Syntax, Variables, and Data Types

David Tucker

CTO Consultant

@_davidtucker_ | davidtucker.net

Syntax

Syntax is a set of rules that defines how the collection of characters and symbols should be structured within a programming language.

JavaScript Statement

index.js

```
console.log("Hello");

console.log("1"); console.log("2"); console.log("3");
console.log("2");
console.log("3");

let firstName = "David";

let sum = 2 + 3;
console.log(sum);
```

JavaScript Keywords

abstract
arguments
await
boolean
break
byte
case
catch
char
class
const
continue
debugger
default
delete
do
double
else
enum
eval
export
extends
false
final
finally
float
for
function
goto
if
implements
import
in
instanceof
int
interface
let
long
native
new
null
package
private
protected
public
return
short
static
super
switch
synchronized
this
throw
throws
transient
true
try
typeof
var
void
volatile
while
with
yield

JavaScript Comments

index.js

```
// Creating the name variables
let firstName = "David";
let lastName = "Tucker";

// Logging name to the console
console.log(`Hello ${firstName} ${lastName}`);
```

Comments

Comments

In programming, a comment is text within the source code that is ignored by the engine running the actual application. As such, changing or removing the comment would have no effect on the actual application.

Why We Use Comments in JavaScript

To help other humans understand the code

To add metadata for tools analyzing the code

JavaScript Comments

index.js

```
// This is a single-line comment
let firstName = "David";

/*
  This is a multi-line comment. You can
  place more information in this syntax.
*/
let lastName = "Tucker";
```

Documentation and Configuration

index.js

```
/**
 * Joins the first and last names together.
 *
 * @param {string} firstName The first name
 * @param {string} lastName The last name
 * @return {string} The full name
 */
function getFullName(firstName, lastName) {
  return `${firstName} ${lastName}`;
};

let lastName = "Tucker"; // eslint-disable-line
```

Robert C. Martin, Clean Code

“Nothing can be quite so helpful as a well-placed comment. Nothing can clutter up a module more than frivolous dogmatic comments. Nothing can be quite so damaging as an old crufty comment that propagates lies and misinformation.”

Variables

Phil Karlton

“There are only two hard things in Computer Science: cache invalidation and naming things.”

Variable Naming Rules

- Names can contain letters, digits, and the $ and _ characters
- Cannot start a name with a digit
- Names are case sensitive

Variable Naming Best Practices

- Variable names usually start with a lowercase letter
- Names leverage camel case

JavaScript Variables

index.js

```
// Assigning variable values
let firstName = "David";
let lastName = "Tucker";
let numYearsEmployment = 6;
let dateBirth = new Date("January 1, 1982");

// Changing a variable created with let
numYearsEmployment = 7;

// Using const to create a constant
const title = "VP of Engineering";
title = "CTO"; // This would cause an error

// Using var to define a variable
var department = "Engineering"; // Not recommended
```

Garbage Collection

Garbage collection (GC) is a form of automatic memory management. The garbage collector attempts to reclaim memory which was allocated by the program, but is no longer referenced; such memory is called garbage.

