NEXT ACADEMY

Introduction to Web Development

Week 5: JavaScript

Today's Plan

Short exercises to discover Javascript and it's applications in web

NEXT ACADEMY

What is Javascript?

And why do we need it?

Why isn't HTML/CSS enough?

HTML/CSS is static.

HTML/CSS just describes content and styles.

It doesn't process or manipulate any of that content. Or allow us to respond to user interactions.

In comes Javascript!

- Handle events (clicks, hovers, selection, etc)
  - e.g. clicking on a button reveals something
  - https://www.airbnb.com/rooms/2615058?s=B8V1_7oX

- Can process and manipulate content
  - e.g. sorting a table in alphabetical order
  - http://tablesorter.com/docs/

Javascript can also

- Build games
  - e.g. 2048 in your browser

- Do everything that you can't do with HTML/CSS

Where does JS go?

Like CSS, it has many places it can be.

Option 1: In the HTML file itself

```
<body>
  <script>
    document.write('Hello world!')
  </script>
</body>
```

Where does JS go?

Option 2: In a separate file

```
<script src="scripts.js"></script>
```

Option 3: Inline Javascript

not recommended, and not covered today

NEXT ACADEMY

An Introduction to Programming with Javascript

what is programming anyway?

What is Programming?

Programming is a **creative process** done by humans **to instruct a computer on how to do a task.**

Like how HTML/CSS tells a browser how to display your website, **Javascript tells a browser a set of instructions to follow** which can be more complex.

NEXT ACADEMY

JavaScript basics

Let the computer talk to you

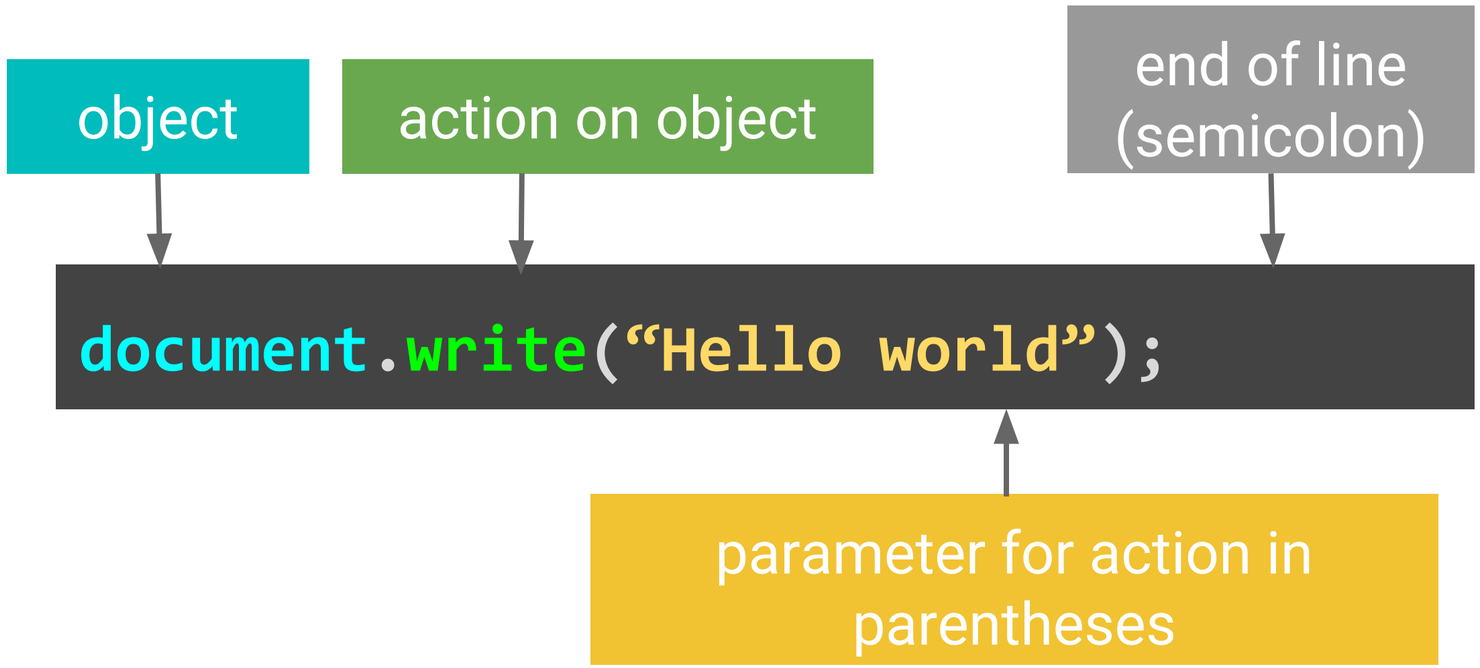
NEXT ACADEMY

Ways to output a message:

