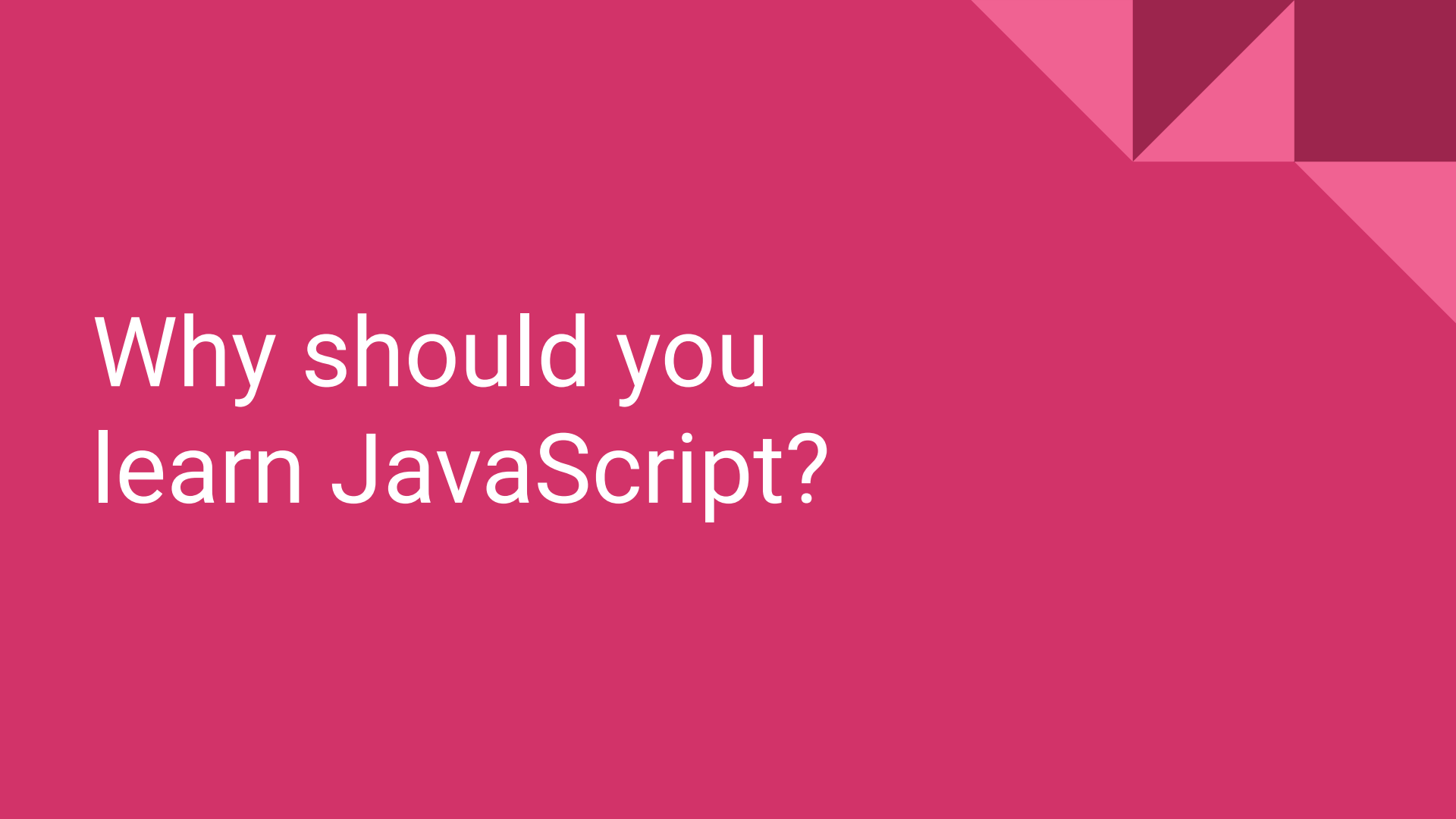




Introduction to Javascript

Getting Started with JavaScript

Chapter 1

The background is a solid pink color. In the top right corner, there is a decorative pattern of overlapping triangles in various shades of pink and magenta, creating a geometric, abstract design.


