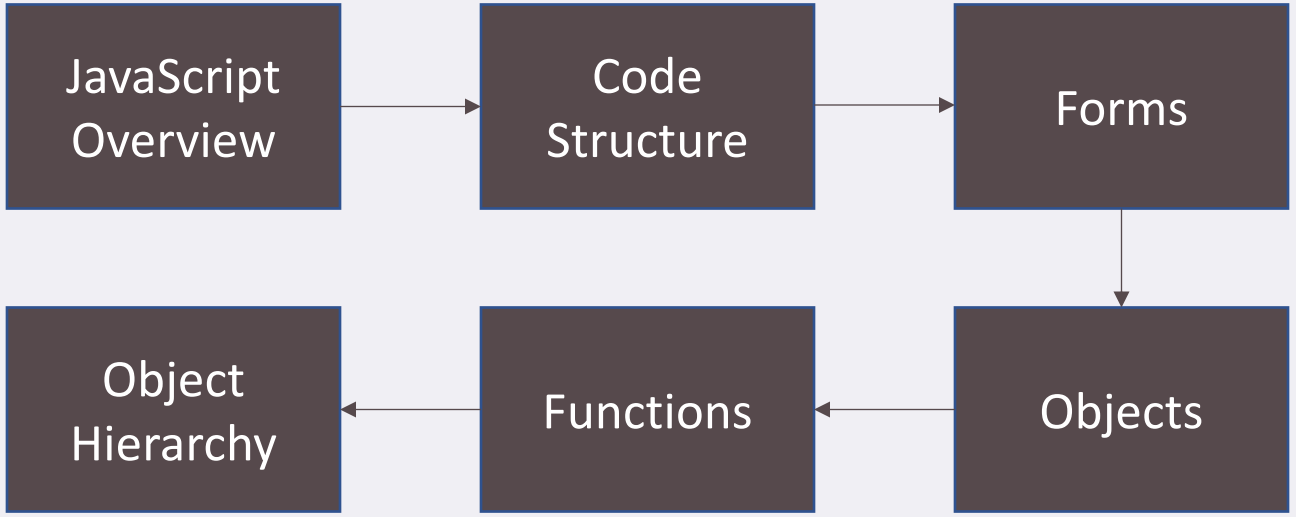
# JAVASCRIPT

THE SCRIPTING LANGUAGE FOR WEB PROGRAMMING

# TRAINING AGENDA



# SOME QUICK CONSIDERATIONS BEFORE WE START

You need to code along with me!

Try all the coding challenges!

If you want the concepts to stick, take notes. Notes on code syntax, notes on theory concepts, notes on everything!

Before moving on from a topic, make sure that you understand exactly what was covered.

If you have an error or a question, start by trying to solve it yourself!

Most importantly, have fun!

# JAVASCRIPT FUNDAMENTALS

A BRIEF INTRODUCTION TO JAVASCRIPT

# WHAT IS JAVASCRIPT?

JAVASCRIPT IS A *HIGH-LEVEL*, *OBJECT-ORIENTED*, *MULTI-PARADIGM PROGRAMMING LANGUAGE.*

We don’t have to worry about complex stuff like memory management

We can use different styles of programming

Based on objects, for storing most kinds of data

Instruct computer to do things

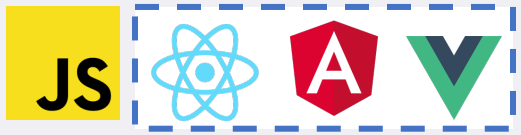
# JAVASCRIPT FEATURES

| | | |
|---|---|---|
| HIGH-LEVEL | PROTOTYPE-BASED OBJECT-ORIENTED | MULTI-PARADIGM |
| INTERPRETED OR JUST-IN-TIME COMPILED | DYNAMIC | SINGLE-THREADED |
| NON-BLOCKING EVENT LOOP | FIRST-CLASS FUNCTIONS | GARBAGE-COLLECTED |

# THERE IS NOTHING YOU CAN'T DO WITH JAVASCRIPT

FRONT-END APPS

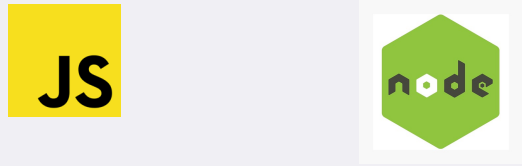
Dynamic effects and web applications in the browser




100% based on JavaScript.
They might go away,
but JavaScript won't!

