



WEBDEVTEACHER.COM

# JavaScript Strings and More

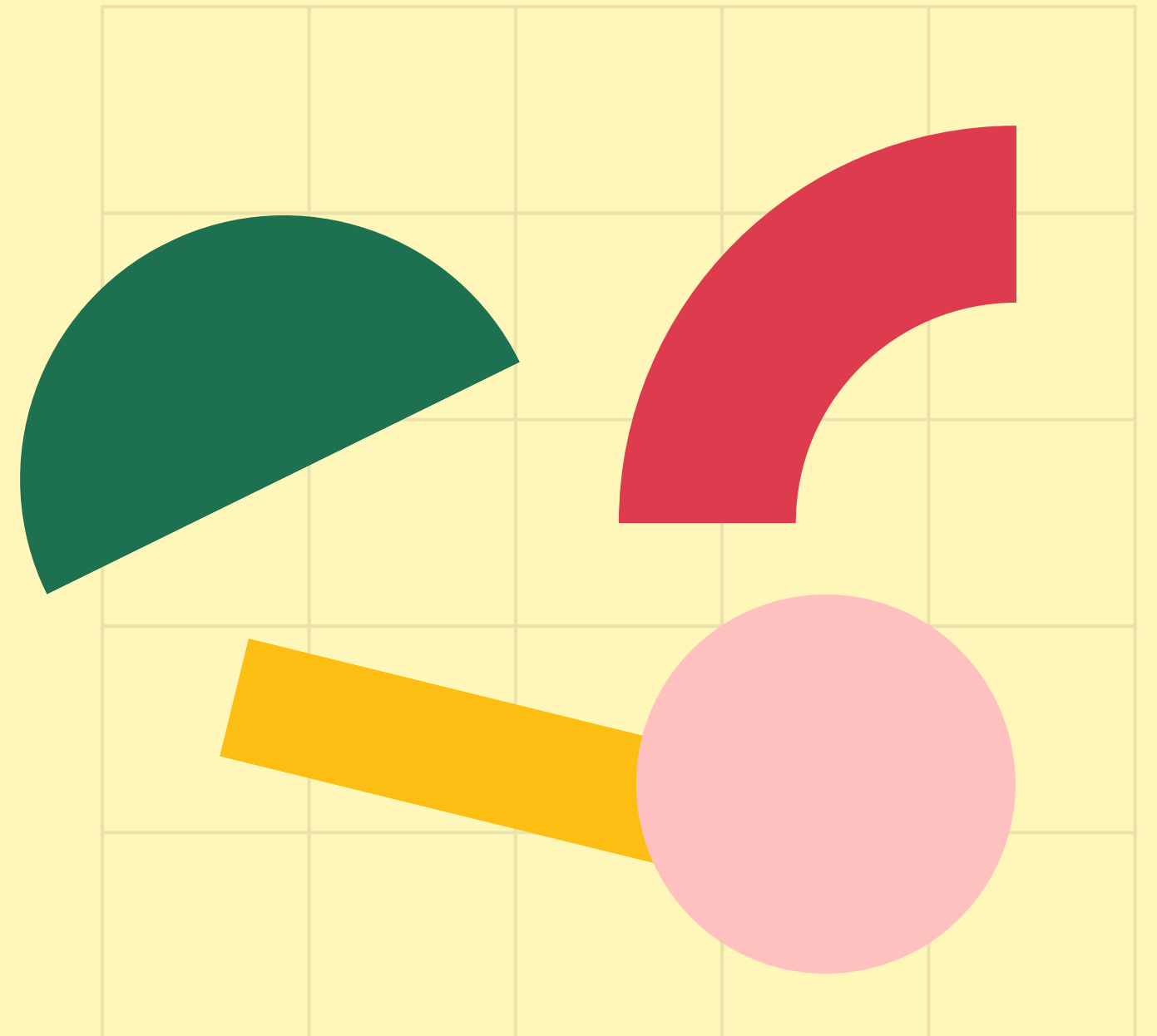
Course for Beginners



# Unit Goals

what we'll cover

- null & undefined
- Math object
- random numbers
- parseInt & parseFloat
- strings
  - methods
    - casing
    - trim
    - indexOf
    - slice
    - replace
  - escapes
  - template literals

