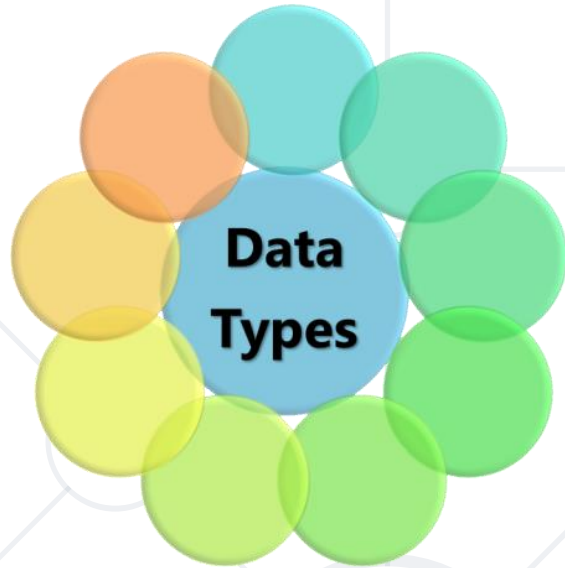


Data Types and Variables

Types of Operators



SoftUni Team
Technical Trainers



SoftUni



Software University

<https://softuni.bg>

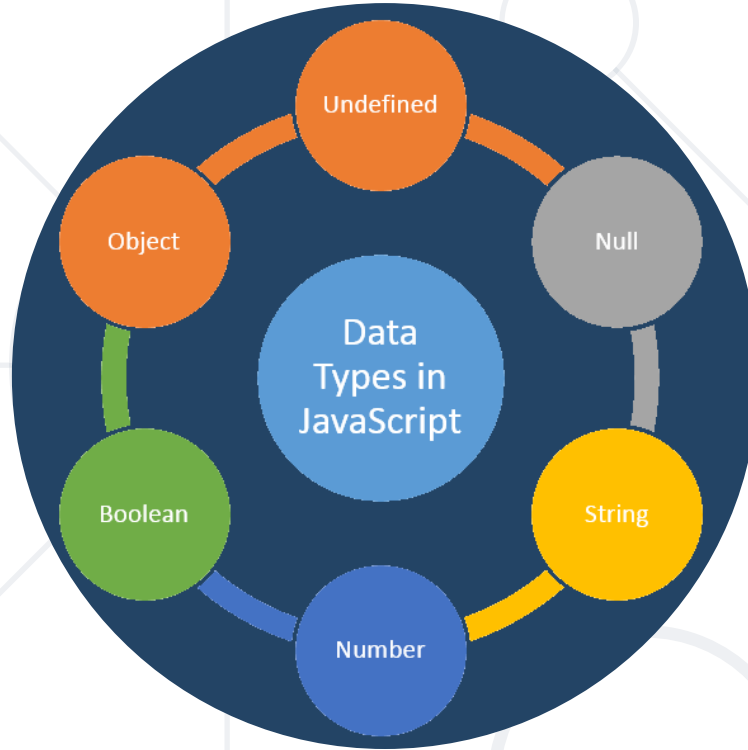
Table of Contents

1. What is a data type?
2. Let vs. Var
3. Strings
4. Numbers
5. Booleans
6. Typeof operator
7. Undefined and Null



sli.do

#fund-js

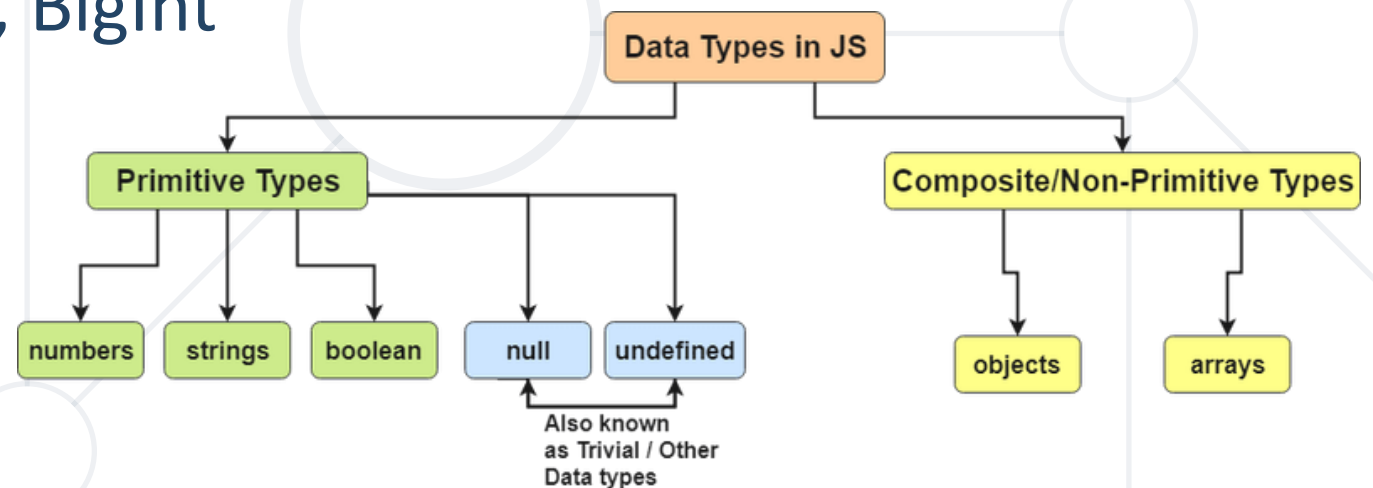


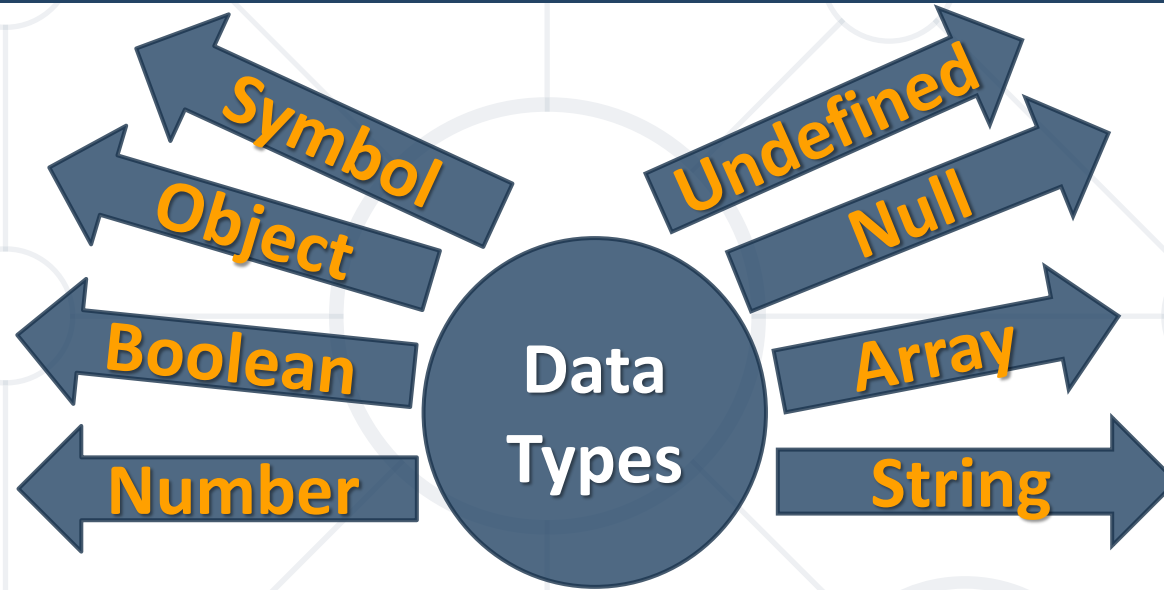
What is Data Type?

Definition and Examples

What is a Data Type?


- Data type **indicates** characteristics of data and tells the compiler whether the value is numeric, alphabetic etc.
- After **ECMAScript** 2015 there are **seven primitive** data types:
 - Seven **primitive**: Boolean, Null, Undefined, Number, String, Symbol, BigInt
 - and **Objects**





```
let number = 10;           // Number
let name = 'George';       // String
let array = [1, 2, 3];      // Array
let isTrue = true;         // Boolean
let person = {name: 'George', age: 25}; // Object
let empty = null;          // Null
let unknown = undefined;   // Undefined
```

Dynamic Typing

- 
- Variables in JavaScript are **not** directly **associated** with any particular **value type**
 - Any variable **can** be assigned (and **re-assigned**) values of all types

```
let foo = 42;      // foo is now a number  
foo = 'bar';      // foo is now a string  
foo = true;       // foo is now a boolean
```

