Codergirl - JavaScript

Class 2

March 10th, 2020

Agenda

- Studio review
- Conditionals and Debugging
- Recap of Exercises
- Studio

boolean

true or false

NOT True or False and

NOT ‘true’ or ‘false’

```
let ateDinner = true;

let readyToLearn = true;

let bored = false;
```

Comparison Operators

- ==
- !=
- >
- <
- >=
- <=

Comparison Operators

Operator	Description	Examples Returning `true`	Examples Returning `false`
Equal (`==`)	Returns `true` if the two operands are equal, and `false` otherwise.	`7 == 7` `"dog" == "dog"`	`7 == 5` `"dog" == "cat"`
Not equal(`!=`)	Returns `true` if the two operands are not equal, and `false` otherwise.	`7 != 5` `"dog" != "cat"`	`7 != 7` `"dog" != "dog"`
Greater than (`>`)	Returns `true` if the left-hand operand is greater than the right-hand operand, and `false` otherwise.	`7 > 5` `'b' > 'a'`	`5 > 7` `'a' > 'b'`
Less than (`<`)	Returns `true` if the left-hand operand is less than the right-hand operand, and `false` otherwise.	`5 < 7` `'a' < 'b'`	`7 < 5` `'b' < 'a'`

Greater than or equal (`>=`)	Returns `true` if the left-hand operand is greater than or equal to the right-hand operand, and `false` otherwise.	`7 >= 5`
		`7 >= 7`
		`'b' >= 'a'`
		`'b' >= 'b'`
			`5 >= 7`
			`'a' >= 'b'`

Less than or equal (`<=`)	Returns `true` if the left-hand operand is less than or equal to the right-hand operand, and `false` otherwise.	`5 <= 7`
		`5 <= 5`
		`'a' <= 'b'`
		`'a' <= 'a'`
			`7 <= 5`
			`'b' <= 'a'`

Loose/Strict Equality

- == versus ===

4 == "4"	**true**
4 === "4"	false
0 == ""	true
0 === ""	false

Use a === strict equality (or !== strict inequality) comparisons to compare values and avoid type conversion issues:

Logical Operators

- && || !
- homeworkCompleted && choresCompleted
- ateFruits || ateVeggies
- !misbehaving

Truth tables for && and ||

For && to be true all conditions should be true

For || to be true any one of the conditions should be true

Operator Precedence

Precedence	Category	Operators
(highest)	Logical NOT	`!`
	Exponentiation	`**`
	Multiplication and division	`*`, `/`, `%`
	Addition and subtraction	`+`, `-`
	Comparison	`<=`, `>=`, `>`, `<`
	Equality	`===`, `!==`, `==`, `!=`
	Logical AND	`&&`
(lowest)	Logical OR	`||`

5.2.2 Truth Tables

if statements

```
if(condition) {
        do X
}
```

if else

```
if(condition) {
        do X
} else {
        do Y
}
```

if else if

```
if(condition) {
        do X
} else if(condition) {
        do Y
}
```

nested if

```
if(condition A) {
        if(condition B) {
                do X
        }
}
```

Errors

- Syntax - error during parsing.
- Runtime - error during execution.
- Logic – unexpected outcome.

JavaScript Error Types

Error Type	Description	Example of code triggering the error	Example description
`SyntaxError`	Occurs when trying to parse syntactically invalid code.	`console.log("hello";`	The call to `console.log` does not have a required close parenthesis.
`ReferenceError`	Occurs when a non-existent variable is used/referenced.	`let firstName = "Jack";` `console.log(firstname);`	The variable `firstname` does not exist; it is a misspelling of `firstName`.
`TypeError`	Occurs when trying to use a value in an invalid way.	`1();`	The numeric value `1` is not a function, so trying to use it as one results in `TypeError: 1 is not a function`.
`RangeError`	Occurs when passing an invalid value to a function.	`let nums = Array(-1);`	The constructor function `Array(n)` creates an empty array of length `n`. It is not possible to create an array with negative length, so the code results in `RangeError: Invalid array length`.
`URIError`	Occurs when improperly using a global URI-handling function. ('URI' = Uniform Resource Identifier)	`decodeURI('%');`	The `%` character is used to encode characters not otherwise allowed in URIs, such as spaces (`%20`). If an invalid character encoding is given, a `URIError` results.
`Error`	The type from which all other errors are built. It can be used to generate programmer-triggered and programmer-defined errors.	`throw Error("Something bad happened!");`	Manually triggers an error with the given message.

Each time you encounter a new error type, take the time to understand what it is, and what JavaScript is trying to tell you.

Remember, **error messages are your friends!**

Debugging

- error messages
- console.log
- google

Best practices

- Tests
- Incremental changes
- Pairing/Mobbing

Code examples and exercises

1. Booleans
2. Equality
3. Comparisons
4. Operators
5. If
6. Nested
7. Syntax, runtime, logic
8. Exercises 12, exercises 34, exercises 56
9. Debugging 1,2,3,4,5

Graded Assignment # 1 part

https://education.launchcode.org/intro-to-professional-web-dev/assignments/candidateQuiz.html

Questions?

Studio time!