

JAVASCRIPT FUNDAMENTALS

Section Summary

JAVASCRIPT

FUNDAMENTALS

What is Javascript?

Explain Javascript

JavaScript is a programming language that is widely used for creating interactive websites and web applications. It allows developers to add dynamic behavior to web pages, such as responding to user input, manipulating content, and updating the display in real-time. JavaScript is often used in conjunction with HTML and CSS to create a complete web application. It is a versatile language that can be used on both the client-side (in the browser) and server-side (on the server). Some popular JavaScript frameworks and libraries include React, Angular, and jQuery. With the continued growth of the web, JavaScript is an important skill for any developer to have.

History of Javascript

JavaScript, a high-level, interpreted programming language, has a rich history and plays various roles in web development. It was initially developed by Brendan Eich at Netscape in 1995 and went through several name changes before settling on JavaScript. Let's summarize the key points about the history of JavaScript:

- JavaScript was created in 1995 by Brendan Eich at Netscape. It was originally named Mocha, then LiveScript, before being renamed JavaScript for marketing purposes, even though it has no technical relation to Java.
- The language was designed to bring interactivity to websites and became the primary scripting language for web development.
- In 1997, JavaScript was standardized by ECMA International, resulting in the ECMAScript standard. This standardization ensured consistent implementation across different browsers and platforms.

- Major updates to JavaScript have been introduced through ECMAScript versions. Notable versions include ES3 in 1999, ES5 in 2009 (which introduced "strict mode" and other improvements), and ES6 in 2015 (which introduced features like let/const, arrow functions, and classes).
- JavaScript is not limited to the browser. It is also used server-side through platforms like Node.js, enabling developers to write JavaScript for backend development as well.
- JavaScript has evolved to become one of the core technologies of the web, alongside HTML and CSS, and is widely used in web development.

Roles of JavaScript in web development:

1. Client-side scripting: JavaScript is primarily used as a client-side scripting language. It runs directly in the user's web browser and allows developers to create interactive and dynamic content, enhancing the user experience of websites.
2. DOM manipulation: JavaScript provides the ability to access and manipulate the Document Object Model (DOM), which represents the structure of a web page. Developers can use JavaScript to add, modify, or delete elements on a web page in real-time without requiring a page refresh.
3. Event handling: JavaScript enables developers to detect and respond to user actions such as clicks, keyboard input, or mouse movements. This capability allows the creation of event-driven applications and provides seamless user interactions.
4. Asynchronous processing: JavaScript supports asynchronous programming, which allows developers to perform tasks in the background without blocking the main execution flow. This feature is essential for creating responsive web applications that can load and update content without interrupting the user experience.
5. Server-side scripting: With the advent of Node.js, JavaScript can also be used for server-side scripting. This allows developers to use JavaScript for both client-side and server-side logic, making it easier to build full-stack applications using a single language.

Benefits of Learning JavaScript:





1. Widespread use: JavaScript is one of the core technologies of the web, and almost every website incorporates JavaScript in some way. Learning JavaScript is essential for web developers due to its high demand in the job market.
2. Versatility: JavaScript can be used for both client-side and server-side development, making it a versatile language. Learning JavaScript enables you to become a full-stack developer, capable of handling front-end and back-end development tasks.
3. Rich ecosystem: JavaScript has a vibrant community that has developed numerous libraries, frameworks, and tools to simplify and expedite web development. By learning JavaScript, you gain access to this extensive ecosystem and can leverage these resources to build high-quality applications.
4. Continuous evolution: JavaScript is a constantly evolving language. Regular updates through the ECMAScript standard ensure that JavaScript remains relevant, incorporating new features and improvements aligned with industry trends and best practices.
5. Transferable skills: JavaScript shares similarities in syntax and programming concepts with other languages like Python, Java, and C#. Learning JavaScript provides a solid foundation that can be transferred to other programming languages, expanding your skill set as a developer.
6. Career opportunities: JavaScript's popularity and versatility open up various career opportunities, including front-end and back-end development, full-stack engineering, mobile app development, and game development.

Setting Up Your JavaScript Development Environment



Setting Up Environment



1. Web browser: Choose a modern browser with good support for web standards and developer tools, such as Google Chrome, Mozilla Firefox, or Microsoft Edge. Keep your browser updated to access the latest features.
2. Text editor: Select a text editor for writing JavaScript code. Popular options include Visual Studio Code, Sublime Text, and Atom. Install your preferred text editor.
3. Developer tools: Modern web browsers come with built-in developer tools for inspecting, debugging, and profiling JavaScript code. Familiarize yourself with the developer tools available in your chosen browser.
4. JavaScript extensions and plugins (optional): Some text editors offer extensions or plugins that enhance the JavaScript development experience. These may include syntax highlighting, code completion, linting, and formatting. Install relevant extensions from the marketplace of your chosen text editor.
5. Version control (optional): Consider using a version control system like Git to track code changes, collaborate with others, and manage different versions of your code. Install Git and explore platforms like GitHub or GitLab for hosting repositories.

Ways to Integrate JavaScript into HTML:

1. Inline Scripting: You can write JavaScript code directly in your HTML file using `<script>` tags. The code is placed between the opening `<script>` tag and the closing `</script>` tag. This method is suitable for small pieces of JavaScript code.

2. External JavaScript file (head tag): Write your JavaScript code in a separate .js file and link to it from your HTML document. Place the `<script>` tag inside the `<head>` section of your HTML file.

