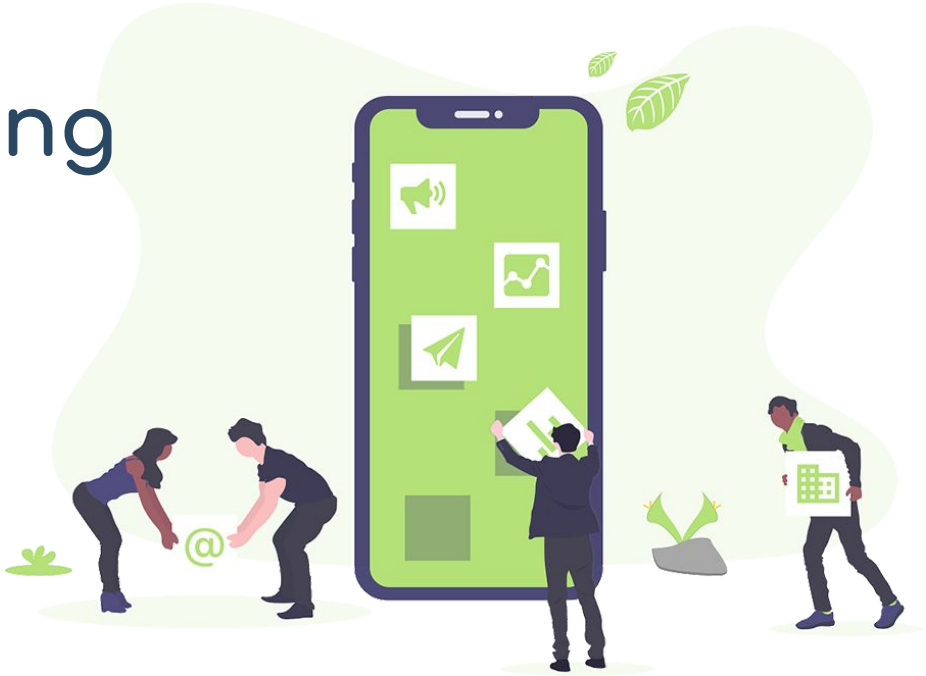
HELLO WORLD

Digital Skills & Coding

Coding for Non-Coders
Beiersdorf

16.09.2019



Agenda

1. <Hello World>
2. Computational Thinking
3. Theory & Terminology
4. Programming Robots
5. Coding & Languages
6. App Prototyping
7. You did it

Many people are unsure about what the digital future brings.

The lack of knowledge often leads to skepticism

Digitalization is way more than:

Lean Startup, Design Thinking & Agile Working.

Digitalization means

Software. Hardware. Technology.

People who understand how to use these new technologies are empowered and are able to navigate our changing world.

To understand the digital world it helps to experience it first hand.

Do it yourself + Knowing Concepts = Understanding Digitalization

In order to create digital products, processes and business models, we need to understand the technological side.

Your trainers

Dr. rer. nat.
Joachim Krois

Areas of interest
Coding, Predictive Modelling, Spatial Data Analytics, Artificial Intelligence and Computer Vision

References
Freie Universität Berlin, Charité – Universitätsmedizin Berlin, Berliner Institut für Gesundheits-forschung

Annemieke
Frank

Areas of interest
Co-Learning & Digitale Kompetenzen, Coding, Design Thinking, Gamification

References
Facebook, Volkswagen, Deutsche Bahn, Code+Design, Technologie Stiftung

Let's connect

https://etherpad.hello-world.academy/p/beiersdorf

Getting to know you

~ 1 min

- Tell us about yourself.
- Which software is most relevant in your day to day life?
- Have you ever coded before?
- What are your expectations for today?


Reality


Reality


Movie


Movie

14 °C | °F

Sonnig

Hamburg, Deutschland

Sonntag, 9:30

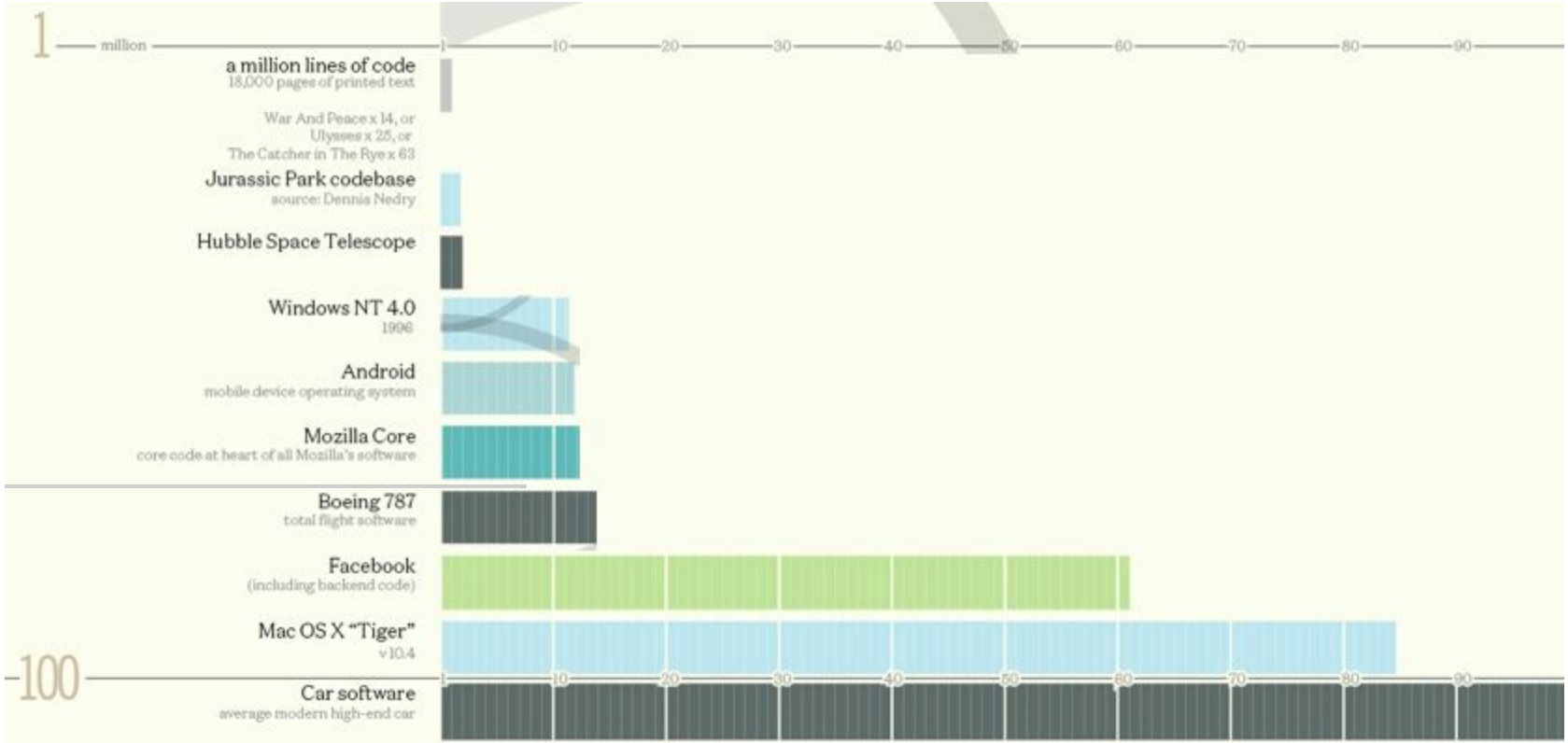
Niederschlag: 0% | Wind: 19 km/h | Luftfeuchtigkeit: 63%

