History of JavaScript

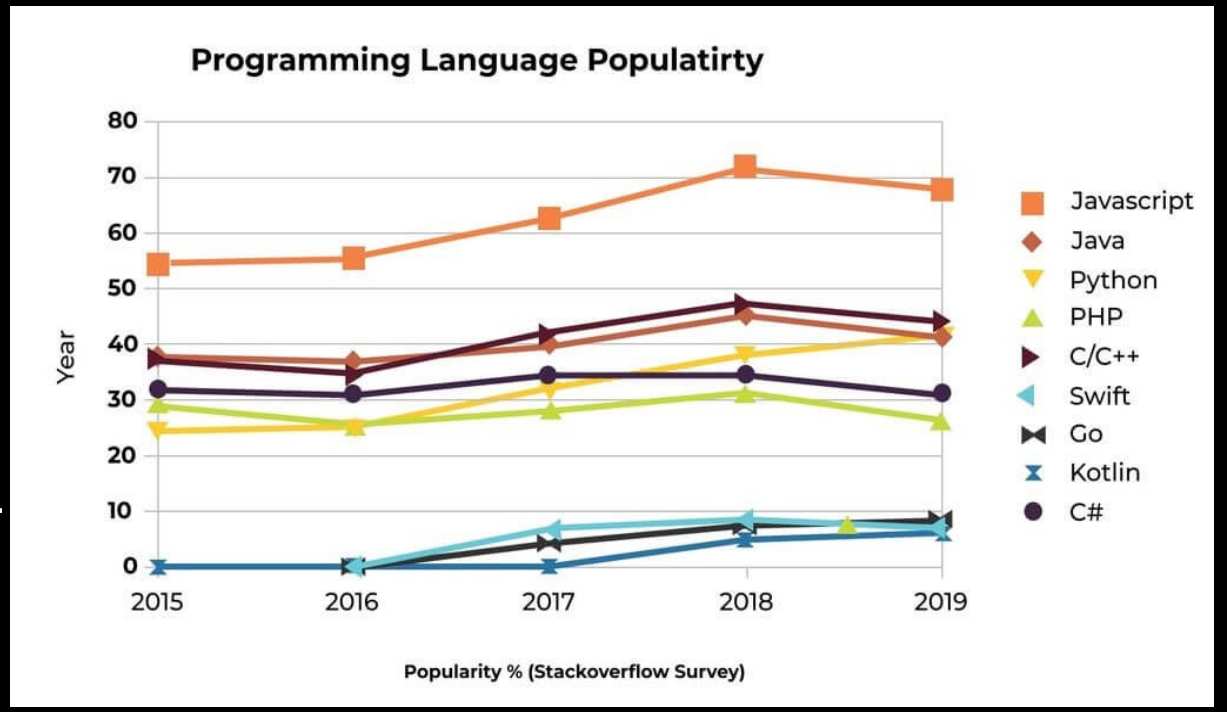
1. JavaScript was originally named Mocha, then renamed to LiveScript, and finally JavaScript to capitalize on the popularity of Java at the time.
2. JavaScript was created by Brendan Eich in 1995 while he was working at Netscape Communications Corporation.
3. JavaScript is an interpreted language, meaning it is executed line by line.

