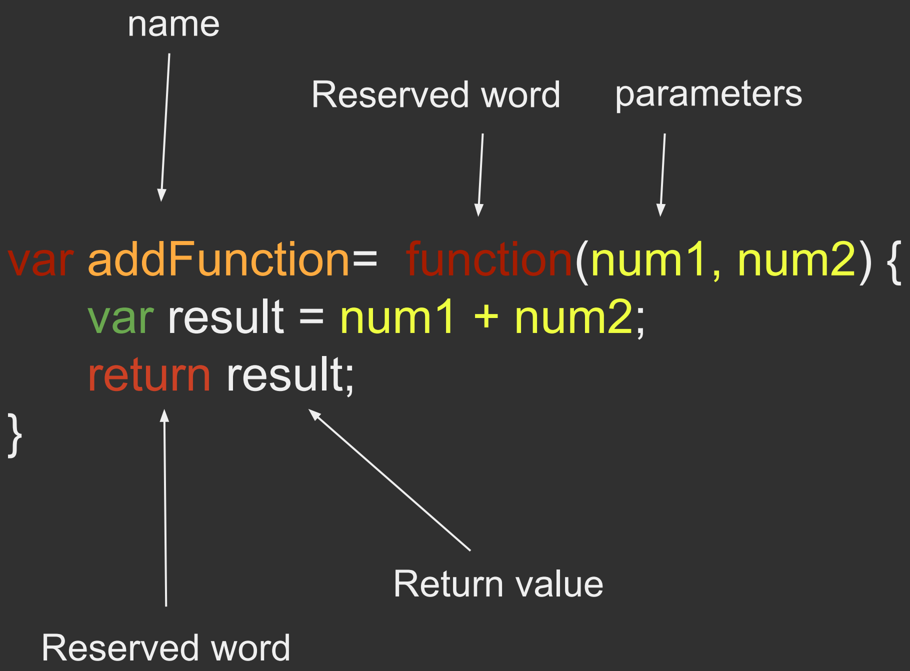
- The reserved word function is what defines the function
- Functions must have a name
- A function can take what is known as parameters
- The brackets define the body of the function, and the code inside the brackets ar what is executed



```
function addFunction(num1, num2) {
    var result = num1 + num2;
    return result;
}
```

Defining a Function as a variable

An alternative way of defining a function is the ability to assign it as a variable.



```
var addFunction=  function(num1, num2) {
      var result = num1 + num2;
      return result;
}
```

Calling A Function

Once a function is defined, it can be called, aka invoked, by () . Any variables passed into the () are called arguments. The amount of arguments should match the amount of parameters.

```
function testFunction1(param1) {
    console.log(param1);
}

testFunction1(“Hello!”);


function testFunction2() {
    console.log(“No Params”);
}

testFunction2();
```

Calling a Function Multiple times

Functions can be called multiple times making the same code reusable.

Also notice how this function does not have a return and does not have any parameters.

```
var ringDoorbell =  function() {
     console.log(“Ding Dong);
}

//Call the function
ringDoorbell();

//Executes the code again
ringDoorbell();

//Same code without rewriting
ringDoorbell();
```

Return Value Assigned to Variable

If the function has a return statement, it can assigned to a variable.

```
function addFunction(num1, num2) {
	var sum = num1 + num2;
	return sum;
}

var result = addFunction(3,5);

console.log(result);
```