



JS

*with*

P

I

N

K

{p r o g r a m m i n g}

# What is programming?

*“The **process** of **designing** and building a computer **program** for accomplishing a specific **computing task**”*

Computers are really good at **computing** and doing **repetitive** things, but we need humans to be **creative** and **tell** the computers what to do.

What does that word mean?

# What is a program?

*“A sequence of **instructions**, written to perform a specific task on a **computer**”*

The instructions has to be written in a language that the **computer** can understand, a **programming language**

Eg: **C#, Python, Java, JavaScript**

# Why learn JavaScript?

</> **JavaScript** has a unique position in internet programming because web browsers can only understand JavaScript

</> It's easy to get started and the results are **visual** and **rewarding**

</> Extremely popular with a **huge community** of developers who can support you






</> There's a great depth to **discover** when you mastered the basics

</> Versatile language: **Web apps, server applications, game development, iot-development** ...and much more!

# What is NodeJs?

The most common engine to run JavaScript as any programming language, without the complexities of a web browser and its document interface. **Runkit** is a convenient tool to put the code into a single document.



-  Code
-  Comments
-  Images
-  Data visualizations
-  Publishing of data endpoints

Which can then be easily embedded to a website or blog

## Fun facts

**A**

**It was developed in only 10 days**

Back in 1995

**B**

**It has very little to do with Java**

Yet the company behind Java keeps licensing the name

**C**

**Mozilla Foundation**

Go to place for JavaScript documentation and the Firefox browser

# Data types

## 1 | Number

The type that helps you do calculating, ie:

- 0, 1, 2, 3 : integers
- -1, -2 : signed integers
- 1.5, -9.0 : float numbers

## 2 | String

Text, a series of characters, are called strings.

- "Hi, my name is"
- "Building 47"

## 3 | Boolean

A data type which value can only be one of two states:

- true
- false

# Data types

## 4 | Object

When data is structured and put inside one variable, it gets this data-type

Example "City":

```
{  
  name: "Stockholm",  
  population: 2 391 990  
}
```

## 5 | Function

The code itself can be a data-type.

Example:

```
function inPercent(fraction) {  
  return fraction*100 + "%"  
}
```

## 6 | undefined

When the computer doesn't understand the data-type, it's called "undefined".

Example:

```
return;
```



# Variable

- A reserved piece of **memory** to hold **values**
- A variable name can be anything - the more descriptive, the better  
Eg. **myName** = 'Marnie' or **myAge** = 9
- Good to remember when naming variables:
  - The name should **start** with a **letter**
  - **Cannot** start with a number
  - **Alphanumeric** characters (A-z & 0-9) and **underscores** (\_)
- Camel case is preferred: pinkProgramming | PinkProgramming | pink\_programming

# What is a variable?

- How much memory is being reserved depends on what value you want to store
- The equal sign (=) is used to assign a value to a variable
  - `myNumber = 30` | variable to the left and value to the right
- The value of a variable can change
  - `myNumber = 40`

```
> console.log(myNumber)
```

```
40
```

**Exercise (1a-c):** Create four variables and assign values of different types to them

# How do we use variables?

```
> firstNumber = 1
```

```
> secondNumber = 2
```

```
> thirdNumber = 3
```

```
> console.log(firstNumber + secondNumber + thirdNumber)  
6
```

**Exercise (2):** Explore with arithmetic operators + - \* / !

# Arrays

- A **data structure** that can store a **collection** of items
- Defined using square brackets [item1, item2, item3, and so on...]
- Each item is separated by a **comma**
- **Index** (position) starts from 0,1,2,...,n (*the number of items minus one*)

**Exercise (example):** *Get the first, second and last value of the array of fruits*



# Arrays

## *Exercise (3a-b):*

*Create a new array with names of your friends.  
Repeat the previous steps.*

# Arrays

- **Count** number of items in the array: `array.length`
- Change the array, **adding** a new item first: `array = [newItem, ...array]`
- Change the array, **removing** the first item: `[item1, ...array] = array`
- Combine two arrays of names, using: `[...array, ...array2]`

These the basic operations.

## More can be found in the documentation

Most of them use methods, ie: `array.indexOf('value')`

**Exercise (4a-d):** *Try some of the basic operations implemented for arrays*

# For-loops

- Useful when you want to go through every item in for example an array

**Exercise:** Console log each name in your array of names

**Exercise:** Add a small change to each name in your array

[Want to know more about for-loops?](#)

# Objects

- A data structure that can store a **collection** of **key-value** pairs
- Defined using curly brackets {key1 : value1, key2 : value 2, ...}
- A **colon** (:) separates each **key** from its associated **value**
- Each key-value pair is separated by a **comma**

***Exercise(6a):** Get the capital of each countries using the object countries*

**Want to know more about objects?**



# Objects

- Add a new key-value pair: `countries.taiwan = "taipei"`
- Update existing key-value pair: `countries["great britain"] = "london"`
- Remove existing key-value pair: `delete countries.sweden` (delete key)

**Exercise(6b-e):** *Add a new key-value pair to countries  
Update existing key-value pair  
Remove existing key-value pair  
Print all countries and capitals in your countries-object*

# Blocks

Curly braces: { } are all over JavaScript.

Keeping track of them is important to get the code to work, but it can be burdensome.

Fortunately, many code editors can help adding indentation and finding missing braces before you get lost.

```
block 1
{
  block 2
  {
    block 3
  }
  block 2 continuation
}
block 1 continuation
```

[Want to know more about block statements?](#)

# Functions

- **What is a function?**
  - A block of code that will run when being called
- **Why do we want to use functions?**
  - Reuse code
  - A function usually performs one action, ie: add two numbers

How do we create a function? → See Runkit Notebook

**Exercise(7a):** *Create three functions that can subtract, multiply and divide two numbers*

# Conditions

In JavaScript, and all programming, mathematical conditions are widely used.

Equals: **a === b**

Not Equals: **a !== b**

Less than: **a < b**

Less than or equal to: **a <= b**

Greater than: **a > b**

Greater than or equal to: **a >= b**

# If statement

If statements often use a logical condition. Depending on the value of the variables we want the program to do different things.

```
let isRaining = true
```

```
if (isRaining === true) {  
  console.log("Take the umbrella!!")  
}
```

```
else {  
  console.log("Don't take the umbrella")  
}
```

# If, else if, else

```
let cats = 6

if (cats === 0) {
  console.log("No cats :( ")
}

else if (elephants < 5) {
  console.log("A very small amount of cats.")
}

else {
  console.log("Lots of cats!!")
}
```

# Tips & Tricks

- There is more than one way to solve the exercises
- Google is your friend
- Pair programming or “Rubber ducking”  
(talking is a big part of problem solving!)
- Ask questions

# More tutorials

<https://www.w3schools.com/javascript>

<https://www.khanacademy.org/computing/computer-programming/programming>

<https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/> (teacher:  
<https://www.youtube.com/watch?v=PkZNo7MFNFg>)

Try something more challenging

<https://edabit.com/challenge/MvZK536X7fyrWH8Qc>

<https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/basic-algorithm-m-scripting/find-the-longest-word-in-a-string>

<https://github.com/getify/You-Dont-Know-JS> (learn things more in detail)