

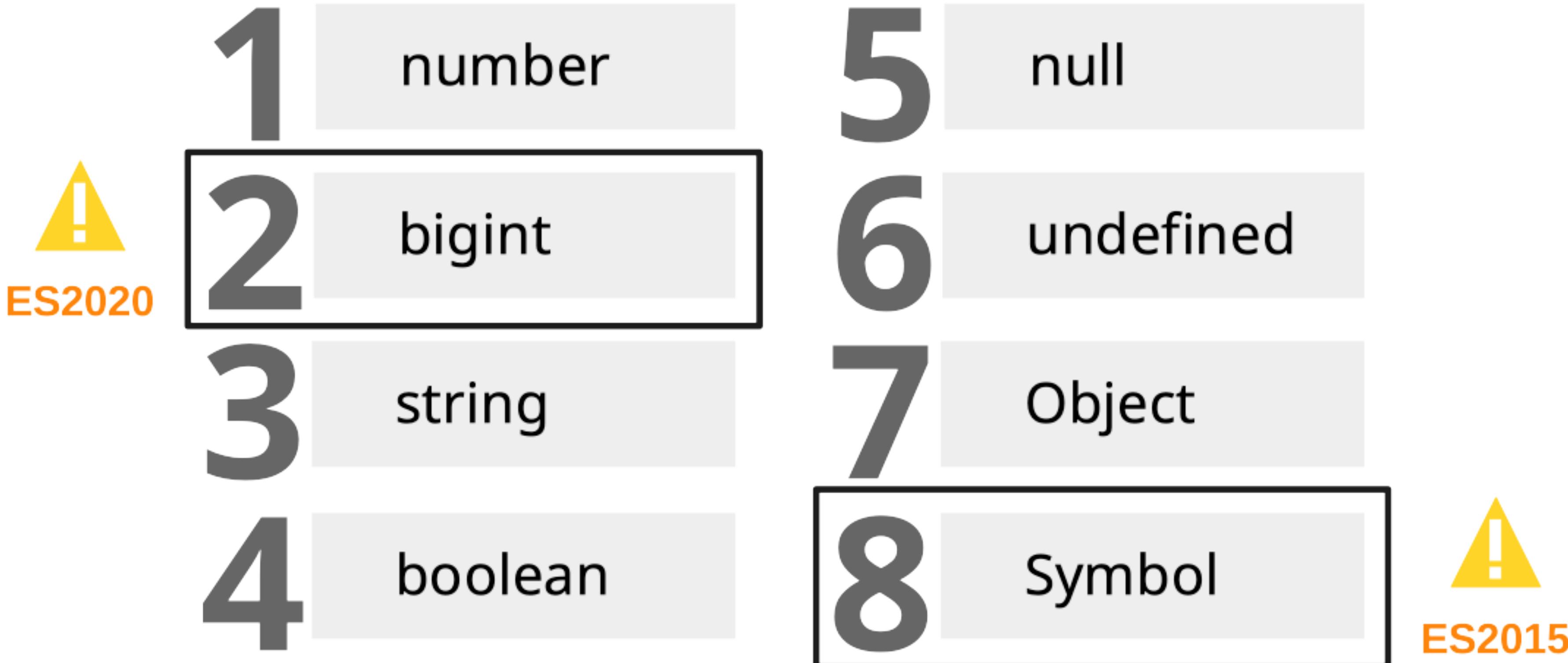
Typy danych

Primitives & Object

Przemysław Maćkowiak

Typy

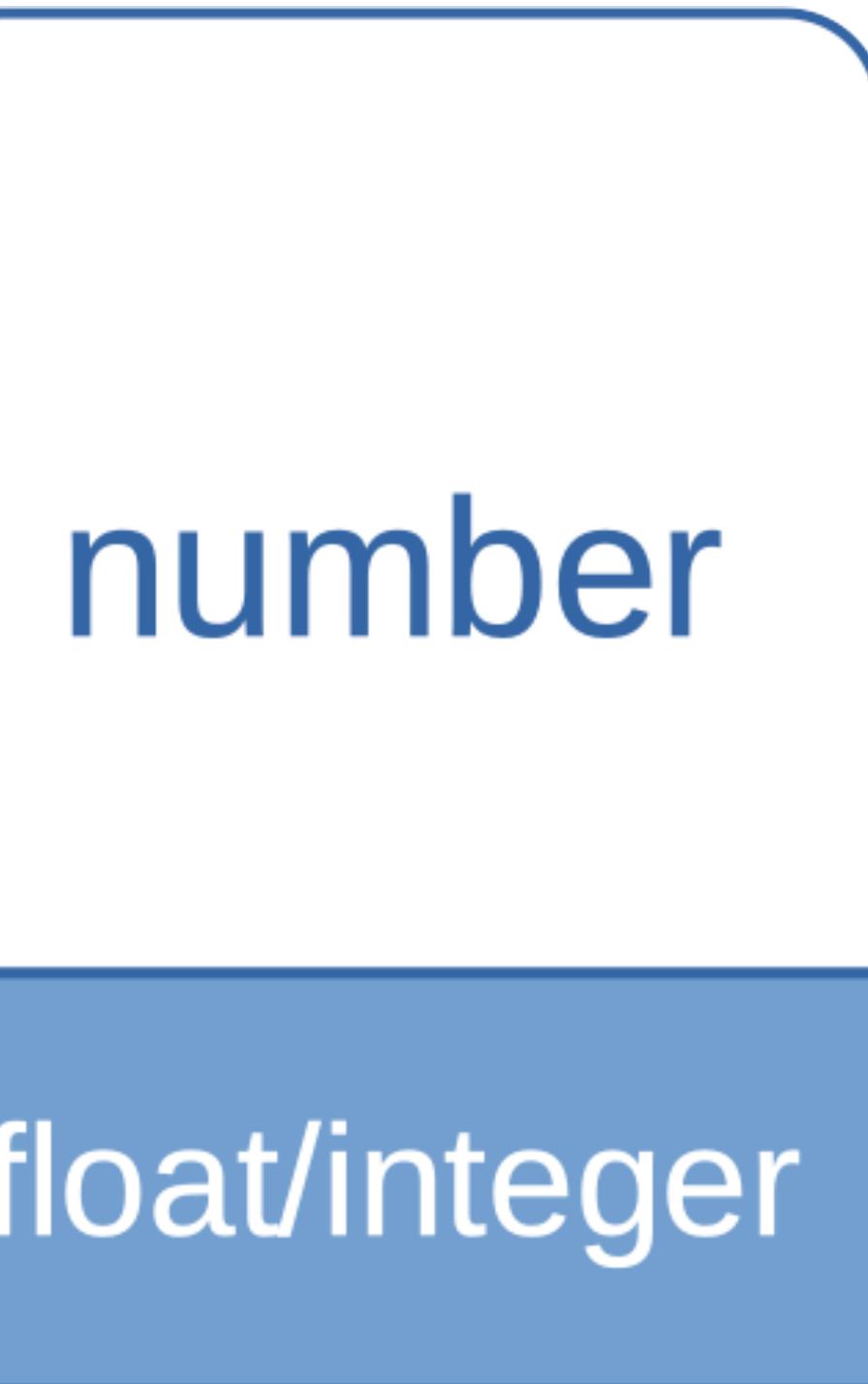
Primitive data types & Object



Number

Float / double / integer

- zarówno liczby całkowite jak i zmiennoprzecinkowe
- przewidziane operacje: arytmetyczne, na bitach, porównania, ...
- specjalne wartości Infinity, -Infinity, NaN
- bezpieczne operacje arytmetyczne (skrypt nie zginie przy dzieleniu przez 0)
- typ opakowujący: Number