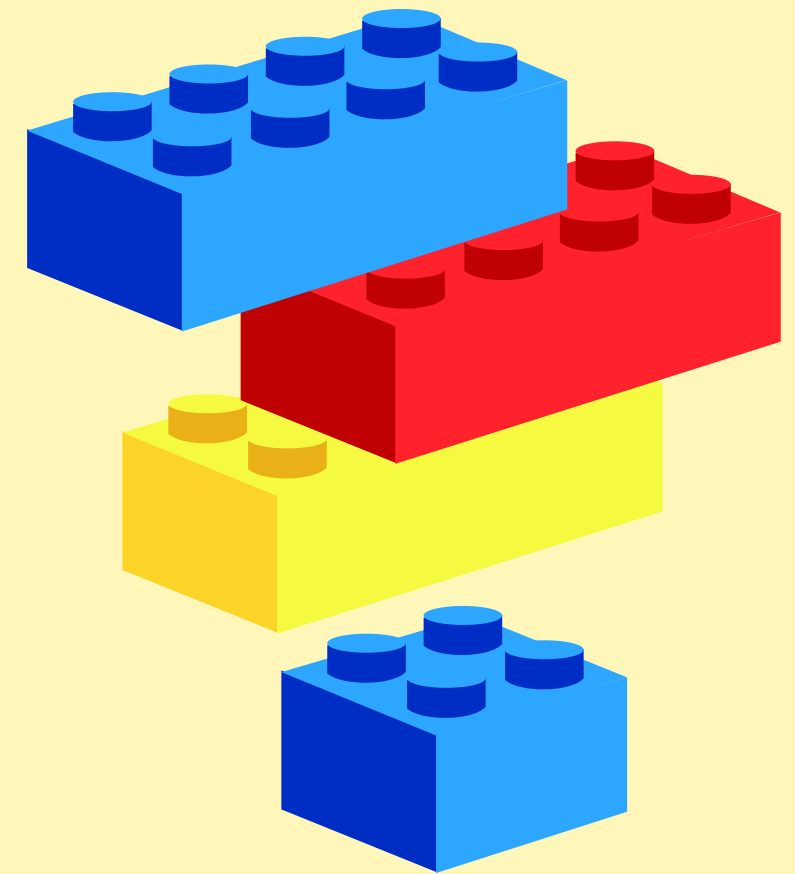


# PRIMITIVE TYPES



The basic building blocks\*:

- Number
- String
- Boolean
- Null
- Undefined



\*technically there are two more others: Symbol and BigInt

# STRINGS



"STRINGS OF CHARACTERS"

Strings are another primitive type in JavaScript. They represent text and must be wrapped in quotes.

abc

# STRINGS



"STRINGS OF CHARACTERS"



```
let firstName = "Ziggy";  
let msg = "Please do not feed the chimps!";  
let animal = 'Dumbo Octopus';  
let bad = "this is wrong";
```

double quotes work

so do single quotes

this **does not work**

It's fine to use either single or double quotes, just be consistent in your codebase.

# STRINGS ARE INDEXED



C

A

R

0

1

2

Each character has a corresponding index  
(a positional number)



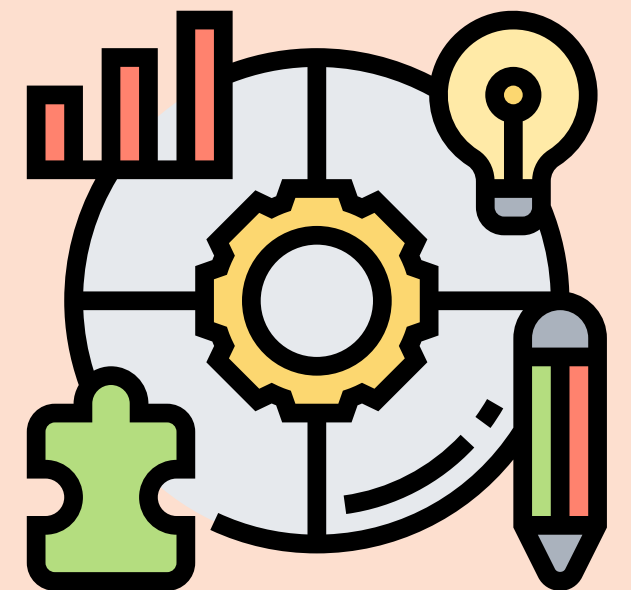
# STIRNG METHODS



METHODS ARE BUILT-IN ACTIONS WE CAN PERFORM WITH INDIVIDUAL STRINGS

They help us do things like:

- searching within a string
- replacing part of a string
- changing the casing of a string





**something.method()**



# CASING



```
let msg = 'I am king';  
let yellMsg = msg.toUpperCase(); // 'I AM KING'  
  
let angry = 'LeAvE mE aLoNe!';  
angry.toLowerCase(); // 'leave me alone!'  
  
//the value in angry is unchanged  
angry; // 'LeAvE mE aLoNe!'
```

# trim()



```
let greeting = '  leave me alone plz  ';  
greeting.trim() // 'leave me alone plz'
```

# `something.method(arg)`

Some methods accept arguments that modify their behavior. Think of them as inputs that we can pass in.