What is JavaScript

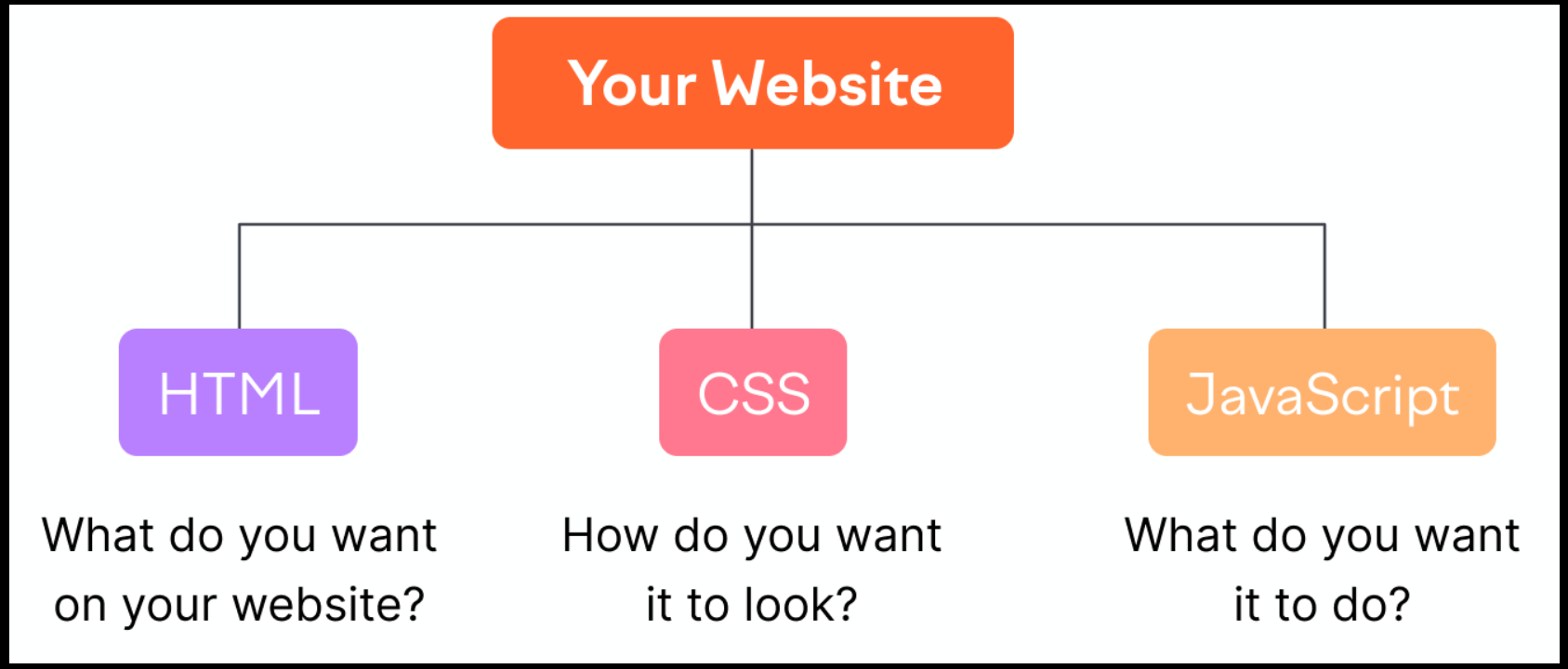
1. JavaScript is a high-level, dynamic programming language commonly used for creating interactive effects within web browsers.
2. Actions: Enables interactivity.
3. Updates: Alters page without reloading.
4. Events: Responds to user actions.
5. Data: Fetches and sends info to server.

Popularity of JavaScript

1. JavaScript is one of the most popular programming languages in the world, consistently ranking at the top in surveys and job listings.
2. Average JavaScript Dev Salary in India:
   - Entry-Level (0-1 year): Around ₹3,50,000 per annum.
   - Mid-Level (2-5 years): Approximately ₹6,00,000 to ₹10,00,000 per annum.
   - Experienced (5+ years): Can exceed ₹10,00,000 per annum, potentially reaching up to ₹20,37,500.

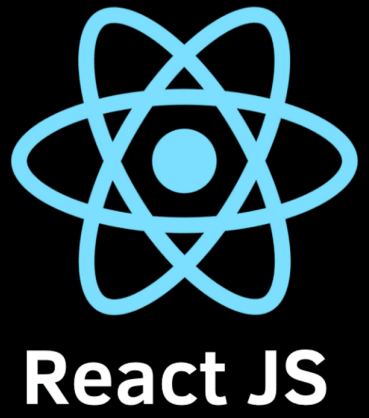


Applications of JavaScript



1. HTML: Defines the structure and content of the website.
2. CSS: Specifies the appearance and layout of the website.
3. JavaScript: Adds interactivity and dynamic behavior to the website.

Applications of JavaScript



Web Applications:

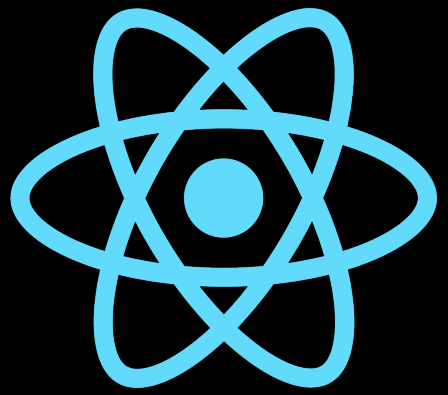
- React: A library for building user interfaces, maintained by Facebook.
- Angular: A platform for building mobile and desktop web applications, maintained by Google.
- Vue.js: A progressive framework for building user interfaces.

Applications of JavaScript

Server-Side:

- Node.js: Allows JavaScript to run on the server, used for building scalable network applications.
- Express.js: A minimal and flexible Node.js web application framework.

Applications of JavaScript



React Native

Mobile Applications:

- React Native: Builds mobile apps using JavaScript and React.
- Ionic: A framework for building cross-platform mobile apps with web technologies like HTML, CSS, and JavaScript.
- NativeScript: Allows building native iOS and Android apps using JavaScript or TypeScript.