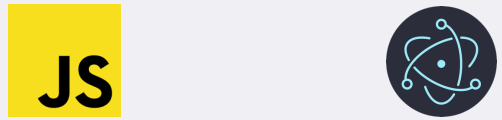
BACK-END APPS

Web applications on web servers

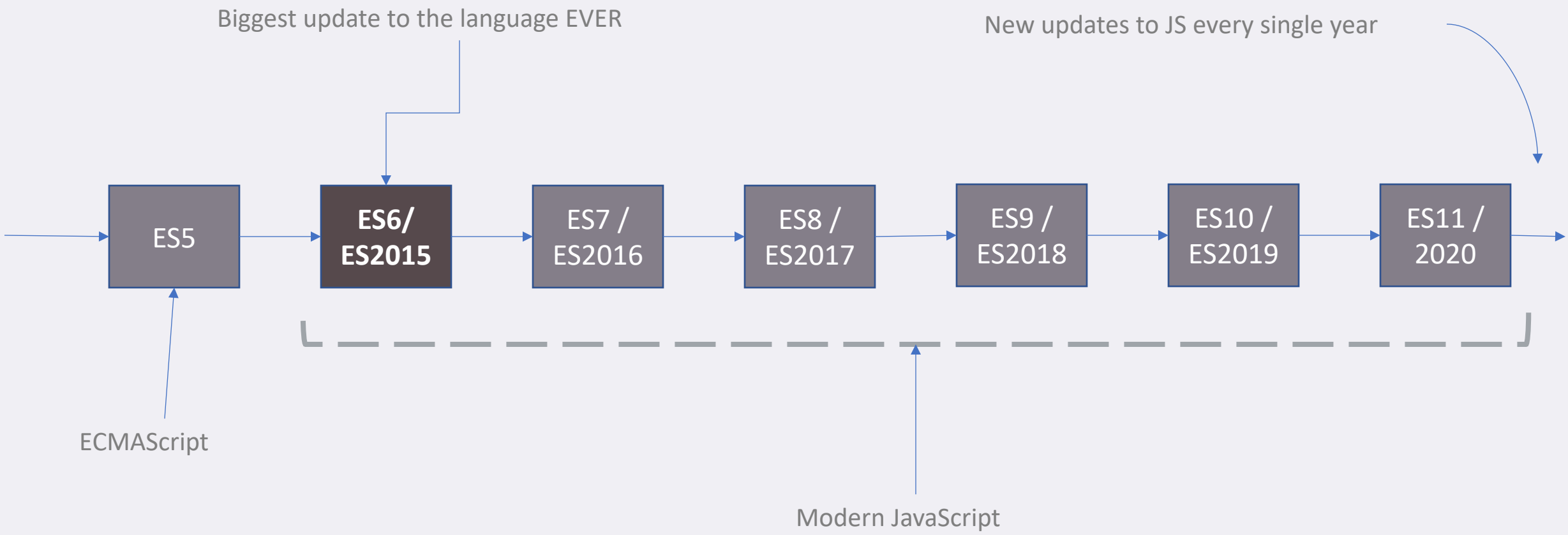



Native mobile applications


Native desktop applications



# JAVASCRIPT RELEASES...



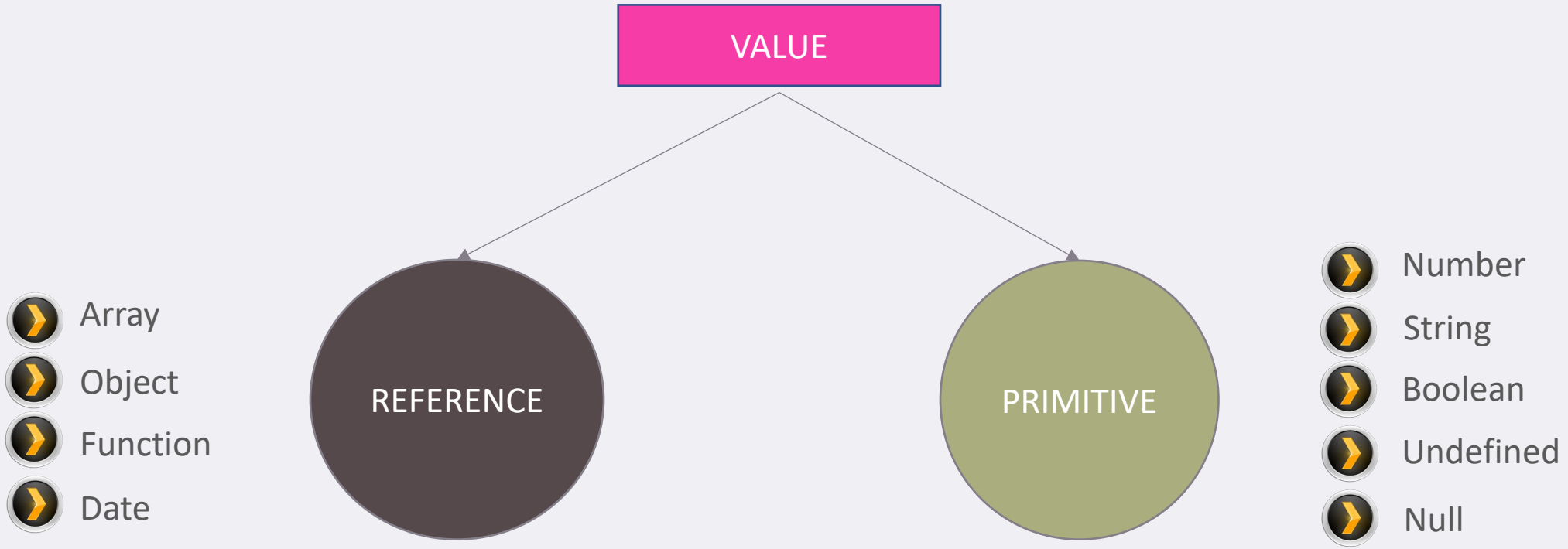
# A BRIEF HISTORY OF JAVASCRIPT

| | |
|---|---|
| 1995 | Brendan Eich creates the very first version of JavaScript in just 10 days. It was called Mocha. |
| 1996 | Mocha changes to LiveScript and then to JavaScript, in order to attract Java developers. |
| | Microsoft launches IE, copying JavaScript from Netscape and calling it JScript; |
| 1997 | With a need to standardize the language, ECMA releases ECMAScript 1 (ES1), the first official standard for JavaScript (ECMAScript is the standard, JavaScript the language in practice) |
| 2009 | ES5 (ECMAScript 5) is released with lots of great new features |
| 2015 | ES6/ES2015 (ECMAScript 2015) was released: the biggest update to the language ever! |