We pass these arguments inside of the parentheses.

# indexOf(elem)



`indexOf(searchElement)`

`indexOf(searchElement, fromIndex)`



```
let tvShow = 'catdog';  
  
tvShow.indexOf('cat'); // 0  
tvShow.indexOf('dog'); // 3  
tvShow.indexOf('z'); // -1 (not found)
```

# slice()



slice()

slice(start)

slice(start, end)



```
let str = 'supercalifragilisticexpialidocious'
```

```
str.slice(0,5); //'super'
```

```
str.slice(5); // 'califragilisticexpialidocious'
```



# replace

`replace(regex, newSubstr)`  
`replace(regex, replacerFunction)`  
`replace(substr, newSubstr)`  
`replace(substr, replacerFunction)`

```
1 let laughter = 'hehee ioh ioh ioh! hehee!';  
2 laughter.replace('hehee', 'haha'); //'haha ioh ioh ioh! hehee!'  
3 //it replaces only the first instance
```

# WHAT IS THE VALUE OF

age?



```
const age = "5" + "4";
```

# WHAT DOES THIS EVALUATE TO?



```
"pecan pie"[7]
```



# WHAT IS THE VALUE OF

## song?



```
let song = "london calling";  
song.toUpperCase();
```

# WHAT IS THE VALUE OF

## cleanedInput?



```
let userInput = "  TODD@gmail.com";  
let cleanedInput = userInput.trim().toLowerCase();
```

# WHAT IS THE VALUE OF

## index?

```
let park = 'Yellowstone';  
const index = park.indexOf('Stone');
```

# WHAT IS THE VALUE OF

## index?

```
let yell = 'GO AWAY!!!';  
let index = yell.indexOf('!');
```

# WHAT DOES THIS EVALUATE TO



```
'GARBAGE!'.slice(2).replace("B",'');
```

# STRING ESCAPES



`\n` - newline

`\'` - single quote

`\"` - double quote

`\\` - backslash

# TEMPLATE LITERALS



VERY USEFUL!

```
⬢ ⬢ ⬢  
`I counted ${3 + 4} sheep`; // "I counted 7 sheep"
```

TEMPLATE LITERALS ARE STRINGS THAT ALLOW EMBEDDED EXPRESSIONS, WHICH WILL BE EVALUATED AND THEN TURNED INTO A RESULTING STRING.

# WE USE BACK-TICKS\*



# NOT SINGLE QUOTES

``I am a template literal``

\*The back-tick key is usually above the tab key



# TEMPLATE LITERALS



```
let item = 'cucumbers';  
let price = 1.99;  
let quantity = 4;
```

```
`You bought ${quantity} ${item}, total price: ${price*quantity}`;  
// "You bought 4 cucumbers, total price: $7.96"
```

# NULL & UNDEFINED



- null
  - "intentional absence of any value"
  - must be assigned
- undefined
  - variables that do not have an assigned value are undefined

# NULL



```
1 // No one is logged in yet...
2 let loggedInUser = null; //value is explicitly nothing
3
4 // A user logs in...
5 loggedInUser = 'Alan Rickman';
```

# UNDEFINED



```
1 let pickles; //We didn't assign a value
2 pickles; //undefined,
3 pickles = 'are very gross'
4
5 //Undefined also comes up in other situations:
6 let food = 'tacos';
7 food[7]; //undefined
```



# MATH OBJECT

Contains properties and methods for mathematical constants and functions

```
Math.PI // 3.141592653589793
```

```
//Rounding a number:  
Math.round(4.9) //5
```

```
//Absolute value:  
Math.abs(-456) //456
```

```
//Raises 2 to the 5th power:  
Math.pow(2,5) //32
```

```
//Removes decimal:  
Math.floor(3.9999) //3
```



# RANDOM NUMBERS

`Math.random()` gives us a random decimal between 0 and 1 (non-inclusive)

```
Math.random();  
//0.14502435424141957  
Math.random();  
//0.8937425043112937  
Math.random();  
//0.9759952148727442
```



# RANDOM INTEGERS

Let's generate random  
numbers between 1 and 10

```
const step1 = Math.random();  
//0.5961104892810127  
const step2 = step1 * 10  
//5.961104892810127  
const step3 = Math.floor(step2);  
//5  
const step4 = step3 + 1;  
//6  
  
Math.floor(Math.random() * 10) + 1;
```



# parseInt & parseFloat

Use to parse strings into numbers, but watch out for NaN!

```
parseInt('24') //24
parseInt('24.987') //24
parseInt('28dayslater') //28

parseFloat('24.987') //24.987
parseFloat('7') //7
parseFloat('i ate 3 shramp') //NaN
```