

💡 thinkdev #2

Values and types

Explore featured courses



The University of Glasgow



The Museum as a Site and Source for Learning

★★★★☆ 4.6 (75 reviews)

Find out more



Sentinel9 & FutureLearn



NEW

Fundamentals of Business Strategy

Find out more



University of Groningen,
University of Cambridge &
University Medical Center
Groningen (UMCG)



Young People and Mental Health

★★★★☆ 4.7 (649 reviews)

Find out more



Part of an ExpertTrack

Coventry University



International Logistics: A Beginner's Guide to Logistics Management

Find out more

A screenshot of the featured courses on *FutureLearn*.

Text

Explore featured courses



The University of Glasgow ☆

The Museum as a Site and Source for Learning

★★★★☆ 4.6 (75 reviews)

[Find out more](#)



Sentinel|9 & FutureLearn ☆

NEW

Fundamentals of Business Strategy

[Find out more](#)



University of Groningen, University of Cambridge & University Medical Center Groningen (UMCG) ☆

Young People and Mental Health

★★★★☆ 4.7 (649 reviews)

[Find out more](#)



Coventry University ☆

International Logistics: A Beginner's Guide to Logistics Management

[Find out more](#)

Numbers

Explore featured courses



The University of Glasgow ☆
The Museum as a Site and Source for Learning
★★★★☆ 4.6 (75 reviews)

[Find out more](#)



Sentinel9 & FutureLearn ☆
NEW
Fundamentals of Business Strategy

[Find out more](#)



University of Groningen, University of Cambridge & University Medical Center Groningen (UMCG) ☆
Young People and Mental Health
★★★★☆ 4.7 (649 reviews)

[Find out more](#)



Coventry University ☆
International Logistics: A Beginner's Guide to Logistics Management

[Find out more](#)

Alternatives

Explore featured courses



The University of Glasgow ☆

The Museum as a Site and Source for Learning

★★★★☆ 4.6 (75 reviews)

[Find out more](#)



Sentinel9 & FutureLearn ☆

NEW
Fundamentals of Business Strategy

[Find out more](#)



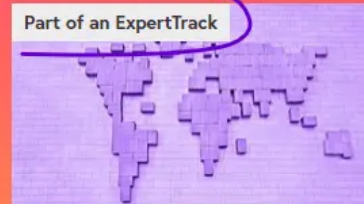
University of Groningen, ☆

University of Cambridge & University Medical Center Groningen (UMCG)

Young People and Mental Health

★★★★☆ 4.7 (649 reviews)

[Find out more](#)



Part of an ExpertTrack

Coventry University ☆

International Logistics: A Beginner's Guide to Logistics Management

[Find out more](#)

How do we represent these values?

- We use *strings* for text:
 - "Explore featured courses"
 - 'Find out more'

- We use *strings* for text:
 - "Explore featured courses"
 - 'Find out more'
- We use *booleans* to choose between alternatives
 - true, false

What about numbers?

What about numbers?

- Many languages differentiate between types of numbers.

What about numbers?

- Many languages differentiate between types of numbers.
- *Integer* types (*int*) for 75, -2, ...

What about numbers?

- Many languages differentiate between types of numbers.
- *Integer* types (*int*) for 75, -2, ...
- *Floating point* types (*float*) for 4.6, -0.789, ...

But in JavaScript...

But in JavaScript...

... a number is just a *number*.

typeof a value

Use the typeof keyword to get the type of a value:

```
typeof "Hi"           // "string"  
typeof 12.34          // "number"  
typeof 3_000_000      // "number"  
typeof false         // "boolean"
```

Comments

- Use comments to explain pieces of your code.
- The language ignores them.

Expressions

Expressions

Things that have value.

- The simplest expressions are literals:
 - 1, "Hi", true.

- The simplest expressions are literals:
 - 1, "Hi", true.
- But they're not so useful alone.

You can use operators to build complex expressions:

```
1 - 2;           // -1
50 * 70 / 67 + 9; // 61.2388...
typeof true;     // "boolean"
```

Wrapping an expression in brackets doesn't change it's value:

```
(1 - 2);           // -1  
(50 * 70 / 67 + 9); // 61.2388...  
(typeof true);   // "boolean"
```

Operator precedence rules apply, even to non-arithmetic operators:

```
(50 * 70 / 67 + 9); // 61.2388...  
50 * 70 / (67 + 9); // 46.0526...  
typeof (2 - 1);    // "number"
```


You can use an expression where a value is expected:

```
typeof (50 * 70 / 67 + 9)  
console.log(typeof true)
```

**What if we wanted to store
the value of an expression?**

Variables

How to name variables

How to name variables

- First character must be a letter, underscore `_`, or dollar sign `$`.
 - E.g, `x`, `$`, `_`

How to name variables

- First character must be a letter, underscore `_`, or dollar sign `$`.
 - E.g, `x`, `$`, `_`
- Following characters may include numbers
 - Valid: `y2`, `first_name`, `_LAST_NAME_`, `$10`
 - Invalid: `2a`, `middle name`

How to name variables

- First character must be a letter, underscore `_`, or dollar sign `$`.
 - E.g, `x`, `$`, `_`
- Following characters may include numbers
 - Valid: `y2`, `first_name`, `_LAST_NAME_`, `$10`
 - Invalid: `2a`, `middle name`
- Names are case-sensitive
 - `message`, `Message`, `MESSAGE` are different variables.

