JAVASCRIPT FUNDAMENTALS - part 1

5.Section Intro:

Start this journey right now....

In this section, you will start learning the very basic JavaScript

6. Hello World:

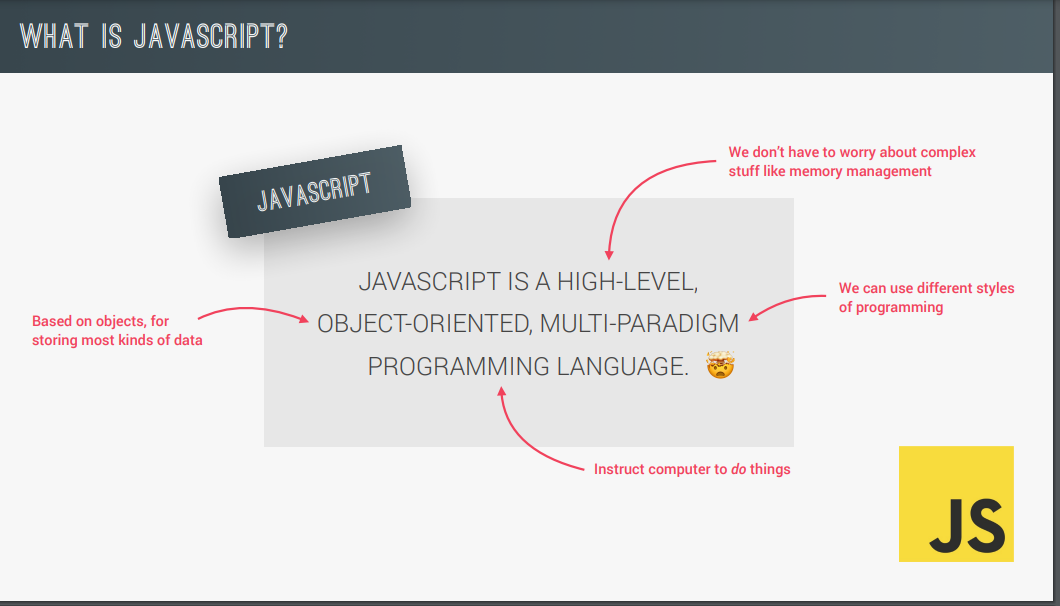
1. alert: The alert () method displays an alert box with a message and an OK button. The alert () method is used when you want information to come through to the user.

Note: alert is so called JavaScript function.

7. A Brief Introduction to JavaScript:

a. what is JavaScript

➡️JavaScript is high-level, object-oriented, multi-paradigm programming language.



b. what is the role of JavaScript in web development?

➡️ Html, CSS and JavaScript all work together to create beautiful, interactive and dynamic website or web application.

c. what can JavaScript do?

➡️ dynamic effect and web application on web browser-

➡️web application on web server.

➡️native mobile application.

➡️native desktop application.

➡️programming-language: A programming language is basically just a tool that allows to write code that will instruct a computer to do something.

d. High-level programming language (HLL) is designed for developing user-friendly software programs and websites. This programming language requires a compiler or interpreter to translate the program into machine language (execute the program).

e. object-oriented: mostly based on the concept of object for storing most kinds of data.

f. multi-paradigm: that it's so flexible and versatile that we can use all kinds of different programming style such as imperative and declarative programming.

g. React native: the ionic frame work tools electron.

h. JavaScript: JavaScript is the real programming language of internet. allows developer to add dynamic effect and interactive effect on any webpage. we also use it to manipulate the content or the CSS, load data from remote server and really build entire application in the browser. we then call web application.

JavaScript releases ... (mote about this later)

Note: JavaScript release.

ES stands for ECMAScript.

ECMA = European Computer Manufacturers Association.

Note: Learn modern JavaScript from the beginning. but without forgetting the older parts! 🔥🔥

8.Linking a JavaScript:

download project file:

Link GitHub: https://github.com/jonasschmedtmann/complete-javascript-course

Note: Console.log () is a special function

Now, I we still want to displays results in the console we need to tell JavaScript to explicitly output that result to the console. and we can do that a special function called console.log |

9.Values and Variables:

a. Value: Value is basically a piece of data. so, value is basically the smallest unit of information that we have in JavaScript.

Example:

'Jonas' = value.

