

JAVASCRIPT

THE SCRIPTING LANGUAGE FOR WEB PROGRAMMING

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

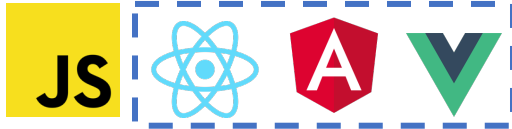
FIRST-CLASS
FUNCTIONS

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!

Native mobile
applications



BACK-END APPS

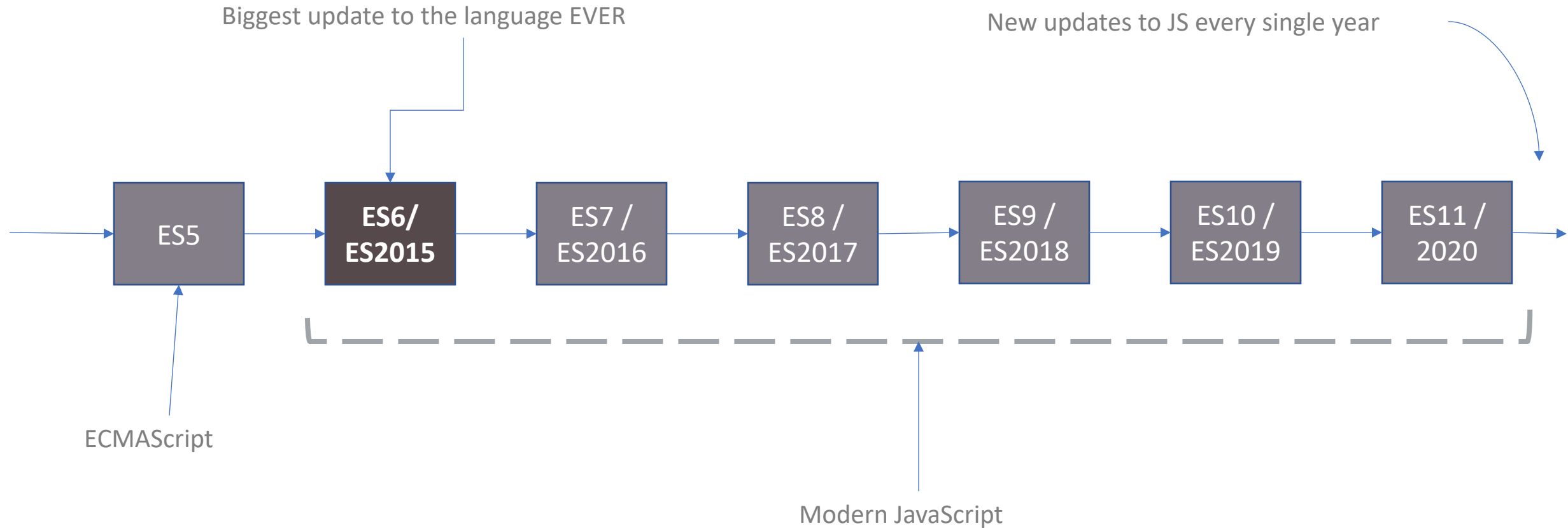
Web applications on
web servers



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

JAVASCRIPT NEW FEATURES

ES6 AND MORE

ES6 & MORE – NEW FEATURES

ARROW
FUNCTION

DESTRUCTURING

REST / SPREAD

TEMPLATE
LITERALS

BLOCK SCOPING

MAP/SET

CLASSES

DEFAULT
PARAMETERS

ARROW FUNCTION =>

Arrow functions are handy for one-liner functions

Arrow Function Flavors

Without Curly braces
(...args) => expression

With curly braces
(...args) => { body

Limitations

Don't have this keyword

Don't have arguments keyword

Cant call with new operator

BLOCK SCOPING

Restricts the scope of variables to the nearest curly braces { }

Variables Types

const: converts the variable to a constant

let: for all type of variables

const !== immutable

REST / SPREAD OPERATOR

REST

A function can be called with any number of arguments, no matter how it is defined.

The rest parameters must be at the end.

Usage: create functions that accept any number of arguments

SPREAD

Spread operator looks similar to rest parameters, also using (...), but does quite the opposite.

It is used in the function call, it “expands” an iterable object into the list of arguments.

Usage: pass an array to functions that normally require a list of many

DESTRUCTURING

Destructuring assignment is a special syntax that allows us to “unpack” arrays or objects into a bunch of variables.

OBJECT DESTRUCTURING

- We have an existing object at the right side, that we want to split into variables.

ARRAY DESTRUCTURING

- The array is destructured into variables, but the array itself is not modified.

NESTED DESTRUCTURING

- If an object or an array contain other objects and arrays, we can use more complex left side patterns to extract deeper portions.