Applications of JavaScript

BuildTools:

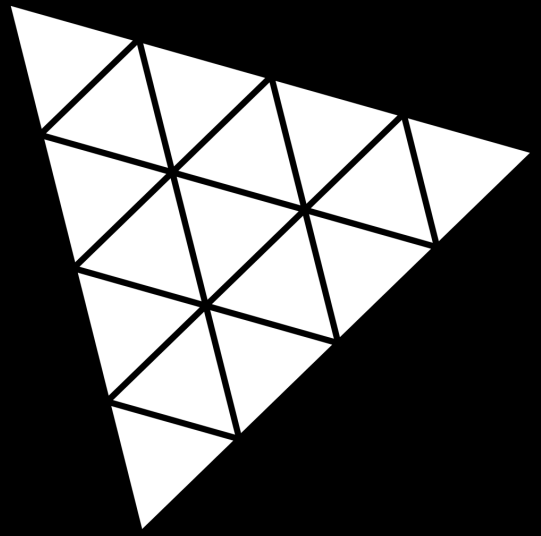
- Webpack: A module bundler for JavaScript applications.
- Parcel: A fast, zero-configuration web application bundler.
- Gulp: A toolkit to automate tasks in your development workflow.

Applications of JavaScript

Desktop Applications:

- Electron: Allows building cross-platform desktop applications using HTML, CSS, and JavaScript.
- NW.js: A framework for building native applications with web technologies.

Applications of JavaScript

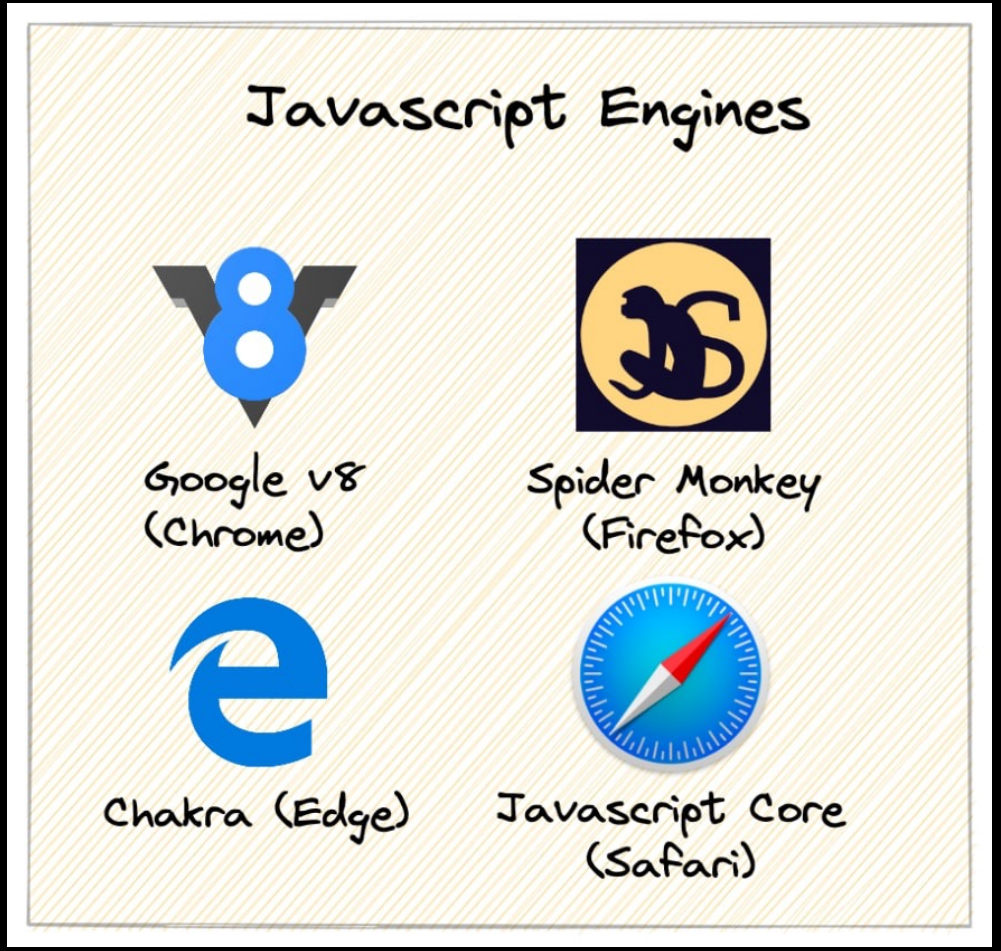


Cameras and Speakers:

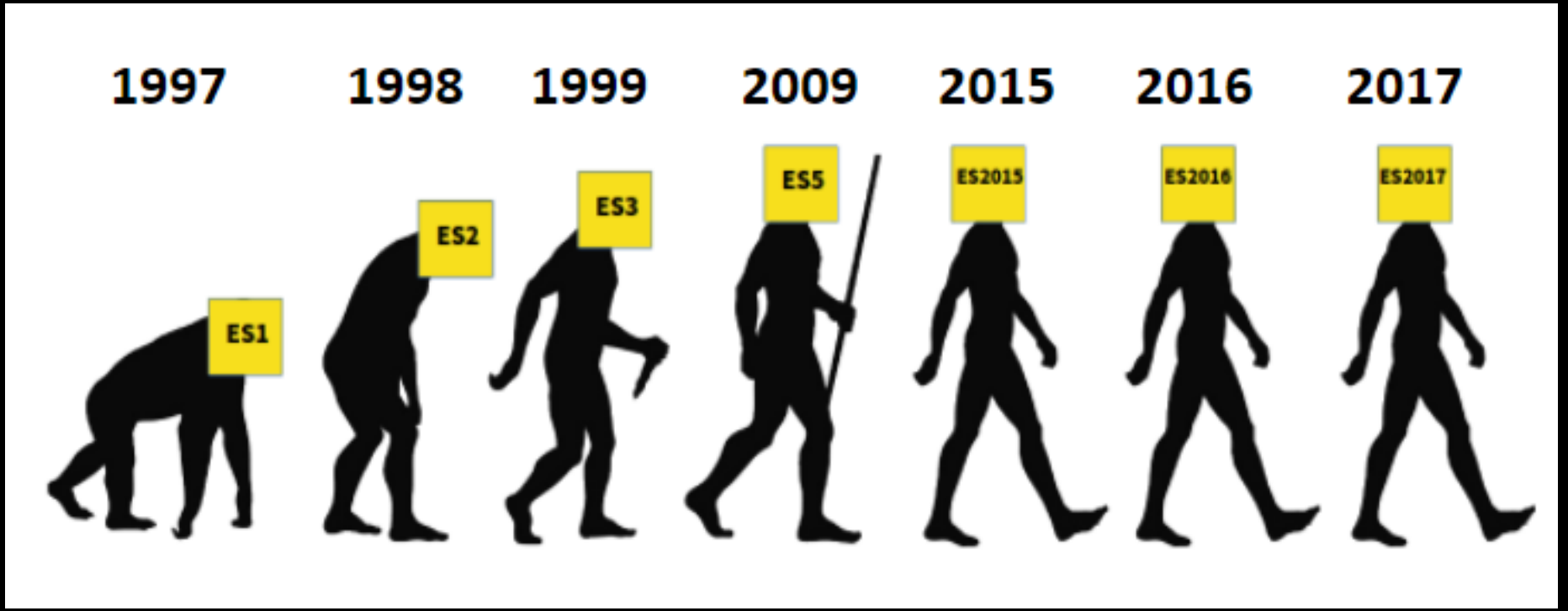
- Three.js: A library that makes WebGL - 3D programming for the web - easier to use.
- WebRTC: A technology that enables peer-to-peer audio, video, and data sharing.
- Howler.js: A JavaScript audio library for the modern web.

Runtime Environment

1. Provides infrastructure to execute JavaScript code.
2. Core: Includes a JavaScript engine (e.g., V8, SpiderMonkey).
3. Browser Environment: Offers APIs for DOM manipulation, events, and network requests.
4. Node.js: Extends JavaScript capabilities to server-side programming.
5. Asynchronous Support: Handles non-blocking operations with event loops, callbacks, and promises.

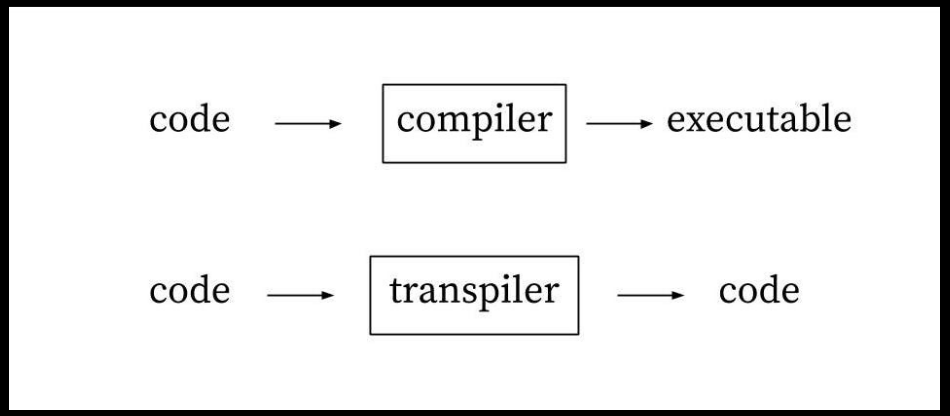


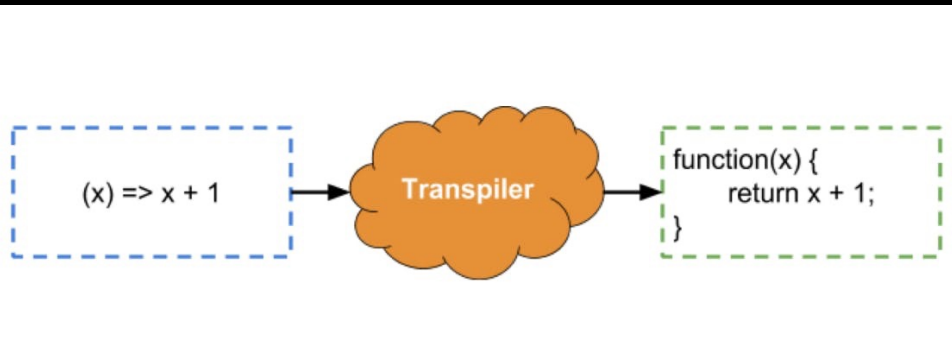
JavaScript vs ECMA



1. ECMAScript is the standardized specification developed by ECMA International that defines the core features, syntax, and functionalities of JavaScript and similar scripting languages.
2. JavaScript is the actual language implementation.

