Introduction

`console.log()`

console is a keyword that references to the console object

The `console.log()` method is used to log or print messages to the console. It can also be used to print objects and other info.

```
console.log('Hi there!');
// Prints: Hi there!
```

JavaScript

JavaScript is a programming language that powers the dynamic behavior on most websites. Alongside HTML and CSS, it is a core technology that makes the web run.

```
//line comment

/*
block comment
*/
```

Primitive data types:
- Numbers
- Boolean
- Strings ('...' or "...")
- Null
- Undefined
- Symbol
- Object

prefer "..." for visibility and using ' in the string

Methods

Methods return information about an object, and are called by appending an instance with a period `.` , the method name, and parentheses.

```
// Returns a number between 0 and 1
Math.random();
```

Libraries

Libraries contain methods that can be called by appending the library name with a period `.` , the method name, and a set of parentheses.

```
Math.random();
// ☝️ Math is the library
```

Numbers

Numbers are a primitive data type. They include the set of all integers and floating point numbers.

```
let amount = 6;
let price = 4.99;
```

String `.length`

The `.length` property of a string returns the number of characters that make up the string.

every string instance has property length

method vs. property:
method = function, property = value
thought we could have a function returning a value...

```
let message = 'good nite';
console.log(message.length);
// Prints: 9

console.log('howdy'.length);
// Prints: 5
```

Data Instances

When a new piece of data is introduced into a JavaScript program, the program keeps track of it in an instance of that data type. An instance is an individual case of a data type.

Booleans

Booleans are a primitive data type. They can be either `true` or `false`.

```
let lateToWork = true;
```

`Math.random()`

The `Math.random()` function returns a floating-point, random number in the range from 0 (inclusive) up to but not including 1.

```
console.log(Math.random());
// Prints: 0 - 0.9
```

notice the chained method calls and arguments
If method needs no argument, MUST put ()

`Math.floor()`

The `Math.floor()` function returns the largest integer less than or equal to the given number.

Number.isInteger(2017), not 2017.isInteger()
isInteger() is not a method of instances of Numbers, but a method of Number that JavaScript provides

```
console.log(Math.floor(5.95));
// Prints: 5
```

Single Line Comments

In JavaScript, single-line comments are created with two consecutive forward slashes `//`.

```
// This line will denote a comment
```

Null

Null is a primitive data type. It represents the intentional absence of value. In code, it is represented as `null`.

```
let x = null;
```

Strings

Strings are a primitive data type. They are any grouping of characters (letters, spaces, numbers, or symbols) surrounded by single quotes `'` or double quotes `"`.

```
let single = 'Wheres my bandit hat?';
let double = "Wheres my bandit hat?";
```

Arithmetic Operators

JavaScript supports arithmetic operators for:

- `+` addition
- `-` subtraction
- `*` multiplication
- `/` division
- `%` modulo

In JS, all Numbers are doubles, there is not Int.
I think / returns "Int" if can be divided exactly, and return "float" otherwise
So to do integer division just do:
Math.floor(x / y)

funny: try "123" + 1, and "123" - 1

```
// Addition
5 + 5
// Subtraction
10 - 5
// Multiplication
5 * 10
// Division
10 / 5
// Modulo
10 % 5
```

JavaScript is weakly typed, which means certain types are implicitly cast depending on the operation used.[38]

The binary + operator casts both operands to a string unless both operands are numbers. This is because the addition operator doubles as a concatenation operator
The binary - operator always casts both operands to a number
Both unary operators (+, -) always cast the operand to a number

Values are casted to strings like the following[38]:

Strings are left as-is
Numbers are converted to their string representation
Arrays have their elements cast to strings after which they are joined by commas (,)
Other objects are converted to the string [object Object] where Object is the name of the constructor of the object

codecademy

Multi-line Comments

In JavaScript, multi-line comments are created by surrounding the lines with `/*` at the beginning and `*/` at the end. Comments are good ways for a variety of reasons like explaining a code block or indicating some hints, etc.

```
/*
The below configuration must be
changed before deployment.
*/

let baseUrl =
'localhost/taxwebapp/country';
```

Assignment Operators

An assignment operator assigns a value to its left operand based on the value of its right operand. Here are some of them:

- `+=` addition assignment
- `-=` subtraction assignment
- `*=` multiplication assignment
- `/=` division assignment

```
let number = 100;

// Both statements will add 10
number = number + 10;
number += 10;

console.log(number);
// Prints: 120
```

String Interpolation

String interpolation is the process of evaluating string literals containing one or more placeholders (expressions, variables, etc).