6:53 19:37

Täglich Stündlich

Sonntag	Montag	Dienstag	Mittwoch	Donnerstag	Freitag	Samstag
19° 11°	15° 11°	13° 9°	14° 7°	16° 7°	16° 6°	18° 8°


60
5
55
10
50
15
45
20
40
25
35
21:16
12
9
6


1
million
1
10
20
30
40
50
60
70
80
90
a million lines of code
18,000 pages of printed text
War And Peace x 14, or
Ulysses x 25, or
The Catcher in The Rye x 63
Jurassic Park codebase
source: Dennis Nedry
Hubble Space Telescope
Windows NT 4.0
1996
Android
mobile device operating system
Mozilla Core
core code at heart of all Mozilla's software
Boeing 787
total flight software
Facebook
(including backend code)
Mac OS X "Tiger"
v 10.4
100
Car software
average modern high-end car

Agenda

1. <Hello World>
2. Computational Thinking
3. Theory & Terminology
4. Programming Robots
5. Coding & Languages
6. App Prototyping
7. You did it

> *Software is eating the world!*

Marc Andreessen
Entrepreneur and IT pioneer

> *To reading, writing, and arithmetic, we should add computational thinking to every child's analytical ability.*

Jeannette M. Wing
Professor for Computer Science

> *Everybody should learn to program a computer, because it teaches you how to think.*

Steve Jobs
former CEO Apple

> *Programming languages should be part of the curriculum. They are at least as important as multiplying, reading and foreign languages.*

Timothy Höttges
CEO Deutsche Telekom

The Bread Machine

1. Give the robot commands so that 3 same-sized pieces of baguette are cut.
2. Write down each command on a post-it.
3. Sort post-its in the right order.

The Bread Machine

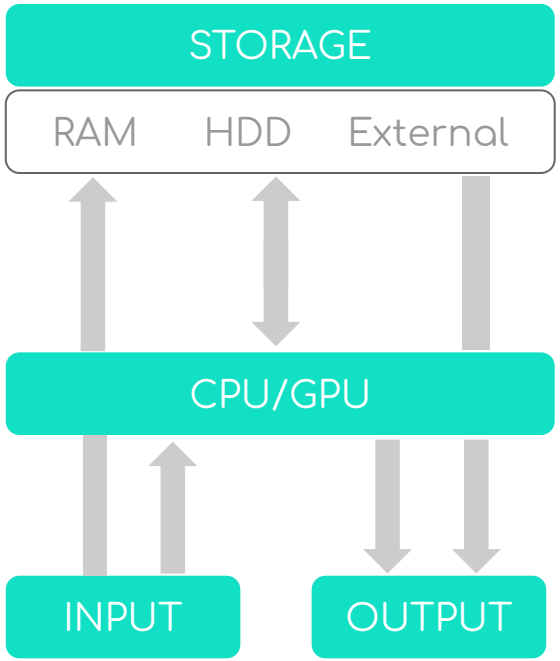
1. Give the robot commands so that 3 same-sized pieces of baguette are cut.
2. Write down each command on a post-it.
3. Sort post-its in the right order.
4. Create a mockup with (a) the type of bread, (b) the number of pieces & (c) the thickness of the slices.

Agenda

1. <Hello World>
2. Computational Thinking
3. Theory & Terminology
4. Programming Robots
5. Coding & Languages
6. App Prototyping
7. You did it

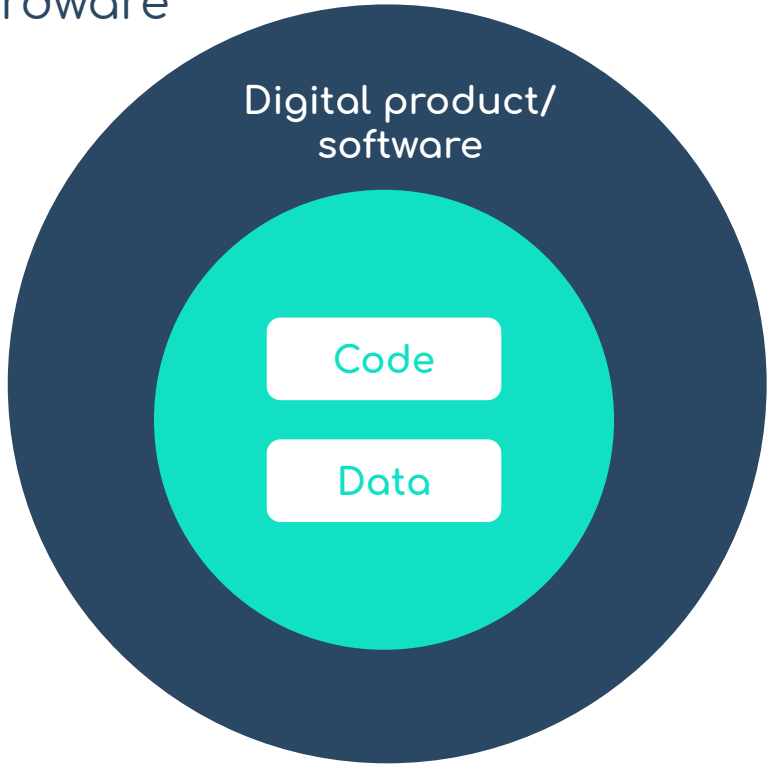
HARDWARE

- Hardware = physical components of a computer
- The CPU (Central Processing Unit, prozessor) executes software
- The GPU (Graphical Processing Unit)
- Storage saves data
- Input: Mouse, keyboard, touchscreen ...
- Output: Monitor, sound, vibration ...
- Analogy: Hardware = skeleton



SOFTWARE

- Software executes commands on hardware
- **Data** is stored information
- **Code** is text-based commands
- Analogy: Software = brain/nerves