3. External JavaScript file (head tag with defer): Use the `defer` attribute within the `<script>` tag to indicate that the script should be executed after the HTML document has been fully parsed. This allows the HTML parsing to continue without blocking.

4. External JavaScript file (head tag with async): Use the `async` attribute within the `<script>` tag to fetch and execute the script asynchronously while the HTML document is being parsed. Execution may start before the HTML parsing is complete.

5. External JavaScript file (below the body tag): Place the `<script>` tag just before the closing `</body>` tag. This allows the HTML document to load and render before executing the JavaScript code, improving perceived load time.

6. JavaScript in HTML Event Handler Attributes: Embed JavaScript code directly into HTML tags as attributes. This method is commonly used for event handling, such as clicks or mouse movements.

7. JavaScript in the URL: Execute JavaScript code from a URL using the `javascript:` protocol. This method is less common and not recommended due to security concerns.

Understanding Variables

- Variables are containers for storing, manipulating, and retrieving data throughout a program.
- Variables are placeholders for values that can change over time or during program execution.
- Variables enable data storage, reuse, readability, and updating.
- Variable declaration reserves memory space for data storage.
- Variables are fundamental building blocks in programming, enabling more complex and dynamic programs.

JavaScript Data Types

- Data types specify the kind of data that can be stored and manipulated.
- There are two categories of data types: Primitive (Basic) and Non-primitive (Reference).

Primitive Data Types

- String: Represents a sequence of characters.
- Number: Represents numeric values.
- Boolean: Represents logical entities, true or false.
- Undefined: Represents a declared but unassigned variable.
- Null: Represents a null or no value.
- Symbol: An immutable primitive value that is unique.

Non-Primitive Data Types

- Object: Represents an instance storing values and functions.
- Array: Represents a group of similar values.
- Function: Represents a set of statements that performs a task or calculates a value.

Examples of Primitive and Non-primitive Data Types

- Examples of primitive data types: 'John Doe', 3050.75, true, undefined, null, Symbol('id')
- Examples of non-primitive data types: user object, transactionAmounts array, greetUser function

Assigning Values to Variables

- Direct assignment: Declare a variable and assign a value simultaneously.
- Assignment after declaration: Declare a variable first, then assign a value.
- Re-assignment: Assign a new value to an already declared variable.

- Assignment via user input: Assign a value to a variable based on user input.

Examples of Variable Assignment

- Examples of direct assignment: 'Apple iPhone 13', 0.07, 15.00
- Examples of assignment after declaration: 'Jane Smith', '12345ABC'
- Example of re-assignment: 72 to 75
- Example of assignment via user input: userFeedback via prompt function

Best Practices for Naming Variables

- Use Camel Case for variable names.
- Use meaningful and descriptive names for variables.
- Start variable names with a letter.
- Avoid using reserved words for variable names.
- Use constants for values that don't change.
- Use all caps for constant values.

Examples of Good and Bad Variable Naming

- Good example: 'Thomas'
- Bad example: 'u' or 'lname' or 'for'
- Example of constant: PI = 3.14
- Example of all caps constant: MAXIMUM_AGE = 100

JavaScript operators:

JavaScript operators are used to perform operations on variables and values. There are several types of operators in JavaScript:

1. Arithmetic Operators: These operators are used to perform mathematical operations similar to those that can be performed in algebra. Examples are +, -, *, /, %, ++, and --.

```
let c = 10;
c += 5; // equivalent to c = c + 5;
console.log(c); // 15

c -= 3; // equivalent to c = c - 3;
console.log(c); // 12

c *= 2; // equivalent to c = c * 2;
console.log(c); // 24

c /= 6; // equivalent to c = c / 6;
console.log(c); // 4

c %= 3; // equivalent to c = c % 3;
console.log(c); // 1
```

2. Assignment Operators: These operators are used to assign values to JavaScript variables. Examples are `=`, `+=`, `-=`, `*=`, `/=`, `%=`, `**=`.

```
let c = 10;
c += 5; // equivalent to c = c + 5;
console.log(c); // 15

c -= 3; // equivalent to c = c - 3;
console.log(c); // 12

c *= 2; // equivalent to c = c * 2;
console.log(c); // 24

c /= 6; // equivalent to c = c / 6;
console.log(c); // 4

c %= 3; // equivalent to c = c % 3;
```

```
console.log(c); // 1
```

3. Comparison Operators: These operators are used to compare two values. Examples are ==, ===, !=, !==, >, <, >=, <=.

```
let d = 10;
let e = "10";

console.log(d == e); // true
console.log(d === e); // false
console.log(d != e); // false
console.log(d !== e); // true
console.log(d > e); // false
console.log(d < e); // false
console.log(d >= e); // true
console.log(d <= e); // true
```

4. Logical Operators: These operators are used to determine the logic between variables or values. Examples are &&, ||, !.

```
let f = true;
let g = false;

console.log(f && g); // false
console.log(f || g); // true
console.log(!f); // false
console.log(!g); // true
```

```
console.log(!g);           // true
```

5. String Operators: The + operator is used to add (concatenate) strings together, and the += operator is used to add (concatenate) strings to the end of an existing string.

```
let h = "Hello";
let i = "World";
console.log(h + i);        // HelloWorld
console.log(h += i);       // HelloWorld
```

Please note that these operators may behave differently depending on the data types of the values or variables they are used with, due to JavaScript's type coercion rules. For instance, using the + operator with a string and a number will result in string concatenation, not numeric addition.