Why should you
learn JavaScript?

Why should you learn JavaScript?

- JavaScript originates from 1995, and is often considered the most widely used programming language.
- JavaScript is the language that web browsers support and understand.
- You have everything you need to interpret it already installed on your computer if you have a web browser and text editor.
- JavaScript is a great programming language for beginners
- Most advanced software developers will know at least some JavaScript because they will have run into it at some point.



Why should you learn JavaScript?

- You can start building really cool apps using JavaScript sooner than you could imagine.
 - JavaScript can be used for programming for the web browser, but also the logic layer of code that we cannot see (such as communication with the database) of an application can be programmed in JavaScript, along with games.
 - JavaScript can also be used for different programming styles, by which we mean ways to structure and write code unlike other languages which restrict the programming paradigm such as **OOP** or **FP**
- 

Why should you learn JavaScript?

- There are a ton of libraries and frameworks you can use once you get the basics of JavaScript down.
- These libraries and frameworks will really enhance your software life and make it a lot easier and possible to get more done in less time.
 - Examples of these great libraries and frameworks include React, Vue.js, jQuery, Angular, and Node.js.
- There won't be a problem for which you cannot find a solution on the internet.




Setting up your environment

Setting up your environment

- There are many ways in which you can set up a JavaScript coding environment.
Such as:
 - Integrated Development Environment (IDE). Example: VS Code, Sublime Text, Atom, etc.
 - Web browser. Example: Chrome, Firefox, etc.
 - Online editor (optional). Example: StackBlitz, Replit, etc.





How does the
browser understand
JavaScript?

How does the browser understand JavaScript?

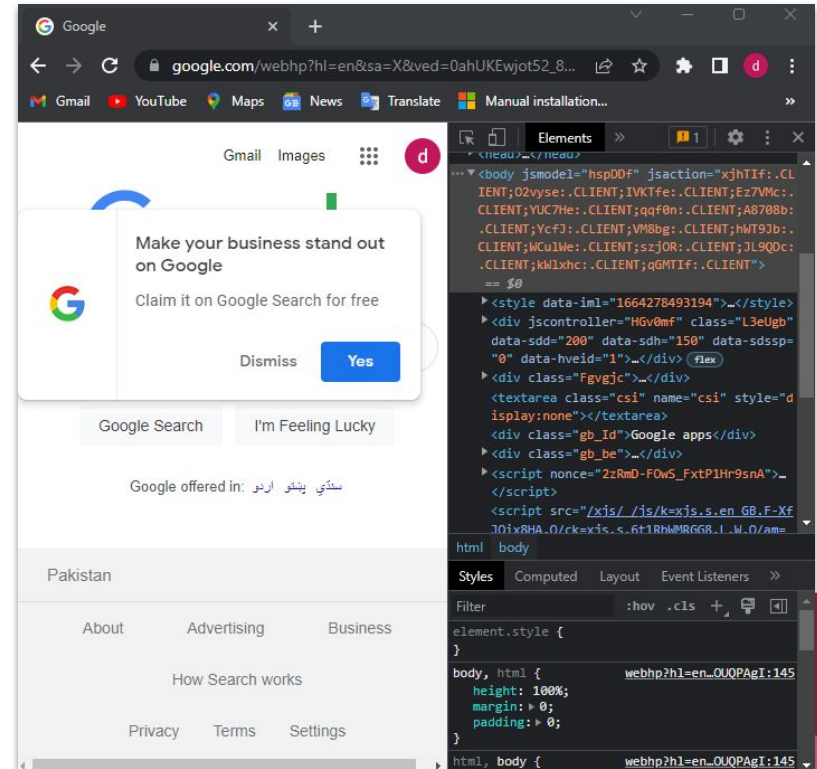
- JavaScript is an interpreted language, which means that the computer understands it while running it. Some languages get processed before running, this is called compiling, but not JavaScript. The computer can just interpret JavaScript on the fly. The "engine" that understands JavaScript will be called the interpreter here.
- Browsers use **ECMAScript** to support JavaScript (in addition to some other topics such as **Document Object Model (DOM)**).



