[Lesson 2]

Roi Yehoshua 2018

[What we learnt last time?]

- What is JavaScript and where is it used
- How to add scripts to your HTML page
- How to perform basic interactions with the user in Browser
- JavaScript basics

[Our targets for today]

- JavaScript basic data types
- How primitive types in Javascript work when combined together and with each other
- Type casting when working with different types

[Data Types]

→ A variable in JavaScript can contain any data

→ A variable can at one moment be a string and later receive a numeric value:

```
let foo = 42;
foo = 'bar';
foo = true;
```

→ Programming languages that allow such things are called “**dynamically typed**”

→ Meaning that there are data types, but variables are not bound to any of them

```
//foo is now a number
//foo is now a string
//foo is now a boolean
```

[Data Types]

- → There are seven basic data types in JavaScript:
  - → **number** for numbers of any kind: integer or floating-point
  - → **string** for strings
    - → A string may have one or more characters, there’s no separate single-character type
  - → **boolean** for true/false
  - → **null** for unknown values
    - → a standalone type that has a single value null
  - → **undefined** for unassigned values
    - → a standalone type that has a single value undefined
  - → **object** for more complex data structures
  - → **symbol** for unique identifiers

[DAN.IT] EDUCATION

Numbers

→ The **number** type serves both for integer and floating point numbers

→ Numbers are stored in memory as double precision 64-bit floating point numbers

→ Besides regular numbers, there are three special symbols which also belong to the number type: Infinity, -Infinity and NaN

→ Infinity represents the mathematical Infinity ∞

```
alert(1 / 0); //   Infinity
alert(Infinity); //    Infinity
```

→ NaN (Not a Number) represents a computational error. It is a result of an incorrect or an undefined mathematical operation, for instance:

```
alert("hello" * 2); // NaN, such division is erroneous
```

→ NaN is sticky. Any further operation on NaN would give NaN:

```
alert("hello" *2 + 5); // NaN
```

Strings

→ A string in JavaScript must be quoted
→ There are 3 types of quotes:
  → Double quotes: "Hello“
  → Single quotes: 'Hello’
  → Backticks: `Hello`

```
let str = "Hello";
let str2 = 'Single quotes are ok too'; let phrase =
`can embed ${str}`;
```

→ There’s no difference between double and single quotes in JavaScript
→ Backticks are “extended functionality” quotes. They allow us to embed variables and expressions into a string by wrapping them in ${…}, for example:

```
let name = "John";
// embed a variable
alert(`Hello, ${name}!`); // Hello, John!
// embed an expression
alert(`the result is ${1 + 2}`); // the result is 3
```

Boolean

→ The boolean type has only two values: true and false
→ This type is commonly used to store yes/no values: true means “yes, correct”, and false means “no, incorrect”
→ For example:

```
let nameFieldChecked = true; // yes, name field is checked let
ageFieldChecked = false; // no, age field is not checked
```

→ Boolean values also come as a result of comparisons:

```
let isGreater = 4 > 1;
alert(isGreater); // true (the comparison result is "yes")
```

Null

→ null forms a separate type of its own, which contains only the null value:

```
let age = null;
```

→ null expresses a lack of identification, indicating that a variable points to no object

  → null is often found in a place where an object can be expected but no object is relevant

→ The code above states that age is known to exist now but it has no type or value

Undefined

→ The special value undefined also makes a type of its own, just like null

→ The meaning of undefined is “value is not assigned

→ If a variable is declared, but not assigned, then its value is exactly undefined:

```
 let x;
alert(x); // shows "undefined"
```

→ Technically, it is possible to assign undefined to any variable:

```
let a = 123;
a = undefined;
alert(a); // "undefined"
```

→ But it’s not recommended to do that

→ Normally, we use null to write an “empty” or an “unknown” value into the variable, and undefined is only used for checks, to see if the variable is assigned

The typeof Operator

→ The typeof operator returns the type of the argument
→ It's useful when we want to process values of different types differently, or just want to make a quick check
→ It supports two forms of syntax:
  → As an operator: typeof x
  → Function style: typeof(x)
→ The call to typeof x returns a string with the type name:

```
typeof 0 // "number"
typeof "foo" // "string"
typeof true // "boolean"
typeof null // "object" null is recognized erroneously by
typeof as an object (historical reasons)
typeof undefined // "undefined"
typeof Math // "object"
typeof Symbol("id") // "symbol"
typeof alert // "function" functions belong to the object
type, but typeof treats them differently
```

Type Conversions

→ Most of the time, data types are converted automatically as needed during script execution

→ For example, alert automatically converts any value to a string to show it

→ Mathematical operations convert values to numbers

→ There are also cases when we need to explicitly convert a value to put things right

→ There are three most widely used type conversions: to string, to number and to boolean

Conversion To String

- → String conversion happens when we need the string form of a value
- → For example, alert(value) does it to show the value
- → We can also call String(value) function for that:

```
let value = true;
alert(typeof value); // boolean

value = String(value); // now value is a string "true"
alert(typeof value); // string
```

Conversion To Number

→ Numeric conversion happens in mathematical functions and expressions automatically

→ For example, when multiplication * is applied to non-numbers:

```
alert('3' * 2); // 6
```

→ Addition (+) is exceptional: if one of the added values is a string, then the other one is also converted to a string and then it concatenates (joins) them:

```
alert(1 + '2'); // '12' (string to the right)
alert('1' + 2); // '12' (string to the left)
```

→ Explicit conversion is usually required when we read a value from a string-based source like a prompt or a text field, but we expect a number to be entered

→ We can use a Number(value) function to explicitly convert a value:

```
let  str = '123';
let  num = Number(str); // becomes a number 123
alert(typeof num); // number
```

[Conversion To Number]

→ If the string is not a valid number, the result of such conversion is NaN, for instance:

```
let age = Number('hello');
alert(age); // NaN, conversion failed
```

Value	Becomes…
undefined	NaN
null	0
true and false	1 and 0
string	Whitespaces from the start and the end are removed. Then, if the remaining string is empty, the result is 0. Otherwise, the number is “read” from the string. An error gives NaN.

Conversion To Boolean

→ Boolean conversion is the simplest one

→ It happens in logical operations, but also can be performed manually with the call of **Boolean(value)**

→ The conversion rule:

  → “Empty” values, like 0, an empty string, null, undefined and NaN become false

  → Other values become true

```
alert(Boolean(1)); // true
alert(Boolean(0)); // false

alert(Boolean("hello")); // true
alert(Boolean("")); // false
```

[Control questions]

1. How many data types there are in Javascript?
2. Name all JavaScript basic types
3. What type shall we use to store a text?
4. What type do we use for calculations?
5. What is the difference between undefined and null?
6. How can we cast one type to another?

[Materials]

Core materials:
https://learn.javascript.ru/types-intro
https://developer.mozilla.org/ru/docs/Web/JavaScript/Guide/Grammar_and_Types
https://learn.javascript.ru/types-conversion
https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Global_Objects/Object/toString
https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Global_Objects/Number/toString
https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Global_Objects/parseInt
https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Global_Objects/parseFloat

Additional materials:
https://developer.mozilla.org/ru/docs/Web/JavaScript/Data_structures
https://dorey.github.io/JavaScript-Equality-Table/
https://learn.javascript.ru/operators

Video materials:
https://youtu.be/5JEkiHHUOFs
https://youtu.be/-GWOP5JdPpo
https://youtu.be/jd7J9NUBJWQ

[DAN.IT] EDUCATION