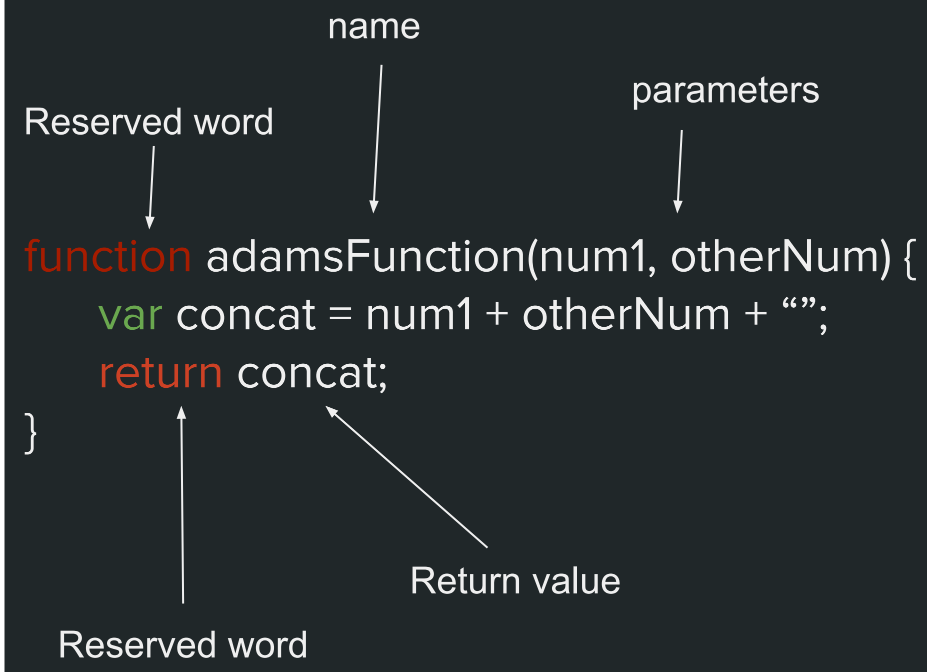
Functions

How functions are *defined*

- Functions are defined with the reserved word “function”
- Defined with:
  - Name
  - Parameters
  - Code block
  - Optional return statement



```
function adamsFunction(num1, otherNum) {
    var concat = num1 + otherNum + “”;
    return concat;
}
```

How functions are *called*

- Functions are called using ( )
- Inputs can be variables or hard-coded values
- The function will be *evaluated* and then “replaced” with the return value

```
function adamsFunction(num1, otherNum) {
    var concat = num1 + otherNum + “”;
    return concat;
}

var someNumber = 44;

var result = adamsFunction(someNumber, 2 )

console.log(result); // “442”
```

Call functions *after* they're defined

- Javascript is an interpreted language
- The script in the right panel ➔ doesn't know what adamsFunction is when it calls it

```
var someNumber = 44;

var result = adamsFunction(someNumber, 2 )
// ERROR!

function adamsFunction(num1, otherNum) {
      var concat = num1 + otherNum + “”;
      return concat;
}
```

Inputs and outputs are “undefined” by default

- If you don’t write a “return” statement, your function will return “undefined”
- If you don’t provide parameters when calling functions, they will be “undefined”

```
function adamsFunction(num1, otherNum) {
    var concat = num1 + otherNum + “”;
}


var result = adamsFunction(); // undefined
```

Pass-by-value

Variables that are passed into functions are unchanged by whatever the function tries to do to them

Pass-by-reference

Variables that are passed into function can be changed by the function itself

So is Javascript pass by value or pass by reference?

Trick question! Turns out it’s kinda both!

How JS functions treat input variables

Strings	Pass by value
Numbers	Pass by value
Arrays	Pass by reference
Objects	Pass by reference
Functions	Pass by reference

Pass by value

- The code to the right can’t change the value of the “outside” variable

```
var outside = 44;


function testFunction(num) {
      num +=  22;
}

var result = testFunction(outside );

console.log(outside);  // 44
```

Pass by reference

- The code to the right can change the key-value pairs of the “outside” variable

```
var outside = {
      num: 44;
};


function testFunction(obj) {
      obj.num +=  22;
}

var result = testFunction(outside );

console.log(outside.num);  // 66
```

Scoping

- Variables that are created *inside* your function are forever trapped in the function

```
function testFunction() {
      var hello = “Hello World!”;
}



console.log(hello);  // undefined
```

Scoping

- Variables that are created *outside* your function are able to come and go as they please

```
var outside = “Hello World!”;

function testFunction() {
	console.log(outside); // “Hello World!!”
}
```