23 = value.

b. variable:

Now, one extremely useful thing that we can do with values is to store them into Variables. and so, this way we can reuse them over and over again.

Example:

let firstName = 'Jonas'; [so what we did here is called declaring a variable. and so, this will actually create a real variable in your computer memory.]

So real world, a box can hold some object, for example, a book and we can write a label on the box to describe what's in it. and then we can find object later when we need it by using that label. variable works exact same way.

c. variable naming conventions and rules for naming variable:

1.camelcase

Example: firstNamePerson

2. underscore:

Example: first\_name\_person

3. illegal variable name:

example: let 3years = 3; [start with a number]

4. variable names only contain number, letters, underscores, and the dollar assign.

5. We should not reserve JavaScript keyword.

Example:

let new = 27;

let function = 33;

5. we should not start with uppercase letters, so we should not do this:

Example: let Person = 'jonas';

6. That’s very important to write cleaner code to make sure variable name are descriptive.

example:

let myFirstJob = "Coder";

let myCurrentJob = "Teacher";

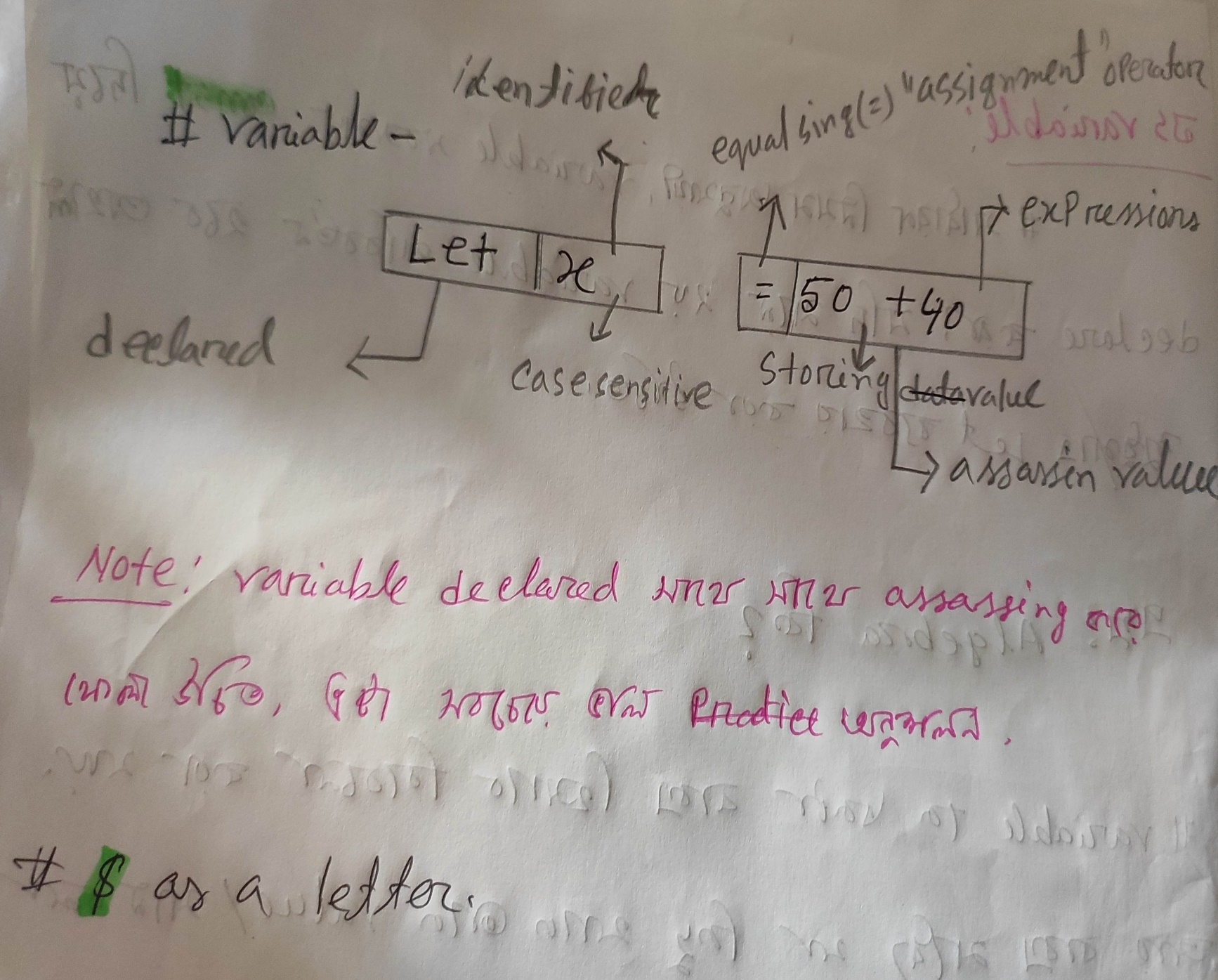
let job1 = "programmer";

let job2 = "teacher";