FUNCTIONS

- Reusable collection of commands

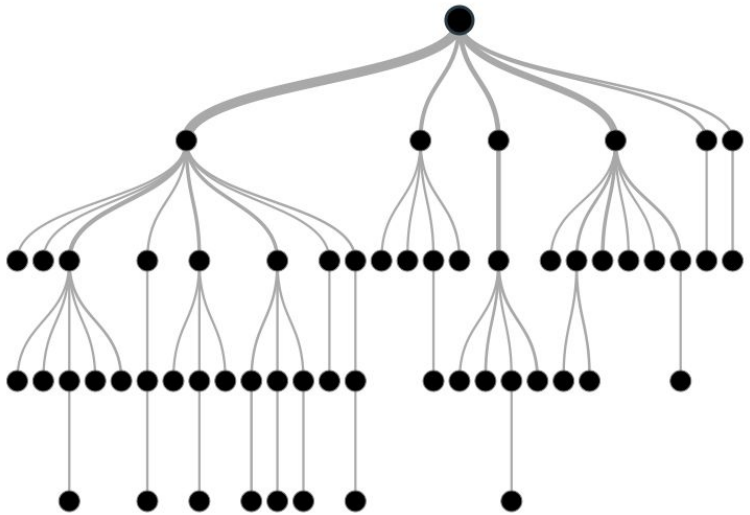
```
function roll (time, speed, dir) {
   setDirection(dir);
   setSpeed(speed);
}

function order(type, client) {
   var pizza = bake(type);
   deliver(pizza, client);
   …
}
```

FLOW CONTROL

- Commands are executed when conditions are met.

If-else statement

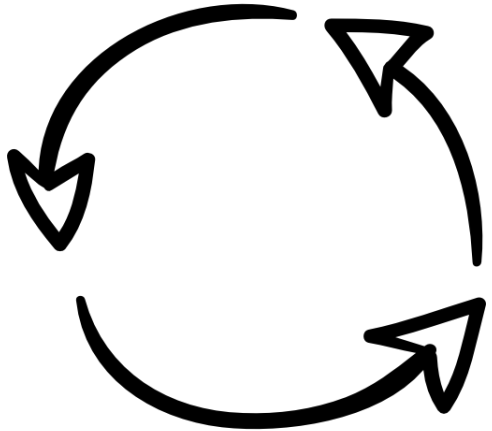
```
if (colour == green) {
    Sound(Boing);
}
if (Waiter.ready == Yes) {
    order();
}
```



LOOPS

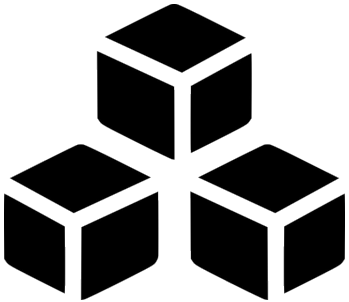
- Commands are repeated as long as condition is met

```
Var speed = 5;
while (speed <= 100) {
    speed = speed + 5;
    role (time, speed, dir);
}

while (orders < hunger) {
    order();
}
```



LIBRARIES / MODULES

- Collection of functions and commands



FRAMEWORKS

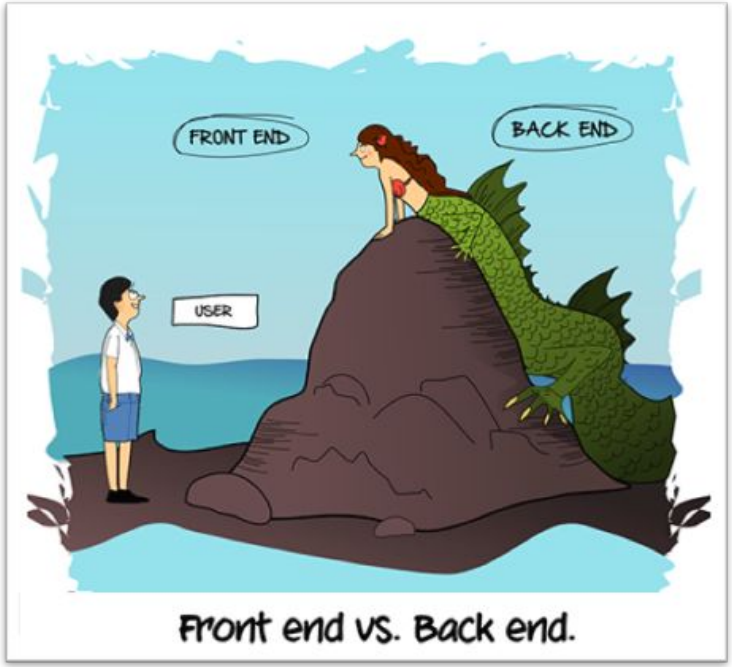
- Collection of functions, commands and rules
- Framework are dependent on language- and application ...

SOFTWARE PHASES

- The software is developed/coded during the **development phase** (programmer's task).
- Once written, the software is executed at **runtime** (user task).
- Analogy: Car during manufacturing & during life cycle.

FRONTEND & BACKEND

- Front-end and back-end describe layers of IT systems
- Front-end is closer to the user and to data inputs, can contain logic
- Back-end is closer to data processing
- Pair of terms is context-related
- Analogy: mimic as front-end, thoughts as back-end

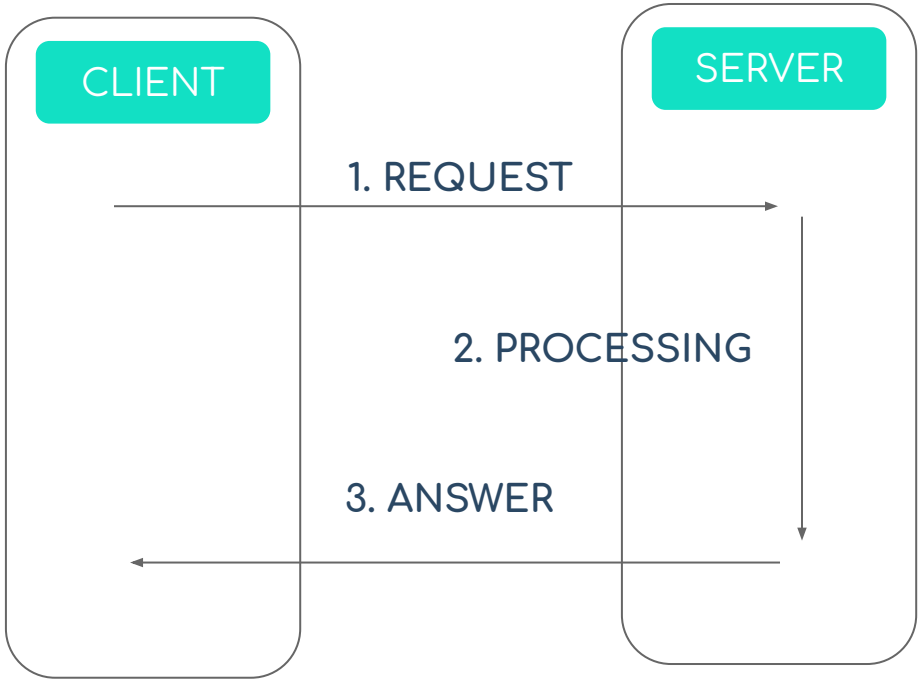


INTERFACE

- Interfaces provide defined functions to for accessing the IT system from an environment.
- Graphical interfaces (also GUI, Graphic User Interface) are used by humans.
- API (Application Programming Interfaces) used by other programs
- Analogy: Power cable in socket, stove-plate and pot and lid

CLIENT & SERVER

- Client-Server describes the distribution of tasks between two software systems.
- A physical device can be client and server at the same time.
- The term pair is context-related
- Analogy: Pizza customer is client, employee is server



LAYERS OF SOFTWARE DEVELOPMENT

- Software is built using programming languages.
- Modern software applications are built upon a stack of different tools and technologies.
- High-level programming languages, such as Python or JavaScript, offer layers of abstraction and provide meaningful concepts.
- Assembler code is very close to the hardware but is cumbersome to read and write.