Using the browser console

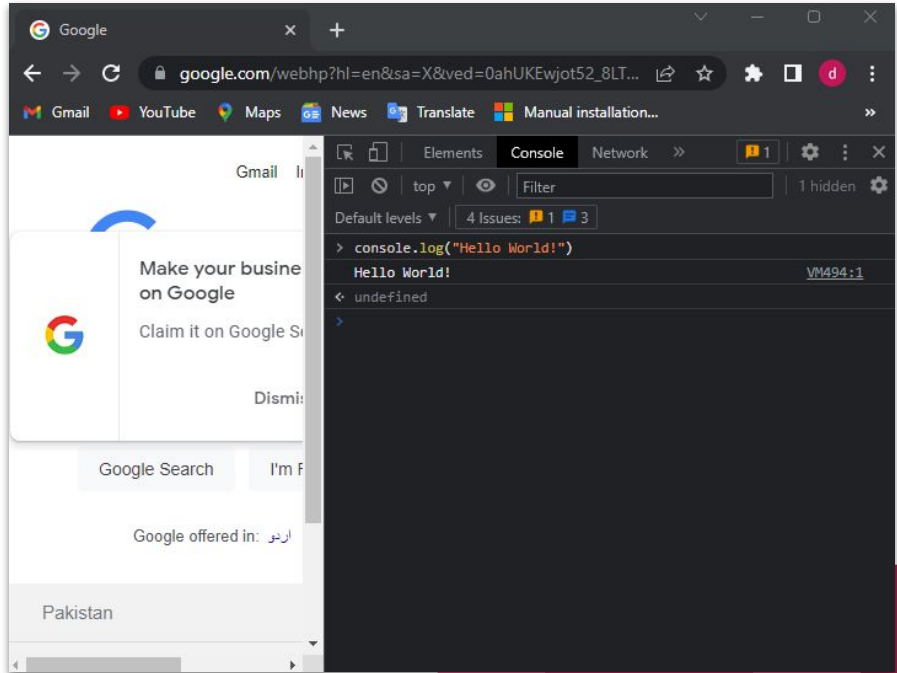
Using the browser console

- If you hit F12 on a Windows computer while you are in the web browser, or you right-click and select Inspect on macOS systems, you will see a screen appear, similar to the one in the following screenshot.
- There are multiple tabs in this panel for inspecting the page.



Using the browser console

- The console is used by developers to log what is going on and do any debugging.
- Click on the **console** tab and type `console.log("Hello world!");` And press Enter key.



Adding JavaScript to a web page

Adding JavaScript to a web page

- There are two ways to link JavaScript to a web page.
 - The first way is to type the JavaScript directly in the HTML between two <script> tags.

```
<html>  
  <script type="text/javascript">  
    alert("Hello World!");  
  </script>  
</html>
```

- The second way is to create a file with extension of .js and link it to our web page.

```
<html>  
  <script type="text/javascript" src="hello_world.js"></script>  
</html>
```



Writing JavaScript code

Writing JavaScript code

- Formatting code:
 - Code needs to be formatted well. If you have a long file with many lines of code and you didn't stick to a few basic formatting rules, it is going to be hard to understand what you've written.
 - The two most important rules for formatting the code are indentations and semicolons.

```
1 var variable1 = 1; var variable2 = 2; var variable3 = 3
2 if (variable1 === 1) {
3   console.log(variable1)
4   if (variable2 === 1) {
5     console.log(variable2)
6   } else {
7     console.log(variable3)
8   }
9 }
```

```
1 var variable1 = 1;
2 var variable2 = 2;
3 var variable3 = 3;
4
5 if (variable1 === 1) {
6   console.log(variable1);
7   if (variable2 === 1) {
8     console.log(variable2);
9   } else {
10    console.log(variable3);
11  }
12 }
```