The JavaScript convention is
camelCase 🐪

Variables that vary



- A `const` variable is *constant*; it always refers to the same value.

Variables that vary

- A `const` variable is *constant*; it always refers to the same value.
- That's usually fine, but sometimes we'd like to reassign a variable to a different value.

Consider an online shopping cart:

1 SHOPPING CART > 2 CHECKOUT DETAILS

PRODUCT	PRICE	QUANTITY	SUBTOTAL	
 	For The Good Of The Nation: Essays and Perspectives - by Sanusi Lamido Sanusi	₦8,500	- 1 +	₦8,500

← CONTINUE SHOPPING

Screenshot of a cart item on *Tarbiyah Books Plus*.

Use the `let` keyword instead:

```
let quantity = 1  
  
quantity = quantity + 1  
  
console.log(quantity) // 2
```


Addition assignment operator

```
let quantity = 1
```

```
quantity += 1
```

```
console.log(quantity) // 2
```

Increment operator

```
let quantity = 1
```

```
quantity++
```

```
console.log(quantity) // 2
```

You can initialize a `let` variable with a value after declaring it:

```
let quantity;  
  
// initialize after declaring  
quantity = 1  
  
quantity++  
  
console.log(quantity) // 2
```

Its value will be undefined until you initialize it:

```
let quantity;  
  
console.log(quantity) // undefined  
  
// initialize after declaring  
quantity = 1  
  
quantity++  
  
console.log(quantity) // 2
```

Absence of value

- Special values: undefined and null.
- null is often used for an intentionally absent value.

Operations on strings

Joining strings

Also known as *concatenation*:

```
const firstName = "Mubaraq"  
const lastName = "Wahab"  
  
const fullName = firstName + lastName  
// "MubaraqWahab"
```


Interpolation

You can use special strings called *template literals* to interpolate:

```
const firstName = "Mubaraq"  
const lastName = "Wahab"  
  
const fullName = `${firstName} ${lastName}`  
// "Mubaraq Wahab"
```

Get a character from a string

Use square brackets to specify an *index* (starting from zero):

```
//           0123456
const firstName = "Mubaraq"

const initial = firstName[0] // "M"
const second = firstName[1] // "u"
// and so on...
```

Get part of a string

Use the `slice` method:

```
//           0123456
const firstName = "Mubaraq"

const firstThreeLetters = firstName.slice(0, 3)
// "Mub"
const thirdToEnd = firstName.slice(2)
// "baraq"
```

Does a string include this?

Use the `includes` method to check if a string includes another:

```
const firstName = "Mubaraq"  
  
firstName.includes('ba')  
// true  
firstName.includes('ab')  
// false
```

How long is a string?

Use the `length` property to get the length of a string:

```
const firstName = "Mubaraq"  
  
firstName.length  
// 7
```

String to number

You need to convert a string to a number sometimes, such as when working with user input:

```
// Assume this is from user input
const input = "20"

// Careful here! Result is "203"
input + 3
```

Use the `Number` function to convert a string to a number:

```
// Assume this is from user input
const input = "20"

// Convert to number first!
const inputAsNumber = Number(input)

// Result is 23
inputAsNumber + 3
```

Or use the + operator:

```
// Assume this is from user input
const input = "20"

// An idiomatic way
const inputAsNumber = +input

// Result is 23
inputAsNumber + 3
```


Number to string

The opposite is possible too, using the `String` function:

```
const num = 20

// Result is "20"
const numAsString = String(num)
```

Or the toString method:

```
const num = 20

// Result is "20"
const numAsString = num.toString()
```

Or even concatenating with an empty string:

```
const num = 20

// Result is "20"
const numAsString = "" + num
```

UPPERCASE, lowercase

```
const firstName = "Mubaraq"
```

```
firstName.toUpperCase()
```

```
// "MUBARAQ"
```

```
firstName.toLowerCase()
```

```
// "mubaraq"
```

Statements

Statements

- A program is a sequence of statements.
- Statements are executed one after another.

```
const name = 'Mubaraq'  
const message = 'Hello ' + name  
typeof message
```

A variable declaration is a statement:

```
const name = 'Mubaraq'  
const message = 'Hello ' + name  
typeof message
```


An expression can act as a statement too:

```
const name = 'Mubaraq'  
const message = 'Hello ' + name  
typeof message
```

You can't use a statement as an expression:

```
// Error!  
const message = 'Hello ' + (const name = 'Mubaraq')  
typeof message
```

An assignment is an expression:

```
let name  
const message = 'Hello ' + (name = 'Mubaraq')  
typeof message
```