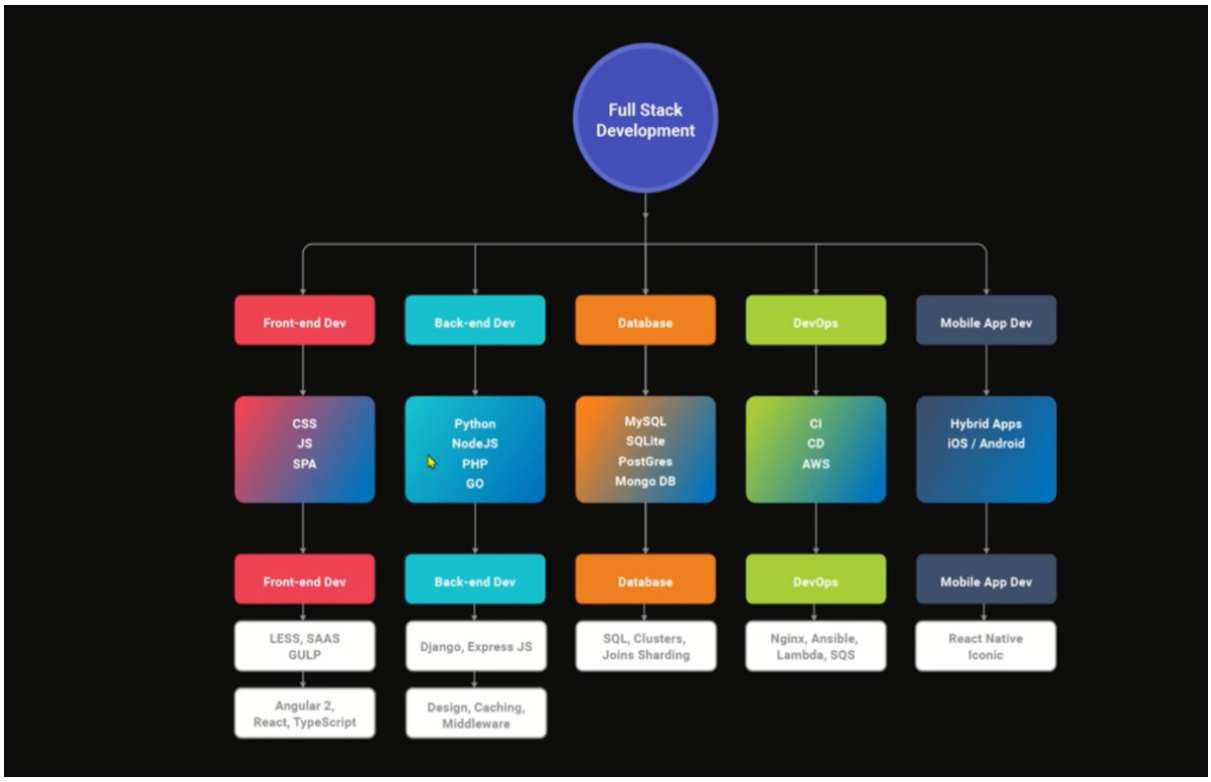
Week4- JavaScript

Introduction



What is JavaScript?

JavaScript was initially created to “make web pages alive”.

The programs in this language are called *scripts*. They can be written right in a web page’s HTML and run automatically as the page loads.

Scripts are provided and executed as plain text. They don’t need special preparation or compilation to run.

In this aspect, JavaScript is very different from another language called Java.

Why is it called JavaScript?
When JavaScript was created, it initially had another name: "LiveScript". But Java was very popular at that time, so it was decided that positioning a new language as a "younger brother" of Java would help.

But as it evolved, JavaScript became a fully independent language with its own specification called ECMAScript, and now it has no relation to Java at all.

Today, JavaScript can execute not only in the browser, but also on the server, or actually on any device that has a special program called the JavaScript engine.

The browser has an embedded engine sometimes called a "JavaScript virtual machine".

Different engines have different "codenames". For example:

- V8 – in Chrome and Opera.
- SpiderMonkey – in Firefox.
- ...There are other codenames like "Chakra" for IE, "ChakraCore" for Microsoft Edge, "Nitro" and "SquirrelFish" for Safari, etc.

The terms above are good to remember because they are used in developer articles on the internet. We'll use them too. For instance, if "a feature X is supported by V8", then it probably works in Chrome and Opera.

What makes JavaScript unique?

There are at least *three* great things about JavaScript:

- Full integration with HTML/CSS.
- Simple things are done simply.

- Support by all major browsers and enabled by default.

JavaScript is the only browser technology that combines these three things.

That's what makes JavaScript unique. That's why it's the most widespread tool for creating browser interfaces.

That said, JavaScript also allows to create servers, mobile applications, etc.

Languages "over" JavaScript

The syntax of JavaScript does not suit everyone's needs. Different people want different features.

That's to be expected, because projects and requirements are different for everyone.

So recently a plethora of new languages appeared, which are *transpiled* (converted) to JavaScript before they run in the browser.

Modern tools make the transpilation very fast and transparent, actually allowing developers to code in another language and auto-converting it "under the hood".

Examples of such languages:

- CoffeeScript is a "syntactic sugar" for JavaScript. It introduces shorter syntax, allowing us to write clearer and more precise code. Usually, Ruby devs like it.
- TypeScript is concentrated on adding "strict data typing" to simplify the development and support of complex systems. It is developed by Microsoft.
- Flow also adds data typing, but in a different way. Developed by Facebook.
- Dart is a standalone language that has its own engine that runs in non-browser environments (like mobile apps), but also can be transpiled to JavaScript. Developed by Google.
- Brython is a Python transpiler to JavaScript that enables the writing of applications in pure Python without JavaScript.
- Kotlin is a modern, concise and safe programming language that can target the browser or Node.

There are more. Of course, even if we use one of transpiled languages, we should also know JavaScript to really understand what we're doing.

Summary

01. What is JavaScript?

- JavaScript is a programming language that adds interactivity to Web pages
- JavaScript is a scripting language
- A JavaScript script is a program that is included on an HTML page
- JavaScript scripts are text on Web pages that are interpreted and run by Web browsers
- JavaScript is initially named and developed as `LiveScript` at Netscape Navigator Corporation
- JavaScript is `not Java`
- Due to Java wave or Java popularity and buzz, `LiveScript` renamed to `JavaScript`

02. What can JavaScript do?

- Create an active User Interface
- Control the user experience based on Day, Date, Time and Browser, etc
- Validate user input on forms
- Create custom HTML pages on the fly/dynamically
- Control Web browsers interactivity and behaviors

03. What can't JavaScript do?

- JavaScript can't talk to a Database (Its possible with NodeJs)
- JavaScript can't write to files (Its possible with NodeJs)
- Keep track of state (except with cookies)

JavaScript basics

Code structure

Variables

Data types

Interaction: alert, prompt, confirm

Basic operators, maths

Functions

Conditional branching: if, '?'

Loops: while and for

Array

Object

Exercice Objects