M02 - JavaScript Fundamentals

•••

Variables

Variables

1. Declaration and Assignment
2. Constants
3. Naming
4. Scope

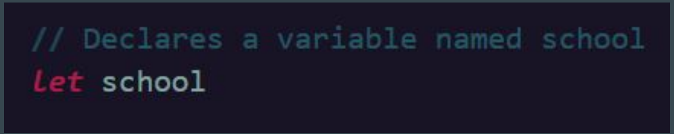
Variables

1. Declaration and Assignment

Variables

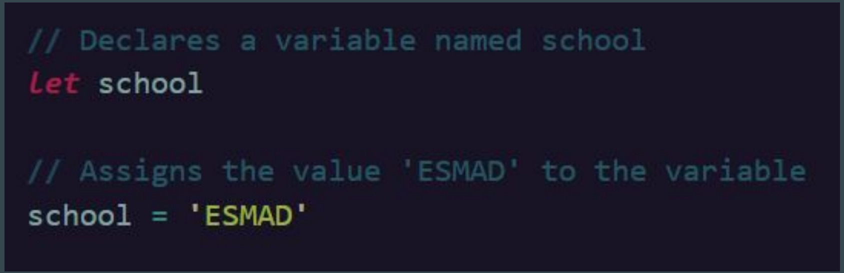
1. Declaration and Assignment

- Variables are named containers for storing data values.
- To declare a variable in JavaScript, use the keyword let



```
// Declares a variable named school
let school
```

- After the declaration, the variable has no value (has the value undefined)
- To assign a value to a variable, use the assignment operator =



```
// Declares a variable named school
let school

// Assigns the value 'ESMAD' to the variable
school = 'ESMAD'
```

Variables

1. Declaration and Assignment

- To be concise, you can combine the declaration and assignment on a single line:

```
// Declares a variable named school and assigns a value to it
let school = 'ESMAD'
```

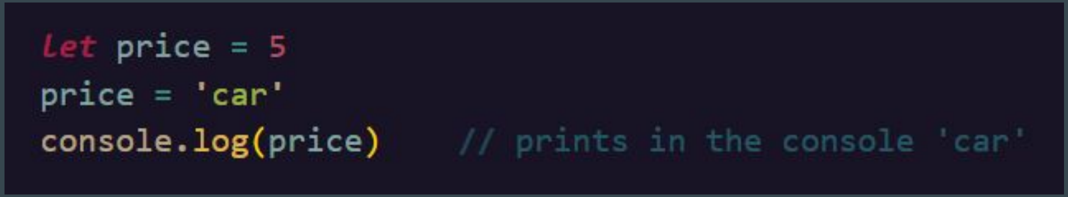
- You can also declare multiple variables on a single line:

```
// Declare and assign multiple variables in the same line
let school = 'ESMAD', city = 'Porto', age = 34
```

Variables

1. Declaration and Assignment

- A variable in JS can have a value of any type
- Unlike other languages, you do not need to tell JavaScript during variable declaration what type of value the variable will keep
  The value type of a variable can change during the execution of a program and JavaScript takes care of that automatically



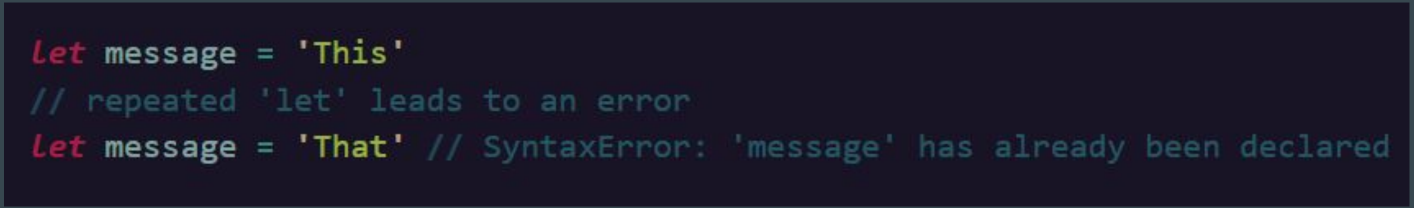
```
let price = 5
price = 'car'
console.log(price)    // prints in the console 'car'
```

- This feature is called dynamic typing

Variables

1. Declaration and Assignment

- A variable should be declared only once
- A repeated declaration of the same variable is an error:



```
let message = 'This'
// repeated 'let' leads to an error
let message = 'That' // SyntaxError: 'message' has already been declared
```