Note: And it’s important to actually start reading these error massages right from the beginning of the course.

e. The general rule is to always declare variables const. Let should only be used if the variable is expected to change.

f. The name of the variable is also called identifier.

g. Name are case sensitive. (A and a are different variables)

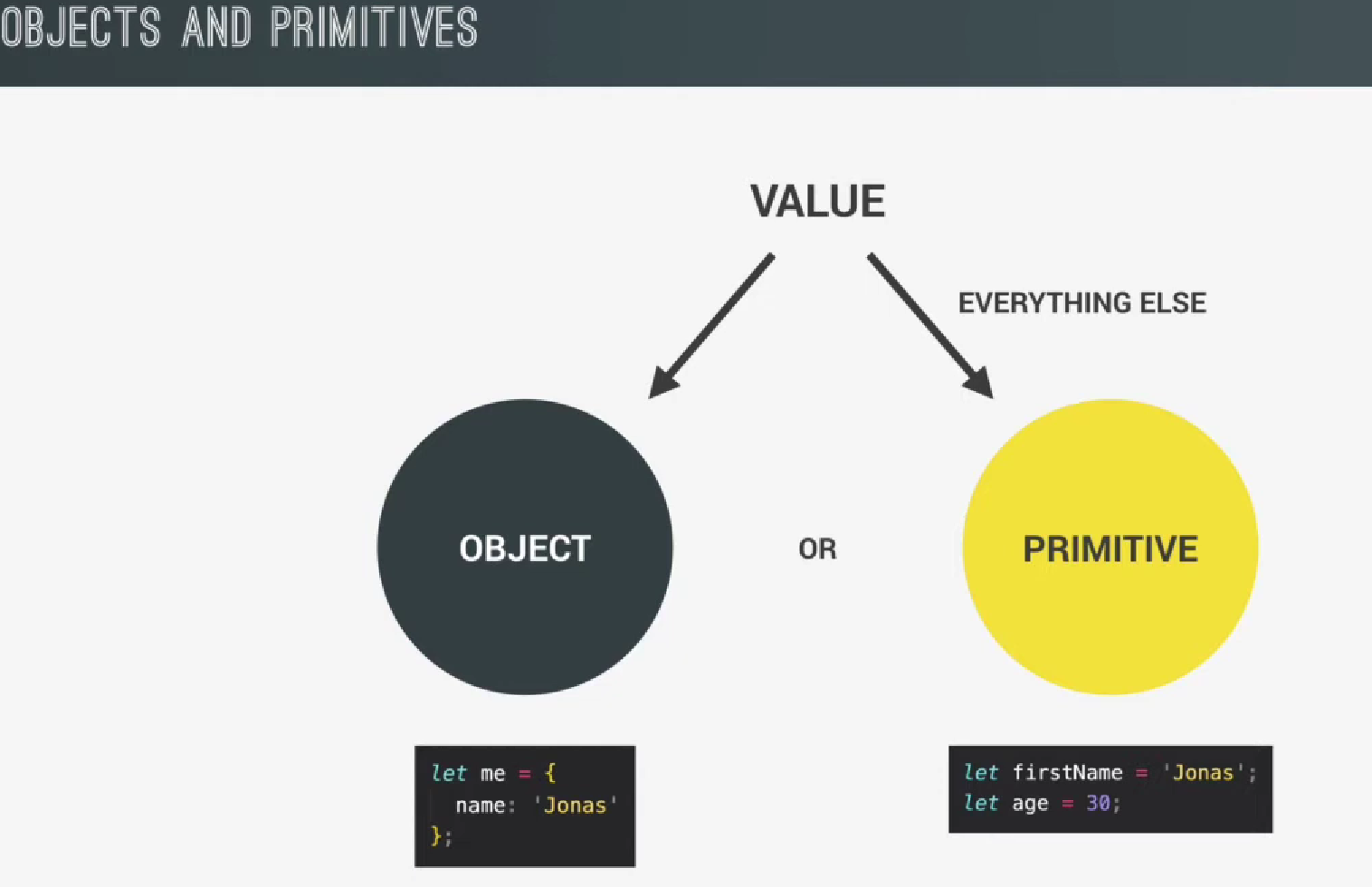
h. never called your variable to reserved keyword.