JavaScript vs TypeScript



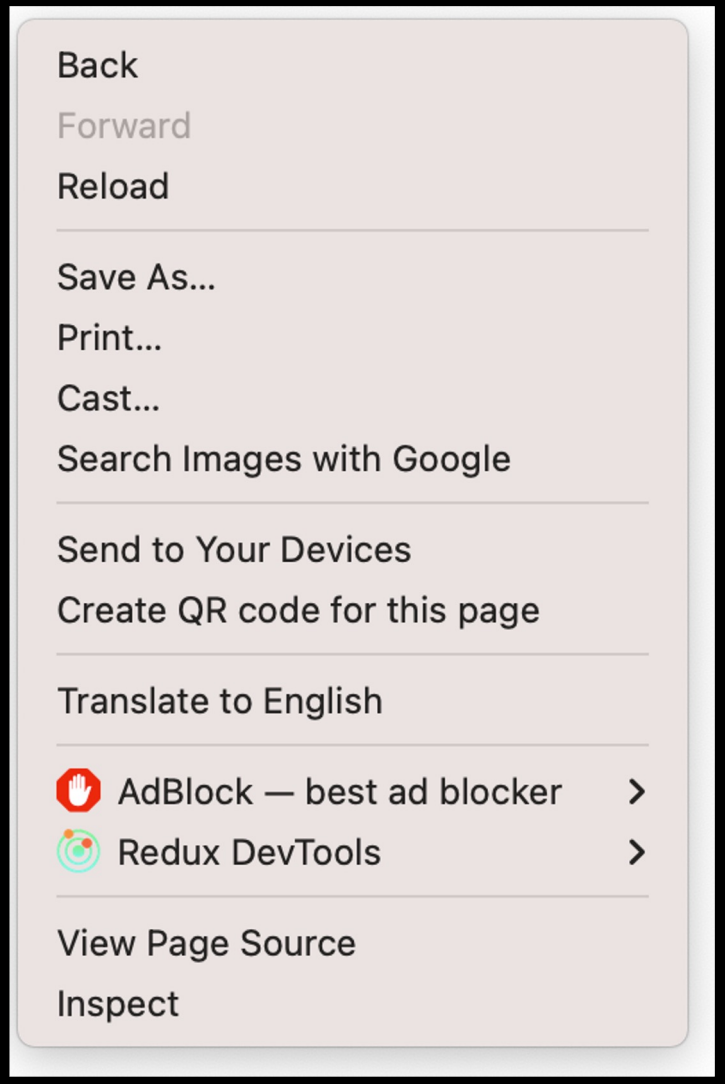


1. JavaScript runs at the client side in the browser.
2. Coffee Script / TypeScript are transpiled to JavaScript.

JavaScript vs TypeScript

Feature	JavaScript (JS)	TypeScript (TS)
Definition	A dynamic, high-level scripting language.	A statically typed superset of JavaScript.
Typing	Dynamically typed.	Statically typed with optional type annotations.
Compilation	Interpreted by browsers.	Transpiles to JavaScript before execution.
Error Detection	Errors detected at runtime.	Errors caught at compile-time.
Tooling Support	Basic tooling, less support for large-scale projects.	Enhanced tooling support with features like IntelliSense.
Learning Curve	Easier to learn for beginners.	Slightly steeper learning curve due to static typing.
Code Maintenance	Can be harder to maintain and debug in large codebases.	Easier to maintain and refactor due to static types.
Development Speed	Faster for small projects and prototyping.	Potentially slower initial development but saves time in the long run with fewer bugs.
Community and Usage	Widely used, especially in web development.	Growing rapidly, especially in large-scale applications.
Example Usage	var x = 10;	let x: number = 10;