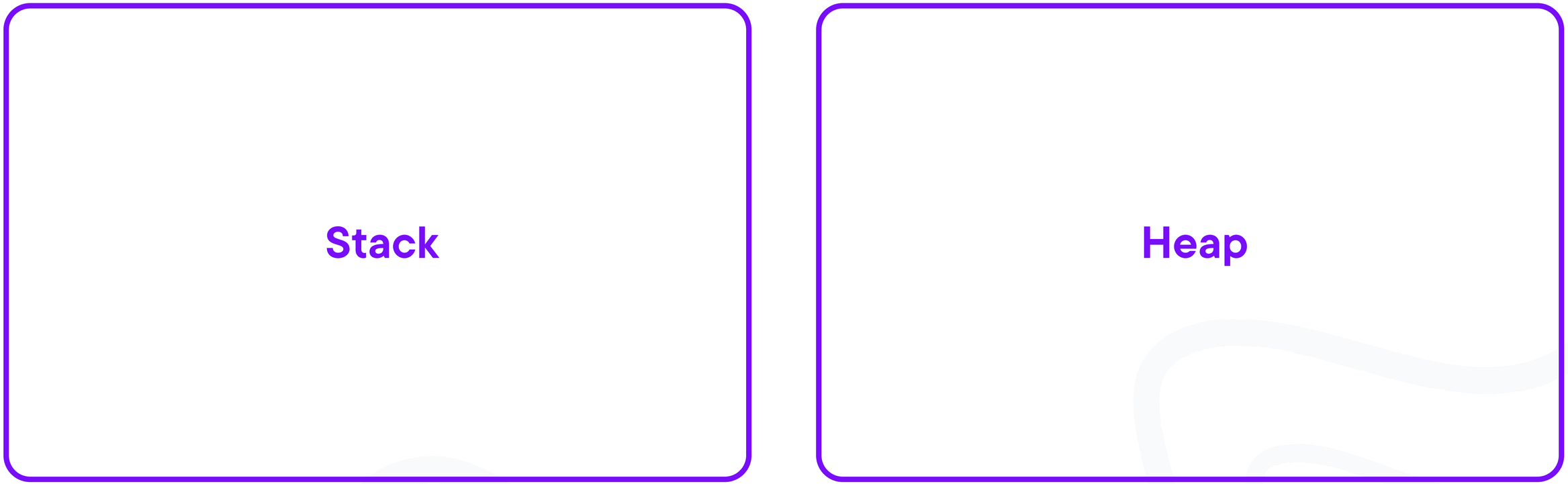
Wikipedia

Data Types

Dynamic Typing

JavaScript is a dynamic language with dynamic types. Variables in JavaScript are not directly associated with any particular value type, and any variable can be assigned (and re-assigned) values of all types.

Memory in JavaScript



JavaScript Data Types

Primitive Types

Objects

Primitives

In JavaScript, primitives are immutable, meaning they cannot be changed. These values have no methods or properties and their data is stored directly on the stack, since the engine knows exactly how much memory to allocate for that piece of data. Primitives are passed by value.

Objects

In JavaScript, objects are mutable, meaning that their contents can be changed without creating a new object. Objects can contain a collection of properties. They are passed by reference and stored on the heap.

Understanding the Data Types

Primitives	VS.	Objects
Boolean		Plain objects
Number		Functions
BigInt		Collection types (Array, Map, Set)
String		Dates
Symbol		Class instances
Null		Errors
Undefined		Promises

MDN Web Docs

“All primitive types, except null and undefined, have their corresponding object wrapper types, which provide useful methods for working with the primitive values.”

Primitive Wrapper Methods

index.js

```
// String
let name = "David";
let upperCaseName = name.toUpperCase(); // DAVID
```

Creating and Using Strings

Using Boolean Values

Storing Numeric Values

Number

The JavaScript Number type is a double-precision 64-bit binary format IEEE 754 value, like double in Java or C#. This means it can represent fractional values, but there are some limits to the stored number's magnitude and precision.

BigInt

BigInt values represent numeric values which are too large to be represented by the number primitive... It is created by appending n to the end of an integer literal, or by calling the BigInt() function (without the new operator) and giving it an integer value or string value.

Understanding Null and Undefined

Undefined and Null

In JavaScript undefined and null both represent something without a value but for different use cases. If you define a variable but don't assign a value, the value is undefined. If you want something to not have a value, you can assign it to null.

JavaScript Objects

Working with Dates

Classes

Classes

Classes are a template for creating objects. They encapsulate data with code to work on that data. Classes in JS are built on prototypes but also have some syntax and semantics that are unique to classes.

Classes

Classes are a template for creating objects. They encapsulate data with code to work on that data. Classes in JS are built on prototypes but also have some syntax and semantics that are unique to classes.

For more examples and explanations, see the Using classes guide.

Description

Defining classes

Classes are in fact "special functions", and just as you can define function expressions and function declarations, a class can be defined in two ways: a class expression or a class declaration.

Supported Class Features

- Constructors
- Static initialization blocks
- Public methods and fields
- Static methods and fields
- Private methods and fields
- Inheritance

Creating a JavaScript Class

index.js

```
class CalendarDay {
  // private fields
  #month;
  #day;
  #year;
  //constructor
  constructor(month, day, year) {
    this.month = month;
    this.day = day;
    this.year = year;
  }
  // public method
  toString() {
    return `${this.year}-${this.month+1}-${this.day}`;
  }
}
```

Using a JavaScript Class

index.js

```
// Using our new class
let day1 = new CalendarDay(2023, 0, 1);
console.log(day1.toString());

// Console returns "2023-1-1"
```