- **NOTE: The use of dynamic typing is considered a bad practice!**



Let vs. Var

Local vs. Global

- **var** – variables use **function scope** – when declared inside a block **{ }** can be accessed from outside the block

```
{  
  var x = 2;  
}  
console.log(x); // 2
```

- **let** – variables use **block scope** – when declared inside a block **{ }** can **NOT** be accessed from outside the block

```
{  
  let x = 2;  
}  
console.log(x); // Error
```



- The scope of a variable is the **region** of the program in which it is defined
 - **Global Scope** – **Global** variables can be accessed from anywhere in a JavaScript function

```
var carName = "Volvo";  
// Code here can use carName  
function myFunction() {  
    // Code here can also use carName  
}
```

- **Function Scope – Local** variables can only be accessed from inside the function where they are declared

```
function myFunction() {  
    var carName = "Volvo";  
    // Only here code CAN use carName  
}
```

- **Block Scope** - Variables declared inside a block **{ }** can **not** be accessed from outside the block

```
{  
    let x = 2;  
} // x can NOT be used here
```

Naming Variables

- Variable names are **case sensitive**
- Variable names must begin with a **letter** or **underscore** (`_`) character



```
firstName, report, config, fontSize, maxSpeed
```

- Variable names **can't** be one of JavaScript's reserved words like: **break**, **const**, **interface**, **typeof**, **true** etc.

```
foo, bar, p, p1, LastName, last_name, LAST_NAME
```

A background network diagram consisting of a grid of light gray lines intersecting at various points. At these intersections, there are several circles of different sizes, some solid light gray and some hollow, creating a web-like structure. The central focus is a large, solid dark blue circle.

typeof

typeof Operator

Checking for a Type

Definition and Examples

- Used to find the **type** of a **variable**
- **Returns** the **type** of a variable or an expression:

```
console.log(typeof "")           // Returns "string"  
console.log(typeof "John")      // Returns "string"  
console.log(typeof "John Doe")  // Returns "string"  
console.log(typeof 0)           // Returns "number"
```

```
let number = 5;  
if (typeof(n) === 'number') {  
    console.log(number); // 5  
}
```





'ABC'

Strings

Sequence of Characters

What is a String?

- Used to represent **textual data**
- Each **symbol** occupies a **position** in the String
- The **first** element is at **index 0**, the next at index 1, and so on
- The **length** of a String is the number of elements in it

```
let name = 'George';  
console.log(name[0]); // 'G'
```

Accessing element at index



Strings Are Immutable

- Unlike in languages like C, JavaScript strings are **immutable**
- This means that once a string is created, it is **not** possible to **modify** it

```
let name = 'George';  
name[0] = 'P';  
console.log(name)  // 'George'
```



String Interpolation

- In JS we can use **template literals**. These are string literals that allow **embedded** expressions



```
let name = 'Rick';  
let age = 18;  
console.log(`${name} = ${age}`);  
// 'Rick = 18'
```

Place your **variables** after the **'\$'** sign

Problem: Concatenate Names

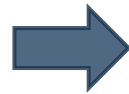
- Receive two **names** as **string parameters** and a **delimiter**
- Print the names **joined** by the delimiter

'John', 'Smith', '->'



John->Smith

'Jan', 'White', '<->'



Jan<->White

```
function solve(first, second, del) {  
    console.log(`${first}${del}${second}`);  
}
```

```
solve('John', 'Wick', '***')
```

Problem: Right Place

- You will receive **3 parameters** (string, symbol, string)
- Replace the underscore '_' in the **first word** with the **symbol**
- Compare both strings and print **"Matched"** or **"Not Matched"**

'Str_ng', 'I', 'Strong'



Not Matched

'Str_ng', 'i', 'String'



Matched

```
function solve(str, symbol, result) {  
  let res = str.replace('_', symbol);  
  let output = res ===  
    result ? "Matched" : "Not Matched";  
  console.log(output);  
}
```

`solve('Str_ng', 'I', 'Strong')`




123

Numbers

Integer, Float, Double – All in One

What is a Number?