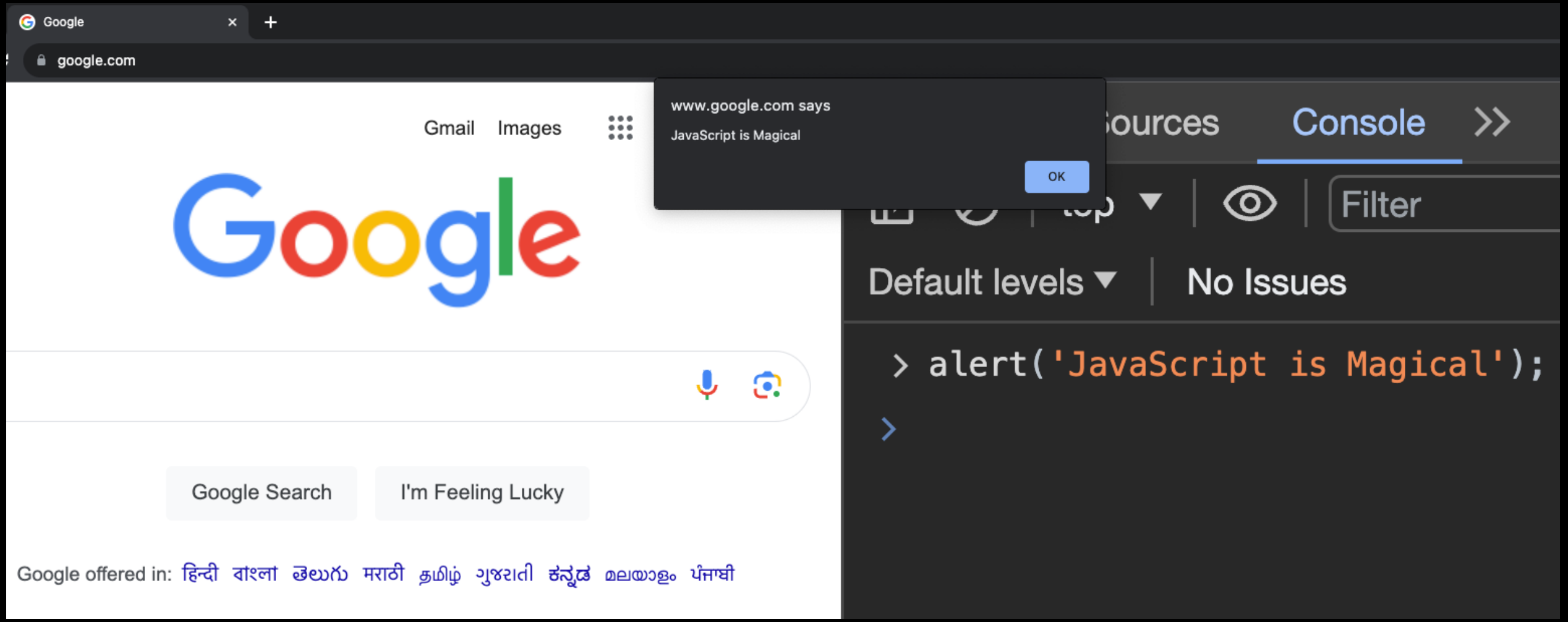
JavaScript in Console (Inspect)



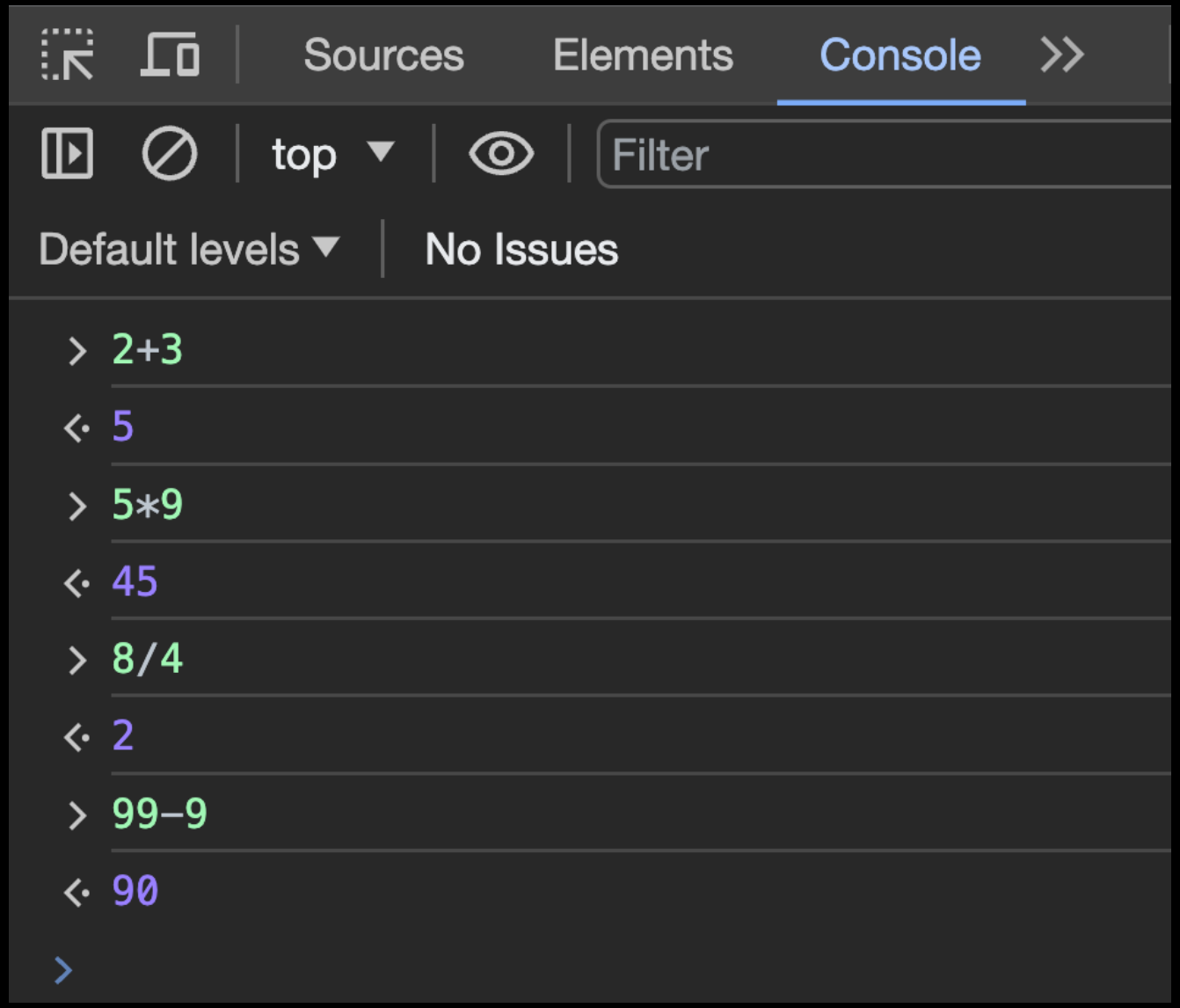
1. Allows real-time editing of HTML/CSS/JS
2. Run Scripts: Test code in console.
3. Debug: Locate and fix errors.
4. Modify DOM: Change webpage elements.