| | ECMAScript changes to an annual release cycle in order to ship less features per update |
| 2016 onwards | Release of ES2016 / ES2017 / ES2018 / ES2019 / ES2020 / ES2021 / … / ES2089 |

# DATA TYPES

LET’S GIVE IT A TYPE

# OBJECTS AND PRIMITIVES

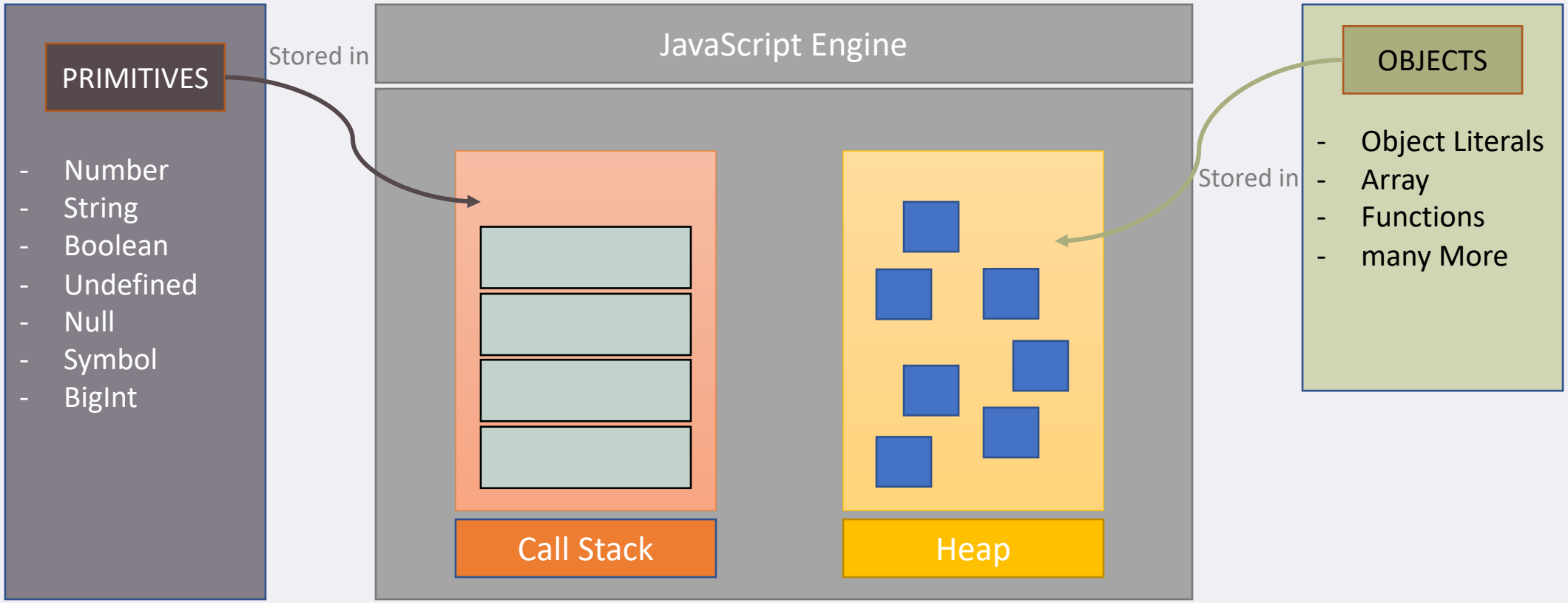


# THE 7 PRIMITIVE DATA TYPES

| | |
|---|---|
| Number | Floating point numbers. Used for decimals and integers |
| String | Sequence of characters. Used for text |
| Boolean | Logical type that can only be true or false. Used for taking decisions |
| Undefined | Value taken by a variable that is not yet defined (‘empty value’) |
| Null | Also means ‘empty value’ |
| Symbol (ES2015) | Value that is unique and cannot be changed |
| BigInt (ES2020) | Larger integers than the Number type can hold |

JavaScript has dynamic typing: We do not have to manually define the data type of the value stored in a variable. Instead, data types are determined automatically.