Agenda

1. <Hello World>
2. Computational Thinking
3. Theory & Terminology
4. Programming Robots
5. Coding & Languages
6. App Prototyping
7. You did it

Roboter Coding

Sphero
Spark+

Roboter Coding

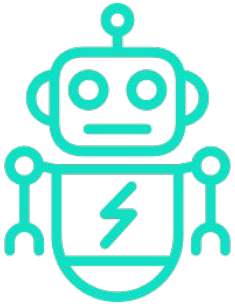
WHY

In order to thrive in the 21st. Century promoting the right skills will play a major role in the success of individuals and corporates.

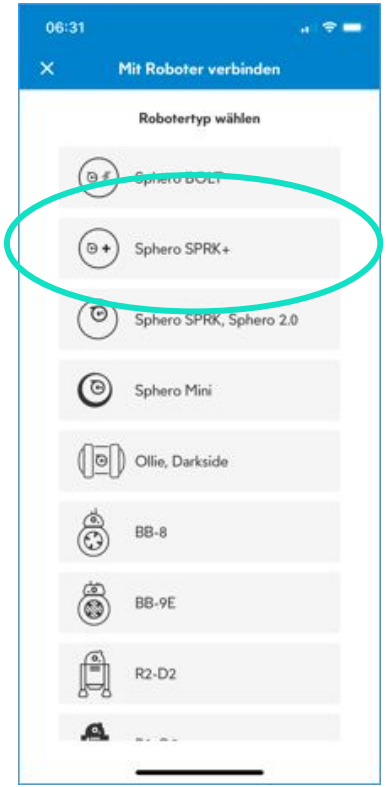
"To be as productive as it could be, this new automation age will also require a range of human skills in the workplace, from technological expertise to essential social and emotional capabilities."

McKinsey - Video:
The digital future of work:
What skills will be needed?

Roboter Coding

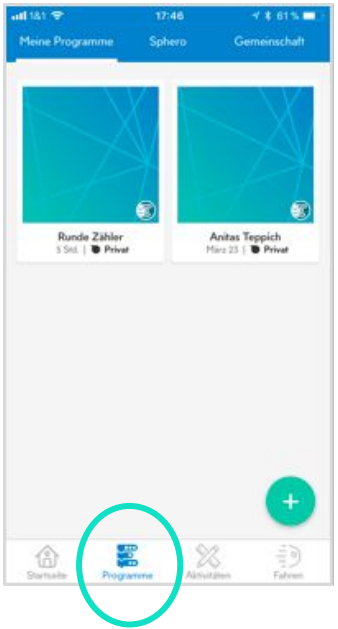


Sphero
Spark+



Roboter Coding

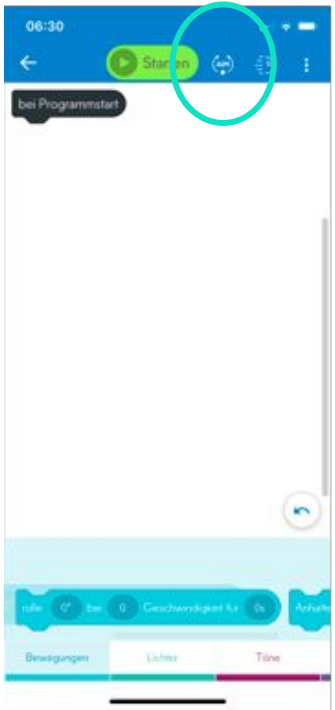
Click
Programmes



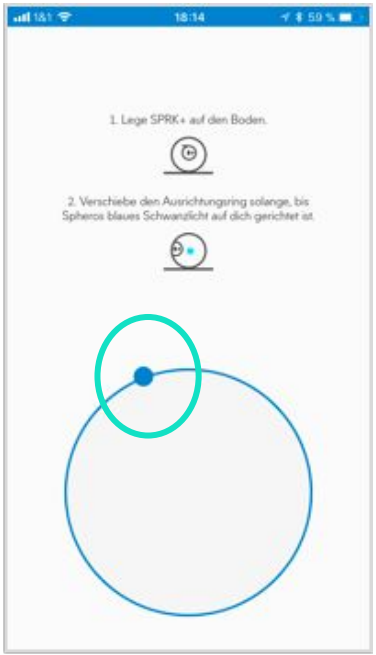
Click „+“ to create a new program and choose “Block“.

Roboter Coding

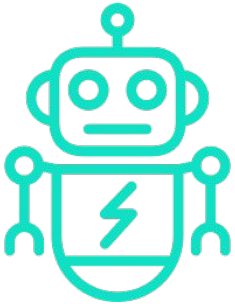
Click „Aim“ to callibrate robot.

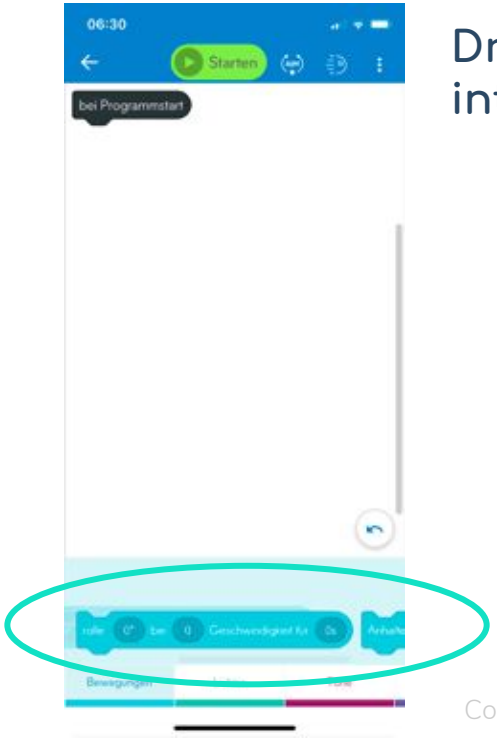


Rotate the blue dot on the smartphone so that your robots blue dot points to you.