Writing JavaScript code

- Code comments:
 - With comments, you can tell the interpreter to ignore some lines of the file. They won't get executed if they are comments.
 - Adding comments to specific parts of the code to explain what is happening or why a certain choice has been made.

```
1  // This line is commented out and the interpreter will ignore it.  
2  
3  /**  
4   * this is a multi-line comment.  
5   * comments to specific parts of the code to explain what is happening  
6   */  
7
```



JavaScript Essentials

Chapter 2

Variables

Variables

- Variable means anything that can vary.
- A JavaScript variable is simply **a name of storage location**.
- A variable must have a unique name.



Variables

- Variables are values in your code that can represent different values each time the code runs.
- The first time you create a variable, you declare it. And you need a special word for that: `let` , `var` , Or `const` .

Example: `let firstname = "Ali";`

- The commonly used naming conventions used for **variables** are camel-case.

Example: `let firstName = "Ali";`



Variables Scope

- **LOCAL**
- **GLOBAL**



Variables Names

- A variable name can't contain any spaces
- A variable name can contain only letters, numbers, dollar signs, and underscores.
- The first character must be a letter, or an underscore (-), or a dollar sign (\$).
- Subsequent characters may be letters, digits, underscores, or dollar signs.
- Numbers are not allowed as the first character of variable.

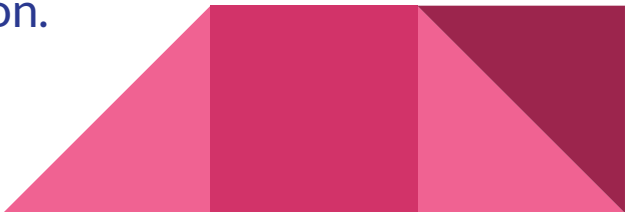


Comments

- Single line Javascript comments **start with two forward slashes (//)**.
- All text after the two forward slashes until the end of a line makes up a comment
- Even when there are forward slashes in the commented text.
- Multi-line Comments
- Multi-line comments start with `/*` and end with `*/`.
- Any text between `/*` and `*/` will be ignored by JavaScript.



Statements

- A computer program is a list of "instructions" to be "executed" by a computer.
 - In a programming language, these programming instructions are called statements.
 - A JavaScript program is a list of programming statements.
 - JavaScript applications consist of statements with an appropriate syntax. A single statement may span multiple lines. Multiple statements may occur on a single line if each statement is separated by a semicolon.
- 

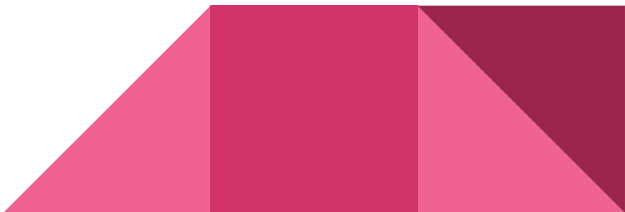
Let, Var, Const

Hoisting of var

Hoisting is a JavaScript mechanism where variables and function declarations are moved to the top of their scope before code execution.

Hoisting of let

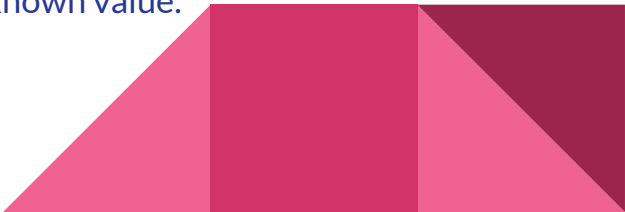
Just like `var`, `let` declarations are hoisted to the top. Unlike `var` which is initialized as `undefined`, the `let` keyword is not initialized. So if you try to use a `let` variable before declaration, you'll get a Reference Error.