# PRIMITIVES, OBJECTS AND THE JAVASCRIPT ENGINE



# MISCELLANEOUS TOPICS

CONCEPTS FROM GROUND ZERO

# MISCELLANEOUS TOPICS

| | | |
|---|---|---|
| let, const & var | Operators & Precedence | Strings & Template Literals |
| Type Conversion & Coercion | Conditional & Loops | Document Object Model |

# DATA STRUCTURE

STORING DATA IN OBJECT AND ARRAY

# OBJECTS & ARRAYS

## Objects represented by Flower Brackets – { }

- The Object type represents one of JavaScript's data types
- It is used to store various keyed collections and more complex entities
- Objects can be created using the Object() constructor or the object initializer / literal syntax

## Arrays represented by Square Bracket - [ ]

- The Array object, as with arrays in other programming languages, enables storing a collection of multiple items under a single variable name, and has members for performing common array operations

# WORKING WITH ARRAY

A COLLECTION OF MULTIPLE ITEMS UNDER A SINGLE VARIABLE NAME

# ARRAY IN JAVASCRIPT

The Array object, as with arrays in other programming languages, enables storing a collection of multiple items under a single variable name, and has members for performing common array operations.

JavaScript arrays are resizable and can contain a mix of different data types

JavaScript arrays are not associative arrays and so, array elements cannot be accessed using strings as indexes

JavaScript arrays are zero-indexed

JavaScript array-copy operations create shallow copies

# WHICH ARRAY METHOD TO USE?

## I WANT …

### TO MUTATE ORIGINAL ARRAY

- Add methods -
  - push()
  - unshift()
- Remove methods -
  - Pop()
  - shift()
  - Splice()
- Others -
  - Reverse()
  - Sort()
  - Fill()

### A NEW ARRAY

- Computed from original-
  - Map()
- Filtered using condition-
  - Filter()
- Portion of original -
  - Slice()
- Adding original to other-
  - Concat()

### AN ARRAY INDEX

- Based on value-
  - indexOf()
- Based on test condition
  - findIndex()
- An array element -
  - Find()

# WHICH ARRAY METHOD TO USE?

## I WANT …

### KNOW IF ARRAY INCLUDES

- Based on value -
  - Includes()
- Based on test condition -
  - some()
  - every()
- Based on separator string-
  - Join()

### TO TRANSFORM TO VALUE

- Based on accumulator
  - reduce()
- Based on callback-
  - forEach()

# FUNCTIONS

FIRST CLASS CITIZENS IN JAVASCRIPT

# FUNCTIONS

**Function declaration**
Function that can be used before it's declared

**Function expression**
Essentially a function value stored in a variable

**Arrow function**
Great for a quick one-line functions. (more later...)

```
function calcAge(birthYear) {
  return 2037 - birthYear;
}

const calcAge = function (birthYear) {
  return 2037 - birthYear;
};

const calcAge = birthYear ⇒ 2037 - birthYear;
```

Three different ways of writing functions, but they all work in a similar way -
Receive input data, transform data, and then output data.

# FIRST-CLASS AND HIGHER-ORDER FUNCTIONS

## FIRST-CLASS FUNCTIONS

JavaScript treats functions as first-class citizens.

This means that functions are simply values

Functions are just another “type” of object

Store functions in variables or properties

Pass functions as arguments to OTHER functions

Return functions FROM functions

## HIGHER-ORDER FUNCTIONS

A function that receives another function as an argument, that returns a new function, or both

