# Lesson 3

# Перетворення типів

Явне перетворення:

- String();
- Number();
- Boolean();
- parseInt(value) - Функція яка приймає строку щоб створити ціле число. Наприклад parseInt("1") = 1;
- parseFloat(value) - Функція яка приймає строку щоб створити дробне число. Наприклад parseInt("1.1") = 1.1;

Не явне перетворення:

- "+" - Попереду від якогось значеня. Наприклад +true = 1
- У математичних виразах. Виняток "+", якщо є строка то результат буде строка;
- У логічних виразах.

# String transform

- String()
- .toString()
- JSON.stringify()

# Number transform

- Undefined = > NaN;
- True => 1;
- False => 0;
- Null => 0;

# Falsy values

- false;
- 0;
- NaN;
- Null;
- undefined;
- '' - пуста строка;

# Рядкові літерали

У alert() чи console.log();

- \n - перенос  на нову строку;
- \\ - екранування;
- \' - одинарна  лапка;
- \" - двойні лаки;
- \t - табуляція;

# Оператори

Унарні (є тільки один операнд):

- " + " (унарний плюс)
- " - "(унарний мінус)
- " ++ " (інкремент)
- (variable)++ - постфіксна форма (нижчий пріоритет)
- ++(variable) - префіксна форма (вищий пріоритет)
- " -- " (декремент)
- (variable)-- - постфіксна форма (нижчий пріоритет)
- --(variable) - префіксна форма (вищий пріоритет)

# Оператори (бінарні)

Арифметичні:

- " + " - додавання
- " - "- віднімання
- " / " - ділення
- " * " - множення
- " ** " - Піднесення до степеня (ES2016)
- " % " - Модуль (залишок). Повертає залишок від ділення.

Пріоритет операторів описує порядок виконання операцій в арифметичному виразі. Усе як і в традиційній шкільній математиці

Якщо у виразі присутня строка то " + " працює як конкатинація.

# Скорочені арифметичні оператори

- += - Додай та присвой значення
- -= - Відніми та присвой значення
- *= - Помнож та присвой значення
- /= - поділи та присвой значення

# Оператори (бінарні)

Порівняння:

- " > " - більше, ніж
- " < " - менше, ніж
- " >= " - більше, ніж або дорівнює
- " <= " - менше, ніж або дорівнює
- " == " - дорівнює
- " === " - рівне значення та рівний тип
- " != " - не дорівнює
- " !== " - не рівне значення або не рівний тип

Оператори порівняння та логічні оператори використовуються для перевірки true або false.

null == undefined => true => undefined === undefined

# Оператори (бінарні)

Логічні:

- " && " - and
- " || " - or
- " ! "  - not
- " ?? " - перевірка на null і undefined

# Оператори (бінарні)

|| робить наступне:

- Обчислює операнди зліва направо.
- Перетворює значення кожного операнда на булеве. Якщо результат true, зупиняється і повертає початкове значення цього операнда.
- Якщо всі операнди були обчисленні (тобто усі були false), повертає останній операнд.

Значення повертається у первісному вигляді без конвертації.

# Оператори (бінарні)

&& робить наступне:

- Обчислює операнди зліва направо.
- Перетворює кожен операнд на булевий. Якщо результат false, зупиняється і повертає оригінальне значення того операнда.
- Якщо всі операнди були обчисленні (тобто усі були правдиві), повертає останній операнд.

Значення повертається у первісному вигляді без конвертації.

Оператор І && має вищий пріоритет за АБО ||.

Отже, код a && b || c && d по суті є таким самим, як код з виразами && у дужках: (a && b) || (c && d).

# Оператори (бінарні)

Оператор ! приймає один аргумент і виконує наступне:

- Перетворює операнд на булевий тип: true/false.
- Повертає зворотне значення.

# Оператори (бінарні)

Оператор ?? повертає перший аргумент, якщо він не null/undefined. Інакше, другий.

Пріоритет оператора ?? такий самий, як у ||. Це означає, що, як і ||, оператор об'єднання з null ?? оцінюється до = та ?, але після більшості інших операцій, таких як +, *.

З міркувань безпеки, JavaScript забороняє використання ?? разом з операторами && та ||, якщо пріоритет явно не вказаний дужками.

```
let x = 1 && 2 ?? 3; // Синтаксична помилка

let x = (1 && 2) ?? 3; // Працює
```

# Дякую за увагу