- There is **no specific** type for integers and floating-point numbers
- To represent floating-point numbers, the number type has three symbolic values:
+Infinity, **-Infinity**, and **NaN** (not-a-number)



```
let num1 = 1;
let num2 = 1.5;
let num3 = 'p';
console.log(num1 + num2) // 2.5
console.log(num1 + num3) // '1p'
console.log(Number(num3)) // NaN
```

Concatenation

Trying to parse a string

Problem: Integer or Float

- You will receive **3 numbers**
- Find their **sum** and print "**{Sum} - {Integer or Float}**"

9, 100, 1.1

➔ 110.1 - **Float**

100, 200, 303

➔ 603 - **Integer**

122.3, 212.3, 5

➔ 339.6 - **Float**

```
function solve(num1, num2, num3) {  
  let sum = num1 + num2 + num3;  
  let output = sum % 1 === 0  
    ? sum + ' - Integer'  
    : sum + ' - Float';  
  console.log(output);  
}
```

`solve(112.3, 212.3, 5)`



true
false

Booleans

Conditions, Truthy and Falsy values

What is a Boolean?

- **Boolean** represents a logical entity and can have two values: **true** and **false**
- You can use the **Boolean()** function to find out if an expression (or a variable) is true:

```
Boolean(10 > 9)    // Returns true
```

- Or even easier:

```
(10 > 9)           // Also returns true  
10 > 9             // Also returns true
```



Comparisons and Conditions

Operator	Description	Example
<code>==</code>	equal to (no type)	<code>if (day == 'Monday')</code>
<code>></code>	greater than	<code>if (salary > 9000)</code>
<code><</code>	less than	<code>if (age < 18)</code>
<code>===</code>	equal to (with type)	<code>if (5 === 5)</code>
<code>>=</code>	greater than or equal (no type)	<code>if (6 >= 6)</code>
<code>!==</code>	not equal (with type)	<code>if (5 !== '5')</code>
<code>!=</code>	not equal (no type)	<code>if (5 != 5)</code>

- Everything **with** a "value" is **true**

```
let number = 1;  
if (number) {  
  console.log(number) // 1  
}
```

true

- Everything **without** a "value" is **false**

```
let number;  
if (number) {  
  console.log(number)  
} else {  
  console.log('false') // false  
}
```

false



Booleans Examples (2)

```
let x = 0;  
Boolean(x);           // false  
let x = -0;  
Boolean(x);           // false  
let x = '';  
Boolean(x);           // false  
let x = false;  
Boolean(x);           // false  
let x = null;  
Boolean(x);           // false  
let x = 10 / 'p';  
Boolean(x);           // false
```



Problem: Amazing Numbers

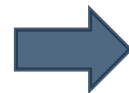
- You will receive **a number**, check if it is **amazing**
- An amazing is a number, whose **sum** of digits includes **9**
- Print it in format "**{number} Amazing? {True or False}**"

1233



1233 Amazing? **True**

999



999 Amazing? **False**

Solution: Amazing Numbers

```
function solve(num) {  
  num = num.toString();  
  let sum = 0;  
  for(let i = 0; i < num.length; i++)  
    sum += Number(num[i]);  
  let result = sum.toString().includes('9');  
  console.log(result ? `${num} Amazing? True`  
    : `${num} Amazing? False`);  
}
```



**Undefined
Null**

Undefined and Null

Non-Existent and Empty

Undefined

- A variable without a value, has the value **undefined**. The **typeof** is also **undefined**

```
let car; // Value is undefined, type is undefined
```

- A variable can be emptied, by setting the value to **undefined**. The type will also be **undefined**

```
let car = undefined;  
// Value is undefined, type is undefined
```



Null

- **Null** is "**nothing**". It is supposed to be something that doesn't exist
- The **typeof** null is an **object**



```
let person = {  
  firstName: "John",  
  lastName: "Doe",  
  age: 50  
};  
person = null;  
console.log(person);           // null  
console.log(typeof(person));  // object
```

Null and Undefined

- **Null** is an assigned value. It means nothing
- **Undefined** typically means a variable has been declared but not defined yet
- **Null** and **Undefined** are **falsy** values
- **Undefined** and **Null** are equal in value but different in type:

```
null !== undefined    // true  
null == undefined     // true
```