- So, we should declare a variable once and then refer to it without let

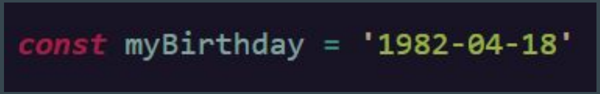
Variables

2. Constants

Variables

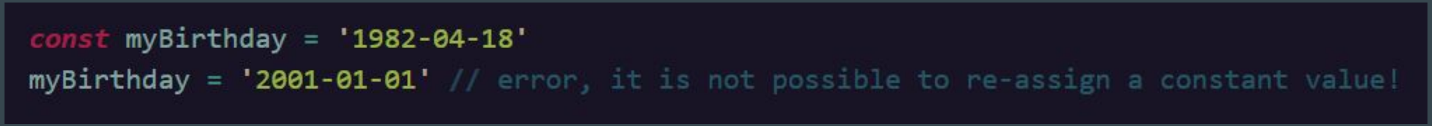
2. Constants

- To declare a constant (immutable) variable, use const instead of let:



```
const myBirthday = '1982-04-18'
```

- Variables declared using const are called "constants"
- They cannot be changed. An attempt to do so would cause an error:



```
const myBirthday = '1982-04-18'
myBirthday = '2001-01-01' // error, it is not possible to re-assign a constant value!
```

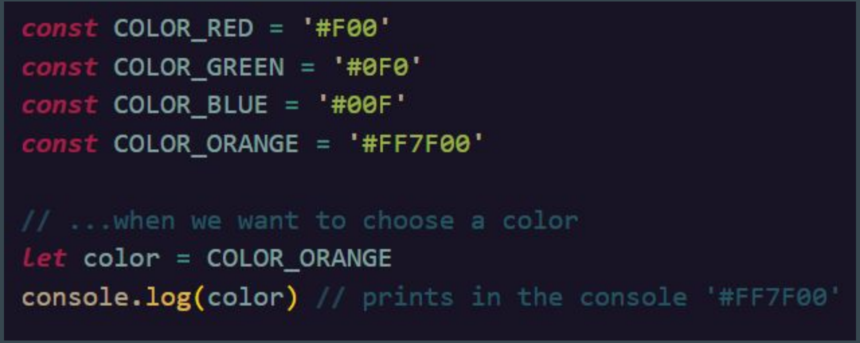
- When a programmer is sure that a variable will never change, he must declare it with const to ensure and clearly communicate that fact to everyone

Variables

2. Constants

- There is a widespread practice of using constants as *aliases* for hard-to-remember values known before execution

- These constants are named using capital letters and *underlines*



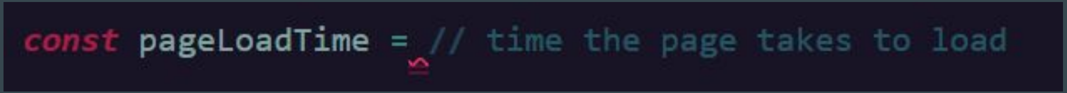
```
const COLOR_RED = '#F00'
const COLOR_GREEN = '#0F0'
const COLOR_BLUE = '#00F'
const COLOR_ORANGE = '#FF7F00'

// ...when we want to choose a color
let color = COLOR_ORANGE
console.log(color) // prints in the console '#FF7F00'
```

Variables

2. Constants

- There are constants that
  - are known before execution (as a hexadecimal value for red)
  - are calculated at run time, but are not changed after the initial assignment



```
const pageLoadTime = // time the page takes to load
```

- The pageLoadTime value is not known before the page loads, so it is named normally. But it's a constant because it doesn't change after the assignment
- In other words, capitalized constants are used only as aliases for *hard-coded* values

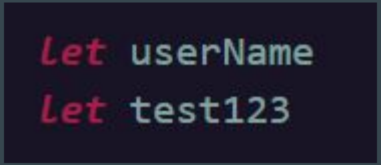
Variables

3. Naming

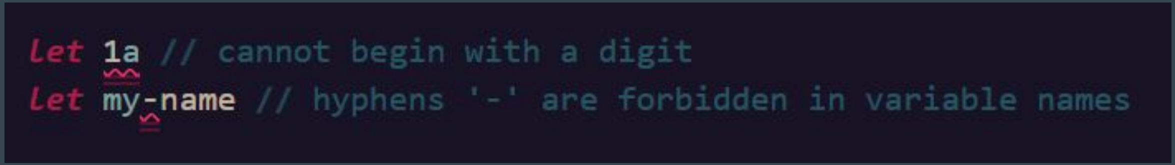
Variables

3. Naming

- There are two limitations to variable names in JavaScript:
  - The name must contain only letters, digits or the symbols $ and _
  - The first character cannot be a digit
- Examples of valid and invalid names:



```javascript
let userName
let test123
```