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

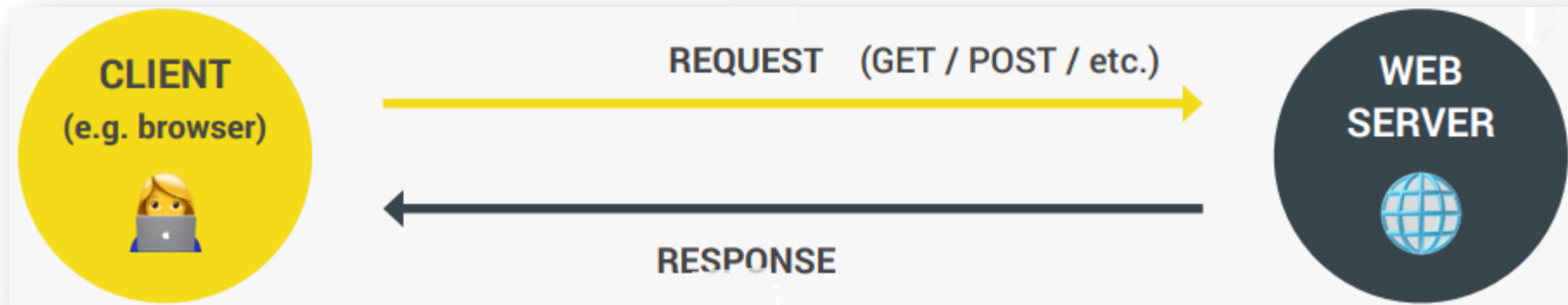
Asynchronous code is non-blocking

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



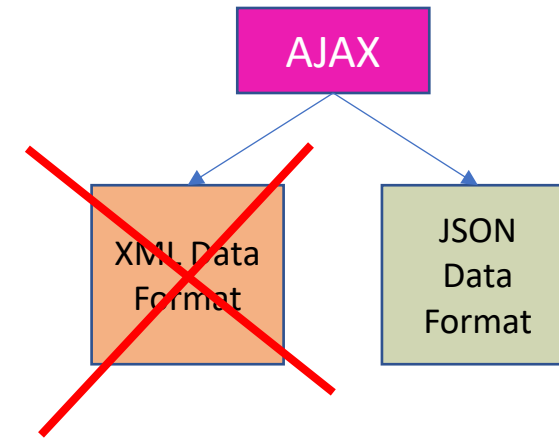
WHAT IS AN API?

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?

Promise: An object that is used as a placeholder for the future result of an asynchronous operation.

Promise: A container for an asynchronously delivered value.

Promise: A container for a future value.

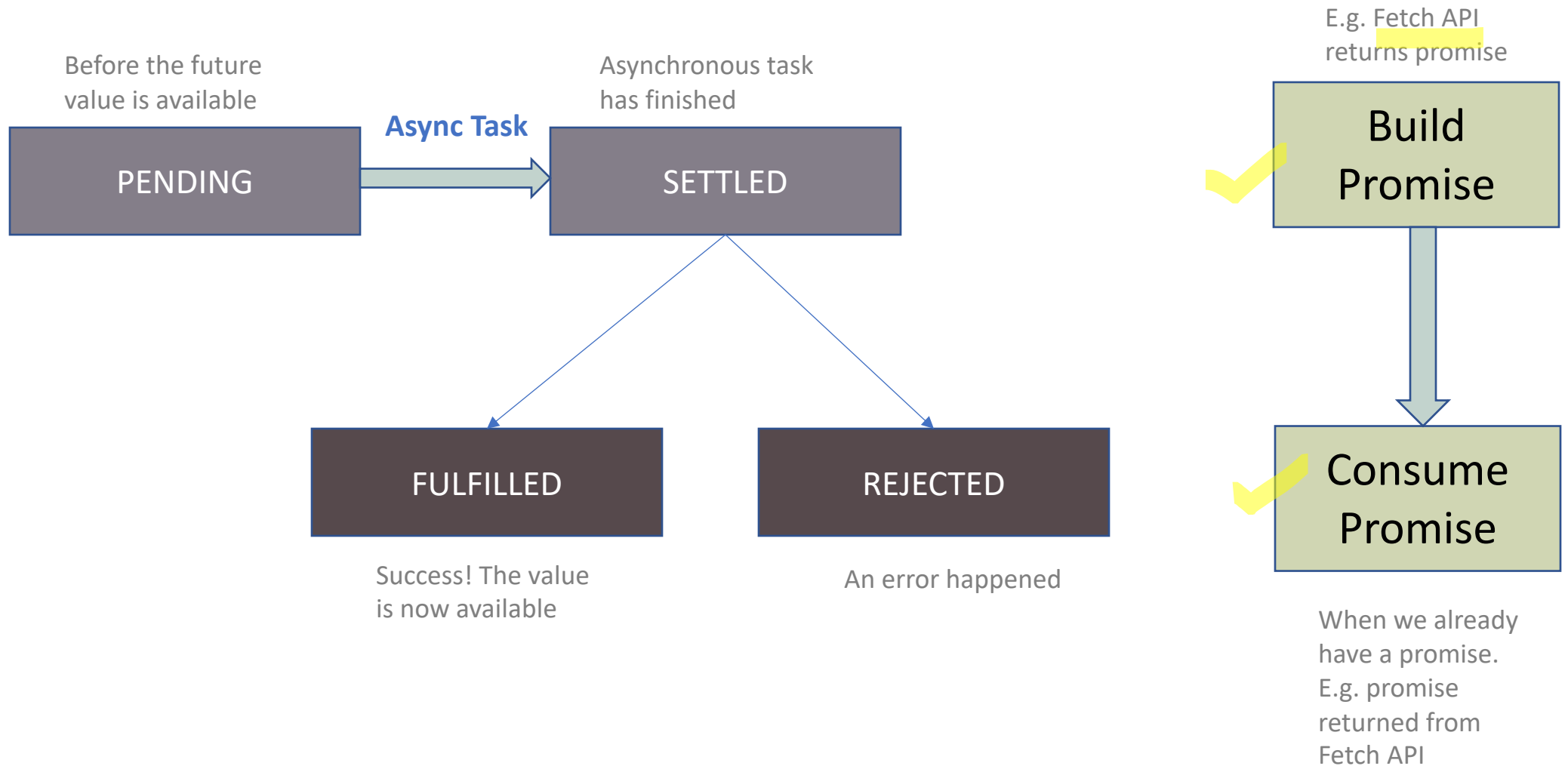
Less Formal

Less Formal

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

VIDEO LINKS

- https://www.youtube.com/watch?v=NCwa_xi0Uuc
- <https://www.youtube.com/watch?v=nZ1DMMsyVyl>