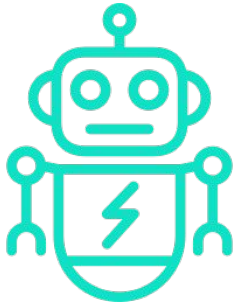
Roboter Coding





Drag & Drop commands into the interface.

The Parcours



- Turn the light to green.
- Drive along the course.
- After the first curve, change the color to red.
- Play a sound after impact.

Agenda

1. <Hello World>
2. Computational Thinking
3. Theory & Terminology
4. Programming Robots
5. Coding & Languages
6. App Prototyping
7. You did it

Coding Origins are Female

Ada Lovelace
(1815-1852)

Developed together with Babbage the preliminary stage to the first programming language.

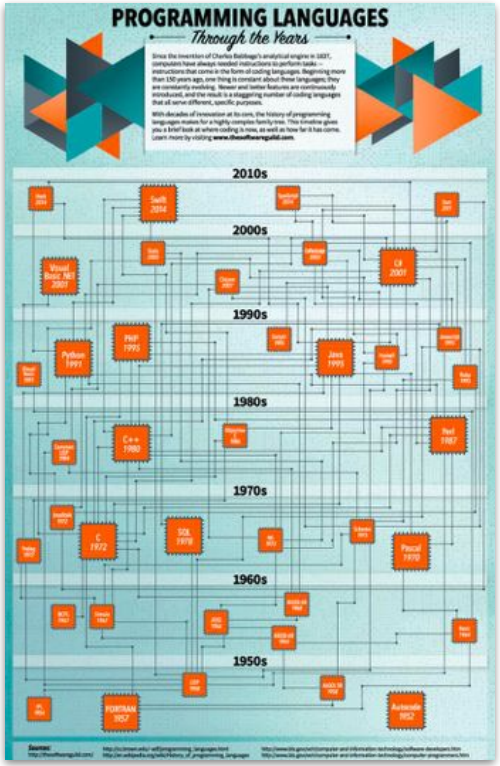
Grace Hopper
(1906-1992)

Involved in the development of the first compiler. This translated source code into a machine code understandable for the processor.

Margret Hamilton
(1936-)

Led the software team on the Apollo 11 mission. And she also established programming principles that are still valid today.

Programming Languages



With decades of innovation at its core, the history of programming languages makes for a highly complex family tree.

This timeline gives you a brief look at where coding is now, as well as how far it has come.

RUBY

```
MyLittleVar = "Hello World!"

5.times{puts MyLittleVar}
```

PYTHON

```
MyLittleVar = "Hello World!"

for _ in range(5):

    print(MyLittleVar)
```

JAVASCRIPT

```
var MyLittleVar = "Hello World!";

for (var x = 1; x <= 5; x++)

    {

    alert(MyLittleVar);

    };
```

JAVA

```
class MyExample {
    public static void main(String[] args)
    {
        String MyLittleVar = "Hello World!";
        for (int x = 0; x < 5; x++)
        {

System.out.println(MyLittleVar);
        }
    }
}
```

C/C++

```
#include<stdio.h>
int main() {
    char MyLittleVar[] = "Hello World\n";
    int x = 0;
    for (x = 0; x < 5; x++)
    {
        printf("%s", MyLittleVar);
    }

    return 0;
}
```

ASSEMBLER

```
section .data
    MyLittleVar  db 'Hello World!', 10
    length equ $ - MyLittleVar;

section .data
 start:
    mov cx, 5 ; fill cx-register with 5
 loop:
    mov eax, 4 ; write(stdout, hello, length)
    mov ebx, 1
    mov ecx, MyLittleVar
    mov edx, length
    int 80h

    loop schleife ; jump to 'loop' as long as cx > 0
and decrease cx by 1
    mov ebx, 0  ; Call: exit
    mov eax, 1
    int 80h
```

MACHINE CODE

```
01001000 01100101 01101100 01101100 01101111 00100000
01010111 01101111 01110010 01101100 01100100 0001010
01001000 01100101 01101100 01101100 01101111 00100000
01010111 01101111 01110010 01101100 01100100 0001010
01001000 01100101 01101100 01101100 01101111 00100000
01010111 01101111 01110010 01101100 01100100 0001010
01001000 01100101 01101100 01101100 01101111 00100000
01010111 01101111 01110010 01101100 01100100 0001010
01001000 01100101 01101100 01101100 01101111 00100000
01010111 01101111 01110010 01101100 01100100
```

=

Hello World Hello World Hello World Hello World Hello World

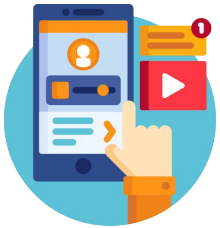
Let's Play

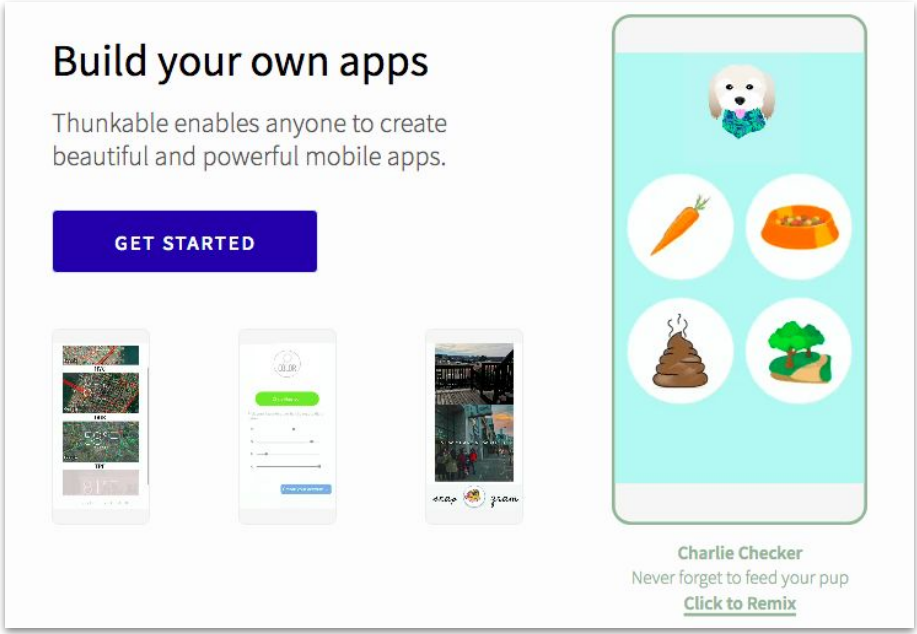
Agenda

1. <Hello World>
2. Computational Thinking
3. Theory & Terminology
4. Programming Robots
5. Coding & Languages
6. App Prototyping
7. You did it

App Prototyping

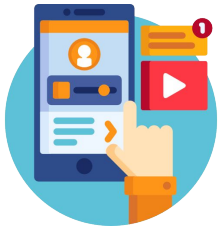
thunkable





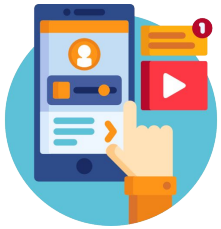
- Block-based visual programming language.
- Lets users create programs by manipulating program elements graphically rather than by specifying them textually.