1. `window.alert("Hello world!")`
2. `document.write("Hello world")`
3. `console.log("Hello world!")`

Examples of usage.

Anatomy of an output statement



Number

data type

Data Type: Number

Numbers in JS can be written in 2 ways:

As integers (no decimal)

```
2, 42, 900001
```

As floats (when you need decimal)

```
42.0, 0.0001, 10.2222
```

More on precision of numbers in JS.

Arithmetic Operators
things you can do with numbers

Addition	42 + 24	
Subtraction	100 - 12	
Multiplication	20 * 4	
Division	100 / 3	
Remainder	42 % 6 42 % 5	Gives you remainder of a division (also called modulo)
Negation	-53	Negative value of number
Increment and Decrement	num++ num--	Increment or decrement number by 1

Operator Precedence

Operators have orders of precedence. Ones with highest precedence will be calculated first.

```
5 + 6 * 3   //is equal to 23

(5 + 6) * 3 //is equal to 33
```

List of precedence

Variables

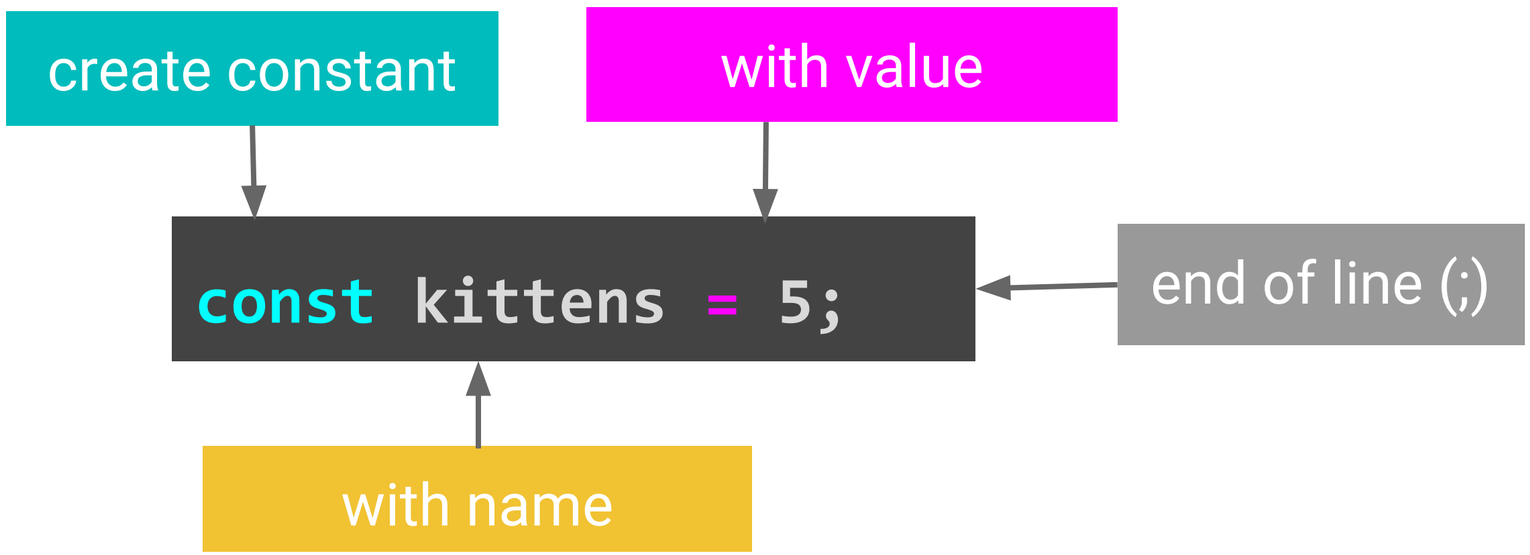
concept

Concept: Variables

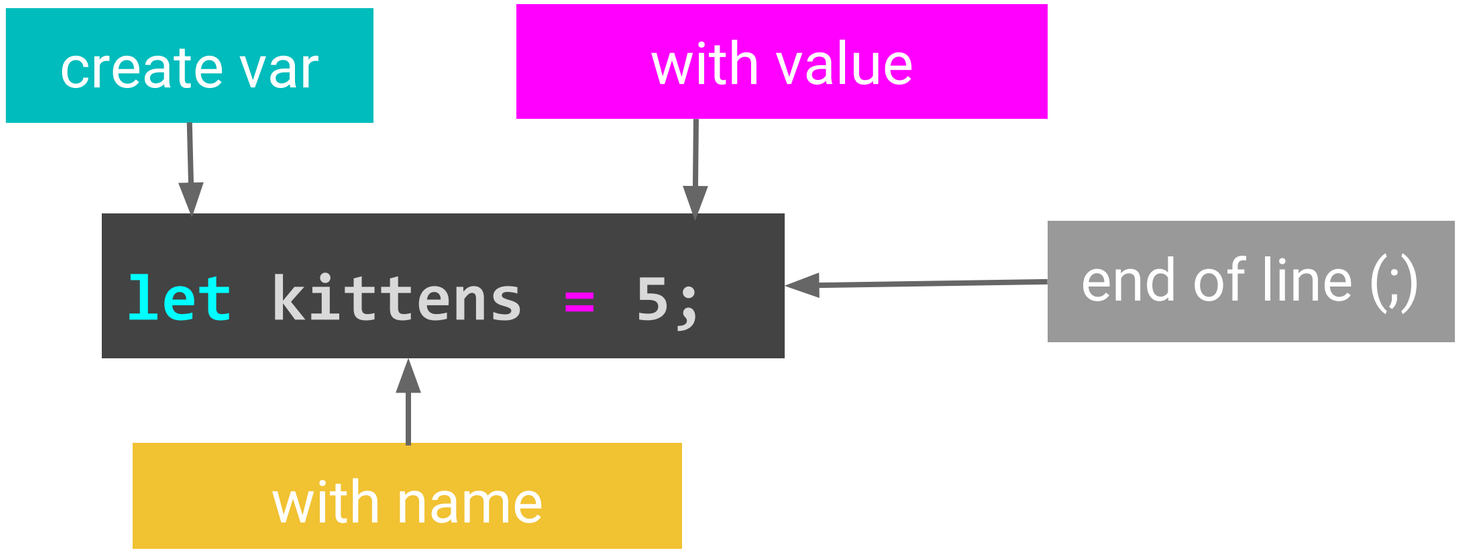
Variables are containers used to store any values, like numbers and strings which we'll see soon

It lets you **name** your values.

Anatomy of a constant



Anatomy of an variable



Variable creation

To create a variable:

```
const puppies = 3;

let kittens;

let food = 10;
```

Variable assignment

Let variables can also be updated.