Note: The best way of learning JavaScript is by practicing it! [\*\*\*]

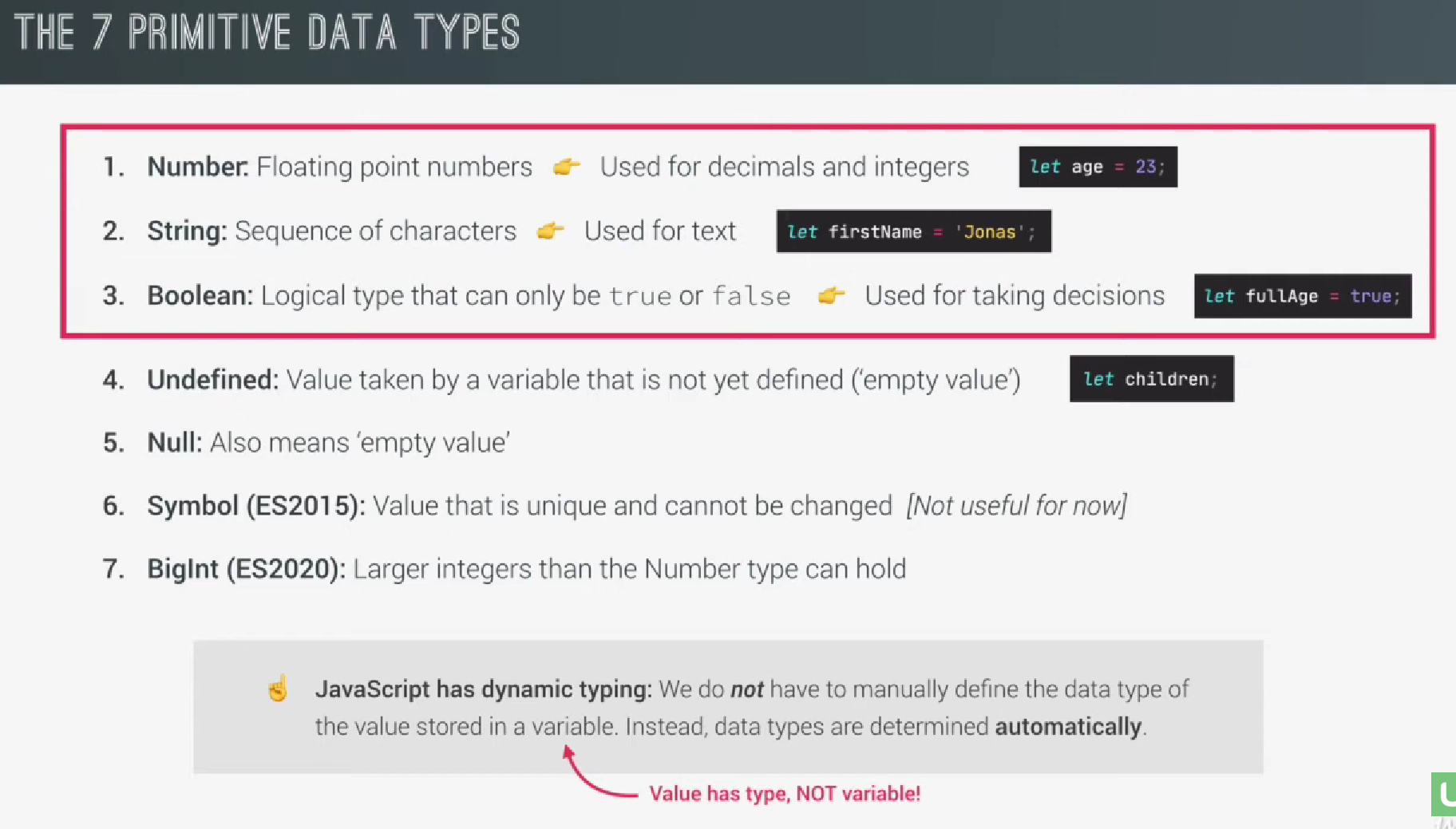
10. Practice Assignments:

11. Data Types:

a. Every value either an object or primitive value. or value is only a primitive when it's not object.



