Brainnest Homework / Terminology & Definitions (JavaScript)
Clémence TAN (group B)

1. Name the three ways to declare a variable? **(pptx p-)**
=> **var** term = xyz
=> **let** term1 = abc
=> **const** term2 = efg

2. Which of the three variable declarations should you avoid and why?
=> the « **var** » declaration should be avoided
=> variables declared with « var » have the global scope

3. What rules should you follow when naming variables?
=> we should avoid using a list of reserved keywords, like « var », « let », « return »...
=> the variable name should be in camelCase, like : « newUser »
=> the variable name should not start with a digit
=> the name must contain only letters, digits, or symbols as « **$** » and « **_** »
=> variables are case-sensitive
=> we should also avoid naming the variables with non-Latin words
=> uppercase constants, like `const COLOR_ORANGE = "#FF7F00";` for difficult-to-remember values
=> make names maximally descriptive and concise, like « currentUser », « newUser » etc

4. What should you look out for when using the + operator with numbers and strings ?
=> it is the « **unary +** » operator
=> if we type `let x = 1 + "hen"`
=> the anwser will be « **1hen** »

5. How does the % operator work?
=> It's about « modulo », for example, 13 % 4 = 1
=> it's about what **remains** after the division operation is done
=> 13 / 4 = 3 (with 1 as remainder)

6. Explain the difference between == and ===
=> == this operator does the type conversion of the operands before comparison
=> === this operator compares the values as well as the **data types** of the operands

7. When would you receive a NaN result?
=> if I do the following operations :

```
b = "NaN" / 2;
console.log(b); // NaN
```

```
c = "NaN" + 99;
console.log(c); // NaN99
```

```
d = "NaN" * 3;
console.log(d); // NaN
```

8. How do you increment and decrement a number?
=> increment : `++`
=> decrement : `--`

9. Explain the difference between prefixing and post-fixing increment/decrement operators.
=> **postfix** (**i++**) : it increments the value and returns the value **before** the increment
=> **prefix** (**++i**) : it increments the value and returns the value **after** the increment

10. What is operator precedence and how is it handled in JS?
=> It determines which operators have higher preceence than others, and tells us the order to perform a given mathematical expression.

11. How do you log information to the console?
=> we type « console.log() », this is a way to print information to the developer console in the browser.

12. What does unary plus operator do to string representations of integers?
=> It can convert string representations of **integers** and **floats**, as well as the non-string values **true**, **false**, and **null**.

13. What are the **eight data types** in JavaScript?
=> numbers / strings / boolean / null / undefined / objects / symbols / bigint

14. Which data type is NOT primitive?
=> the « **object** » is a non-primitive data type.

15. What is the relationshiop between null and undefined?
=> « **null** » : is the intentional absence of the value
=> « **undefined** » : means the value does not exist in the compiler.

16. What is the difference between signle, double, and backtick quotes for strings?
=> There is no special difference between them, simply it's better to stick to the same format throughout a project.
=> « single quotes » : `'string'`
=> « double quotes » : `"string"`
=> « backticks » : `` `string` ``

17. What is the term for embedding variables/expressions in a string?
=> it is about « **string interpolation** »

18. Which type of quote lets you embed variables/expressions in a string?
=> « **backticks** »

19. How do you embed variables/expressions in a string?

```
let times = 3;
console.log(`I went to Japan ${times} times in my life.`);
console.log("I went to Japan" + " " + times + " times in my life.");
```

Both results are the same : **I went to Japan 3 times in my life.**

20. How do you escape characters in a string?

=> « **backslash** » ****

21. What is the difference between the **slice**/**substr**/**substring** methods?

=> string.prototype.**substr** : considered as deprecated, not suitable for new website use
substr() method is even **crossed out** in vsCode

```
const chaîne = 'Twas the night before Christmas';
let start = -9;
let length = 9;
chaîne.substr(start, length); // 'Christmas'
'Christmas'.length; // 9
```

=> string.prototype.**substring** : there is a start_index and an end_index
if start_index is not mentioned,
the rest of the string is returned,
starting from the start_index

```
const str = 'Twas the night before Christmas';
let indexStart = 0;
let indexEnd = 4;
console.log(str.substring(indexStart, indexEnd)) // 'Twas'
console.log(str.substring(5, 14)) // 'the night'
console.log(str.substring(5)) // 'the night before Christmas'
```

=> string.prototype.**slice** : seems to be the **best choice** among the three
the name is shorter and clearer,
it also upports negative indices

```
const wordStr = 'Twas the night before Christmas';
console.log(wordStr.slice(0, 4)); // Twas
console.log(wordStr.slice(5, 14)); // the night
console.log(wordStr.slice(-16, -10)); // before
console.log(wordStr.slice(-9)); // Christmas
```

22. What are the three logical operators and what do they stand for?

=> **||** (OR)

=> **&&** (AND)

=> **!** (NOT)

23. What are the comparison operators?

Operator	<	>	<=	>=	==	!=
Meaning	less than	greater than	less than or equal to	greater than or equal to	equal to	not equal to

24. What are truthy and falsy values?

truthy	'0' / 'false' / [ ] / { } / function(){}
falsy	False / 0 / -0 / 0n (BigInt) / '' `` (empty string) / null / undifined / NaN

25. What are the falsy values in JavaScript?

==	true	false	0	''	null	undefined	NaN	Infinity	[]	{}
true	true	false	false	false	false	false	false	false	false	false
false	false	true	true	true	false	false	false	false	true	false
0	false	true	true	true	false	false	false	false	true	false
''	false	true	true	true	false	false	false	false	true	false
null	false	false	false	false	true	true	false	false	false	false
undefined	false	false	false	false	true	true	false	false	false	false
NaN	false	false	false	false	false	false	false	false	false	false
Infinity	false	false	false	false	false	false	false	true	false	false
[]	false	true	true	true	false	false	false	false	false	false
{}	false	false	false	false	false	false	false	false	false	false

26. What are conditionals?
=> the syntax `if () {}`
=> the conditional statements in JavaScript control and determine whether or not pieces of code can run

27. What is the syntax for an if/else conditional?

```
let hours = 4; // 0 - 24
if (hours < 13) { // 0-12
  console.log("Good morning");
} else { // 13-24
  console.log("Good afternoon");
}
```

28. What is the syntax for a switch statement?

```
switch(expression) {
  case x:
    // code block
    break;
  case y:
    // code block
    break;
  default:
    // code block
}
```

29. What is the syntax for a ternary operator?

```
condition ? expressionIfTrue : expressionIfFalse
```

30. What is nesting?
=> a function nested inside another function, for example :

```
function foo() {
  function bar() {
    return 1;
  }
  return bar();
}
```

31. What are functions useful for?
=> functions are useful because they can perform a task or calculate a value

32. How do you invoke a function?
=> it means to « call a function »
=> below, **myFunction(10, 2) ;** is calling of the function

```
function myFunction(a, b) {
  return a * b;
}
myFunction(10, 2);
```

33. What are anonymous functions?
=> it's a function that does not have any name associated with it

34. What is function scope?
=> the « scope » determines the accessibility of variables and other resources in the code, like functions and objects.
=> JavaScript function scopes can have two different types, the **local** and the **global** scope.

35. What are return values?
=> the values that a function returns when it completes

36. What are arrow functions?
=> an arrow function is a compact alternative to a traditional function expression, with some semantic differences and deliberate limitations in usage.
=> arrow function sdon't have their own bindings and arguments and should not be used as methods.