This is only possible because of first-class functions

Function that receives another function

Function that returns new function

# DOM AND EVENTS

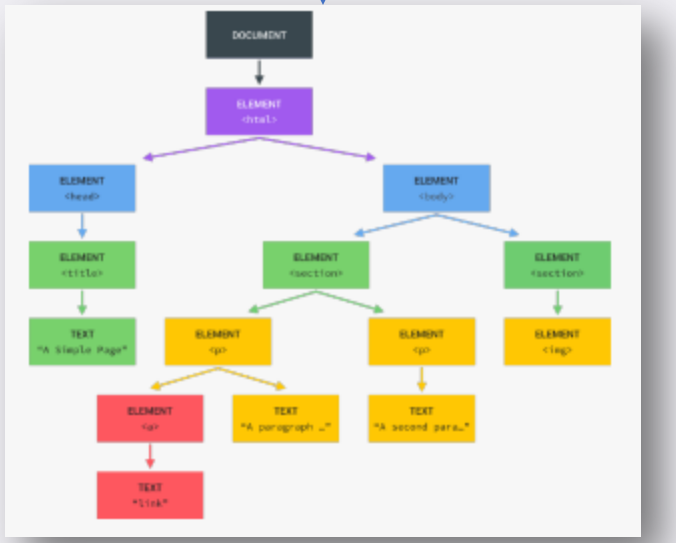
JAVASCRIPT IN THE BROWSER

# WHAT IS THE DOM?

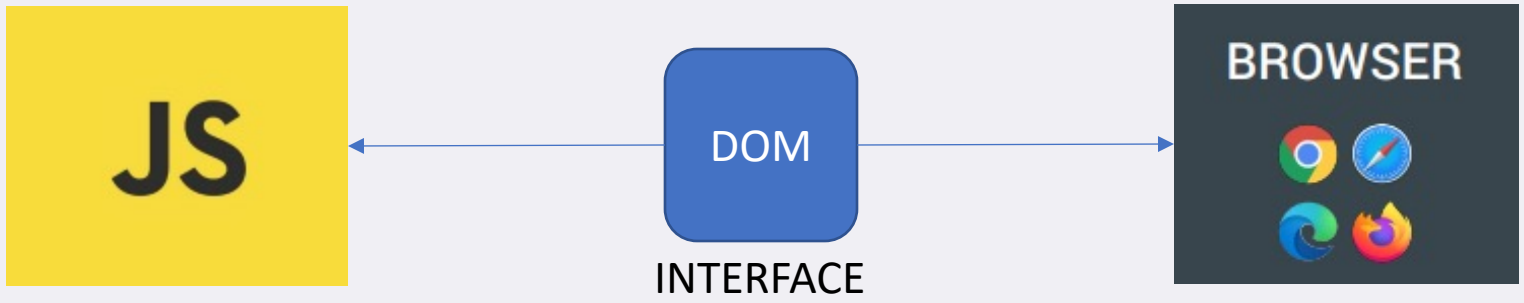
**DOCUMENT OBJECT MODEL:** STRUCTURED REPRESENTATION OF HTML DOCUMENTS. ALLOWS JAVASCRIPT TO ACCESS HTML ELEMENTS AND STYLES TO MANIPULATE THEM

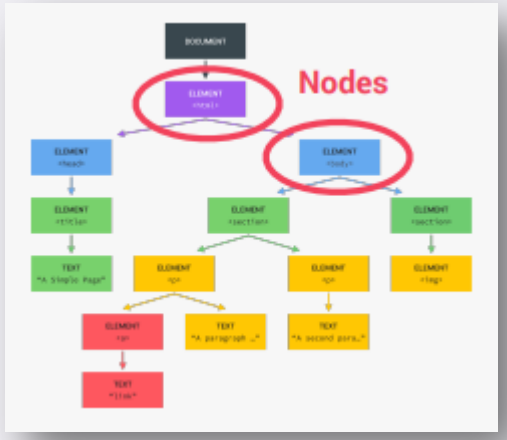
Change text, HTML attributes, and even CSS styles