1. primitive data Types: Value is only primitive when it’s not an object.





1. Number: always called floating point number. which means always have decimals.

Note: even if we don’t see them or don’t define them.

Example:

let age = 23;/ let age = 23.0; [same]

1. string: sequence of characters - use for text. always put them quotes no matter if double quotes {" "} or single quotes {' '}.

example:

let firstName = 'nahid';/ let firstName = "nahid";

1. Boolean: logical types that can only be true or false. use for taking decision.

4.Undefined: value take by a variable that is not yet define. [empty]

example:

let children; [without Assigning value]

5.Null: also means ('empty value'). Very similar, but it used different circumstance.

6.symbol: (es 2015): value that is unique and cannot be changed [not useful for now]

7. BigInt: (es 2020): Larger integer than the number type can be hold.

C. When a new variable is declared, it is not possible to set the data type manually. Setting is possible from other programming languages but not in JavaScript. JavaScript automatically determines the data type when it is assigned to a variable.

Note: value has type not variable.

d. Typeof: typeof is a special operator. so typeof it’s an operator that we just like the plus or minus operator that we can show the type of value.

why is JavaScript is called Dynamic typing.

e. Dynamic typing: Dynamic typing simply means that we can changed the type of a value that is hold by variable.

example:

let x = 50; [typeof == number]

x = 'nahid'; [typeof == string]

Note: JavaScript evaluates expression from left to right. different sequence can produce different result.

example:

let x = 14 + 4 + 'Volvo'; output -- 18 Volvo

let x = 'Volvo' + 14 + 4; output -- volvo144

The above two examples show how JavaScript works.

Note: Code can be used inside a sting if it does not match the code used at the beginning of the sting.

Example:

let answer = "It's alright"

f. typeof null:

typeof null is an object. [typeof null];

It is considered a defect. Although this error is not fixed due to inheritance.

12.let, const and var:

when, where and how to use:

a. let:

example:

let age = 30;

age = 31; //mutate variable.

It’s means, we can create an empty variable. then we can reassign value anywhere.

example:

let age; [empty value]

age = 50; [reassigning value or mutate variable]

b. const: When a variable is made with const, value must be assigned immutable. Const cannot be empty variable and cannot be reassigned later. A const value can never change.

Example:

const birthyear = 1990; //immutable

so const variable that cannot be mutated.

Note: recommend to used const by default and only when you are really sure that the variable need to change at some point in the future.

Note: changing variable has chance of creating problem. \*\*\*

d. var: completely avoid [old way] \*\*\*

e. without variable:

example:

lastYear = 2022;

JavaScript is valid. variables can be declared like this but it shouldn't. variable should always be declared in the correct way.

f. JavaScript program executed top to bottom.

13.Basic Operators:

Operators: so, an operator basically allows us to transform values or combine multiple values and really do all kinds of work with values.

There are different types of JavaScript operators:

• Arithmetic Operators (+, -, /, \*)

• Assignment Operators (=)

• Comparison Operators (==, ===)

• String Operators = Concatenation: **+**

• Logical Operators (&&, ||,!)

• Bitwise Operators

• Ternary Operators

• Type Operators

b. The console actually has access to all the variable that is running in the current browser.

c. we can log different values at the same time in one console.log |

d. equal sing: {=} is an operator.

e. plus {+} There is some variation in its usage as it helps to concatenate two strings.

e. Typeof: that would then give us the typeof the value.

f. Assignment operator: Assignment operator is just {=} sing;

g. comparison operator: produce Boolean values. yes = true, no = false;