Number

Przeglądarka vs Node

```
20:43:16.777 >> 34/34
```

```
20:43:16.864 <- 1
```

```
20:43:27.275 >> 12/0.34
```

```
20:43:27.365 <- 35.29411764705882
```

```
20:43:33.264 >> 4 + 5 * 6
```

```
20:43:33.341 <- 34
```

```
20:43:47.663 >> -2 * (4 - 9) * 3.14
```

```
20:43:47.735 <- 31.40000000000002
```

```
20:43:53.666 >> 12 / 0
```

```
20:43:53.733 <- Infinity
```

```
20:43:56.867 >> -120 / 0
```

```
20:43:56.938 <- -Infinity
```

```
20:44:02.877 >> "er" / 5
```

```
20:44:02.955 <- NaN
```

```
20:44:07.852 >> NaN + 12
```

```
20:44:07.964 <- NaN
```

```
20:44:12.872 >> Infinity - 30
```

```
20:44:12.950 <- Infinity
```

```
>  
> 12 ** 2 + 1  
145  
> 34 - 90  
-56  
>  
> 45 % 7  
3  
> 120 + 123 * 2 / 3  
202  
>  
> "test" / 3  
NaN  
> 5 / 0  
Infinity  
>
```

browser console

node console

Number

Zakres, błąd precyzji

- IEEE 754 (64bits, double precision)
- $(-2^{1024}; -(2^{-1074})) \text{ AND } (2^{-1074}; 2^{1024})$
- Automatyczna konwersja poza zakresem
- Błąd precyzji dla większych liczb

Number.MAX_SAFE_INTEGER

Number.MAX_VALUE



...

Zadanie

Wykonaj proste operacje matematyczne
(włączając: +, -, *, **, %, /)

Stwórz liczbę poza dopuszczalnym zakresem

Sprawdź czy $0.1 + 0.2 = 0.3$

Zobacz jaki jest wynik dodawania liczby 2
oraz największej bezpiecznej liczby naturalnej

Jaki jest wynik dzielenia przez 0 ?



BigInt

ES 2020

- Operuje bezpiecznie na dużych liczbach, które nie mogą być reprezentowane przez typ *number*
- *Możesz używać tego typu gdy jest potrzebne by operować na liczbach $> 2^{53}$*
- Dodaj na końcu “n”: *456n*
- Nie mogą być używane w metodach obiektu Math ani nie mogą być mieszane z operacjami na zwykłych liczbach

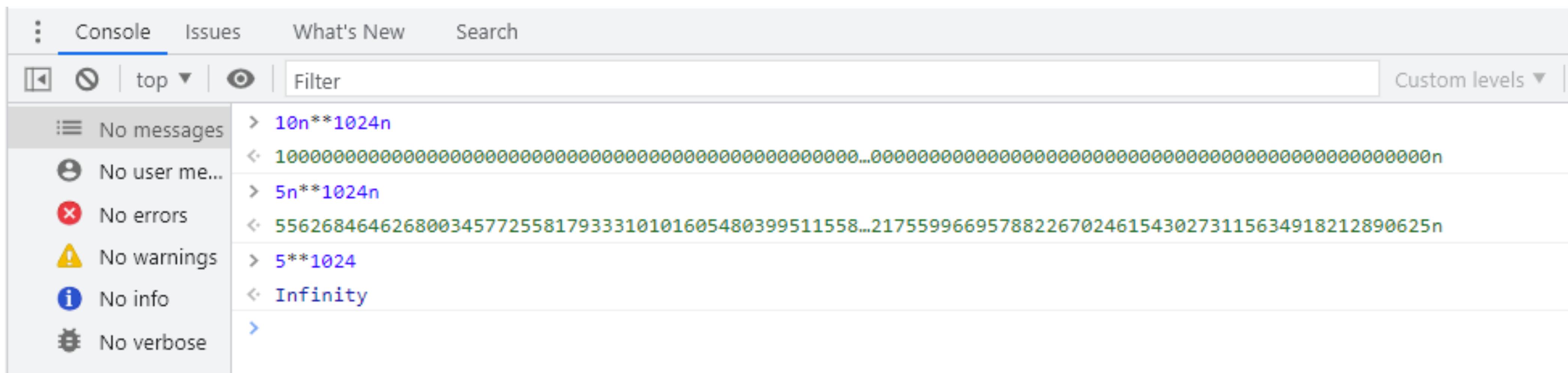
```
> 4n + 8
Uncaught TypeError: Cannot mix BigInt and other types, use explicit conversions
> 123123n * 890
Uncaught TypeError: Cannot mix BigInt and other types, use explicit conversions
> 123n - 90 + 6n
Uncaught TypeError: Cannot mix BigInt and other types, use explicit conversions
> -
```

bigint

inf. precision

BigInt

Przeglądarka vs Node



String

Reprezentacja tekstu

- Tworzenie tekstu:

```
const txt1 = "Hello Guys !";  
const txt2 = 'Hello Guys !';  
const txt3 = `Hello Guys !`;
```