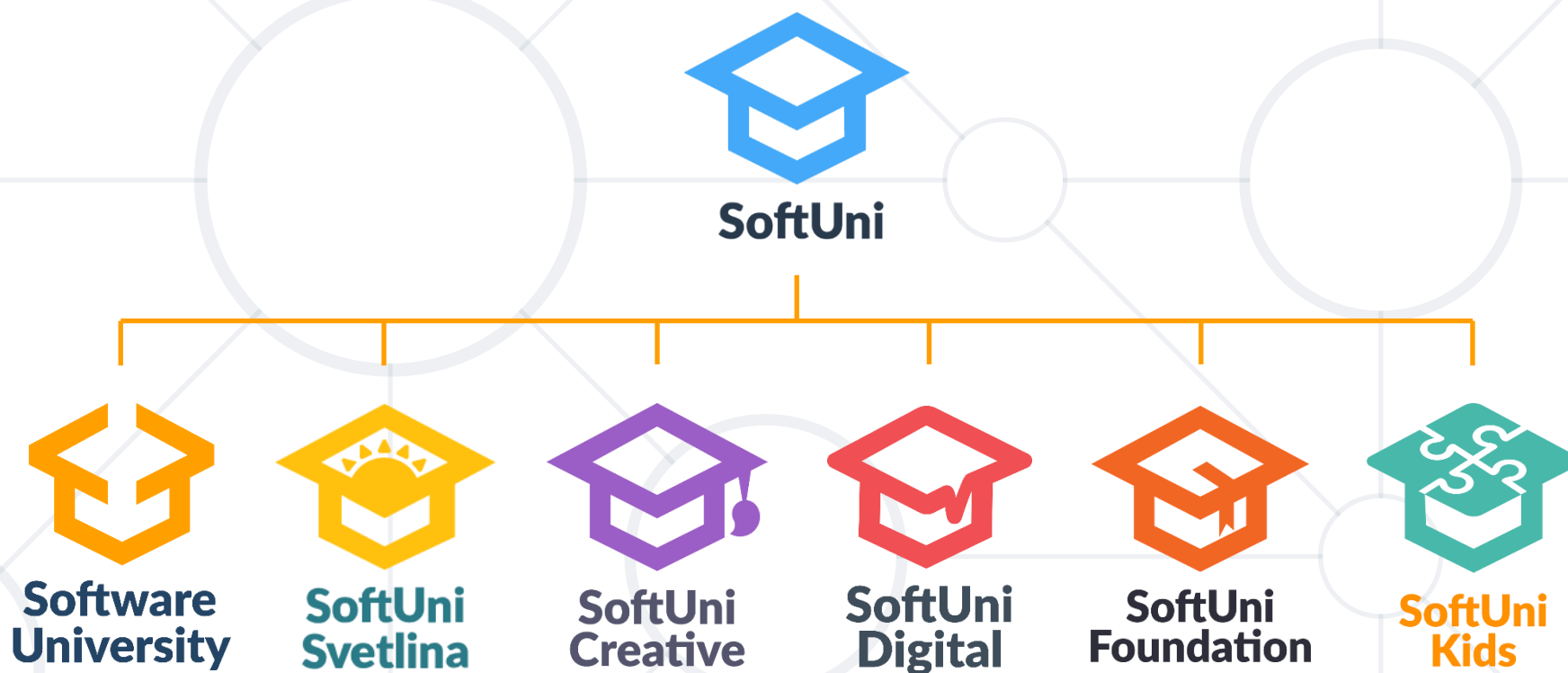


Live Exercises

- There are **7 data types** in JavaScript: **Number**, **String**, **Symbol**, **Null**, **Undefined**, **Object**, **Boolean**
- **let** has block scope, **var** has function scope
- With **typeof** we can receive the type of a variable
- **Null** is "nothing", **undefined** exists, but is empty



Questions?



SoftUni Diamond Partners

**SUPER
HOSTING
.BG**

INDEAVR
Serving the high achievers

 **SmartIT**


SOFTWARE

zühlke
empowering ideas

 **INFRAGISTICS®**



**Coca-Cola HBC
Bulgaria**



Postbank

Решения за твоето утре



 **DRAFT
KINGS**



**SOFTWARE
GROUP**



VIRTUAL RACING SCHOOL



codexio

- Software University – High-Quality Education, Profession and Job for Software Developers
 - softuni.bg
 - Software University Foundation
 - softuni.foundation
- Software University @ Facebook
 - facebook.com/SoftwareUniversity
- Software University Forums
 - forum.softuni.bg



- This course (slides, examples, demos, exercises, homework, documents, videos and other assets) is **copyrighted content**
- Unauthorized copy, reproduction or use is illegal
- © SoftUni – <https://about.softuni.bg/>
- © Software University – <https://softuni.bg>