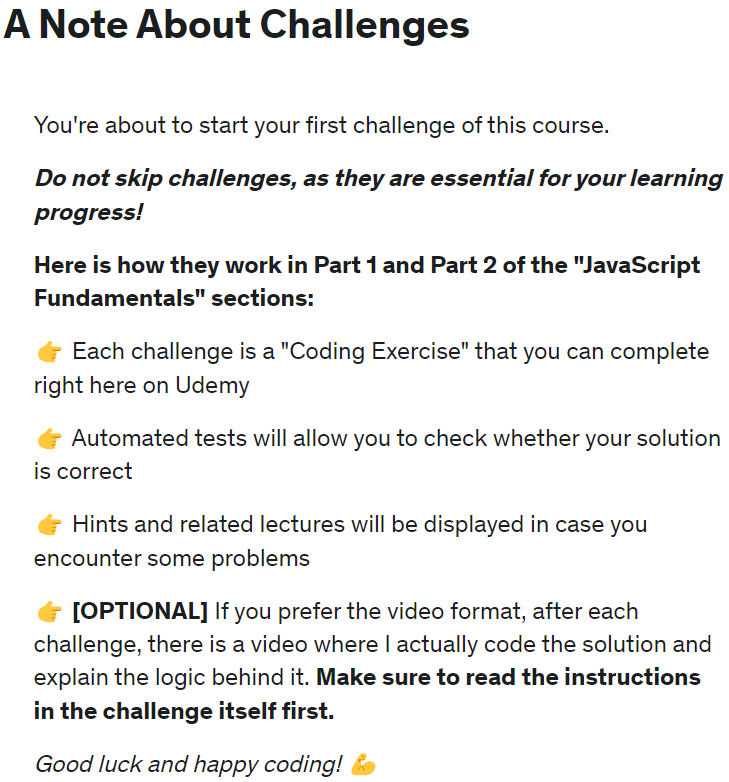
14. Operator Precedence

=> See operator precedence table

1. JavaScript has a well define order of operator precedence.

Note: viewing the seat of the operator precedence table.

15. A Note About Challenges



16. Coding Challenge #1:

17. Strings and Template Literals:

*const* firstName = 'Jonas';

*const* job = 'teacher';

*const* birthYear = 1991;

*const* year = 2037;

1. Normal strings
2. *const* jonas =
3. "I'm " + firstName + ', a ' + (year - birthYear) + ' year old ' + job + '!';
4. console.log(jonas);

b. Template Literals(es6): Template Literals we can write a string in a more normal way and then basically insert the variables directly into the string. [assemble multiple pieces into one final strings]

*const* jonasNew = `I'm ${firstName}, a ${year - birthYear} year old ${job}!`;

console.log(jonasNew);

*// # backticks symbol= {``};*

c. tells JavaScript that were writing a template string, the single quote or double quote won’t work....

console.log (`Just a regular string...`);

d. multiline string: you would have to write, string with and you needed this backslash, n, backslash, means new line. so, this is like a special character in programming, which is string always means that a new line should be created.

example:

console.log ('String with \n\

multiple \n\

lines');

e.

console.log (`String

multiple

lines`);

18. CHALLENGE #2: Video Solution

19. Taking Decisions: if / else Statements

a. if Statements: all we have to do is write {if} the open parenthesis and then in here goes a condition that is evaluated, and this condition turns out to be true, then this block will be executed.

b. else: this else block will basically be executed whenever this condition here false.

example:

*const* age = 15;

if (age >= 18) {

  console.log("Sarah can start driving license 🚗");

} else {

*const* yearsLeft = 18 - age;

  console.log(`Sarah is too young. Wait another ${yearsLeft} years :)`);

}

c. we take decision with code all the time which is essentially what we did here. so, we can execute certain parts of our program based on certain condition.

d. If/ else: if / else Statement which is the official name of this kind of construction here is called control structure.

if (){

}

else {

}

e. control structure: it is called control structure because this structure actually allows us to have more control over the way that our code is executed. For example, in this, 'if', 'else' statement the whole code does not just execute in a linear way. So, JavaScript does not execute all of this code here, linearly. Instead, we can now control blocks of code that should execute and blocks that should not execute.

f.

another example:

1. created a variable conditionally

2.not just always use "console.log"

3. which will basically contain the century in which this person was born.

*const* birthYear = 2012;

*let* century;

if (birthYear <= 2000) {

  century = 20;

} else {

  century = 21;

}

console.log(century);

note: what you need to know is that this is because a variable that we define inside of any code block, which is what this year is so this is a code block, and this is a code block. And any variable that we declare inside of one of these blocks will not be accessible outside of the block.

20. Type Conversion and Coercion:

a. types are one of the fundamental aspects in programming, and converting between types is something that we do in every programming language.

example:

1.converting string to a number;

2. a number into Boolean;

b. In JavaScript, there is type Conversion and type coercion.

Type conventions: when we manually convert from one type to another type.