App Prototyping



- Go to www.thunkable.com and sign up
- Download Thunkable App
- Open the Thunkable Live app and log in
- On your computer, click the "Live Test" button
- When you make changes to your app on the computer, they will update on your mobile device.

App Prototyping



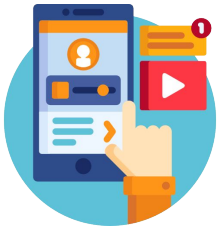
DESIGN

Here we will insert the components.

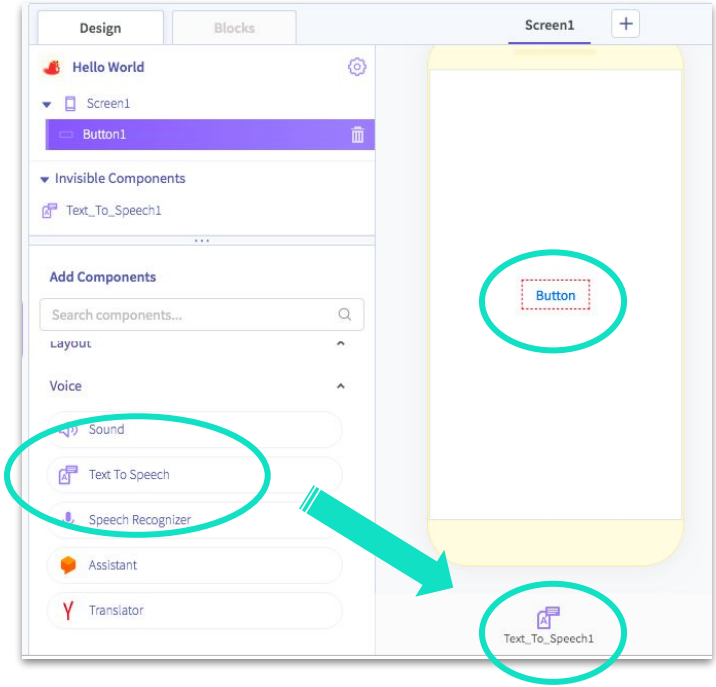
BLOCKS

Here we will code using block-coding.

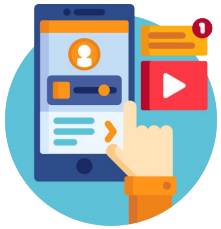
App Prototyping



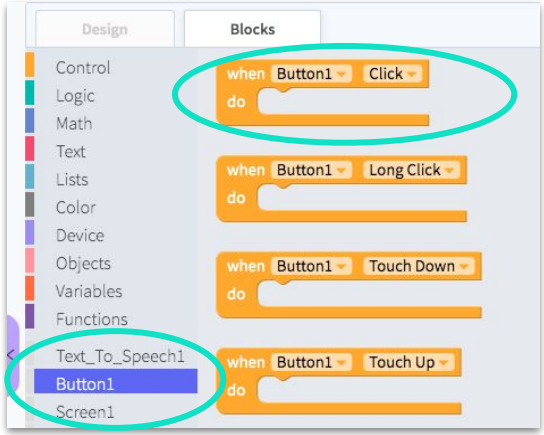
- In the Design-Area add the following two components: Buttons & Text to Speech



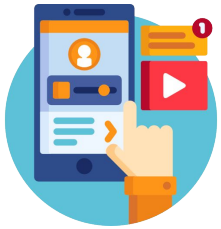
App Prototyping



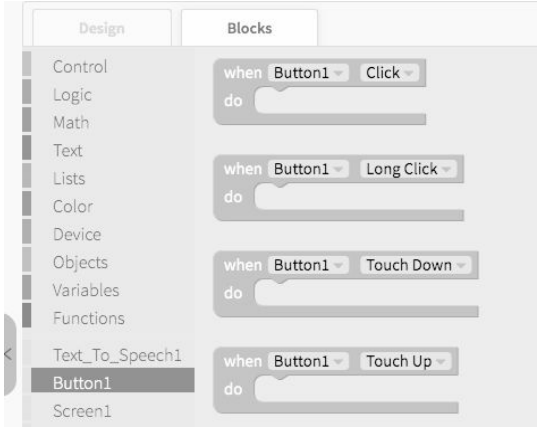
- In Block-Area on the left side click on Button1.
- Drag & drop the when Button1 Click function onto screen.

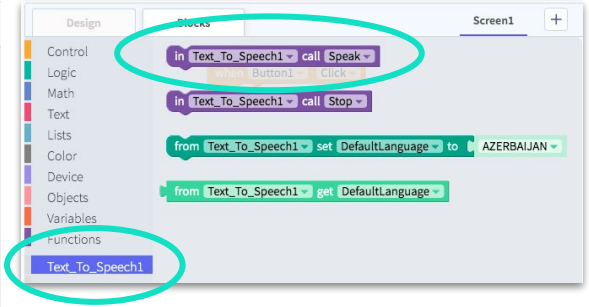


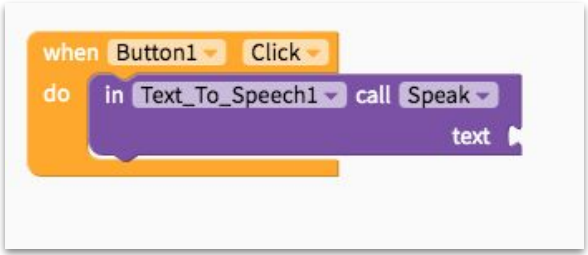
App Prototyping



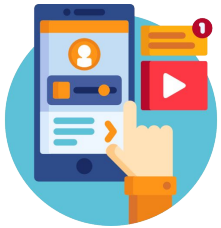
- In Block-Area on the left side click on Text_To_Speech1.
- Drag & drop the in_Text_To_Speech1 function into yellow brackets.