   Errors: View error messages.

JavaScript in Console (Alert)

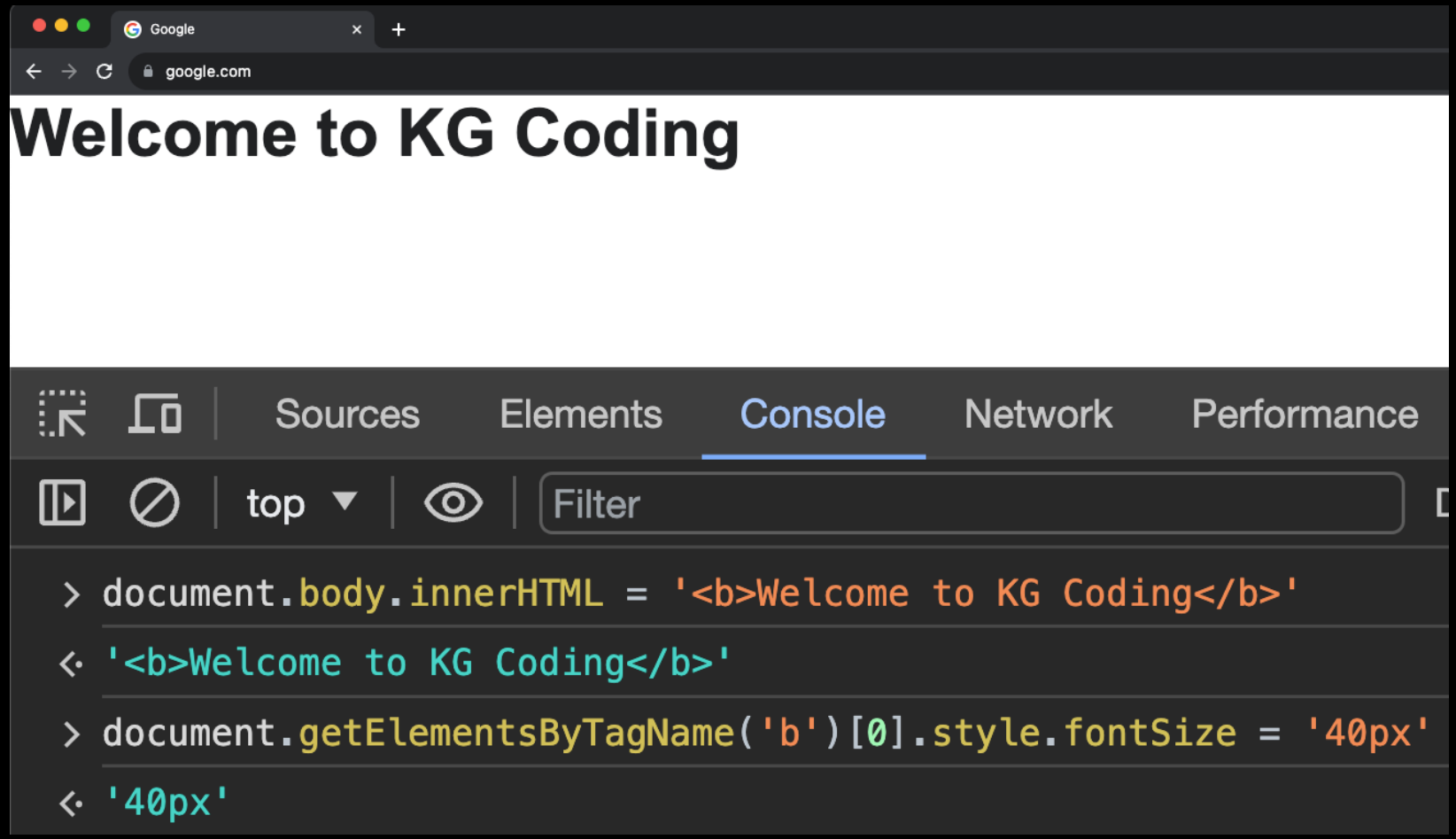


JavaScript in Console (Math)