Type coercion: when JavaScript automatically convert types behind the scenes for us.

example:

const inputYear = '1991';

type coercion: is when JavaScript automatically converts types behind the scenes for us.

c. NaN = whenever an operation that involves number fails to produce a new number. so basically, not a number actually means invalid number.

d. JavaScript can only convert to three types.

1. number to string to Boolean

e. cannot convert undefined or null.

21. truthy and falsy values:

In JavaScript, the following values are considered falsy, meaning they are evaluated as false in boolean contexts:

1. `false`: The boolean value `false`.

2. `0`: The number zero.

3. `""` (empty string): An empty string with no characters.

4. `null`: The absence of any object value.

5. `undefined`: A variable that has been declared but has not been assigned a value.

6. `NaN`: Not-a-Number, resulting from an invalid mathematical operation.

These values are considered falsy because when they are evaluated in a boolean context, they are converted to the boolean value `false`. It's important to note that any other value not listed above is considered truthy and will be evaluated as true in boolean contexts.

When you use these values in conditional statements or logical expressions that expect a boolean result, they will be treated as false.

Keep in mind that the concept of truthy and falsy values is important when working with conditional statements and boolean expressions in JavaScript.

truthy: everything else are our so called truthy values. So, values that will be converted to true.

any number that is not zero or any string that is not an empty string will be converted to true.

Note: but in practice, I actually never did this in my life. I never used this function here in practice. This was just to show you the concept of truthy and falsy values.

22.equality operators: == vs. ===

In JavaScript, there are two equality operators: `==` (loose equality) and `===` (strict equality). They are used to compare values for equality, but they have different behaviors and comparison rules.

1. Loose Equality Operator (`==`):

- The `==` operator performs type coercion, which means it converts the operands to a common type before comparison.

- If the operands have different types, JavaScript tries to convert them to a common type based on certain rules.

- After the type coercion, the loose equality operator compares the values.

- It performs implicit type conversions, which can lead to unexpected results in certain cases.

console.log(5 == "5"); *// Output: true (type coercion: string "5" converted to number 5)*

console.log(1 == true); *// Output: true (type coercion: boolean true converted to number 1)*

2. Strict Equality Operator (`===`):

- The `===` operator compares the values of the operands without performing any type coercion.

- It checks both the value and the type of the operands for equality.

- The strict equality operator is more reliable and avoids unexpected results due to type coercion.

console.log(5 === "5");  *// Output: false (different types: number vs. string)*

console.log(1 === true); *// Output: false (different types: number vs. boolean)*

In general, it is recommended to use the strict equality operator (`===`) in JavaScript because it ensures both the value and the type are equal. It leads to more predictable and less error-prone code. The loose equality operator (`==`) can be useful in specific cases where type coercion is desired, but it requires careful consideration to avoid unintended consequences.

Remember that when comparing values, it's essential to understand the difference between the loose and strict equality operators and choose the appropriate one based on your specific requirements.

23. Boolean logic:

24.logical operators:

Boolean logic, also known as logical operations or boolean algebra, is a branch of mathematics and computer science that deals with the manipulation and evaluation of boolean values. In programming, boolean logic is used to make decisions, control program flow, and perform logical operations.

Boolean values represent the concept of true or false. In JavaScript, the boolean type has two possible values: `true` and `false`.

Boolean logic includes three fundamental logical operations:

1. AND (`&&`):

- The AND operator returns `true` if both operands are `true`, otherwise it returns `false`.

- Example: `true && true` returns `true`, while `true && false` or `false && true` or `false && false` all return `false`.

2. OR (`||`):

- The OR operator returns `true` if at least one of the operands is `true`, otherwise it returns `false`.

- Example: `true || false` or `false || true` or `true || true` all return `true`, while `false || false` returns `false`.

3. NOT (`!`):

- The NOT operator negates the boolean value of the operand. If the operand is `true`, it returns `false`, and if the operand is `false`, it returns `true`.

- Example: `!true` returns `false`, and `!false` returns `true`.

These logical operations can be combined and nested to form complex boolean expressions. Parentheses can be used to group operations and specify the order of evaluation.