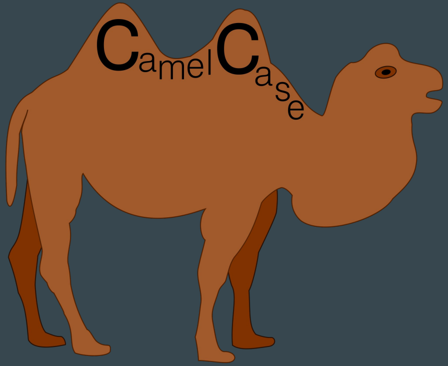
```javascript
let 1a // cannot begin with a digit
let my-name // hyphens '-' are forbidden in variable names
```

Variables

3. Naming

- Do not use names that are too short or names that are too long
- When the name contains multiple words, camelCase is commonly used
  - Words go one after the other, each beginning with a capital letter
  - Example: myVeryLongName

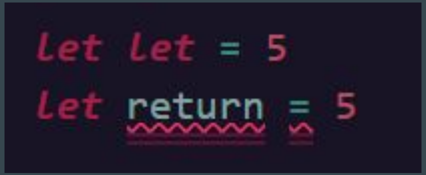
- Other naming conventions: snake_case



Variables

3. Naming

- Upper and lowercase
  - Variable names are case sensitive
  - The apple and AppLE variable names are 2 different variables
- Reserved words
  - There is also a list of reserved words, which cannot be used as variable names because they are used by the language itself
  - For example: let, class, return and function are reserved



```
let let = 5
let return = 5
```

Variables

3. Naming

- Good practices:
  a. Use readable names like userName or shoppingCart
  b. Don't use abbreviations or short names like a, b, c, unless you really know what you're doing
  c. Create descriptive and concise names. Examples of invalid names are data and value. These names say nothing. There is no problem using them if the context of the code makes it exceptionally obvious what data or values the variable is referencing
  d. Take into account the terms used by the development team. If a site visitor is called a user, we should name related variables currentUser or newUser instead of currentVisitor or newManInTown
  e. Don’t mix natural languages (portuguese with english)
     - Use evenly only one
     - Suggestion: english (reserved words of the language already in english, world wide readable code, etc.)

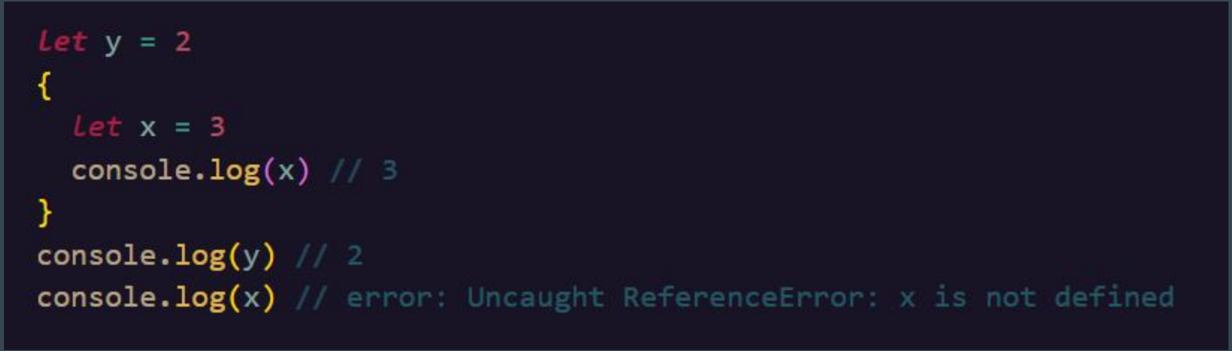
Variables

4. Scope

Variables

4. Scope

- The scope of a variable is the context where it was defined
- Naming variables via let or const restricts the variable's access to the nearest surrounding block
- A JavaScript block is represented by { ... }



```
let y = 2
{
  let x = 3
  console.log(x) // 3
}
console.log(y) // 2
console.log(x) // error: Uncaught ReferenceError: x is not defined
```