Note: You only need to specify ‘let’ the first time you name the variable.

```
let kittens = 22;

kittens = 22 + 3;       //I picked up 3 more kittens

document.write(kittens);  //This will display 25
```

Variable use

To use a variable:

```
let cuteThings = kittens + puppies

document.write(cuteThings)
```

undefined - variable default value

If values are not assigned to a variable, it has a default value of `undefined`.

```
let mystery;

document.write(mystery); //what is this?
```

Variables
or I can't remember what is what!

```
//What is this??

document.write(4 * 3 + 1);

//Compared to this:

const cookiesPerBox = 4;

const boxes       = 3;

const randomCookie    = 1;

const totalCookies    = cookiesPerBox * boxes + randomCookie;
```

Variable Naming

- Names can contain **letters** (a-Z), **digits** (0-9), _ and **$**
- Names **cannot** begin with a digit
- Names are **case sensitive**
  - myName and myname are two different variables

Variable naming cont'd

- Reserved words (like JavaScript keywords) cannot be used as names
  - words like let, const, if, else, true, false, etc
- Good naming
  - Use camelCase
  - Use descriptive names

Comments

Concept: comments

Single line comment

```
// comments one line

// yeah, like only

// one line at a time
```

Multi-line comment

```
/*

  comments

  multiple

  lines

*/
```

String

data type

Data Type: String

A string is a line of text.

