* JS Flavours

Graphical user interface, application

Description automatically generated

* Navigating the JS landscape
* Selecting transcript lines in this section will navigate to timestamp in the video
* .
* First off, at the core of all of this sits JavaScript itself, often colloquially referred to as **Vanilla JavaScript .**
* This is the scripting language itself and the scripting language for the web .
* This is what you write when you write JavaScript and what we'll cover in this course, everything else is built on top of JavaScript itself, Vanilla JavaScript or JavaScript Proper .
* The browser implementation specification for JavaScript is called **ECMAScript,** named after the European Computer Manufacturers Association, ECMA .
* This group is in charge of defining the specification for JavaScript .
* Now ECMAScript is not the language itself, but the official description of how the language should be interpreted by browsers .
* This is meant to ensure that when you write JavaScript, it behaves the same way no matter what browser is used .
* *When people say they write ECMAScript or ES6 or 2015 or 2017 or 2020 or something like that, they usually mean they're writing JavaScript that is at the cutting edge of development, effectively, the JavaScript of the future .*
* When people write some version of ECMAScript, they typically use a tool called **Babel** to convert their code back into JavaScript the current browser can read .
* JavaScript is an opinionated coding language and some developers either don't like JavaScript's opinions or they need specific features not provided by JavaScript .
* To get around this issue, they've developed abstracted versions of JavaScript with these additional features added .
* The most popular of these is TypeScript .
* TypeScript is heavily used in modern JavaScript frameworks .
* And you can usually spot TypeScript by looking at the file name .
* TypeScript files typically have the .
* ts file name extension .
* Think of TypeScript and CoffeeScript and all the variations of JavaScript as dialects of JavaScript .
* The modern web runs on JavaScript frameworks .
* These are tools written in JavaScript to render content on the web in the form of applications .

Graphical user interface, text

Description automatically generated

* The most popular of these frameworks are React or React .
* js, Vue or Vue .
* js and Angular or AngularJS .
* They all add an abstraction layer on top of JavaScript to do things in a more streamlined and efficient way .
* As an example, React uses JSX or JavaScript XML which simplifies how we mix JavaScript and HTML .
* With JavaScript frameworks come **build the tools .**
* These are helper applications, typically automated that run on your computer or in the cloud to turn your human written and readable JavaScript code into code optimized for the browser .

Graphical user interface, text

Description automatically generated

* I've already mentioned Babel .
* Other common tools include NPM, WebPack, Grunt, Gulp and so on .
* Finally, in the last several years, **JavaScript has migrated from the browser to the server,** so we can now write Java JavaScript to run on the backend as well as the front end of our applications .

Graphical user interface, text

Description automatically generated

* Node .
* js is the ubiquitous JavaScript server runtime used for this purpose .
* And when you work with modern JavaScript, you'll be interfacing with Node .
* js on your computer all the time, through the Node package manager or NPM .
* Considering all of this, here is your takeaway .
* **JavaScript is the core underlying language for anything and everything we've just talked about .**
* And understanding pure vanilla JavaScript is the key to understanding all this other stuff .
* So learn JavaScript first, then specialize and optimize your knowledge .