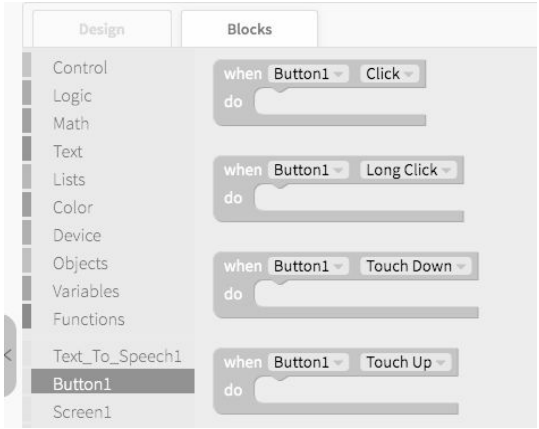


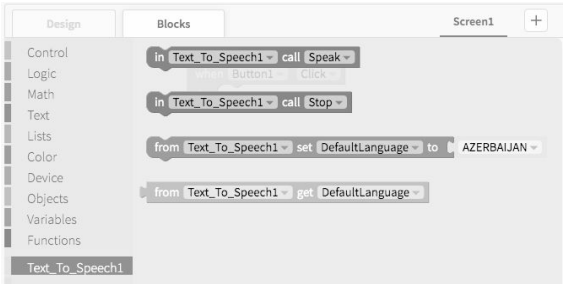


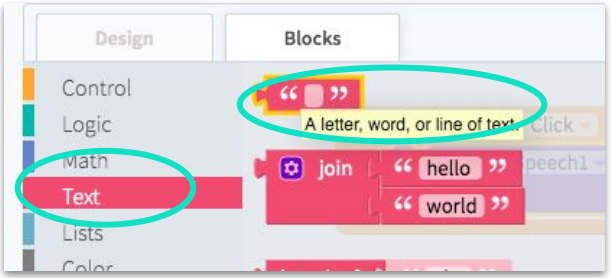
App Prototyping

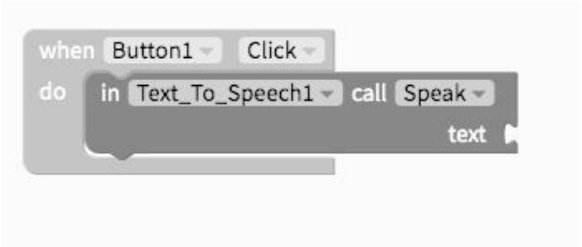


- Im Block-Modus click on Text and choose field with “ ”
- Add next to purple text funtion



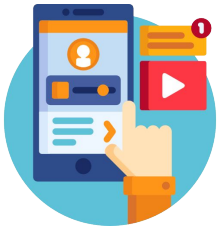


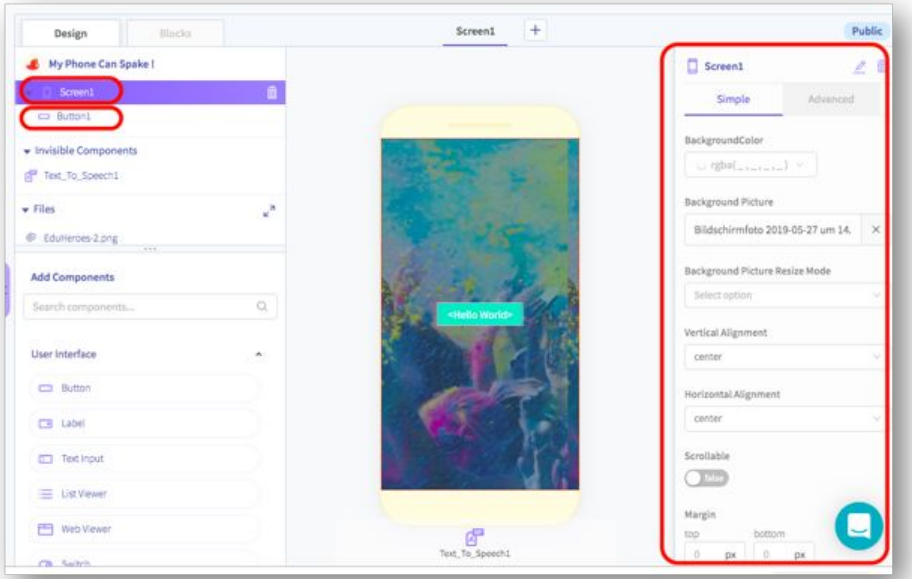




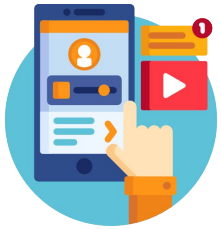
App Prototyping

- And now add some color.





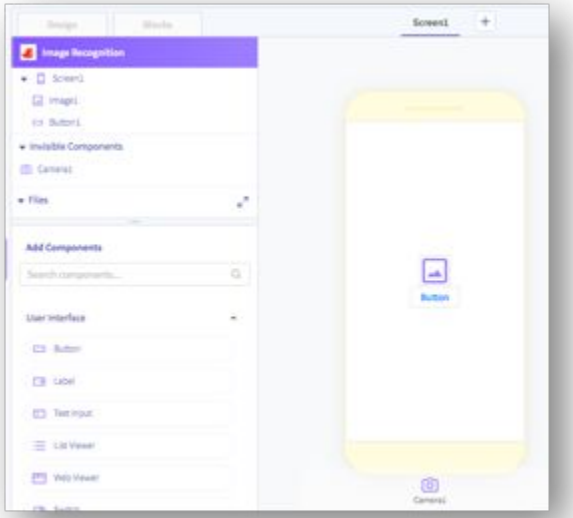
APP PROTOTYPING



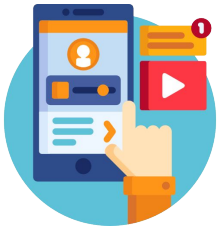
- Let's build an AI-based app

In the Design area add the following components:

- Button
- Image
- Camera



APP PROTOTYPING

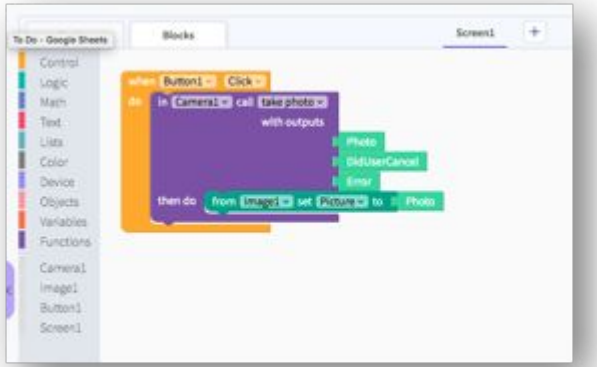


In the Design area add the following components:

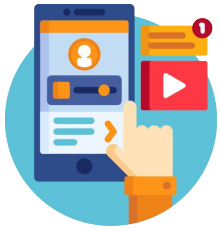
- Button
- Image
- Camera

In the Block area:

1. Button Function:
   When Button1 Click
2. Camera Function:
   In Camera1 Call Take Photo
3. Image Function:
   Image1 set Picture to
4. Move Photo to “ ! “