Here's an example to illustrate boolean logic in JavaScript:

*var* x = 5;

*var* y = 10;

*var* z = 15;

*// Logical AND (&&)*

console.log(x < y && y < z); *// Output: true (both conditions are true)*

*// Logical OR (||)*

console.log(x > y || y < z); *// Output: true (at least one condition is true)*

*// Logical NOT (!)*

console.log(!(x > y)); *// Output: true (negating the condition)*

Boolean logic allows you to make decisions based on conditions, combine conditions using logical operators, and control the flow of your program. It forms an essential part of programming and is used extensively in conditional statements, loops, and other control structures.

25.coding challenge #3

26.The switch Statement:

The `switch` statement is a control flow statement in JavaScript that allows you to select one of many code blocks to execute based on the value of an expression. It provides a way to simplify multiple `if...else if...else` statements when you have a specific value to match.

Here's the basic syntax of a `switch` statement:

switch (expression) {

  case value1:

*// Code to execute when expression matches value1*

    break;

  case value2:

*// Code to execute when expression matches value2*

    break;

  case value3:

*// Code to execute when expression matches value3*

    break;

*// Add more cases as needed*

  default:

*// Code to execute when none of the cases match*

    break;

}

Here's how the `switch` statement works:

1. The `expression` is evaluated once, and its value is compared with the values specified in the `case` clauses.

2. If a match is found between the expression and a `case` value, the corresponding code block is executed.

3. The `break` statement is used to exit the `switch` statement once a match is found. Without the `break` statement, the execution will continue to the next `case` block, regardless of whether it matches or not (known as "fall-through").

4. If none of the `case` values match the expression, the code block under the `default` clause is executed (optional). It acts as a default case if no other cases match.

Here's an example to illustrate the usage of the `switch` statement:

*var* day = 2;

*var* dayName;

switch (day) {

  case 1:

    dayName = "Monday";

    break;

  case 2:

    dayName = "Tuesday";

    break;

  case 3:

    dayName = "Wednesday";

    break;

  case 4:

    dayName = "Thursday";

    break;

  case 5:

    dayName = "Friday";

    break;

  case 6:

    dayName = "Saturday";

    break;

  case 7:

    dayName = "Sunday";

    break;

  default:

    dayName = "Invalid day";

    break;

}

console.log(dayName); *// Output: "Tuesday"*

In this example, the `switch` statement is used to assign a corresponding day name to the `dayName` variable based on the value of the `day` variable.

Note that the `switch` statement works with strict equality (`===`), so it compares both the value and the type of the expression with the `case` values.

The `switch` statement is a useful tool when you have multiple conditions to check against a single expression. However, it's important to use it appropriately and consider alternative control flow structures when needed.

27.Statements and Expressions:

a. Expressions: An Expressions is a piece of code the produces a value.

example: 3 + 4 = 7;// Expressions

1999;// Expressions

true && false &&! false//Expressions

Note: will produce a value in JavaScript is an Expressions.

b.

statement: statement is like bigger piece of code that is executed and which does not produce a value on itself.

c.

we can compare this with normal spoken language.

d.

A declaration is like a complete sentence and Expressions are like the words that make up the sentence. Now, basically we write our whole programs as a sequence of actions. And these actions are statement.

example:

if (23 > 10){

const str = '23 is bigger';

console.log(str);}

Note: whenever something ends with a semicolon, that's then a statement.

e. now this difference between expressions and statements is important to know Because JavaScript expects statement and expressions in different places.

example:

in a template literal, we can only insert expressions, but not statement.

28. The Conditional (Ternary):

a. so Conditional operator allows us to something similar to an if/else statement but all in one line.

example:

const age = 23;

const check = age >= 18 ? console.log('I like to drink wine 🍷') : console.log('I like to drink water 💧');

and now let's write, age, grater or equal 18. so, this is your or test to check whether the age is at least 18. after this Condition we use a question mark. and then here we write essentially or if block. so that's the code that we want to be executed in case that dies Condition here is true. so, let's say console.log {I like to drink milk} the we also need basically a mandatory else block. and that goes after this {:} colon.

b. so that’s essentially writing an if/else statement all in one line. But instead of using of using a statement, we use this conditional operator. and the conditional operator is also called the ternary operator. because it has three parts. Unlike another operator.

1. the Condition

2. then if parts

3. then the else part.

c. an operator always produces a value. so, in other words an operator is an expression right. so, what this means is that if we have a value, we can then assign to a variable. so, with this we can make the ternary operator really useful to basically conditionally declare variables.

29.Coding Challenge #4

30.JavaScript Releases: ES5, ES6+ and