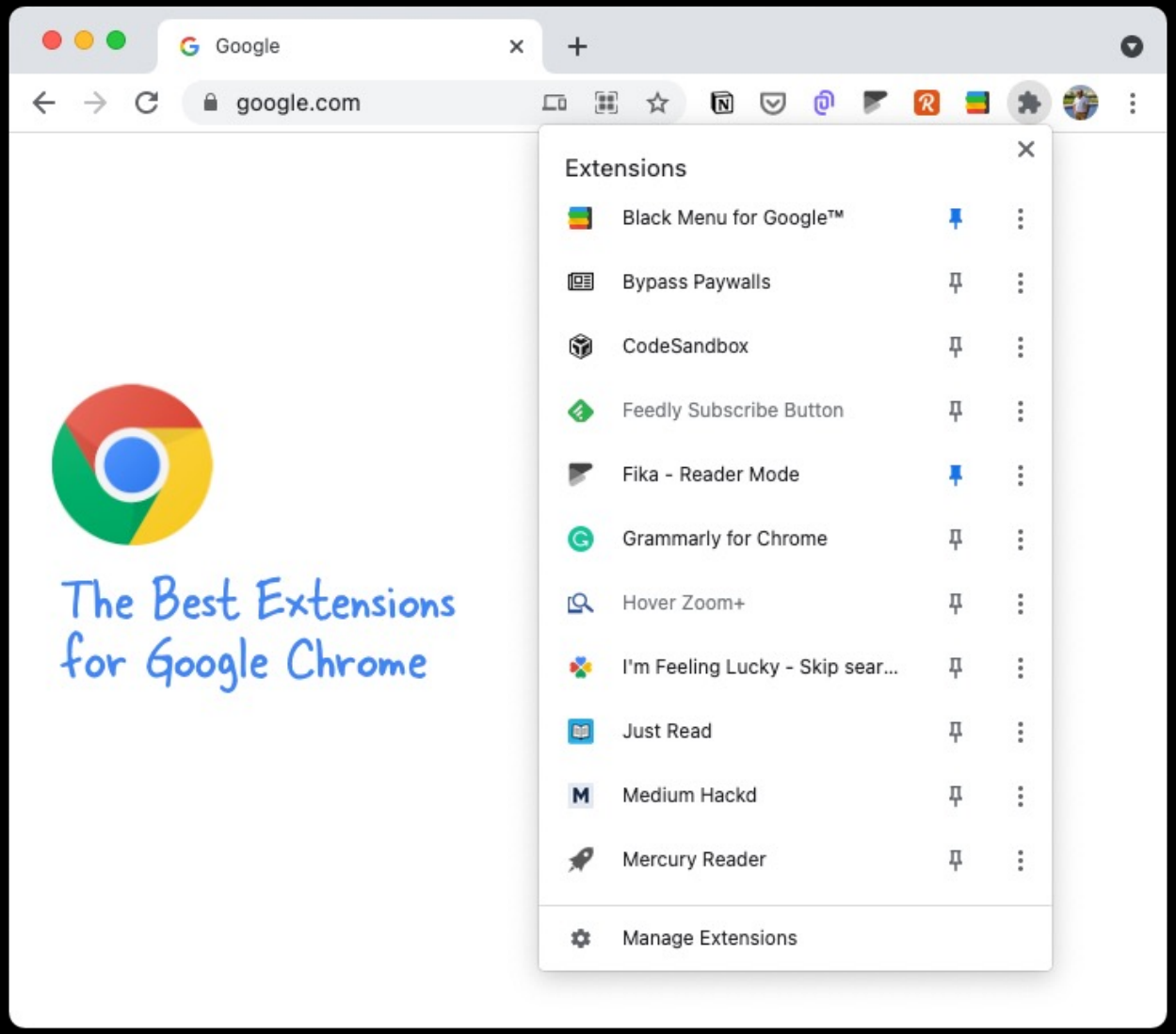
Console can be used as a Calculator

DOM Manipulation



1. Change HTML
2. Change CSS
3. Perform Actions

Chrome Extensions



1. Create Features: Add new functionalities to Chrome.
2. Interact with Web: Modify or read webpage content.
3. API Access: Use Chrome's built-in functions.
4. User Experience: Enhance or customize browsing.

Practice Exercise

1. Use an alert to display Good Morning.
2. Display your name in a popup.
3. Using Math calculate the following:
   => 75-25
   => 3+3-5
4. Change Facebook page to display “I am Learning JS”