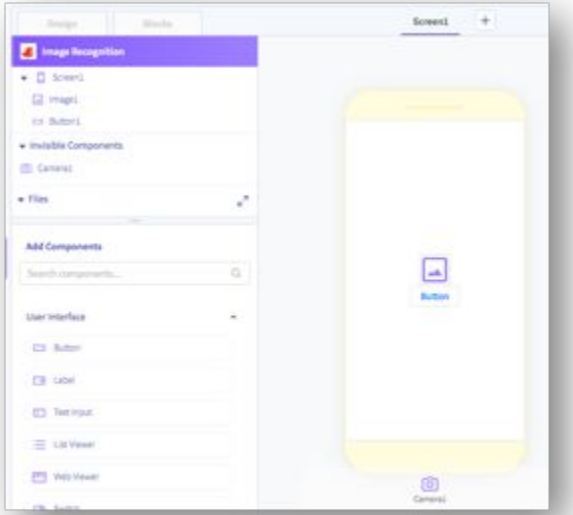


App Prototyping



In the Design area add the following components:

- Button
- Image
- Camera



In the Block area:

1. Button Function:
   When Button1 Click

2. Camera Function:
   In Camera1 Call Take Photo

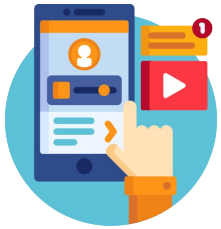
3. Image Function:
   Image1 set Picture to

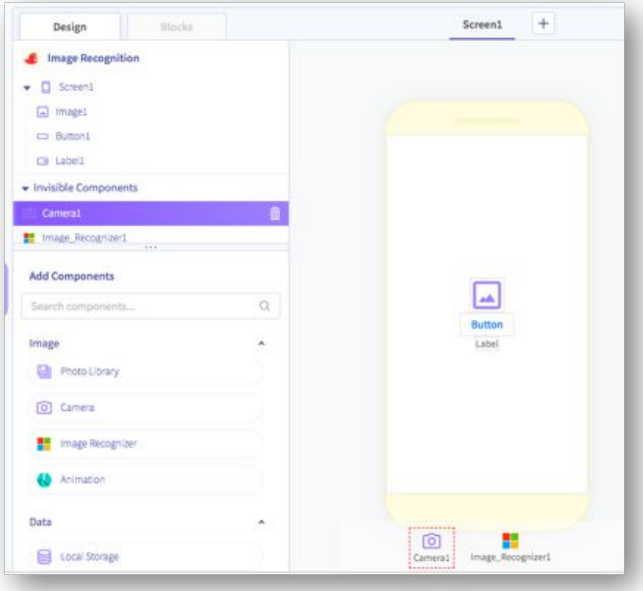
4. Move Photo to “ ! “

Open Thunkable App and test.

Test to see if the image on the screen is set to the picture that you took with the camera.

App Prototyping

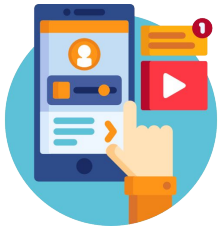




Im Design area:

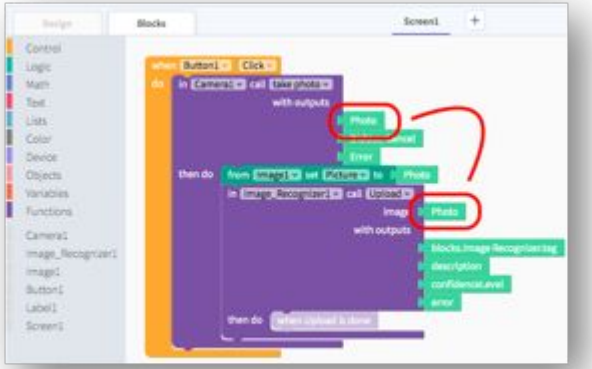
- Add component Label
- Move Label under Button
- Add component Image Recognizer

App Prototyping

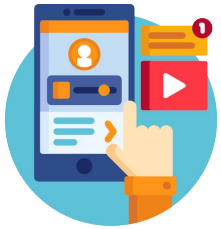


In Block area:

- Click Image_Recognizer1
- Drag & drop Image_Recognizer1 call Upload into the in Camera1 call TakePhoto bracket
- Drag and drop the "photo" block from the Camera1 block into the "image" socket on the Image_Recognizer1 block.

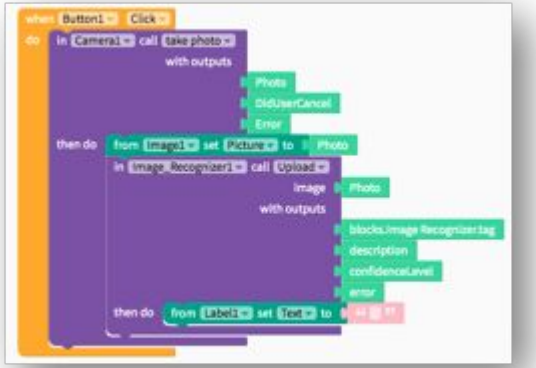


App Prototyping

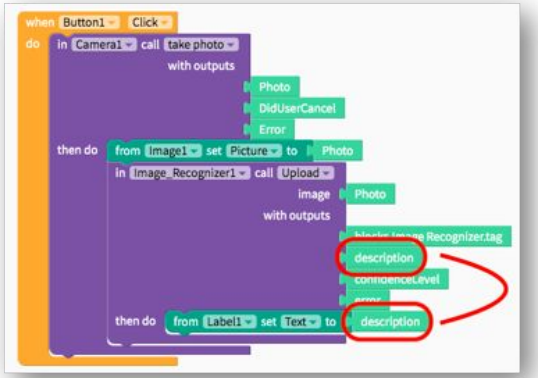


In Block area:

- Open the drawer for Label1.
- Drag and drop from Label1 set Text to block inside the in Image_Recognizer1 call Upload block.



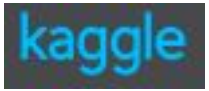
- Drag and drop the "description" block from the Image_Recognizer1 block into the opening of the "from Label1 set Text to" block.



Software check (optional)

Coding Resources

Online



Coding Resources

Bootcamps

Communities

Fun

Agenda

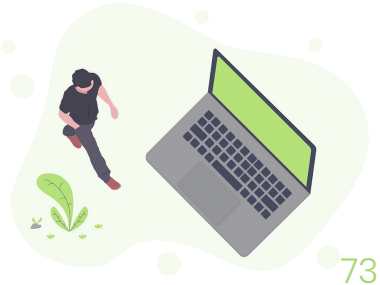
1. <Hello World>
2. Computational Thinking
3. Theory & Terminology
4. Programming Robots
5. Coding & Languages
6. App Prototyping
7. You did it

Wrap Up

- Did we meet your expectations?
- What went particularly well?
- What shall be improved?

Coming Up...

- A deep dive into Python
- Basics of Web Development (HTML, CSS und JS)
- Webscraping
- Excel was yesterday - Data Analysis with Pandas
- Automate the boring stuff
- ...



Annemieke Frank
Dr. Joachim Krois

www.hello-world.academy
hello@hello-world.academy