Types & Operators, Variables, and Control Flow

Module – Advanced JS

Dr. Andrew Errity

iadt
DUN LAOGHAIRE

Activity

- Complete part 1:
  - https://www.codecademy.com/learn/introduction-to-javascript
- You may need to refer to the Mozilla JS docs
  - https://developer.mozilla.org/en-US/docs/Learn/JavaScript

Types and Operators

- Four essential data types in JavaScript include strings, numbers, booleans, and null.
- Data is printed, or logged, to the console with console.log().
- Four built-in mathematical operators include +, -, *, and /.
- JavaScript associates certain properties with different data types.
- JavaScript has built-in methods for different data types.
- Libraries are collections of methods that can be called without an instance.
- You can write single-line comments with // and multi-line comments between /* and */.

Variables

- Variables hold reusable data in a program.
- JavaScript will throw an error if you try to reassign `const` variables.
- You can reassign variables that you create with the `let` keyword.
- Unset variables store the primitive data type undefined.
- Mathematical assignment operators make it easy to calculate a new value and assign it to the same variable.
- The `+` operator is used to interpolate (combine) multiple strings.
- In JavaScript ES6, backticks (`) and `${}` are used to interpolate values into a string.

`var`, `let`, and `const`

- `var`
  - scope of a variable declared with **`var`** is its current *execution context*, which is either the enclosing function or, for variables declared outside any function, global.
- `let`
  - declares a block scope local variable
- `const`
  - are block-scoped, much like variables defined using the **`let`** statement. The value of a constant cannot change through re-assignment, and it can't be redeclared

Assignment Operators

```
let x = 4;
x += 2; // x equals 6
let y = 4;
y -= 2; // y equals 2
let z = 4;
z *= 2; // z equals 8
```

String Interpolation

```
let myName = 'Andrew';
let myCity = 'Seoul';
console.log(`My name is ${myName}. My
favorite city is ${myCity}.`)
```

Control Flow

- if/else statements make binary decisions and execute different code based on conditions.
- All conditions are evaluated to be truthy or falsy.
- We can add more conditional statements to if/else statements with else if.
- switch statements make complicated if/else statements easier to read and achieve the same result.
- The ternary operator (?) and a colon (:) allow us to refactor simple if/else statements.
- Comparison operators, including <, >, <=, and >= can compare two variables or values.
- After two values are compared, the conditional statement evaluates to true or false.

Control Flow – Logical Operators

- The logical operator && checks if both sides of a condition are truthy.
- The logical operator || checks if either side is truthy.
- The logical operator !== checks if the two sides are not equal.
- An exclamation mark (!) switches the truthiness / falsiness of the value of a variable.
- One equals symbol (=) is used to assign a value to a variable.
- Three equals symbols (===) are used to check if two variables are equal to each other.

Falsy values

- false
- 0 and -0
- "" and '' (empty strings)
- null
- undefined
- NaN (Not a Number)
- document.all (something you will rarely encounter)

Ternary (?) Operator

```
let isNightTime = true;
if (isNightTime) {
   console.log('Turn on the lights!');
} else {
   console.log('Turn off the lights!');
}
```

```
isNightTime ? console.log('Turn on the lights!') : console.log('Turn off the lights!');
```

Switch Statements

```
let moonPhase = 'full';
switch (moonPhase){
   case 'full':
      console.log('Howl!');
   break;
   case 'mostly full':
      console.log('Arms and legs are getting
hairier');
   break;
   case 'mostly new':
      console.log('Back on two feet');
   break;
   default:
      console.log('Invalid moon phase');
   break;
}
```

Equality

- Note we will use the strict === and !== to check for equality rather than the looser == and !=

- Why?