It can be performed using template literals: `text ${expression} text`.

```
let age = 7;

// String concatenation
'Tommy is ' + age + ' years old.';

// String interpolation
`Tommy is ${age} years old.`;
```

^^called a "template literal"

use ` not ' or " when using string interpolation

Variables

Variables are used whenever there's a need to store a piece of data. A variable contains data that can be used in the program elsewhere. Using variables also ensures code re-usability since it can be used to replace the same value in multiple places.

```
const currency = '$';
let userIncome = 85000;

console.log(currency + userIncome + ' is
more than the average income.');
// Prints: $85000 is more than the
average income.
```

Undefined

`undefined` is a primitive JavaScript value that represents lack of defined value. Variables that are declared but not initialized to a value will have the value `undefined`.

```
var a;

console.log(a);
// Prints: undefined
```

"Tammy" is the value assigned to variable name, name is initialized with value "Tammy"

codecademy

Learn Javascript: Variables

A variable is a container for data that is stored in computer memory. It is referenced by a descriptive name that a programmer can call to assign a specific value and retrieve it.

```
// examples of variables
let name = "Tammy";
const found = false;
var age = 3;
console.log(name, found, age);
// Tammy, false, 3
```

We can initialize w/ or w/o value with var, let
We can only initialize w/ value with const
If we initialize w/o value, it's assigned to undefined

Declaring Variables

To declare a variable in JavaScript, any of these three keywords can be used along with a variable name:

- `var` is used in pre-ES6 versions of JavaScript.
- `let` is the preferred way to declare a variable when it can be reassigned.
- `const` is the preferred way to declare a variable with a constant value.

```
var age;
let weight;
const numberOfFingers = 20;
```

var is function scope.
let and const are block scope.

Function scope is within the function.
Block scope is within curly brackets.

if var is outside any function, it's global

To prevent leakage to parent environment, let is the keyword of choice

Template Literals

Template literals are strings that allow embedded expressions, `${expression}` . While regular strings use single `'` or double `"` quotes, template literals use backticks instead.

```
let name = "Codecademy";
console.log(`Hello, ${name}`);
// Prints: Hello, Codecademy

console.log(`Billy is ${6+8} years
old.`)
// Prints: Billy is 14 years old.
```

let Keyword

`let` creates a local variable in JavaScript & can be re-assigned. Initialization during the declaration of a `let` variable is optional. A `let` variable will contain `undefined` if nothing is assigned to it.

```
let count;
console.log(count); // Prints: undefined
count = 10;
console.log(count); // Prints: 10
```

const Keyword

A constant variable can be declared using the keyword `const` . It must have an assignment. Any attempt of re-assigning a `const` variable will result in JavaScript runtime error.

```
const numberOfColumns = 4;
numberOfColumns = 8;
// TypeError: Assignment to constant
variable.
```

String Concatenation

In JavaScript, multiple strings can be concatenated together using the `+` operator. In the example, multiple strings and variables containing string values have been concatenated. After execution of the code block, the `displayText` variable will contain the concatenated string.

```
let service = 'credit card';
let month = 'May 30th';
let displayText = 'Your ' + service  + '
bill is due on ' +  month + '.';

console.log(displayText);
// Prints: Your credit card bill is due
on May 30th.
```

typeof NAME will return the type of NAME

Notice JS is dynamically typed, so you can reassigned a variable of type Boolean to a value of type Number.