Primitive data types

Primitive data types

- String
 - A string is used to store a text value.
Example: `let firstName = "Ali";`
- Number
 - A number is used to store a numeric value.
Example: `let score = 25;`
- Boolean
 - A boolean is used to store a value that is either `true` or `false`.
Example: `let isMarried = false;`
- Undefined
 - An undefined type is either when it has not been defined or it has not been assigned a value.
Example: `let unassigned;`
- Null
 - null is a special value for saying that a variable is empty or has an unknown value.
Example: `let empty = null;`



Template Literals

A new and fast way to deal with strings is **Template Literals** or **Template String**.

How we were dealing with strings before ?

```
var myName = "daniyal" ;
```

```
var hello = "Hello " + myName ;
```

```
console.log(hello); //Hello daniyal
```



Template Literals

What is Template literals ?

As we mentioned before , it's a way to deal with strings and specially dynamic strings ; so you don't need to think more about what's the next quote to use single or double.

How to use Template literals

It uses a `backticks` to write string within it.



Analyzing and modifying data types

Analyzing and modifying data types

- You can check the type of a variable by entering `typeof`.

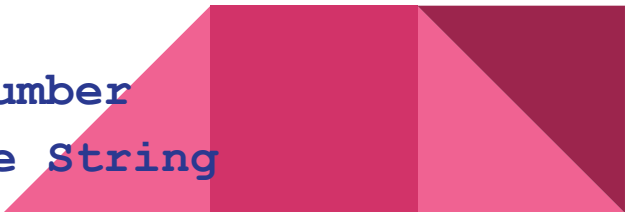
Example:

```
let testVariable = 1;  
console.log(typeof testVariable);
```

- The variables in JavaScript can change types. Sometimes JavaScript does this automatically.

Example:

```
let v1 = 2;  
let v2 = "2";  
console.log(v1 * v2); // 4 ← Type Number  
console.log(v1 + v2); // "22" ← Type String
```



Analyzing and modifying data types

- There are three conversion methods:
 - `String()` ← converts to string type
 - `Number()` ← converts to number type
 - `Boolean()` ← converts to boolean type



Operators

Operators

- Arithmetic operators:

- Addition

Example:

- ```
let n1 = 1;
let n2 = 2;
console.log(n1 + n2); // 3
```
  - ```
let str1 = "1";  
let str2 = "2";  
console.log(str1 + str2); // "12"
```



Operators

- Arithmetic operators:

- Subtraction

Example:

- ```
let n1 = 5;
let n2 = 2;
console.log(n1 - n2); // 3
```

- Multiplication

Example:

- ```
let n1 = 5;  
let n2 = 2;  
console.log(n1 * n2); // 10
```



Operators

- Arithmetic operators:

- Division

Example:

- `let n1 = 4;`
`let n2 = 2;`
`console.log(n1 / n2); // 2`

- Exponentiation

Example:

- `let n1 = 2;`
`let n2 = 2;`
`console.log(n1 ** n2); // 4`



Operators

- Arithmetic operators:

- Modulus

Example:

- ```
let n1 = 10;
let n2 = 3;
console.log(n1 % n2); // 1
```



# Operators

- Assignment operators:
  - Assignment operators are used to assign values to variables.

Example:

- ```
let n = 5;  
console.log(n); // 5  
n += 5;  
console.log(n); // 10  
n -= 5;  
console.log(n); // 5
```



Operators

- Comparison operators:

- Comparison operators are used to compare values of variables.

Example:

- ```
let n = 5;
console.log(n == 5); // true
console.log(n === 5); // true
console.log(n != 5); // false
console.log(n > 8); // false
console.log(n < 8); // true
console.log(n >= 8); // false
console.log(n <= 8); // true
```



# Operators

- Logical operators:

- Logical operators are used to combine multiple conditions in one.

Example:

- ```
let n = 5;
console.log(n >= 5 && n < 10); // true
console.log(n > 5 && n < 10); // false
console.log(n >= 5 || n < 10); // true
console.log(n > 5 || n < 10); // true
console.log(!(n < 10)); // false
console.log(!(n > 10)); // true
```



Thank You