Tree structure, generated by browser on HTML load



# DOM IN DETAIL



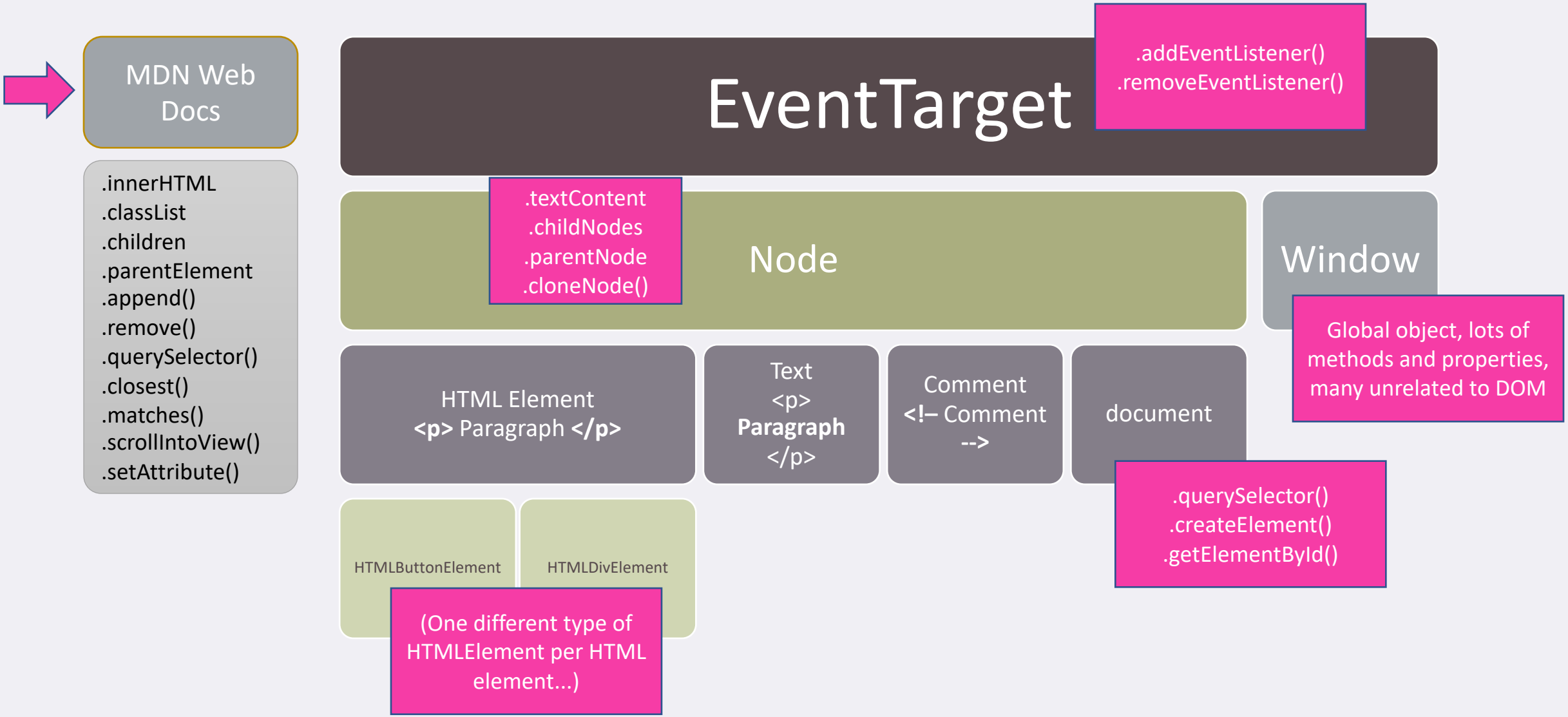

Allows us to make JavaScript interact with the browser

We can write JavaScript to create, modify and delete HTML elements set styles, classes and attributes; and listen and respond to events

DOM tree is generated from an HTML document, which we can then interact with

DOM is a very complex API that contains lots of methods and properties to interact with the DOM tree

# HOW THE DOM API IS ORGANIZED BEHIND THE SCENES



# ASYNCHRONOUS JAVASCRIPT

PROMISES, ASYNC/ AWAIT AND MORE

# SYNCHRONOUS CODE

```
const p = document.querySelector("#paragraph");
p.textContent = "Hello World";
alert("Who's this?");
p.style.color = "red";
```

Most code is synchronous

Synchronous code is executed line by line

Each line of code waits for previous line to finish

Long-running operations block code execution

# ASYNCHRONOUS CODE

```
const p = document.querySelector("#paragraph");
setTimeout(() => {
    alert("Who's this?")
}, 1000);
p.style.color = "red";
```

Asynchronous code is executed after a task that runs in the “background” finishes

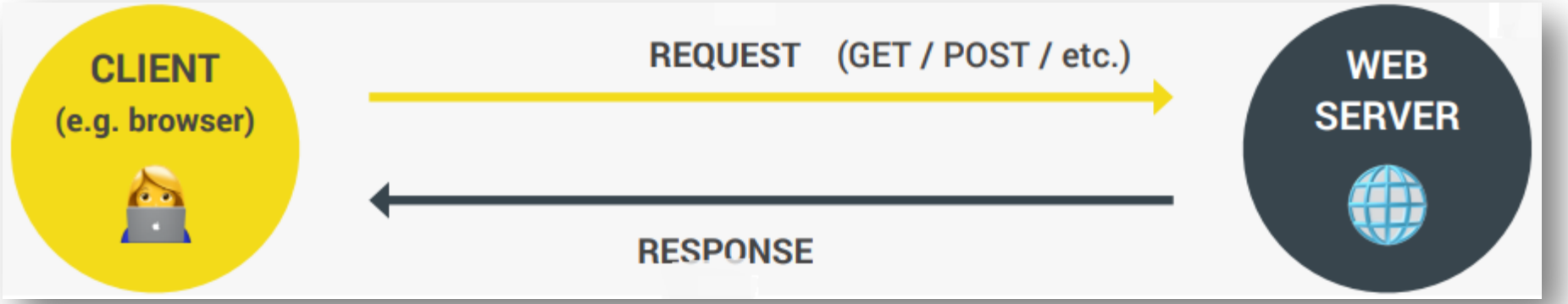
Asynchronous code is non-blocking

Execution doesn’t wait for an asynchronous task to finish its work

Callback functions alone do NOT make code asynchronous

# WHAT ARE AJAX CALLS?

Asynchronous JavaScript And XML: Allows us to communicate with remote web servers in an asynchronous way. With AJAX calls, we can request data from web servers dynamically



