

💡 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



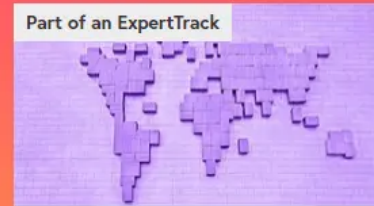
University of Groningen,
University of Cambridge &
University Medical Center
Groningen (UMCG)



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



Part of an ExpertTrack

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



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



Part of an ExpertTrack

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

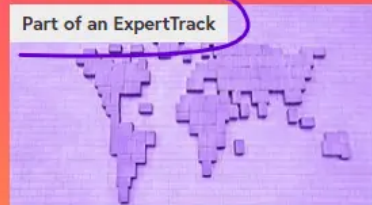


University of Groningen,
University of Cambridge &
University Medical Center
Groningen (UMCG) ☆

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?

How do we represent these values?

- We use *strings* for text:
 - "Most popular"
 - 'Installed'

How do we represent these values?

- We use *strings* for text:
 - "Most popular"
 - 'Installed'
- 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 20, -7, ...

What about numbers?

- Many languages differentiate between types of numbers.
- *Integer* types (*int*) for 20, -7, ...
- *Floating point* types (*float*) for 3.2, -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.

Comments

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

```
// Line comment
```

```
/* Block comment  
can span  
multiple lines. */
```

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

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

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

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

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


You can use an expression where a value is expected:

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

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

Variables

Variables

```
1 // Declare a variable  
2 const costPerItem = 3000
```

Variables

```
1 // Declare a variable
2 const costPerItem = 3000
3
4 // Use the variable
5 console.log(costPerItem * 10)
```

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

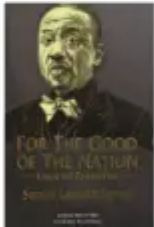
PRODUCT

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:

```
1 let quantity = 1
2
3 quantity = quantity + 1
4
5 console.log(quantity) // 2
```


Addition assignment operator

```
1 let quantity = 1
2
3 quantity += 1
4
5 console.log(quantity) // 2
```

Increment operator

```
1 let quantity = 1  
2  
3 quantity++  
4  
5 console.log(quantity) // 2
```

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

```
1 let quantity;  
2  
3 // initialize after declaring  
4 quantity = 1  
5  
6 quantity++  
7  
8 console.log(quantity) // 2
```

Its value will be undefined until you initialize it:

```
1 let quantity;  
2  
3 console.log(quantity) // undefined  
4  
5 // initialize after declaring  
6 quantity = 1  
7  
8 quantity++  
9  
10 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*:

```
1 const firstName = "Mubaraq"  
2 const lastName = "Wahab"  
3  
4 const fullName = firstName + lastName  
5 // "MubaraqWahab"
```


Interpolation

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

```
1 const firstName = "Mubaraq"  
2 const lastName = "Wahab"  
3  
4 const fullName = `${firstName} ${lastName}`  
5 // "Mubaraq Wahab"
```

Get a character from a string

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

```
1 //          0123456
2 const firstName = "Mubaraq"
3
4 const initial = firstName[0] // "M"
5 const second = firstName[1] // "u"
6 // and so on...
```

Get part of a string

Use the `slice` method:

```
1 //          0123456
2 const firstName = "Mubaraq"
3
4 const firstThreeLetters = firstName.slice(0, 3)
5 // "Mub"
6 const thirdToEnd = firstName.slice(2)
7 // "baraq"
```

Does a string include this?

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

```
1  const firstName = "Mubaraq"  
2  
3  firstName.includes('ba')  
4  // true  
5  firstName.includes('ab')  
6  // false
```

How long is a string?

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

```
1 const firstName = "Mubaraq"  
2  
3 firstName.length  
4 // 7
```

String to number

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

```
1 // Assume this is from user input
2 const input = "20"
3
4 // Careful here! Result is "203"
5 input + 3
```

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

```
1 // Assume this is from user input
2 const input = "20"
3
4 // Convert to number first!
5 const inputAsNumber = Number(input)
6
7 // Result is 23
8 inputAsNumber + 3
```

Or use the + operator:

```
1 // Assume this is from user input
2 const input = "20"
3
4 // An idiomatic way
5 const inputAsNumber = +input
6
7 // Result is 23
8 inputAsNumber + 3
```


Number to string

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

```
1 const num = 20
2
3 // Result is "20"
4 const numAsString = String(num)
```

Or the toString method:

```
1  const num = 20
2
3  // Result is "20"
4  const numAsString = num.toString()
```

Or even concatenating with an empty string:

```
1 const num = 20
2
3 // Result is "20"
4 const numAsString = "" + num
```

UPPERCASE, lowercase

```
1  const firstName = "Mubaraq"  
2  
3  firstName.toUpperCase()  
4  // "MUBARAQ"  
5  
6  firstName.toLowerCase()  
7  // "mubaraq"
```

Statements

Statements

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

```
1 const name = 'Mubaraq'  
2 const message = 'Hello ' + name  
3 typeof message
```

A variable declaration is a statement:

```
1 const name = 'Mubaraq'  
2 const message = 'Hello ' + name  
3 typeof message
```


An expression can act as a statement too:

```
1 const name = 'Mubaraq'  
2 const message = 'Hello ' + name  
3 typeof message
```

You can't use a statement as an expression:

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

An assignment is an expression:

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