Strings are noted by “double-quotes” or ‘single-quotes’ around them.

```
//Displays Sher Minn (without quotes)

const name = “Sher Minn”;

document.write(name);
```

You can do things to Strings

Combine strings:

```
“Hi ” + name + “!” //is equal to “Hi Sher Minn!”
```

Get a length of a string:

```
name.length //is equal to 9
```

You can do things to Strings

Get a character at a position in a string:

```
name[5] //is equal to “m”.positions start at 0.
```

S	h	e	r		M	i	n	n
0	1	2	3	4	5	6	7	8

You can do things to Strings

Convert to uppercase

```
name.toUpperCase() // “SHER MINN”
```

More String functions

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/String#String_generic_methods

Boolean

data type

Data Type: Boolean

We can use booleans to determine conditions.

A boolean is a value that has 2 states:

`true` or `false`

Concept: Boolean expressions

A combination of values and operators that evaluate to either true or false

Examples in English:

- today is Saturday ✅ → true
- (today is Saturday) ✅ or (yesterday was Wednesday) ❌ → true
- (today is Saturday) ✅ and (we are in Malaysia) ✅ → true
- (today is Monday) ❌ or (we are not in Malaysia) ❌ → false

Boolean Operators in JS

Instead of using ‘and’, ‘or’, ‘not’, we use these symbols:

And, &&	cold && rainy	is true if both are true
Or, ||	warm || dry	is true is at least one is true
Not, !	!green	is opposite of boolean

A && B Truth Table

A	B	A && B
true	true	true
true	false	false
false	true	false
false	false	false

A || B Truth Table

A	B	A || B
true	true	true
true	false	true
false	true	true
false	false	false

!A Truth Table

A	!A
true	false
false	true

Boolean Comparators in JS

We can also compare values. More comparators here.

Less than, <	100 < 3	true if left value is less than right value.
Less than or equal, <=	100 <= 3	true if left value is less than or equal to right value.
Greater than, >	kittens > puppies	true if left value is more than right value.
Greater than or equal to, >=	kittens >= puppies	true if left value is more than or equal to right value.
Equal to, ==	sherminn == sherminn	true if values are identical in value.
Not equal to, !=	cookie != pizza	true if values are not identical in value.

Functions

concept

brushTeeth()

1) open the cabinet,

2) grab your brush and paste,

3) put some paste on your brush,

4) open your mouth,

5) push the brush across your teeth.

6) ...

introduceSelf(name, jobTitle)