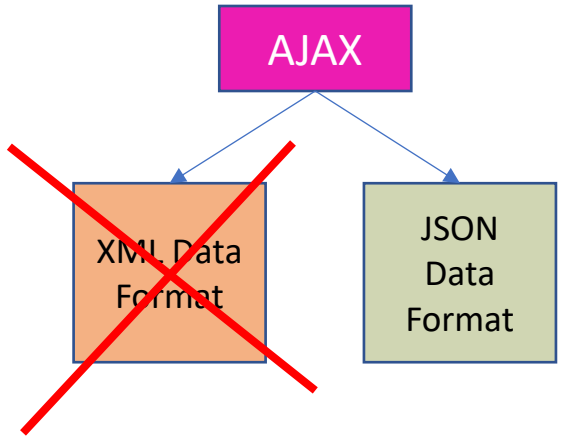
# WHAT IS AN API?

Piece of software that can be used by another piece of software, in order to allow applications to talk to each other

There are be many types of APIs in web development: DOM API, Geolocation API, Books API, Online API etc

“Online” API: Application running on a server, that receives requests for data, and sends data back as response

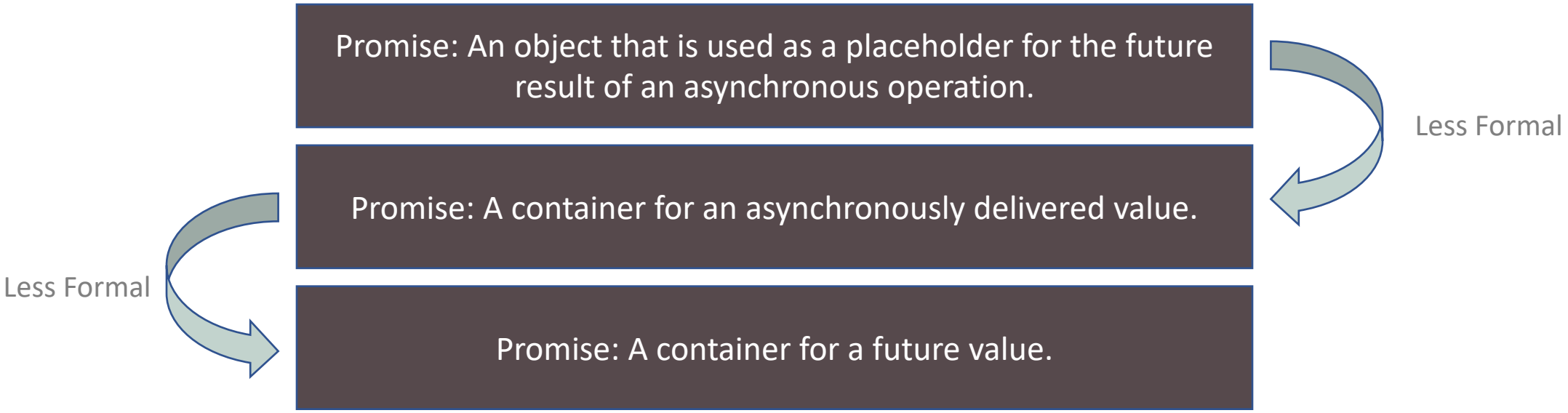
We can build our own web APIs (requires back-end development, e.g. with node.js) or use 3rd-party APIs




Examples of 3rd Party APIs

- Weather data
- Data about countries
- Flights data
- Currency conversion data
- APIs for sending email or SMS
- Google Maps
- Millions of possibilities.

