2-The Basics

Chapter 2: Understanding the Basics

JavaScript is built on fundamental concepts that are easy to grasp yet powerful enough to unlock endless possibilities. This chapter introduces essential building blocks such as variables, data types, and operators, laying the groundwork for more advanced topics later.

2.1 Debug Print

When your program doesn't work as expected you will need to "debug" the program to find the problem. Printing the value of variables and objects to the terminal allows you to check values and hopefully find the place in the code that is not working properly. One way is to use `console.log()` to display values on the terminal.

Create a file called debug-print.js in Chap2_Basics

```
const a_variable = 5;
console.log(a_variable); // Output: 5
console.log("The value of a_variable is: " + a_variable); // Output: The value of a_variable is: 5
console.log("The value of a_variable is: ", a_variable); // Output: The value of a_variable is: 5
console.log(`The value of a_variable is: ${a_variable}`); // Output: The value of a_variable is: 5
```

Open the Terminal

- Change terminal directory to Chap2_Basics
- node `debug-print.js`

Terminal Output:

```
5
The value of a_variable is: 5
The value of a_variable is:  5
The value of a_variable is: 5
```

2.2 Variables

Variables are containers for storing data values. In JavaScript, you can declare variables using `var`, `let`, or `const`.

- `let` **and** `const`**:** Introduced in ES6, these are recommended for modern JavaScript.
- Use `let` for variables whose values might change.
- Use `const` for variables that should remain constant.
- If you use let (or var), you can change the value of the variable later in the code

Create a file called variables.js in Chap2_Basics

```
// 2.1 variables
let username = "John Doe";
let age = 30;
const pi = 3.14;
age = 34; // Reassigning a variable is allowed, but not for constants

// Print the variables to the console
console.log(`Username: ${username}`); // Output: Username: John Doe
console.log(`Age: ${age}`); // Output: Age: 30
console.log(`Pi: ${pi}`); // Output: Pi: 3.14
```

- `var`**:** The older way to declare variables, but less commonly used now due to scope limitations.

Open the Terminal

- No need to change directory, terminal will run programs in the Chap2_Basics director
- node `variables.js`

Terminal output

```
Username: John Doe
Age: 34
Pi: 3.14
```

Pro Tip:
Don't forget to save your code after changes. VSCode puts a white dot on unsaved filenames. If your program doesn't work as expected, check that the modified files are saved.

2.3 Data Types

JavaScript has several primitive data types, broadly categorized as:

- **Primitive Types:** `string`, `number`, `boolean`

Create a file called data-types.js in Chap2_Basics

```
// 2.2 primitive data types
let number = 42;
let isBoolean = true;
let string = "Hello, World!";

// Print the variables to the console
console.log(`Number: ${number}`); // Output: Number: 42
console.log(`Boolean: ${isBoolean}`); // Output: Boolean: true
console.log(`String: ${string}`); // Output: String: Hello, World!
```

Run the program in the terminal as before

```
Number: 42
Boolean: true
String: Hello, World!
```

2.3 Operators

Operators are symbols or keywords that perform operations on variables and values.

1. **Arithmetic Operators:**
   - Addition: `+`
   - Subtraction: `-`
   - Multiplication: `*`
   - Division: `/`
   - Modulus: `%`

Create a file called operators.js in Chap2_Basics

```
// 2.3 operators
let sum = 5 + 3; // result is 8
let difference = 10 - 4; // result is 6
let product = 7 * 6; // result is 42
let quotient = 20 / 4; // result is 5
let remainder = 10 % 3; // result is 1
```

```
let exponent = 2 ** 3; // Same as 2^3 and result is 8
let increment = 5; // result is 5
increment += 2; // same as increment = increment + 2 and result is 7
let decrement = 5; // result is 5
decrement -= 2; // same as decrement = decrement - 2 and result is 3
increment++; // same as increment = increment + 1 and result is 8
decrement--; // same as decrement = decrement - 1 and result is 7

// Print the results
console.log("Sum:", sum); // Output: Sum: 8
console.log("Difference:", difference); // Output: Difference: 6
console.log("Product:", product); // Output: Product: 42
console.log("Quotient:", quotient); // Output: Quotient: 5
console.log("Remainder:", remainder); // Output: Remainder: 1
console.log("Exponent:", exponent); // Output: Exponent: 8
console.log("Increment:", increment); // Output: Increment: 8
console.log("Decrement:", decrement); // Output: Decrement: 2
console.log("Increment after incrementing:", increment++); // Output:
Increment after incrementing: 8
console.log("Decrement after decrementing:", decrement--); // Output:
Decrement after decrementing: 2
```