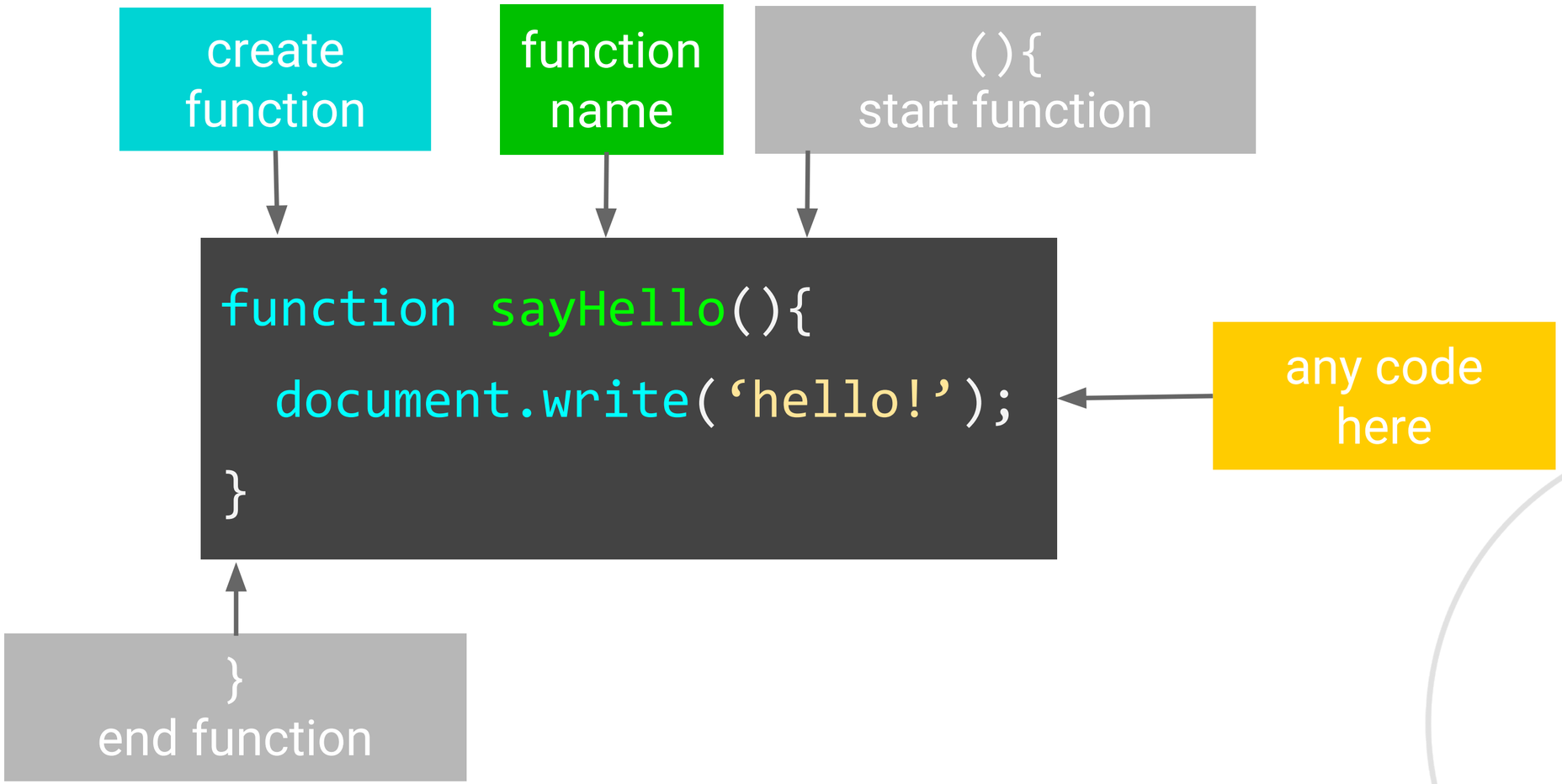
```
1. const introduceName = “Hi, my name is ” + name +
   “!” ;
2. const introduceJob = “I work as a ” + jobTitle;
3. document.write(introduceName);
4. document.write(introduceJob);
5. document.write(“How about you?”)
```

Concept: functions

Functions are a way to group code together. It can be good for:

1. writing reusable code
2. making code easier to read
3. abstracting, or giving a name to a set of instructions

Anatomy of a function



creating a function

Functions are created with the `function` keyword

```
function sayHello(){
  document.write('hello!');
}
```

calling (using) a function

Functions are used by writing the name, followed by ()

```
function sayHello(){
  document.write('hello!');
}

sayHello(); //this will print 'hello' to the console
```