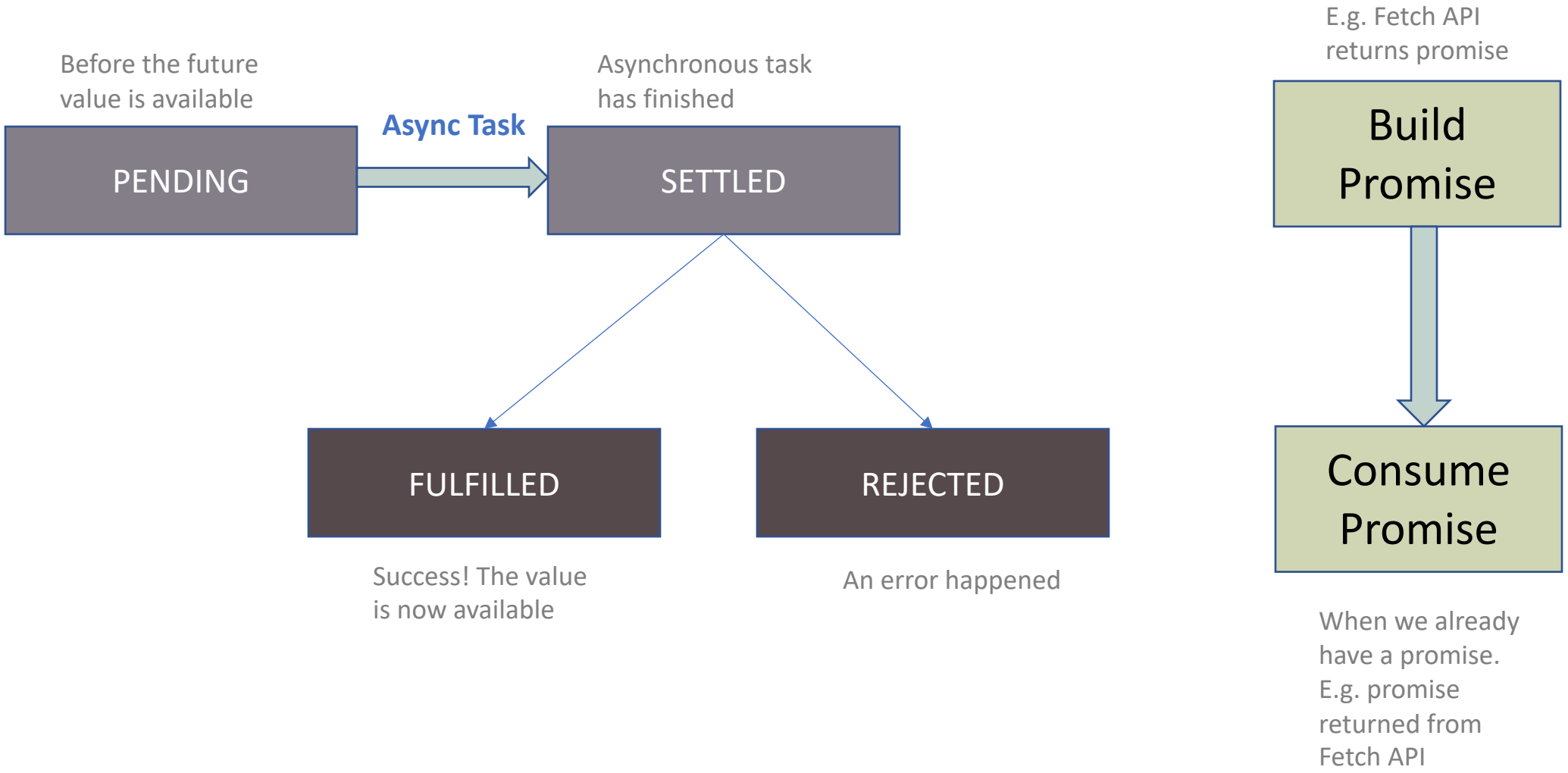
# WHAT ARE PROMISES?




We no longer need to rely on events and callbacks passed into asynchronous functions to handle asynchronous results;

Instead of nesting callbacks, we can chain promises for a sequence of asynchronous operations: escaping callback hell

# THE PROMISE LIFECYCLE



# MODERN JAVASCRIPT DEVELOPMENT

WRITING CLEAN AND MODERN JAVASCRIPT

# REVIEW: MODERN AND CLEAN CODE

## READABLE CODE

- Write code so that others can understand it
- Write code so that you can understand it in 1 year
- Avoid too “clever” and overcomplicated solutions
- Use descriptive variable names: what they contain
- Use descriptive function names: what they do

## FUNCTIONS

- Generally, functions should do only one thing
- Don’t use more than 3 function parameters
- Use default parameters whenever possible
- Generally, return same data type as received
- Use arrow functions when they make code more readable

# REVIEW: MODERN AND CLEAN CODE

## GENERAL

- Use DRY principle (refactor your code)
- Don’t pollute global namespace, encapsulate instead
- Don’t use var
- Use strong type checks (=== and !==)

## OOP

- Use ES6 classes
- Encapsulate data and don’t mutate it from outside the class
- Implement method chaining
- Do not use arrow functions as methods (in regular objects)

# REVIEW: MODERN AND CLEAN CODE

## AVOID NESTED CODE

- Use early return (guard clauses)
- Use ternary (conditional) or logical operators instead of if
- Use multiple if instead of if/else-if
- Avoid for loops, use array methods instead
- Avoid callback-based asynchronous APIs

## ASYNCHRONOUS CODE

- Consume promises with async/await for best readability
- Whenever possible, run promises in parallel (Promise.all)
- Handle errors and promise rejections

# REFERENCES

**READING MATERIAL**

- https://developer.mozilla.org/en-US/docs/Web/JavaScript
- https://javascript.info

**VIDEO LINKS**

- https://www.youtube.com/watch?v=POPLF-Qc0OU&list=PLsyeobzWxl7rrvgG7MLNIMSTzVCDZZcT4&index=2
- https://www.youtube.com/watch?v=chx9Rs41W6g