JS - Intro

Overview

JavaScript (JS) is a versatile programming language with a wide range of capabilities. It's lightweight, interpreted, or just-in-time compiled, and has first-class functions. As a high-level, prototype-based, multi-paradigm, dynamic language, JavaScript supports various programming styles including object-oriented, imperative, and declarative.

Basic Characteristics

- **Lightweight:** Every browser has a JavaScript engine which interprets the code.
- **Loosely Typed:** JavaScript is not strictly typed.
- **Dynamically Typed:** You can store any type of data in a variable.
- **Browser Specific Engines:** Each browser has its own JavaScript engine. For example, Chrome uses V8, Firefox uses Spidermonkey, Edge uses Chakra, and Safari uses JavaScript Core.

Comparison with Java

JavaScript	Java
var brandname = “PE”	String brandname = “PE”;
var price = 399	int price = 399;
var rating = 4.5	float rating = 4.5;
rating = “great”	rating = “five star”; (throws error)

Types of Programming Languages

Scripting language	Functional PL	Object-oriented PL

JS, TS, Perl, VB	Python, C	Java, C++

Languages where variables must be declared to contain a specific type of data are strictly typed. For example, `String myString = "Fred";` means "myString" can only contain string data. If the following works: `x = 10; x = "Fred";`, it's a loosely typed language since two different types of data can be stored in the same variable.

Features

- **Scripting Language:** JavaScript is a powerful scripting language designed for immersive web applications. It is lightweight, browser-side, and packed with tailored libraries for a variety of scenarios.
- **Interpreter Based:** JavaScript code is interpreted line by line in real-time by the browser, reducing development complexity without sacrificing speed or power.
- **Validation of User's Input:** Form validation ensures accuracy when users are entering data into forms, improving user experience.
- **In-build Function:** JavaScript has a range of helpful built-in functions like `isNAN()`, `Number()`, `parseFloat()`, and `parseInt()` for data manipulation.
- **Case Sensitive:** JavaScript is case sensitive. The same code with different capitalization can yield different results.
- **Light Weight and Delicate:** JavaScript allows for efficient code creation without the need to declare variables. Operations can be carried out with just objects.
- **Statements Looping:** Looping executes the same code multiple times, automating tedious tasks. It can be used with specified or indefinite repetition.
- **Control Statements:** JavaScript's control statements like `if-else-if` and `switch-case` or loops like `for`, `while`, and `do-while` allow tasks of any complexity to be conquered.
- **Dynamic Typing:** JavaScript's dynamic typing allows for variable declarations that are not explicitly stated, making it a unique and highly versatile language.

```
var eid = 101  // variable

function cal_Age(){ //  function

}

cal_Age()

class Employee{ // class

}

var product = {
                                pid : 101,
                                pname : ' marker '

                            }
```

In the above block var, function, class, var are all data types

eid, ename, cal_Age, product are all Identifiers

Rules for Identifiers

Identifiers in JavaScript must follow these rules:

- They can include lowercase letters (`a-z`), uppercase letters (`A-Z`), numbers (`0-9`), or the dollar sign (`$`).
- They should not start with a number (`0-9`).
- They should not contain any special characters, except for the dollar sign (`$`).
- They should not be any keyword or reserved word.
- They are case-sensitive.

Keywords

JavaScript keywords are reserved words that are used to control the flow of the program, make declarations, define functions and classes, handle errors, and more.

Control Flow

Keywords used for controlling the flow of the program include `break`, `continue`, `if...else`, `switch`, `throw`, `try...catch`.

Declarations

Keywords used for making declarations include `var`, `let`, `const`.

Functions and Classes

Keywords used for defining functions and classes include `function`, `return`, `class`, `extends`, `super`.

Iterators and Generators

Keywords used for iterators and generators include `do...while`, `for...in`, `for...of`, `while`, `yield`.

Modules

Keywords used for modules include `export`, `import`.

Other

Other keywords include `debugger`, `new`, `this`, `typeof`, `delete`, `instanceof`.

Error Handling

Keywords used for error handling include `throw`, `try...catch...finally`.

Strict Mode

The keyword used for strict mode is `strict mode`.