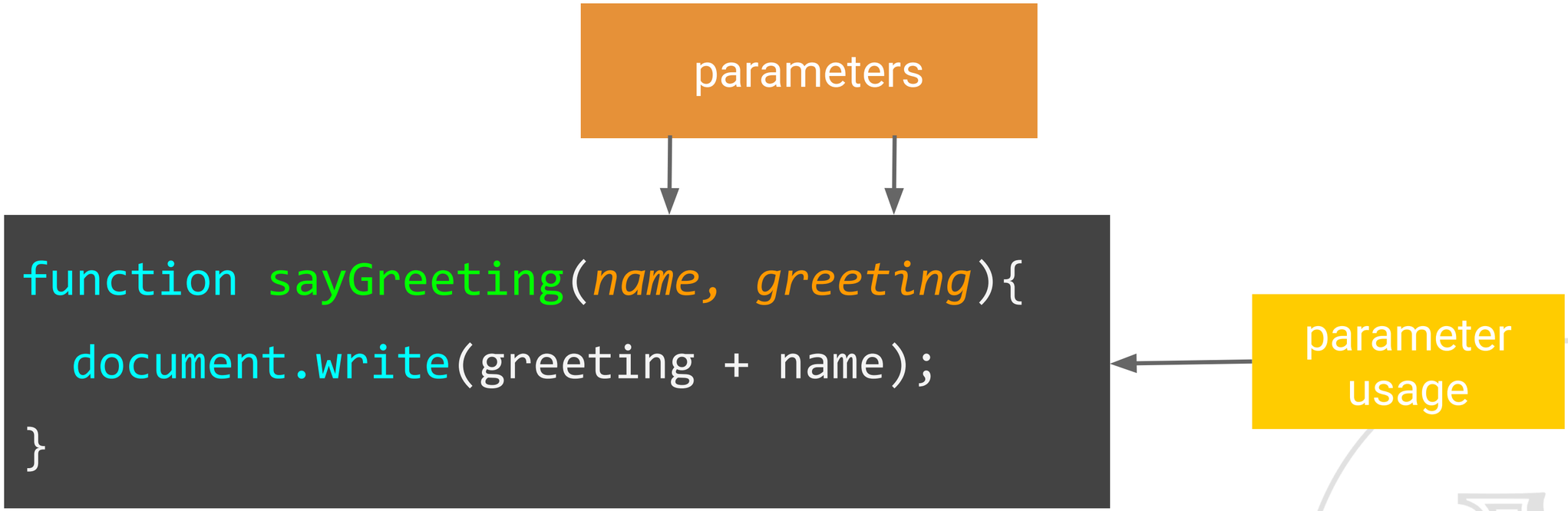
what are the () for?

Functions can also take in **parameters.** Parameters allow you to customize the behavior of a function.

```
function sayHello(name){
  document.write('hello ' + name '!');
}

sayHello('Sher Minn'); //this will print 'hello Sher Minn!'
sayHello('Audrey'); //this will print 'hello Audrey!'
```

Anatomy of a function: parameters



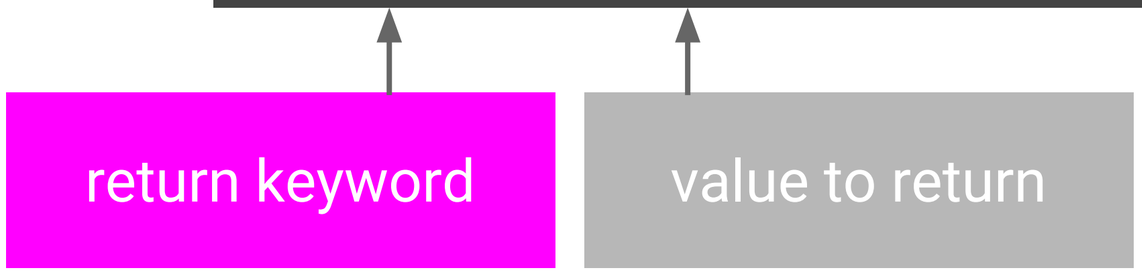
Functions can also return values

What goes in can come out different! Your function can return a value to be used elsewhere.

```
function addExcitement(name){
  return name + ‘!!’;
}
// save the return value in variable
const newName = addExcitement(‘Sher Minn’); // newName is now ‘Sher Minn!!’
document.write(newName); // this will display the name to us
```

Anatomy of a function: return

```
function sayGreeting(name, greeting){
  const result = greeting + name
  return result;
}
```



function scope

Variables that are created in functions only live within it. This is called a **local scope**.

```
function totalPets(numCats, numDogs){
   const total = numCats + numDogs;
   document.write(total);
}

totalPets(2, 3);
document.write(total); // what happens here?
```

function scope

Variables created outside of functions are in the **global scope**

```
let total;
function totalPets(numCats, numDogs){
  total = numCats + numDogs;
  document.write(total);
}
totalPets(2, 3);
document.write(total); // displays 5
```

function rules

1. Function naming rules are the same as variables
2. Functions must be created before they are used
3. Functions must be called with parentheses `()`
4. Function parameters and return statements are optional

Digresion: semicolons;

Do we always need them?

What do semicolons mean?

Recap

Concepts

- arithmetic operators
- variables
- boolean expressions
- functions

Data Types

- numbers
- strings
- boolean

The End

Thank you for coming!