- Dostęp:

```
const txt = "AA BB";  
const firstChar = txt[0];  
const secondChar = txt.charAt(1);
```

String

Cechy

- brak oddzielnego typu danych dla znaku (zobacz *char* w C++)
- immutable (nie można zmienić)
- każda litera to 16bit unsigned integer (UTF16)
 - 你好,
- łączenie stringów
 - “Hello ” + “Guys” = “Hello Guys”



Zadanie

- połącz różne teksty
- dodaj liczbę do stringa, przeanalizuj zachowanie
- tym razem odejmij oraz pomnóż liczbę przez tekst
- sprawdź działanie *typeof*



null oraz undefined

Typ null ma jedną wartość: **null**

Oznacza “wartość nieznana” lub “pusta”

Typ undefined ma jedną wartość: **undefined**

Oznacza “wartość nie jest przypisana”

Zmienna została zdeklarowana ale nie została zainicjalizowana

null

pusta
wartość

undefined

not initialized

13:23:16.856 » `typeof null`

13:23:16.934 ← "object"

»

13:43:43.377 » `let test;`

13:43:43.458 ← "undefined"

13:43:47.576 » `typeof test`

13:43:47.633 ← "undefined"

»

Boolean

- Reprezentuje wartość *true* albo *false*

```
> 5 > 90
```

```
< false
```

```
> 10 > 5
```

```
< true
```

```
> 2 === 2
```

```
< true
```

```
> typeof true
```

```
< 'boolean'
```

```
const condition = false;  
if (condition) {  
  console.log("Works");  
}
```

boolean

binarna
wartość

Object

Przykładowa instancja

Zaprojektowana by przechowywać grupę danych (proste lub obiekty)

```
const obj = {  
    type: "DS4",  
    company: "Citroen",  
    segment: "B",  
    year: 2012,  
    engine: 1.6,  
    sellerInfo: {  
        name: "Joseph",  
        surName: "Smith",  
        age: 40,  
        phone: "123123123",  
    },  
};
```

object

struktura
złożona

Object

Dostęp do pól

```
const car1 = {  
    type: "DS4",  
    company: "Citroen",  
    segment: "B",  
    year: 2012,  
    engine: 1.6,  
    sellerInfo: {  
        name: "Joseph",  
        surName: "Smith",  
        age: 40,  
        phone: {  
            mobile: "123123123",  
            home: "321321321",  
        },  
    },  
};
```

```
console.log("Car year", car1.year);  
console.log("Seller name", car1.sellerInfo.name);  
console.log("Seller phone", car1.sellerInfo.phone.mobile);  
console.log("Seller phone", car1.sellerInfo.phone.home);
```

Zadanie

Dostęp do pól

Wyświetl w konsoli informacje na temat silnika i segmentu.

Które z poniższych odniesień do pola *mobile* są prawidłowe ?

- obj.mobile
- obj.phone.mobile
- obj.engine.sellerInfo.phone.mobile
- obj.sellerInfo.phone.mobile

```
const obj = {  
    type: "DS4",  
    company: "Citroen",  
    segment: "B",  
    year: 2012,  
    engine: 1.6,  
    sellerInfo: {  
        name: "Joseph",  
        surName: "Smith",  
        age: 40,  
        phone: {  
            mobile: "123123123",  
            home: "321321321",  
        },  
    },  
};
```

Zadanie

Zbadaj działanie operatora `typeof`

`typeof true`

`typeof 56.34`

`typeof "sth"`

`typeof undefined`

`typeof NaN`

`typeof Infinity`

`typeof {}`

`typeof 456n`

`typeof null`

Zadanie

Zbadaj działanie operatora `typeof`

boolean

number

string

undefined

number

number

object

bigint

object

Dziękuję