Run the program in the terminal

```
Sum: 8
Difference: 6
Product: 42
Quotient: 5
Remainder: 1
Exponent: 8
Increment: 8
Decrement: 2
Increment after incrementing: 8
Decrement after decrementing: 2
```

2. **Comparison Operators:** Used for comparing two values and often used in if-else flow control structures (See)
   - Equal: `==` or `===` (strict comparison)
   - Not equal: `!=` or `!==`
   - Greater than: `>`
   - Less than: `<`

Create a file called comparison_operators.js in Chap2_Basics

```
// 2.3 Comparison operators
let age = 18; // Example age variable
if (age == 18) {
    console.log("You are 18 years old.");
} else if (age > 18) {
    console.log("You are older than 18 years old.");
} else if (age < 18) {
    console.log("You are younger than 18 years old.");
} else if (age != 18) {
    console.log("You are not 18 years old.");
} else if (age === 18) {
    console.log("You are exactly 18 years old.");
}
```

Run the program with age = 18

```
You are 18 years old.
```

Run the program with age = 25

```
You are older than 18 years old.
```

Run the program with age = 12

```
You are younger than 18 years old.
```

3. **Logical Operators:** Combine multiple conditions.
   - AND: `&&`
   - OR: `||`
   - NOT: `!`

Create a file called logical_operators.js in Chap2_Basics

```
// 2.3 Logical operators
let age = 25; // Example age variable
let isAdult = age >= 18 && age <= 65; // true if age is between 18 and 65
let orState = age < 18 || age > 65; // true if age is less than 18 or
greater than 65
let isSenior = age > 65; // true if age is greater than 65
let isMinor = age < 18; // true if age is less than 18
let isNotAdult = !isAdult; // true if age is not between 18 and 65

// Print the variables to the console
console.log("isAdult:", isAdult); // Output: true or false based on age
```

```
console.log("orState:", orState); // Output: true or false based on age
console.log("isSenior:", isSenior); // Output: true or false based on age
console.log("isMinor:", isMinor); // Output: true or false based on age
console.log("isNotAdult:", isNotAdult); // Output: true or false based on age
```

Run the program

```
isAdult: true
orState: false
isSenior: false
isMinor: false
isNotAdult: false
```

2.4 Comments

Comments are lines in your code that JavaScript ignores when executing. They are incredibly useful for:

- Documenting your code.
- Explaining complex logic.
- Temporarily disabling parts of your code during testing.

JavaScript supports two types of comments:

Create a file called logical_operators.js in Chap2_Basics

```
// 2.4 Comments
// This is a single-line comment
let isReady = true; // Comment at the end of a line

/* This is a multi-line comment
   that spans multiple lines */
let count = 0;
```

Pro Tip:
Comments should enhance clarity, not clutter your code. Be concise and focus on explaining *why* the code exists rather than *what* it does (unless it's complicated).

2.5 Writing Cleaner Code

At this stage, adopt habits that make your code readable and maintainable:

- Use meaningful variable names.
- Add comments to document tricky sections.
- Keep your code tidy with proper indentation and spacing.

Create a file called cleaner_code.js in Chap2_Basics

```
let totalScore = 100; // Initial game score
console.log(`Total Score: ${totalScore}`);
```

Run the program

```
Total Score: 100
```

Explorer at the End of this Chapter if you are following along.